Abstract

BACKGROUND: Tangible user interfaces and end user development are two increasing research areas in software technology. Physical representation promote opportunities to ease the use of technology and reinforce personality traits as creativeness, collaboration and intuitive actions. However, designing tangible user interfaces are both cumbersome and require several layers of architecture. End user development allows users with no programming experience to create or customize their own applications. OBJECTIVE: This dissertation aims to ease development of tangible user interfaces by creating a toolkit, called RAPT (Rapid Arduino-Prototyping Toolkit), on the Arduino platform. RAPT will be developed as an Android application to support mobility and accessibility. The requirements of the toolkit are based on three scenario analysis; home assistance, restaurant and buddy notifier. RESEARCH METHOD: First, this dissertation will explore the underlying concepts of tangible user interfaces and end user development. Second, examine the state of the art within the domains of tangible user interfaces and end user development. Third, we will look at benefits of using a software toolkit when developing a tangible user interface through an end user development interface. To solve these enquiries, we have conducted a systematic mapping study of tangible user interface and end user development domains. Finally, requirement specifications have been formed based on the scenario analysis. MEASUREMENTS: To evaluate the toolkit, we have conducted a usability test with ten non-programmers as the target group. RESULTS: As a prototype, basic functionality is implemented and the usability test confirms that the toolkit is heading in the right direction by letting non-technical individuals create and modify simplistic tangible user interface applications. Main improvements are to support other hardware than Arduino and making it easier to add new devices. The usability test illustrates that parts of the toolkit's user interface can be hard to understand immediately, yet, it is easy to master.

Keywords: tangible user interface, physical user interface, end user development, end user programming, prototype development, arduino toolkit, toolkit, android, rapt.

Sammendrag

Fysiske brukergrensesnitt og sluttbruker utvikling er forskningsområder med økende fokus innen applikasjonsutvikling. Fysiske brukergrensesnitt fremmer mulighetene for å forenkle bruk av teknologi og fremheve personlige egenskaper som kreativitet, samarbeidsevne og intuitivitet. En utfordring med fysiske brukergrensesnitt er selve utviklingen. Den er både tungvinn og krever flere arkitektur- og design lag, som gjør det både kostbart og krevende å utvikle. Sluttbrukerutvikling kan gi brukere uten programmeringserfaring muligheten til å lage eller endre sine egne applikasjoner.

Masteroppgavens mål er å forenkle utvikling av applikasjoner som bruker fysiske brukergrensesnitt ved å lage et verktøysett til maskinvareplattformen Arduino. Vi har laget et verktøysett som kalles RAPT (Rapid Arduino-Prototyping Toolkit) som er en Android applikasjon. En Android applikasjon ble valgt på grunnlag av mobilitet og tilgjengelighet. Kravene til verktøysettet er blitt utformet ved hjelp av en analyse av tre scenarioer; "home assistance", "restaurant" og "buddy notifier". Denne masteroppgaven vil starte med å utforske de underliggende konseptene til domenene fysiske brukergrensesnitt og sluttbrukerutvikling. Deretter vil vi gjennomføre en systematisk undersøkelse for å samle den nyeste kunnskap om disse domenene. Videre vil denne masteroppgaven omhandle fordeler ved å bruke et ferdiglaget verktøysett ved utvikling av fysiske brukergrensesnittapplikasjoner gjennom et sluttbrukergrensesnitt. En scenarioanalyse vil så blie gjennomført for å forme kravene som stilles til verktøy-settet. Til slutt vil vi evaluere designet til verktøysettet gjennom en brukertest med 10 brukere uten programmeringserfaring som målgruppe.

RAPT er en prototype og har basisfunksjonaliteten implementert. Brukertesten bekrefter at brukergrensesnittet er på rett vei, til å la brukere uten programmeringserfaring lage og endre applikasjoner som bruker fysiske brukergrensesnitt. I forhold til funksjonalitet så bør det vurderes å implementere støtte for flere typer maskinvareplattformer enn Arduino, samt gjøre det enklere å legge til nye fysiske enheter gjennom verktøysettet. Brukertesten illustrerte at RAPT sitt brukergrensesnitt har forbedringspotensiale på spesifikke elementer, men at brukerne forstod konseptet veldig raskt.

Preface

This submission is Anders Palfi, Haakon Svendsen Sønsteby and Daniel Tandberg Abrahamsen's Master dissertation in Informatics at Norwegian University of Science and Technology (Department of Computer Science and Information Science). Our master specialisation is software.

The assignment was given by our supervisors Babak A. Farshchian and Monica Divitini, and is due June 2014.

We would especially like to thank Babak A. Farshchian for his engagement, guidance and feedback throughout the entire project, as well as giving us access to needed hardware and labs. We really appreciate your time and commitment.

Additionally we are grateful for Terje Røsand's help and time during user evaluation of the prototype application. Moreover, we would like to thank Simone Mora and Alfredo Perez Fernandez for help and advice regarding hardware.

Trondheim, May 30, 2014

Daniel Tandberg Abrahamsen

Anders Palfi

Haakon Svendsen Sønsteby

Terminology

Acronyms

Acronyms	
APK	Android Application Package
EUD	End User Development
PUI	Physical User Interface
RAPT	Rapid Arduino-Prototyping Toolkit
RFID	Radio-frequency identification
TUI	Tangible User Interface

Table 1: Acronyms

Definitions

Actuator: Type of device that handles digital output or environmental actions. Example: Led.

Android: "Mobile operating system developed by Google. Android phones typically come with several built-in applications and also support third-party programs. Developers can create programs for Android using the free Android SDK (Software Developer Kit). Android programs are written in Java and run through Google's "Davlik" virtual machine, which is optimized for mobile devices." [1]

Arduino: "Tool for making computers that can sense and control more of the physical world than your desktop computer. It's an open-source physical computing platform based on a simple microcontroller board, and a development environment for writing software for the board." [2]

Arduino board: The physical (hardware) microcontroller board.

Board representation: The representation, illustration, of an Arduino board within RAPT.

cpp-code: The code format (a variation of C++) that an Arduino IDE can compile to run at Arduino boards [3].

Device: A representation of a hardware device attached to an Arduino board. Example: RFID reader.

End User Development: "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 artifact." [Lieberman et al 2006] in [4].

Installation (of a RAPT app): The process of translating rules to a sketch, compile the sketch, and upload the compiled sketch to an Arduino board.

Modes: There are two modes in RAPT: drag and connect. Drag is used to move a device, while connect is used to draw arrows between devices to connect them and to open a device's configuration dialog.

RAPT: This dissertation's implementation of an Android application, which is a prototype toolkit that creates applications for Arduino.

RFID: "Automatic identification technology which uses radio-frequency electromagnetic fields to identify objects carrying tags when they come close to a reader." [5]

Rule: A connection of a configured sensor and one or several actuators (with one or more configurations).

Rule container: The white space, where the user can drag, drop and configure devices and create rules.

Sensor: Type of device that handles digital input, or physical input from a human or the environment. Example: RFID reader.

Sketch: Code (cpp-code) that can be compiled into code that can be run on an Arduino application. Usually mentioned together with "generator" that produces the sketch within RAPT.

Social computing: "Social computing has to do with digital systems that support online social interaction." Even though the contact is not directly directed towards another person, "online activity also count as social. Actions may not involve people we know, and may not lead to interactions, but nevertheless they are social because we do them with other people in mind." [6]

Tabletop computing: "The definition states that an interactive tabletop is a large surface that affords direct, multi-touch, multi-user interaction. (..) In tabletops, the display is the interactive surface itself." [7]

Tangible user interface, also known as physical user interface and graspable user interface: The crossing point, interface, between users and hardware. Unlike a graphical user interface, the interaction happens through physical objects or hardware. The input and the output device can often be the same. These terms are used interchangeably throughout the report.

Ubiquitous computing: "The word ubiquitous can be defined as existing or being everywhere at the same time, constantly encountered, and widespread. When applying this concept to technology, the term ubiquitous implies that technology is everywhere and we use it all the time." [8]

Uploading (a RAPT app): The upload (of compiled sketch) to an Arduino board.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Figure 1.1: Report structure: Introduction

1.1 Problem Definition

End User Development Toolkit for Developing Physical User Interface Applications

Physical user interaction (PUI), also called tangible interaction or tangible user interfaces (TUI) is an emerging field within human computer interaction. As opposed to desktop computing, PUI utilizes physical form factor and affordance in physical objects in order to support a more natural and fluid interaction with computer systems. PUI is inspired by ubiquitous computing and product design.

In order to develop good PUI we need to be able to construct PUI prototypes easily and rapidly and test these with real users. There is a growing number of hardware toolkits for constructing physical prototypes, e.g. the open source framework Arduino. What is missing is a well-constructed and easy-to-use software toolkit to bridge the gap between the hardware prototype and the software system to interact with. This dissertation will investigate different models for such a software toolkit. The expected deliverables are the following:

- An analysis of the underlying concepts of PUI.
- A state of the art survey of existing systems and toolkits.
- An open source software toolkit with focus on Arduino as the hardware prototyping platform.

- A prototype set of applications to demonstrate the utility of the toolkit.

All software will be written in Java for the Android OS and Arduino toolkit. All software will be released as open source under Apache 2.0 and as part of the UbiCollab.org project.

1.2 Problem Domain

The research area of tangible user interfaces (TUI) and the possibilities to extend the software centred domain, have the recent years increased by a huge extent. We will examine scenarios of how TUI, with Arduino as hardware prototyping platform, can utilize physical objects. Additionally, end users desire flexibility and customization of applications to a greater extent than before. The challenge is how end users can develop TUI applications without needing any programming expertise. Mobility and accessibility have also become wanted functionalities, if not requirements. Moreover, sharing and learning from the environment and peers are emerging through app stores, interactive information sharing and virtual environments.

To research and solve these enquiries, we are going to develop a prototype toolkit for non-programmers to create their own app running on an Arduino board. Moreover, to attend accessibility and mobility, the prototype will be a mobile application. To investigate the current state of TUI and end user development (EUD), we will conduct a pre-study through a systematic mapping. The pre-study will additionally explore how the toolkit can make use of current end user development techniques. Thereafter, a usability test will evaluate the software toolkit.

The problem domain covers end users (non-programmers) and their capability to develop applications for a programmable device given a proper tool. Explicit, the domain of the assignment: We will develop a toolkit as an Android application with Arduino as hardware prototyping platform.

The dissertation is part of the UbiCollab.org project. “UbiCollab (Ubiquitous Collaboration) is about supporting natural collaboration using mobile and ubiquitous computing technologies.” [9]. UbiCollab projects provide code as open source. These projects focuses primarily on Java programming using technologies such as Android. In addition, the focus of UbiCollab is device-centric, and most of the development is done on mobile devices.

1.3 Motivation

User interaction through tangible interfaces is an increasing research area in software technology. Physical representation promotes opportunities to ease the use of technology and reinforce personality traits as creativeness, collaboration and intuitive actions [10], [11]. End user development aims to allow users to create, customize and tailor applications, in contrast of hiring a professional software developer. Demanding users and a vast increase of data availability have made it

important to adapt applications to users and not the other way around [12]. Thus, this dissertation will investigate opportunities for a non-programmer and how he or she can benefit and make use of physical objects in an end user environment. The main focus is to create a toolkit for non-programmers to easy development of physical user interaction applications.

1.4 Research Questions

First, we will explore the underlying concepts of tangible user interfaces and end user development. Second, examine the state of the art within the domains of tangible user interfaces and end user development. Third, we will look at benefits of using a software toolkit when developing a tangible user interface through an end user development interface. We will conduct a systematic mapping study of tangible user interface and end user development domains. Finally, scenario analysis will contribute to form requirement specifications. To evaluate the toolkit, the team will conduct a usability test with non-programmers as the target group. The pre-study aims to answer the research questions presented in Figure 1.1.

Research Questions	
RQ-1	In which fields of social computing are tangible user interfaces developed or suggested, and what characterises these tangible user interfaces?
RQ-2	What are the characteristics, challenges and advantages of tangible user interfaces in social computing?
RQ-3	What types of user interfaces in end user development exist and what are their weaknesses and strengths?

Table 1.1: Research Questions

The research questions, the results from the pre-study, user evaluation and scenario analysis will together form this report.

However, after evaluating the results of the systematic mapping study, we decided not to pursue the element of social domain. Further explanation can be found in Section 3.2.1.4.

1.5 Methodology

1.5.1 Lean Development

1.5.1.1 Background

Lean was originally a manufacturing process developed by Toyota for use in their car making industry [13]. Lean software development derived from lean manufacturing process in the early 1990s by taking the industry specific elements and

adapting them to software development practices. “Lean targets to reduce the unnecessary overhead activities and outputs as well as wastes from the production line“ [14].

Five principles are specified in lean practice [14]:

1. Customer value – Focus on what the customer wants and how to make the product a success.

2. Value stream – Knowledge of the complete system is necessary to make the best decisions.

3. Flow - Minimize overhead and waste.

4. Pull – Pull work from a queue, not push.

5. Perfection – Cheaper to fix errors early than late.

A software development project consists of several phases. Situational specific knowledge is often forgotten by the authors [15]. Errors discovered in an early phase requires the authors to regain that specific knowledge, making the cost of errors that is undiscovered until late, high. This is why optimal flow and communication between each step in the development process important. Starting with a minimum viable product (MVP) will either prove or disprove the solution in the beginning of the development. No time and effort is wasted creating features that end up not being used due to structural, technical or architectural changes.

One of lean's key features is postponing decisions until latest possible time without affecting the rest of the production line [14]. The decisions can be made based on facts and experience, rather than assumptions and uncertainty. By maintaining a smooth flow, errors can be spotted and addressed, and quickly be sent back in the value chain to be solved. Implementing lean has a high entry barrier and does not only require identifying and implementing lean principles. It also requires an organisational change and commitment [15].

Lean is an agile development method, and in true agile spirit another key feature is the ability for early and continuous deliveries. This minimises possible misunderstandings during requirements gathering and any corrections early in the development process. Continuous deliveries enforce frequent communication between customer and development team, proving easy and quick access to any information needed by the team members. Progress in a development project is primarily measured in working software [16].

Lean acknowledges that few people are able to complete a task optimally the first time. Therefore, refactoring has high focus. The idea is to rewrite old code with new knowledge to optimise current code and improve code quality. Old code could also need refactoring as an effect of architectural changes or restructuring. The most efficient refactoring is done with knowledge of the code fresh in mind and lean encourages quickly and rapidly refactoring [17].

1.5.1.2 Approach

Figure 1.2 presents the working process in this dissertation. The team has adopted lean's principles, however, a flexible working policy has been applied to eliminate waste. All principles have been tailored in order to suit the team in a best possible way. An MVP was created to prove the different parts of the system and the connections between them. Figure D.1 is a screenshot of the first MVP. This provided confidence that the main base of architecture and technology worked, and focus was moved to expanding, refactoring and optimise the application.

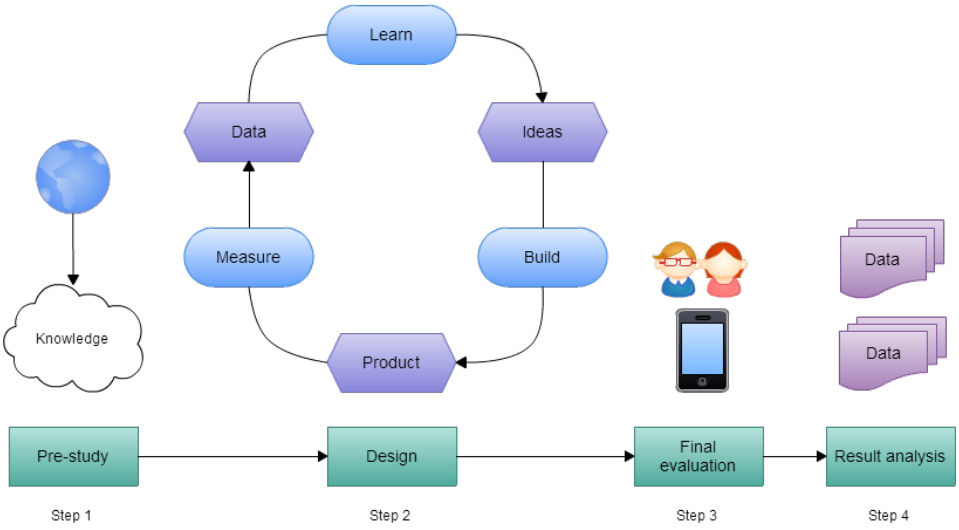

Figure 1.2: Working process

The perfection and flow principles have been assessed through working iterative. A total of eleven iterations have been completed. Two iterations on the systematic mapping, five development iterations and four report iterations. Each iteration has been between 2-4 weeks and included an overall goal. An example of an iteration goal is "Implement 1st draft of generator to translate from GUI elements to cpp-code". After each iteration, a prototype or a technical demonstration has been presented to the supervisor. The frequent demonstrations of the prototypes have minimised misunderstandings between the team and supervisor, and always ensured a working product available. Customer value is based on the supervisor's feedback after each iteration, in addition to the problem definition. Due to unknown aspects in the problem definition, the value stream has been in constant development when acquiring new knowledge.

The team created a lean board for the project through the issue tracking software JIRA [18]. This helped organizing and keeping track of both technical tasks related to development, and research and report tasks. The team members picked their desired tasks. In addition to using Jira, the team held short daily meetings during the development periods. These short meetings can be compared to the

agile development method, SCRUM's stand-up meetings.

1.5.2 Final Evaluation Approach

The final evaluation was a usability test. The test group consisted of ten subjects in the age from 19-49 years old. The location of the usability test was in a lab with two remotely controlled cameras and sound recorders. The equipment was used to record the subjects. Each subject was interviewed before the test to collect demographic data. The test consisted of six tasks with an increasing rate of complexity. Observations made during task executions were presented and discussed in the evaluation interview. The subjects were encouraged to explain their experiences during the interview. Finally, the subjects answered a System Usability Scale (SUS) questionnaire. The results from the final evaluation are based on video and sound recordings, observations, task complete rate and the SUS questionnaire. For further details see Section 6.3.2.

1.6 Expected Results

From the systematic mapping pre-study, we expect to gain knowledge and understanding of the two fields of research; tangible user interfaces and end user development. It is also expected to acquire insight of existing toolkits and applications related to these fields, regarding to techniques and solutions selected. This knowledge, together with a requirements analysis, will create the foundation of a toolkit, allowing non-programmers to rapidly create tangible user interface applications through an end user development environment. The toolkit will be evaluated through a usability test where non-programmers from the target group are given a variety of tasks. The toolkit will be released as open source under Apache 2.0 license and as part of the UbiCollab.org project.

Expected Deliverables	
D-1	An analysis of the underlying concepts of PUI and EUD
D-2	A state of the art survey of existing systems and toolkits
D-4	A prototype of a toolkit (RAPT) based on requirement specification and related work
D-5	Evaluation of RAPT including a usability test

Table 1.2: Expected Deliverables

1.7 Report Outline

Chapter 2: Problem Elaboration

This chapter explores the domain of both tangible user interfaces and end user development. Furthermore, toolkit requirements are created based on three presented

scenarios.

Chapter 3: Research Study

This chapter contains a summary of the systematic mapping "Systematic Mapping Study of Tangible User Interfaces in Social Computing, and End User Development" that were done as a pre-study. Moreover, this chapter presents similar and relevant toolkits and ideas. This chapter discusses why the toolkits and ideas are relevant, in addition to what lessons can be learnt.

Chapter 4: Proposed Solution

This chapter presents the proposed solution RAPT (Rapid Arduino-Prototyping Toolkit) at a high level. In addition to explain the conceptual model of RAPT, this chapter describes RAPT's functionality. Design decisions are presented and explained.

Chapter 5: Development

This chapter contains technical aspects of the proposed solution RAPT (the implementation of an android application for developing Arduino applications). More specific, this chapter provides an overview of the architecture, including data flow and class diagrams. Lower level design of the different parts of the toolkit is presented. Furthermore, presumptions are explained.

Chapter 6: Evaluation and Validation

This chapter presents several steps of evaluation during the development process. Versions of RAPT from earlier iterations are presented, which is followed up by a description of a usability test conducted of the final version. Usability test results are presented, and recommendations are made on the basis of result findings. In addition, created apps for the scenarios are presented with screenshots.

Chapter 7: Conclusion

This chapter summarises the dissertation, and discusses results of the usability test while also adding reflections in hindsight. Moreover, recommendations for further work and development are presented.

Chapter 2

Problem Elaboration

2.1 Introduction

This chapter explores the domain of both tangible user interfaces and end user development. Furthermore, toolkit requirements are created based on three presented scenarios.

Figure 2.1: Report structure: Problem Elaboration

2.2 Background of Physical User Interface

The evolution of user interaction is continuously: From mid- 1960 with command based user interface (CUI) to 1970 with graphical user interface (GUI) and the latest type, physical user interaction (PUI) [19]. All three of these solutions are aiming for one goal: To help users interact with computers in a more intuitive and effective way. The idea of PUI is to take advantage of human's ability to sense, feel and manipulate physical environment [19, 20]. People are familiar with handling physical objects to accomplish tasks. Everyday life surround people with physical objects.

There are many definitions of PUI. [21] states the following: "This field of research (PUI) relies on tangibility and full-body interaction and gives computational resources and data material form". [22] divides PUI into 4 characteristics:

1. Physical representations are linked to the digital information.
2. Physical representations include mechanisms for interactive control.
3. Physical representations are perceptually attached to digital representations.

4. The physical state of the objects are associated with the state of the digital system.

[23] explains PUI with the iceberg metaphor. Only a small percentage of the iceberg's mass is visible over the surface. The rest of the mass, is below the surface. PUI technology can be viewed the same way. Most of the functionality is hidden from the user, and only a limited set of objects are available for interaction. How the iceberg looks below the surface is irrelevant for the user.

The key idea of PUI is to manipulate digital information through physical interaction [22]. People generally have a natural touch with physical objects. These objects represent digital information and create new possibilities compared to command and graphical user interfaces. "The intuitive perception of the tangible products could help to reduce spatial cognition load and thus enhance design creativity" [24]. Ubiquitous computing is the name of the research area where all these acronyms belong.

One of the major challenges with PUI is how to integrate it seamlessly into people's daily life. When adapting a system based on the users' routines and movements, the user will not experience the system as a burden, rather a useful and intuitive tool to help to perform a task. The information flow in a general PUI system can be simplified as follows: The user sends an input through a physical object. The input is being processed by the underlying computer. In case of a feed back or response, output is returned to the user. For example, a user claps, and the information is sent to a computer (e.g. "John clapped"). Based on the information and configuration set up within the system, the computer replies with turning the lights on in John's room. Distinguishing between input and output is one of the biggest challenges when developing PUI applications [22].

Due to the tight coupling between the physical and digital world, a downside of PUI is the challenging process of developing a good system [19]. Advanced sensors and feedback systems are necessary to realize the tight coupling. A typical factor for success in PUI applications are the metaphor and analogy to the physical objects used by the user to communicate with the system. Due to the challenges, many PUI systems have a fixed location or context requirement. Tabletops are an example of this. Physical user interaction is not necessarily the best way to handle interactions with the user. Some examples of aspects that should be taken into consideration: type of task, group of users, evaluation of efficiency, and intuitiveness [23]. When creating a physical user interface, the designers are facing a new set of requirements compared to the development of a traditional GUI. While GUI is usually targeting one single user, with a single device, at one time, PUI is trying to target several users concurrently, multiple sensors, displays, and input types, all at the same time [25]. "The computer is embedded into the environment, becoming invisible" [22].

2.3 Background of End User Development

According to Lieberman et al "The main goal of End-User Development is to study and develop techniques and applications for empowering users to develop and adapt systems themselves" [26].

Furthermore, Lieberman et al, defines EUD 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 or modify a software artifact" [27].

"End-User Development (EUD) is a study within the field of computer science and human-computer interaction which explains the activities or techniques that allow amateur developers to produce or modify a software artifact. Early attempts in End-user development were concentrated in adding simple scripting programming languages to expand and familiarize oneself to an existing application, such as an office suite" [28].

"According to Nardi, three criteria are essential for end user programming to achieve a success level. The following three criteria are considered to be essential" [28]:

- A visual environment.
- A tack specific language.
- Support for collaborative practices.

Even though the criteria above is met, end user development can suffer from disadvantages. For instance, it generally produces narrow inflexible systems, can cause loss of quality of data, duplication of effort and incompatibles that can prevent sharing [28]. All of the mentioned factors, both success criteria and disadvantages, must be carefully considered and remembered when creating end user programming systems.

2.4 Requirements Analysis

2.4.1 Introduction

Three scenarios have been used to create requirements for RAPT. This section presents all three scenarios. Each scenario is described, and requirements for each are created. At the end of the section, an overview of all requirements for RAPT is presented.

2.4.2 Home Assistance

In today's society, many people need special care which often needs to be personalised. The target group in this domain consists of care takers and family members. Care takers and family members know their patients' needs, which make them the most viable users of the toolkit.

There are many possible usages for RAPT within the home assistance domain. This paragraph focuses on one specific application used to conclude the requirements for RAPT. A device that can sense when the stove is powered, in addition to being able to turn the stove off, is placed in the kitchen. A light bulb is placed on the wall above the patient's bed in the bedroom. An additional pressure sensor device is placed in the patient's bed. The application of these devices are when

Home assistance scenario requirements	
HAR-1	Power device can sense that stove is turned on.
HAR-2	Power device can turn the stove off.
HAR-3	Light device can turn light on/off.
HAR-4	Pressure sensor device that can sense someone laying down in bed.
HAR-5	Sending messages between devices.

Table 2.1: Home assistance scenario requirements

the stove is turned on, the light bulb device over the patient's bed lights up. If the stove is turned off, the light is also turned off. If the patient forgets to turn off the stove before going to bed, the patient will see the light above the bed and will hopefully be reminded of the powered stove. If the patient still does not turn the stove off before going to bed, the pressure sensor senses somebody laying on the bed. The pressure sensor then sends a message to the device in the kitchen, turning the stove off.

The requirements created for this scenario is found in Table 2.1.

2.4.3 Restaurant

Food serving at pubs, bars and restaurants often require customers to approach the counter to place orders. The target group in this scenario is employees that create and maintain the restaurant's menu, e.g. the restaurant owners. The restaurant owners should be able to create their own tangible ordering system.

Restaurant scenario requirements	
RSR-1	RFID reader device being able to read RFID tags.
RSR-2	Led device can both blink slow and fast.
RSR-3	Led device can be turned on for 5 seconds.
RSR-4	Screen device. Write text on screen device.
RSR-5	Button device that can sense if someone presses it for 3 seconds.
RSR-6	Pair different types of dishes to RFID tags.
RSR-7	Sending messages from RFID device to kitchen server.
RSR-8	Screen device receiving messages from kitchen server.
RSR-9	Led device receiving messages from kitchen server.

Table 2.2: Restaurant scenario requirements

A customer enters a restaurant, and wants to order food. The customer sits down at a table in the restaurant and can use small blocks to order food. On top of these blocks is a picture of the dish the block represents. Inside the block is an RFID tag. In addition to the blocks, the table includes two led devices in green and yellow colour, a small screen device, an RFID reader device, and a button device. A green led device will blink slowly to signal that the system is ready to take orders. The customer chooses which dishes he/she would like to order, and register the order by hovering the block over the RFID reader. When a dish

is registered, the led will increase blinking speed indicating that the order is in progress. For each dish registered, a message will be sent to the kitchen server containing a table number and dish number. When the kitchen server receives the order of a dish, it sends a message to the screen on the restaurant table displaying the dishes ordered. When all the dishes are registered, the customer presses the confirmation button for three seconds on the restaurant table. The green led will now have a constant green led signalling that all the orders have been placed. When the food is ready, the kitchen server sends a message to the yellow led turning it on for 5 seconds, indicating the food is ready. In addition, the kitchen server sends messages resetting the green led and the screen to default state (green led blinking slowly, and screen clear).

The requirements created for this scenario is found in Table 2.2.

2.4.4 Buddy Notifier

Pudge and Blitz are friends. Pudge is always struggling to keep track of time. They both live by the same lane and has decided to take the same bus. Blitz enters the bus two stops before Pudge. Due to his laziness, Pudge is often not on this bus. Blitz is starting to get annoyed and has implemented a system to help his friend. When Blitz walks from his house, he presses a button, which notifies Pudge by sound. Pudge now gets a reminder that Blitz has left the house and that Pudge needs to leave his house to be able catch the bus.

The requirements created for this scenario is found in Table 2.3.

Buddy Notifier scenario requirements	
BNR-1	Button device can register button clicks.
BNR-2	Speaker can play a sound.
BNR 3	Devices should be able to send messages between each other.

Table 2.3: Home assistance scenario requirements

2.4.5 Requirements Overview

2.4.5.1 Tangible User Interface Requirements

The toolkit's requirements can be split in two parts; end user development requirements and tangible user interface requirements. The tangible user interface requirements are both hardware required and the hardware's functionality. Tangible user interface requirements are found in Table 2.4.

2.4.5.2 End User Development Requirements

End user development requirements focus on functionalities that allow users to create tangible user interfaces. These requirements can be found in Table 2.5.

Note that not all requirements are directly extracted from the scenarios. For example, EUDR-5 is created to simplify the process of installing the menu to all

Tangible user interface requirements	
TUIR-1	Arduino microcontroller controlling devices.
TUIR-2	Power device can sense that stove is turned on.
TUIR-3	Power device can turn the stove off.
TUIR-4	Light bulb can turn light on.
TUIR-5	Pressure sensor device can sense someone laying down in bed.
TUIR-6	Sending messages between Arduino boards.
TUIR-7	RFID reader device can read RFID tags.
TUIR-8	Led device can both blink slow and fast.
TUIR-9	Led device can be turned on for 5 seconds.
TUIR-10	Small screen and the possibility to write text on it.
TUIR-11	Button device that can sense if someone presses it for 3 seconds.
TUIR-12	Send different types of messages based on what RFID tags are read.
TUIR-13	Devices should be able to send messages between each other.
TUIR-14	Devices should be able to send messages to external sources.
TUIR-15	Devices should be able to receive messages to external sources.

Table 2.4: Tangible user interface requirements

End user development requirements	
EUDR-1	Upload compiled sketches from Android to Arduino board through Bluetooth.
EUDR-2	Wireless communication between Arduino boards hidden from the user.
EUDR-3	Specify actions based on which RFID tag is read.
EUDR-4	Generate sketches based on user graphical GUI elements.
EUDR-5	Install compiled sketches on to several Arduino boards at once.
EUDR-6	App store which users can publish apps and loads apps.
EUDR-7	Load/save drafts when creating an app.
EUDR-8	Look up an Arduino board's IP address after an app has been installed.
EUDR-9	Specify IP address, port and body of messages sent to external sources.

Table 2.5: Functional requirements

tables is the restaurant. EUDR-6 is a requirement the team developed together with the supervisor. This feature will allow more advanced users to publish their apps and other users can use these apps for learning and inspiration. App store is explained further in Section 4.9. EUDR-2, EUDR-4, EUDR-7 and EUDR-8 are requirements aimed at improving usability by hiding technical aspects.

2.4.5.3 Non-Functional Requirements

The non-functional requirements for the toolkit can be found in Table 2.6. These requirements are based on the problem definition presented in Section 1.1.

Non-functional requirements	
NFR-1	Users should not need programming skills.
NFR-2	Users should be given proper feedback, and not be able to cause errors.
NFR-3	Provide an efficient way of creating tangible for non-programmers using an Android phone.
NFR-4	Provide flexibility to support further development and implementation of new devices, and hardware platforms.
NFR-5	Run on any Android device with Android version >= 4.3 (API level 18).

Table 2.6: Non-functional requirements

Chapter 3

Related Work

3.1 Introduction

This chapter contains a summary of "Systematic Mapping Study of Tangible User Interfaces in Social Computing, and End User Development" that was done as a pre-study. The entire study is attached in Appendix A. Moreover, this chapter discusses relevant toolkits and ideas.

Figure 3.1: Report structure: Related Work

3.2 Research Method

3.2.1 Systematic Mapping Study

3.2.1.1 Background

In order to acquire knowledge of the domains physical user interaction and end user development, the team conducted a systematic mapping study during the first semester. The full study is attached in Appendix A. Below is a summary of the findings and how it affected the work with RAPT.

"The main goal of a systematic mapping studies is to provide an overview of a research area, and identify the quantity and type of research and results available within it" [29]. While it is often neglected in software engineering, the method is often used in medical research [29]. The usual outcome is a visual map classifying the results. It requires less effort than a systematic literature review while providing a more coarse-grained overview [30].

The steps within in a systematic mapping study can be as follows [29]:

1. Definition of Research Questions
    - Outcome: Review Scope
2. Conduct Search for Primary Studies
    - Outcome: "All" possible papers that could be relevant
3. Screening of Papers for Inclusion and Exclusion
    - Outcome: Relevant Papers
4. Key wording using Abstracts
    - Outcome: Classification Scheme
5. Data Extraction and Mapping Process
    - Outcome: Systematic Map

Systematic mapping and systematic review are two acknowledged research methods for exploring a specific domain. Both study types share the aim of identifying research gaps, though with slightly different focus.

	Systematic Mapping	**Systematic Review**
Goals	Classification and conducting thematic analysis and identifying trends.	Publication for identifying best practices.
Process	Thematic summary, and not evaluation regarding quality.	Evaluated with focus on quality.
Breadth and Depth	More articles, due to no need to evaluate each article in detail.	More specific focus.
Classifying the Research Approach	Classification has to be high level.	Could be very specific (low level).
Validity Consideration	A high level evaluation could lead to a higher error rate. Can consider more papers.	Detailed evaluation of the research methodology.
Industrial accessibility and relevance	Easier to spark interest, visual appeal.	Focus on depth and empirically validated results. This gives higher importance for practitioners.

Table 3.1: Systematic Mapping comparison to Systematic Review

In accordance to [29], there are six main differences between a systematic mapping and a more commonly used, within software engineering, systematic review.The distinctions are illustrated in Table 3.1. Note that only the differences are listed, and that they have several universal elements not present in the table.

Our findings are that the study methods differ in terms of goals, breadth and depth. We have chosen the systematic mapping approach to better get a complete picture, though not necessarily in detail, of existing papers and toolkits within TUI and EUD.

3.2.1.2 Tangible User Interface Findings

The pre-study revealed the following: Tangible user interfaces are a new research area, 97 % of the articles were published later than year 2000, and the majority of documents were published in 2011. Tabletop is the most frequent researched type of tangible user interface, whereas children (and the support of children's learning) is the most focused target group. General characteristics are still not absolutely defined, however, some general trends were found. Real time feedback, one input/out channel and space-multiplexed are some mentioned characteristics that apply to several tangible user interfaces. Advantages and challenges are even more context dependent than characteristics. Still, as for advantages, tangible user interfaces tend to support creativeness, they are often and should be easy to use, they often provide collaborative benefits and perhaps the most important, the significance of the cognitive mind. Studies in the pre-study state that tangible objects have a bigger impact when users can relate to the object, and make use of already known knowledge. When a tangible object does not represent an intuitive or graspable idea, tangible user interfaces become harder to grasp and use. Except the latter described situation, there is none absolute challenges, but many elements to be aware of in different contexts. For example, how to standardize common functions, hardware limitations, seamlessness, lack of TUI modelling capability and that finger gestures may not be the best alternative compared to physical objects.

An aspect that should be emphasised is that the tangible research area is still evolving, and there is no final solution at the time being. Today's research in this field consists of analysis and empirical studies that can point studies, including this dissertation, in the right direction. The team should carefully select the type of physical interface to ensure to mitigate cognitive load. Interfaces with actual physical objects would be to prefer, and keep in mind that users are more creative when given the opportunity.

3.2.1.3 End User Development Findings

The idea to offer end users possibilities to create and customize own software is not completely new, however, the domain is increasingly in focus. The pre-study, in Appendix A, identified a great escalation of research in this domain from 2006.

End user development is a rapidly increasing area of research. Many of the studies specialises in providing interfaces for a specific, narrow, target group. In

general, children and elderly are heavily represented, and the desired outcome of prototypes are often increased learning experience or ease of everyday tasks for elderly or in smart-homes.

Two categories of interfaces stand out: Component based interfaces and customization based interfaces. Several techniques, within each category mentioned above, are considered viable, which means that selecting an interface type depends on context. With correct set of tools, users have no problems developing, tailoring or customizing their applications. The challenge is to provide the correct tools. Within end user development, component based interfaces and customization based interfaces are the most attractive. Both categories supports intuitive and easy-to-use interfaces which provide visual advantages. Additionally, there is no need for programming expertise or even knowledge, and it is easier to control and track legal and illegal operations which leads to giving users proper feedback while making user validation less complicated.

3.2.1.4 Refinement of Problem Definition

The systematic mapping study revealed that the social domain of tangible user interfaces still is at an early research stage. Tangible user interfaces were often directed towards handicapped users, children or elderly. Additionally, by applying both a tangible user interface and an end user development technique combined with Android development, the scope became large. We decided to not focus on the social domain, other than implementing a stubbed app store for sharing knowledge between users. However, Section 7.3 and 7.4 suggest several elements within the social domain.

3.2.2 Manual Search

In addition to the systematic mapping in Appendix A, the team has used Google Play [31] and Google Scholar [32] for manual searches. The reasons were to ensure that most relevant studies were found, even though they were not present in the databases/search engines of the systematic mapping, and to specifically search for existing toolkit applications.

3.3 Related Toolkits

3.3.1 ECCE Toolkit

The ECCE (Entities, Connections, Couplings and Ecologies) toolkit aims to "(...) seamlessly manage interactions between heterogeneous networked devices without necessarily having to deal with low level implementation details" [33]. ECCE uses a custom made XML to define the software logic. The software logic is used to handle the digital and physical input/output. This is a toolkit originally aimed to allow users to rapidly setup 'device ecologies'. This is "collections of devices interacting synergistically with one another, with users, and with the Internet resources" [33].

MyMemodules [34] is a very similar toolkit with the overall same architecture. The team has used both of these toolkits for inspiration.

The ECCE toolkit is relevant for RAPT in many ways. First, the overall goal similar to RAPT's goal. Second, the structure and flow in ECCE, by using the custom XML language to generate the software logic. Finally, end user development where users create the coupling between different devices themselves through an interface.

Lessons learnt: Other people are trying to solve a similar problem as RAPT. XML files to generate system logic and GUI is a flexible solution. An example of end user development where users configure the physical and digital input devices themselves.

3.3.2 GALLAG Strip

GALLAG Strip is a mobile application that combines physical interfaces with end user programming [35]. GALLAG Strip has a 'recording mode' where the system listens for senor events that is triggered by user actions. When an event is triggered, the user can configure appropriate actions through an intuitive GUI. This action will be executed next time the sensor is triggered. E.g. if GALLAG Strip is in recording mode and the TV is turned on. This triggers an event and the user can configure that a sound should be played.

GALLAG Strip is relevant for RAPT in two ways. The type of EUD technique with sensors and actuators used is very similar to the one the team has pictured RAPT will be based on. In addition, the concept of users creating rules is used. *Lessons learnt: The system listens for sensor events which users can configure a corresponding action to. The rule-based end user customization was proven very well by GALLAG Strip's user testing.*

3.3.3 Modkit

Modkit is a web-browser based tool for novice and experienced programmers. It provides the functionality to program an Arduino board using a graphical, block based, interface [36]. Each block represents a simplified Arduino programming command, very similar to the Scratch [37] programming environment. The blocks can be removed and regular source code displayed. This feature is intended for more experienced programmers. The codes is transferred to the Arduino board through a USB cable.

Modkit is relevant for RAPT in that it used a block based programming environment.

Lessons learnt: Block based programing is best suited for motivated end users with some technical interest.

3.3.4 Tangible Learning Framework

Tangible Learning Framework (TLF) "(. . .) enables teachers, caretakers, and therapists of disabled children to create tangible learning experiences for this target

group without the need for programming expertise“ [38]. The creation of these applications are done through a GUI on a computer. The EUD techniques are pattern based, component based and model based. The user selects the digital representation and binds this to a physical object, e.g. a RFID tag.

TLF is relevant for RAPT due to its complete implementation of EUD. Everything from linking the physical and digital objects together to create actions on these objects are implemented. TLF has received positive feedback over a three years evaluation period at a primary school. A useful feature for RAPT is to support registration of new RFID. TLF provides one possible solution.

Lessons learnt: Registration of RFID tags through a TLF application has in this case proven to be very user friendly.

3.4 Related Ideas

3.4.1 ArduinoCommander

ArduinoCommander is an Android application that allows the user to control an Arduino board through Bluetooth, Ethernet or USB [41]. The application consists of a WYSIWYG (What You See Is What You Get) interface where the user can configure the Arduino board through a mobile device in real time.

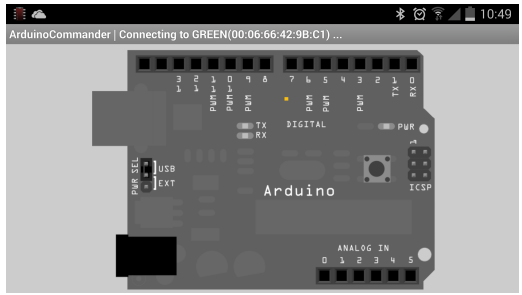

Figure 3.2: Screenshot of ArduinoCommander

The configuration is a graphical user interface that presents many of the features the Arduino Programming language offers trough the Arduino IDE. The most common usage is to upload the Arduino code through the IDE and then control it by using ArduinoCommander. Figure 3.2 displays a screenshot of the application.

The application is relevant for RAPT mainly because of its concept: To program an Arduino board through a mobile device. The fact that Bluetooth is a possible communication protocol between mobile device and Arduino board is also very interesting.

Lessons learnt: Existing possibilities regaring communication between mobile device and an Arduino board, in addition to how one end user development interaction is designed when configuring an Arduino board.

3.4.2 Amarino

Amarino is a toolkit that allows the user to connect the phone to an Arduino board and use the features the Arduino library provides in real time [42]. The library is developed by Amarino and consists of three modules. First, is the Bluetooth Manager. It provides the possibility to pair and connect the phone with an Arduino board through Bluetooth. Second, the EventManger offers handling of both internal phone events, collection of events and custom events. For example phone receiving an SMS or an alarm goes off, can easily be selected and configured. The configuration can then be connected to an action on an Arduino board. Third and finally, is the Monitoring module. This module is very useful for debugging and allows the developer to see received messages and if the events are sent. 3.3 presents a screenshot of the application.

The application is relevant for RAPT because of its concept: To program an Arduino board from a mobile device through Bluetooth. The end user interaction design can be used as inspiration.

Lessons learnt: One example on how to integrate an end user development environment, Bluetooth as communication protocol provides flexibility.

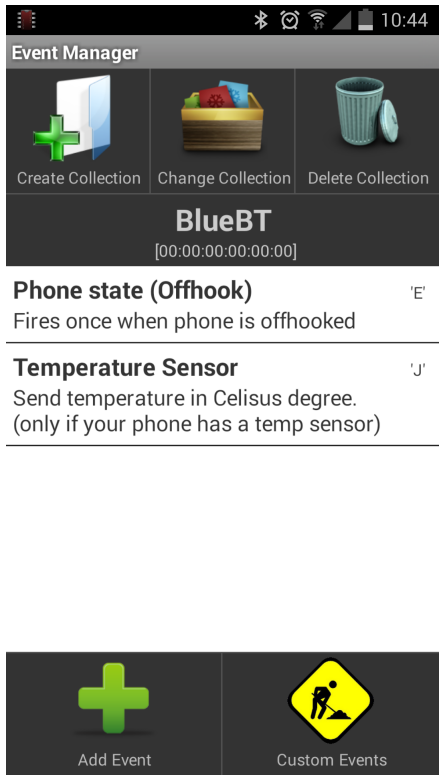

Figure 3.3: Screenshot of Amarino

3.4.3 ArduinoDroid

ArduinoDroid is an Android application that provides functionality to program an Arduino board through an Android phone [40]. The user uses the built-in keyboard to write Arduino code and upload through USB. Offered functionality is, offline opening, editing with syntax highlighting, compiling and upload of Arduino sketches. The user interface is similar to the Arduino Sketch IDE. Figure 3.4 displays a screenshot of the application.

ArduinoDroid is relevant to RAPT mainly because of the concept of programming an Arduino board on Android.

Lessons learnt: Not very user-friendly to write code on a mobile phone. Compiling Arduino sketches on an Android phone might set limitations on phone storage due to size on SDK.

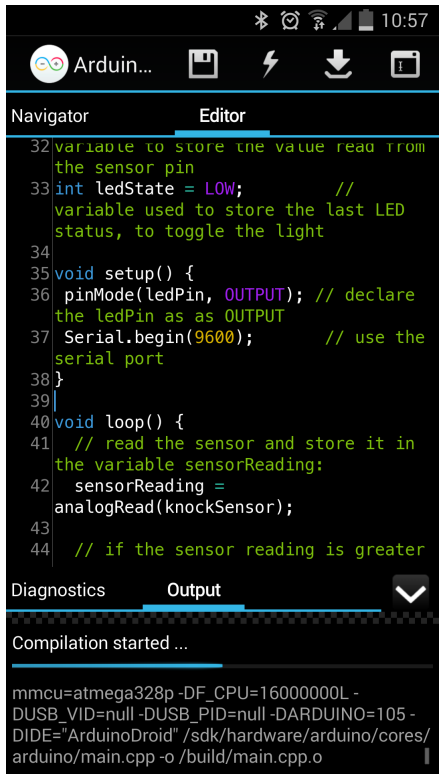

Figure 3.4: Screenshot of ArduinoDroid

Chapter 4

Proposed Solution

4.1 Introduction

This chapter presents the proposed solution RAPT (Rapid Arduino-Prototyping Toolkit) at a high level. In addition to explain the conceptual model of RAPT, this chapter describes RAPT's functionality, a technical overview and design decisions.

Figure 4.1: Report structure: Proposed Solution

To accomplish rapid and convenient prototyping of physical user interaction applications for non-programmers, a mobile application has been developed. This provides ease of access and makes use of the increasingly extensive number of smartphones and mobile applications available. The idea is that a mobile application will facilitate and lower the bar for creating prototypes involving Arduino boards. The connection to UbiCollab [43] requires the use of Android. There is no need for USB cables, downloading additional software, libraries, or learning a new programming language.

4.2 Conceptual model

To illustrate how RAPT can be used to create tangible user interfaces, the home assistance scenario (Section2.4.2) has been selected for demonstration. A care worker has gotten the task to install a new TUI in a care patient's home using RAPT and a hardware kit that includes two Arduino boards equipped with different hardware devices. The care worker places the two Arduino boards in the necessary location, one in the bedroom and the other by the stove. Thereafter, the care worker starts RAPT and creates a new app. The Arduino boards found nearby is presented in RAPT as board representations. Each board representation has a

set of corresponding devices, which matches the hardware devices on the Arduino boards. Each board representation got a board type. What board type a board representation has are decided by its devices and how they are connected to the Arduino board. The devices of each board representations are divided into sensors and actuators. Sensors are devices that handles digital input, or physical input from a human or the environment. Actuators are devices that handles digital output or environmental actions.

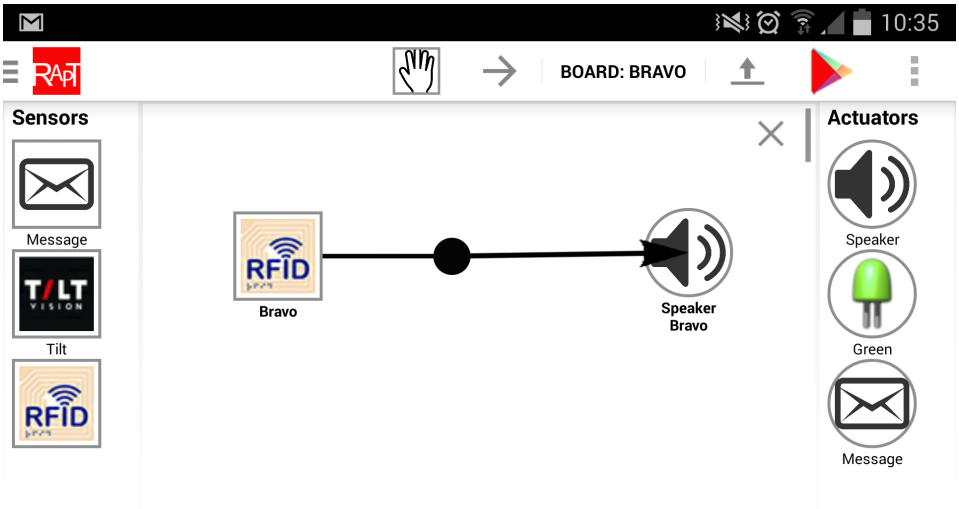

Figure 4.2: RAPT with a rule created in rule container

In RAPT, the devices can be dragged by the care worker into an area called rule container. This is the area between the sensor and actuator containers. When the care worker drags a device into the rule container, an alert dialog appears. This alert dialog can be used to configure the dragged device. Examples of device configurations are to read a specific RFID tag (configuration of a RFID reader device), or play a melody (configuration of a speaker device). A rule is created when connecting two devices. A rule is illustrated in Figure 4.2 and implies the following: *When a specific RFID tag is read by the RFID device, play a melody through the speaker.* After the care worker have created all necessary rules the required functionality of the tangible user interface, the app can be installed. An overview of concepts are found in Table 4.1.

RAPT concepts	
App	Application created within RAPT that consists of one or more rules.
Board representation	The representation, illustration, of an Arduino board within RAPT.
Device	A representation of a hardware device attached to an Arduino board. Example: RFID reader.
Device configuration	A function of a device that can be chosen by the user. Example: Blink led.
Rule	A connection of a configured sensor and one or multiple actuators. Example seen in Figure 4.2.
Sensor	Type of device that handles digital input, or physical input from a human or the environment. Example: RFID reader.
Actuator	Type of device that handles digital output or environmental actions. Example: Led.
Board type	What type of Arduino board the board representation represents. What board type a board representation has are decided by its devices and how they are connected to the Arduino board.

Table 4.1: RAPT concepts

4.3 Functionality

4.3.1 Introduction

RAPT is used for creating, and installing apps on Arduino devices. In addtion an app can be published to an app store. The main functions that are utilized when creating, publishing and uploading apps are introduced in the following sections. The main functions are reflected in the choices that can be made in the main menu screen. The main menu screen is presented in Figure 4.3.

4.3.2 Creating apps

The easiest way to create an app is to click on the button “Create New App” in main menu. This part of RAPT is designed to be as easy as possible for the user. When clicking “Create New App”, representations of all nearby Arduino boards will be listed in the navigation drawer illustrated in Figure 4.4. In this view, the user can choose the active board representation. Depending on active board representation, different sensors and actuators are available.

After the user has chosen the active board representation, the user is presented the most important screen of RAPT, presented in Figure 4.5. An Arduino board can have multiple devices, and these are listed on the left side, the right side or on both sides depending type of device. It can be a sensor, which is used to collect

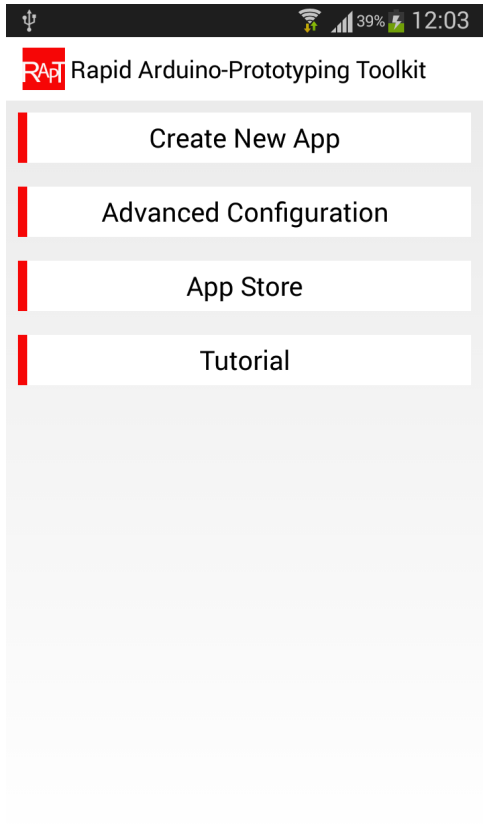

Figure 4.3: Main menu

data or trigger on events, it can be an actuator, which are used to execute actions, or it can be both. In this view, the user can drag devices into the rule container. The available actuators and sensors changes based on which board representation the user has chosen as the active board representation. If the user wants to change active board representation he/she can select a new board representation from the navigation drawer, which can be opened by clicking on the icon at the top left corner.

The active board representation's name is displayed on the toolbar as shown in Figure 4.5. If the user wants to change the name of the board representation, it can be done by clicking the name in the toolbar. This will not change the name of the Arduino board, only what the board representation is called in current the RAPT session. The user might want to use this functionality when creating apps for two Arduino boards with similar board types. If the boards are of similar type they will get the same name with a number in RAPT. This ensures that the name will be unique, but it can be hard for the user to distinguish between which Arduino

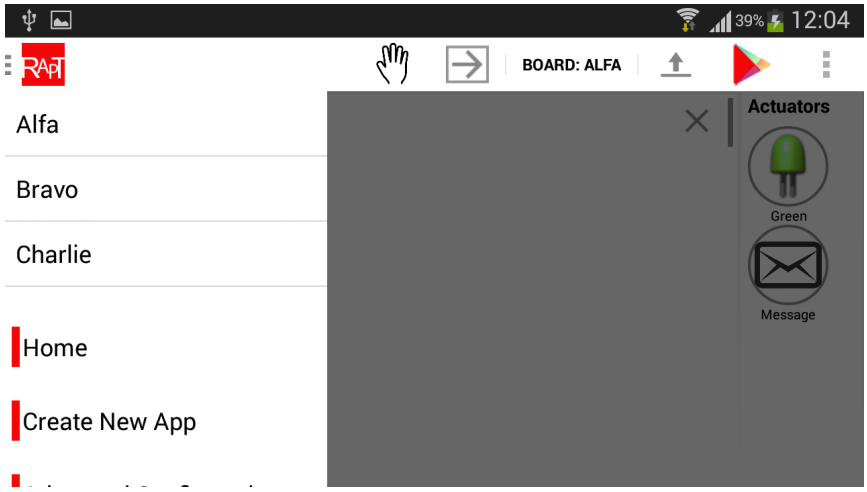

Figure 4.4: Navigation drawer

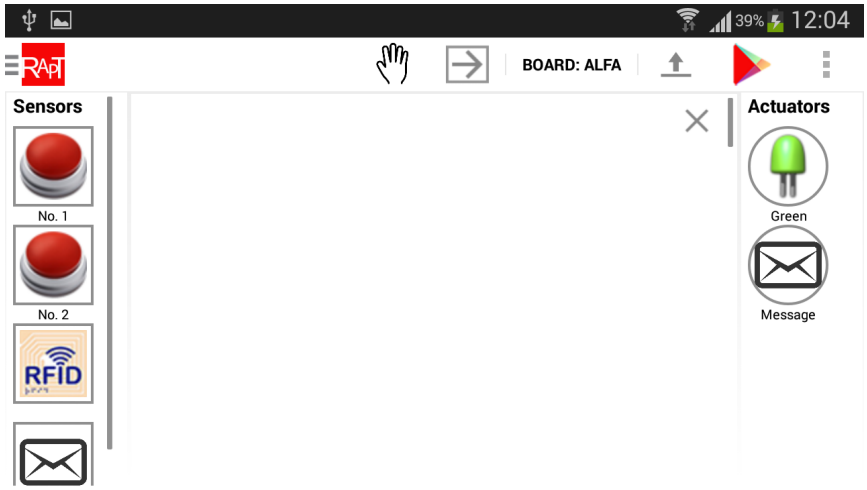

Figure 4.5: Screen for creating an app

board is mapped to which board representation. Giving the board representations new names can help if the user are having a troubles with names.

When dragging a device from either the sensors or the actuators container into the rule container an alert dialog will open, prompting the user to configure what the dragged device should do. If the dragged device is a led, the user can choose between different functions such as led blinking and turning the led on. Another example is when the user drags a WiFi device it can either send a message or receive a message. These functions can be changed at any time by clicking on the icon of the device when in connect mode. When creating apps, two different

modes are used: connect mode and drag mode. Buttons used to toggle these modes can be found on the toolbar, marked in Figure 4.6. Connect mode is chosen by clicking the arrow button, and drag mode is chosen when clicking the hand button. The connect mode is used to connect sensors to actuators. When the user touches a sensor device in the rule container and continue to hold, an arrow is drawn between the sensor and location of the user's finger. When releasing this arrow on an actuator device, the user connects the sensor and the actuator. An arrow with a black dot, called connector, will be drawn. If the user wants to connect a sensor to multiple actuators, the user has two options; Draw two arrows from the sensor, or draw an arrow from the sensor to the first actuator and draw another arrow from the connector created to the second actuator. The second option is illustrated in Figure 4.5. Drag mode is used when the user is not pleased with the layout of the devices and arrows in rule container. When in drag mode, the user can drag devices and connectors around which gives the user the freedom to organise the layout.

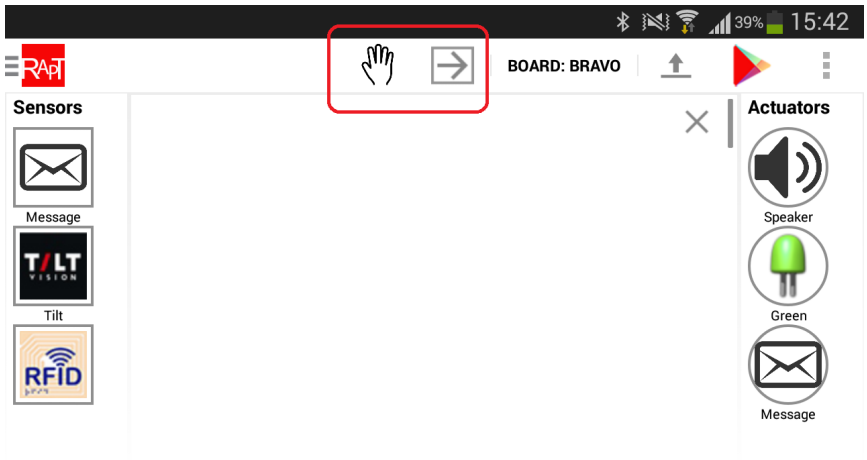

Figure 4.6: Mode buttons

To erase all rules, there is a functionality that clears the rule container. This functionality can be used by clicking the X in the top right corner of rule container. RAPT supports saving and loading of drafts when creating apps. As of current version of RAPT, these drafts do persist multiple sessions, but will be deleted every time the user manually closes RAPT. If the user has created an app which the user is satisfied with, it can be saved as a draft by clicking the save draft option in RAPT's settings menu. The settings menu is presented in Figure 4.7. Loading is also done from this settings menu.

When the user is installing an app that requires the Arduino boards to have a WiFi connection the app will need to know the SSID and password to the local WiFi network. This can either be set during the start of the installation process, after the user have clicked the install button, or it can be set in network settings

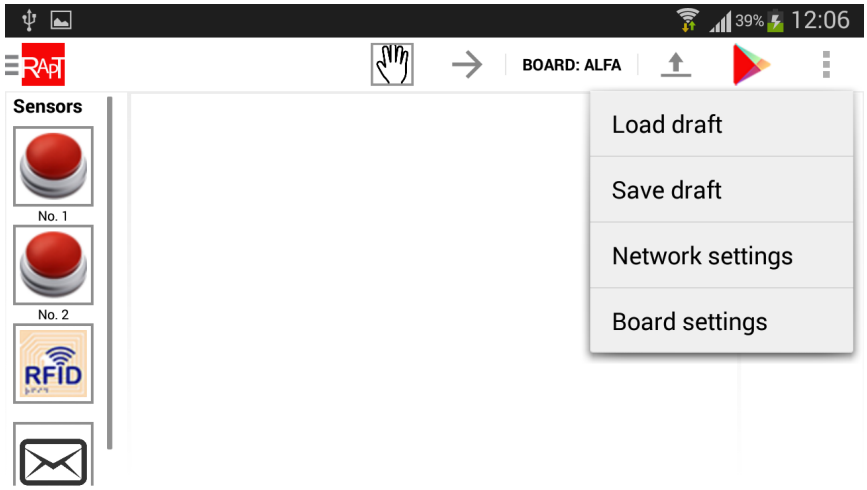

Figure 4.7: Settings menu

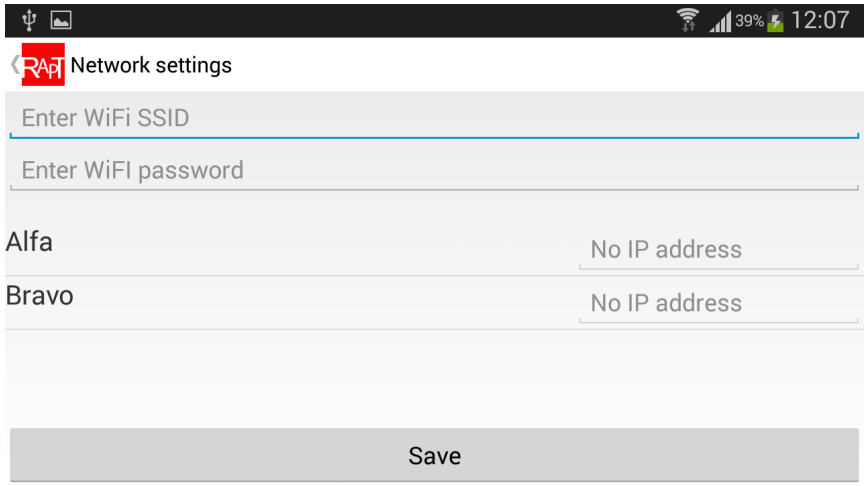

Figure 4.8: Network settings

from the settings menu. The network settings are shown in Figure 4.8. After an installation, all the Arduino boards that are connected to the WiFi network will have its IP addresses displayed in the network settings. This is useful if the user has installed an app that includes an Arduino board receiving messages from external systems. When sending packages from external systems to Arduino boards, the external systems need to know the IP addresses.

The board settings can be entered from the settings menu in RAPT. This functionality is used when the user wants to pair a board representation with an Arduino board. When the user starts to create an app the board representations

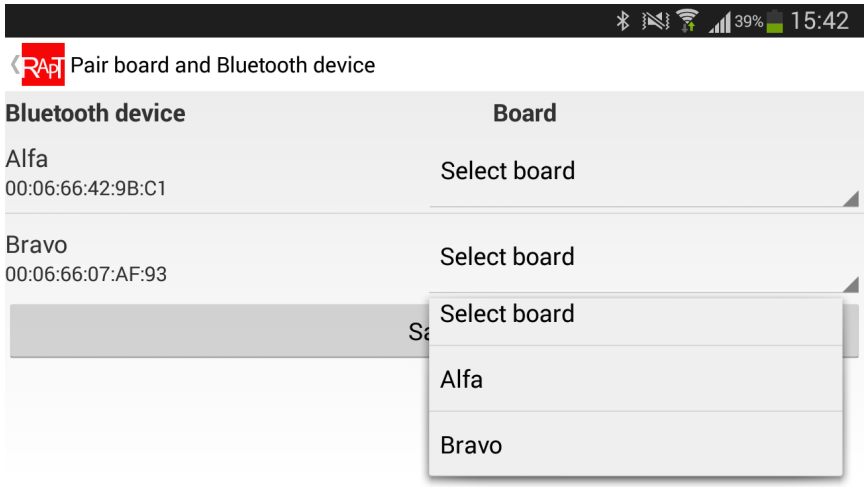

Figure 4.9: Board settings

are already paired with an Arduino board, but this can be overridden from board settings. Board settings screen is shown in Figure 4.9.

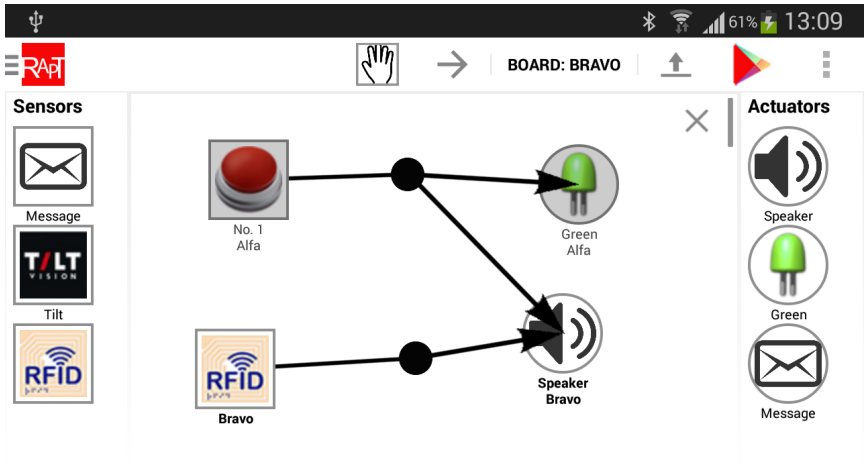

Figure 4.10: App including both Arduino board Alpha and Arduino board Bravo

When the user clicks the installation button, the app created will automatically be installed. The app will be installed on all the Arduino boards that are required for the app to work. Figure 4.10 illustrates an app including devices from two board representations. If the user clicks the installation button, both of the board representations used will get their corresponding sketch.

Users who desire more complex functionality can select “Advanced Mode” in-

stead of “Create New App” in the main menu. In advanced mode the board types used to create an app is not automatic chosen by RAPT. The user has to choose board type. The installation process in advanced mode differentiates from normal installation. In normal installation, the app is installed automatically to the Arduino boards represented by board representations in RAPT. In advanced mode the user has to manually select which Arduino boards that corresponds to the different board representations. An additional feature is that the user can install an app to multiple Arduino boards with only one installation process.

4.3.3 App Store

RAPT users can take advantages of the app store especially in terms of learning experiences. When loading an app from app store, the end user can either use the app as it is, or further customize the app. The end user has the same features as if he/she created the app from the beginning. Allowing a new user to explore how more experienced users have put together apps could be incredibly helpful.

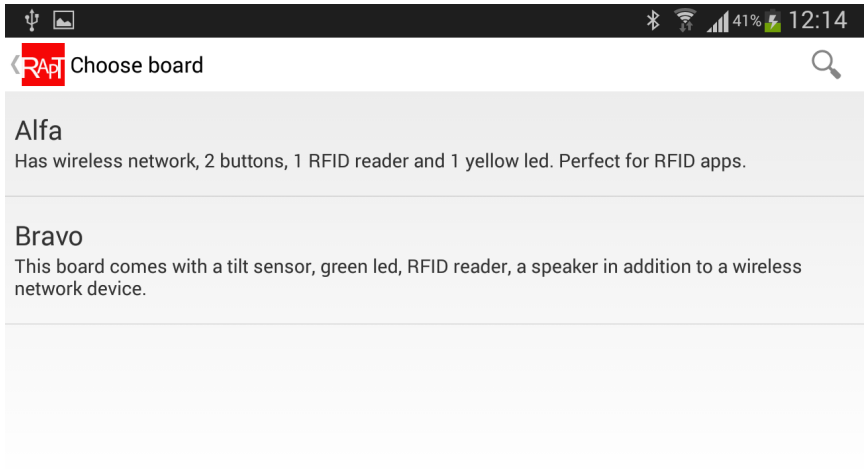

Figure 4.11: Selecting board type in app store

When the user enters the app store from the main menu, the user needs to select a board type. This selection screen is presented in Figure 4.11. All apps that includes the selected board type are displayed on the next screen, shown in Figure 4.12. By selecting an app from this screen, the user will be redirected to the view where the user can create rules.

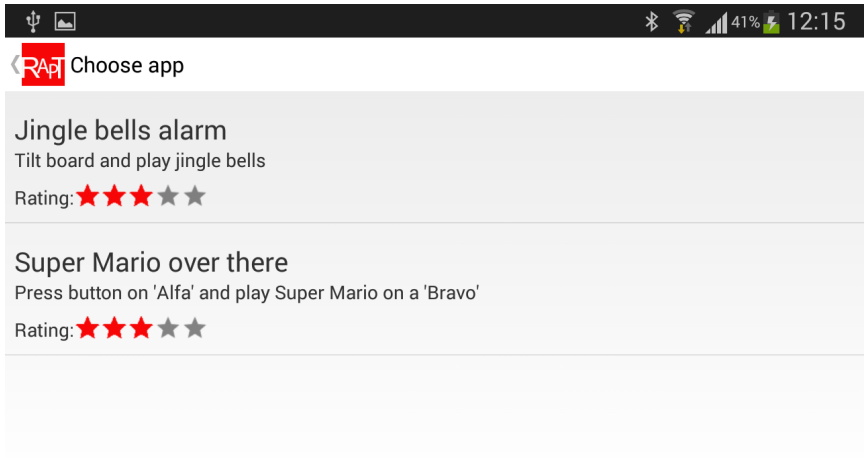

Figure 4.12: Selecting app in app store

4.3.4 Tutorial

RAPT comes with a tutorial to assist users on how to create apps. The tutorial gives a step-by-step guide on how to create an app using the toolkit. The guide is a series of pictures, which all consists of a screenshot from RAPT with descriptive text.

4.4 Technical Overview

RAPT has been designed for a wide spectre of users. This is because of RAPT's main goal: To be a toolkit for non-programmers to create physical user interaction applications. Anyone can buy a hardware board and RAPT should therefore be designed for all potential users.

Using Arduino boards require understanding of electrical engineering and specific wire instructions based on selected components. In the scenarios the team has envisioned, the Arduino boards are bought pre-wired from a retailer. An Arduino board can consist of several devices. XML [44] is used to specify the hardware attached to an Arduino board. XML is a common mark-up language, easily read and great extension possibilities. RAPT makes use of the XML files when defining devices in the graphical user interface.

The graphical user elements is based on drag and drop functionality. The user selects desired device from its container and drags it into rule container. The device can be configured to achieve the appropriate functionality. Two connected devices is a valid rule and the app can be installed to the Arduino board.

Programming an Arduino board is based on the programming language called Arduino programming language [45]. It is a variation of the programming language C++. Arduino code, from here on called a sketch, requires to be compiled before

uploaded to an Arduino board. When a user creates a new app in RAPT, a Bluetooth scan is started. The scan will discover all nearby Arduino boards that are paired with the mobile device running RAPT. When the user has created or modified the app he/she would like to install on an Arduino board, the app needs to be parsed from the rules created by the user into a sketch. The team has created a generator that translates these rules into a sketch. The sketch is sent to an external compile server through WiFi that returns the compiled sketch. The compiled sketch is then transferred to the Arduino board through Bluetooth. See Figure 4.13 for a simplified flow diagram.

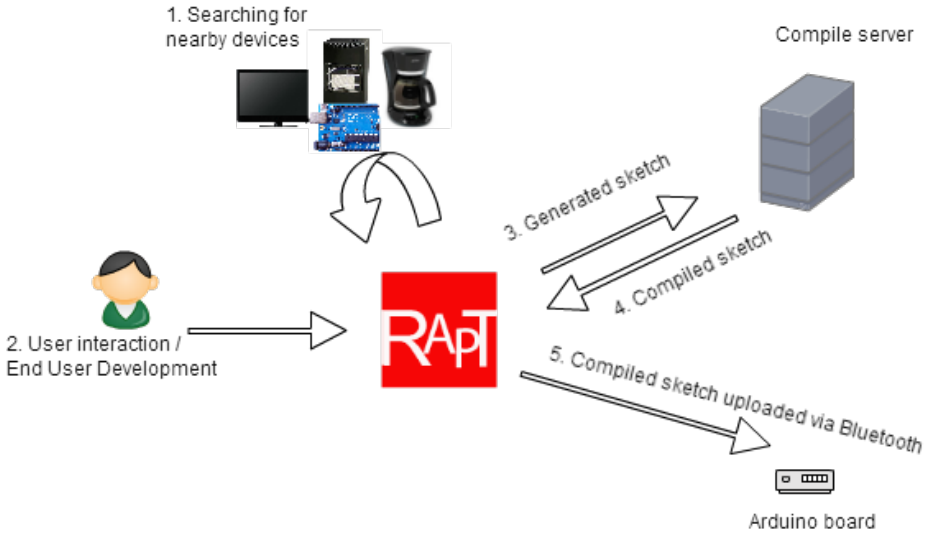

Figure 4.13: Simplified flow diagram of the system

RAPT communicates with Arduino boards through Bluetooth. Bluetooth is a commonly used technology for wireless transfer of data over short distances. Bluetooth was the team's preferred choice due to its widespread implementation in devices, low energy consumption and support in Android [46]. In addition, many existing solutions trying to program an Arduino requires the code to be transferred using a USB cable. Wireless communication is crucial for extensive usage of the app among non-programmers because of the ease of access it provides. WiFi was another technology the team considered for communication between RAPT and Arduino boards (EUDR-1, see Table 2.5). Due to the task's scope, the team quickly explored the java implementation of the STK500 protocol [47], and concluded that the implementation provided the needed functionality. The java implementation provides a working Bluetooth connection between Android and an Arduino board. TUIR-13, see Table 2.4, specifies communication between Arduino boards as a requirement. WiFi was selected as the communication protocol. WiFly RN-XV Module [48] and the belonging WiFly library provided the functionality. WiFi

not only allows communication between Arduino boards, but communication with anything that is connected to Internet. This provides the possibility to create apps integrated with external systems. See Appendix B for further information about hardware selection.

4.5 Rules Pattern

During the pre-study, the team discovered that several of the applications using physical user interfaces were using the same pattern. Two of these articles will be presented with a simplified example to illustrate the pattern. Example 1: [49] presents a wireless remote communication system called "BuddyWall". On a wall, there are mounted multiple removable 'buddies' (round, light emitting balls), where each buddy represents a friend. Each buddy has one BuddyWall. If buddy-A would like to talk to buddy-B, buddy-A picks up buddy-B's ball and squeezes it. This will make buddy-A's ball play a sound and flash a light in buddy-B's home indicating that buddy-A is trying to communicate.

Example 2: Shoppers often have the need to gather additional information about a product before buying it [50]. Each product has product information that can be transferred to a NFC [51] token the shopper is carrying. If the shopper would like the information to be displayed, he/she places the token on a kiosk, where the product information is transferred and displayed.

Both of these examples are using a pattern where you have a condition and one or more actions. If the condition is fulfilled, the corresponding actions are executed. In example 1: Squeezing the buddy is the condition. If the buddy is squeezed, another buddy plays a sound and flashes. Here playing a sound and flashes are actions that are executed when the condition is fulfilled. In example 2, the placing of the NFC token on the kiosk is the condition. If this token is read, the product information is displayed. The action is displaying the product information. In RAPT, a condition is configured sensor and an action is a configured actuator. A condition with one or many actions are called a *rule*. In RAPT, a rule is created when one condition is connected to at least one action (see Figure 4.14).

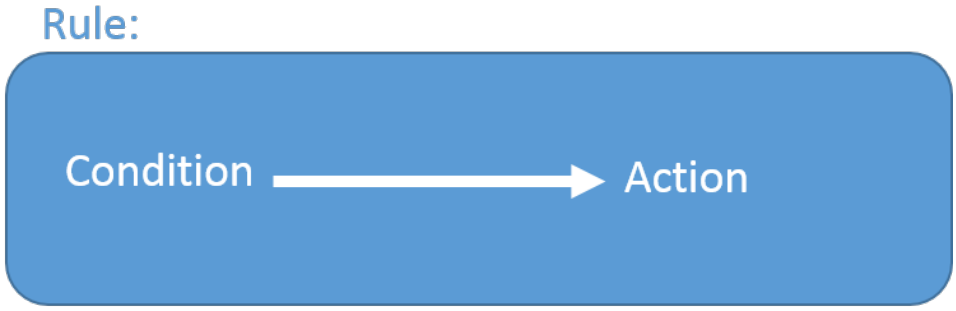

Figure 4.14: Illustration of a rule – a condition is connected to one action

4.6 Sketch Compiling

An external compile server is used when compiling a generated sketch. After the sketch has been compiled, it is returned to RAPT and installed on the Arduino board. There were one architectural decision regarding the compilation flow. The sketch could either be compiled on the Android phone or be compiled on an external server. The Android application ArduinoDroid described in Section 3.4.3 compiles sketches on Android. This requires ArduinoDroid to download an sdk, making the size of the app several hundred megabytes. If this solution were adopted to RAPT, the current requirement of a working Internet connection would be eliminated. A disadvantage would be the time required to download the sdk and the large size of the RAPT. An external compile server was chosen, because the benefits of compiling on Android does not outweigh the disadvantages.

4.7 Installing Sketches on Arduino Boards

One issue that required the team's attention is how to tell RAPT which board representation that corresponds to which Arduino board. If an app includes multiple board representations, multiple sketches are generated. One for each board representation. Two solutions has been implemented to solve this problem. 1) The user manually selects which board representation that corresponds to which Arduino board. 2) RAPT starts a Bluetooth scan, maps the XML configuration file names with the name of the Arduino boards nearby. The second solution requires no interaction from the user. When the user clicks the button "Create New App" it starts a Bluetooth scan. If an Arduino board is found during the scan it will be displayed as a board representation in RAPT. Solution 2 increases usability because the user does not need to manually map Arduino board with the board representation. However, this has a great set of functionality limitations. The user is limited to only create apps of nearby Arduino boards. In addition, solution 2 lacks the important functionality for users to upload the same app to multiple Arduino boards. This is specified as a requirement EUDR-5 in Table 2.5. To maintain the benefits of solution 2 and keep the desired functionality from solution 1, it was decided to split the functionality. By creating an additional 'advanced mode', this would give experienced users freedom to use more complex functionality, without confusing the normal user.

4.8 Selection of End User Development Techniques

Acquiring knowledge of different types of end user development techniques were primarily done through the systematic mapping study found in Appendix A. A characteristic among many of the studied applications was that they all tend to include more than one EUD technique. For example, wizard driven technique is often combined with the component tailoring. Even though many applications are using the same techniques, they often vary in level of integration. One aspect of end user development not considered in the systematic mapping study, is the fact

that the team is developing EUD on a mobile device. Ideally, this should have been included in the search string to narrow the number of results and increase the precision and recall.

When selecting end user development techniques there were several aspects demanding the team's focus. First, the limitation due to device restrictions. No traditional mice and keyboard forces touch elements to be bigger and minimize text input from a touch keyboard. Second, the limited screen size requires thought-through solutions for optimal touch interaction and usability. Third, the performance aspect where users expect applications to be quick and responsive, few bugs and no application crashes. Finally, the challenge with several mobile platforms and many screen resolutions. The problem definition specifies to make use of Android. Regarding the screen resolution, the team decided that it is easier to scale up than down. By using dynamic dimensions RAPT will work fine on a tablet, but not optimal.

By developing a mobile application instead of a desktop or website solution, it limits suitable type of end user development technique. However, the process of selecting techniques started with brainstorming and wireframe drafts. Due to general population's intuitiveness and familiarity with touch on a mobile device, drag and drop were selected as a technique.

4.9 App Store

According to [52], an app store (application store) "refers to an online shop where customers can purchase and download various software applications. The apps sold through app stores are usually intended for mobile devices. App stores are cloud-based in the sense that users access the content via free client software or a Web browser." Moreover, "Apps are extremely popular among smartphone users (...). The mobile app market reached nearly $7 billion in 2010 and is expected to grow to $15 billion by 2015."

The precursors of app store was among others packet mangers, mainly connected to Linux distributions [53]. However, Apple originally created the term "app store" in 2007. Later, the court deemed the term as a descriptive term rather than Apple specific [54]. Not only is app store a concept of revenue and proliferation, but it has become a way for users to take part in the development, increase diversity and introduce aspects that corporations behind the app store ignored or forgot.

Having a RAPT app store would greatly increase the functionality of RAPT. Being able to download apps other people already have created gives RAPT another field of application. An app store could be used to share apps with friends, used as inspiration for new apps, or used by large groups or companies to publish updates to their apps.

The team had to make a decision on what information to store in app store. During the early stages of development of RAPT, the idea was to upload compiled, generated sketches to the app store. This idea was rejected because users would not be able to look at or modify the app's rules when loading an app from app store. Another problem was IP addresses of the Arduino boards included in the

app, as IP addresses are found before compiling each sketch. These IP addresses would be hard-coded into the apps in app store, making the app unusable for the users installing the app from app store.

Based on requirements from the scenarios analysis, the team decided to go for an app store where only rules were stored. Users can publish apps to app store without generating or compiling the app first. This is great for users who do not want to install the app on his/her device, but only share it with others.

Chapter 5

Development

5.1 Introduction

This chapter will provide technical aspects of the proposed solution RAPT. First, the architecture including data flow and description of particular parts and class diagrams. Second, this chapter provides more in-depth details of selected technical solutions in the section Design. Third, presumptions are explained, and finally, this chapter looks into software and compatibility issues.

Figure 5.1: Report structure: Development

5.2 Architecture

5.2.1 Introduction

This chapter will describe how RAPT is constructed from a detailed perspective. However, instead of going into detail about each class and activity in the system, this chapter will explain data flow, graphical user interface, generator, compilation server and Bluetooth communication. Data flow between these parts is first described and illustrated. Thereafter, each of the mentioned elements are given attention in a separate section. At the end, we find the most detailed diagram, the class diagram.

5.2.2 Data Flow

The high-level client architecture is based a layered approach. This type of architecture splits the responsibility of each layer preventing high coupling and increasing

reusability. This makes it easier to develop and deploy parts of the system if needed, as well as change and further develop only selected components.

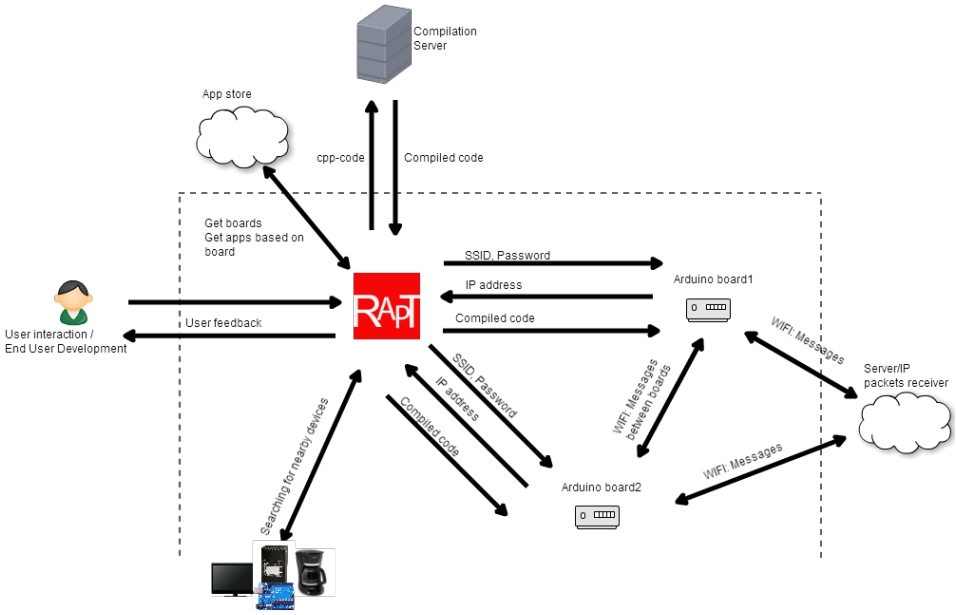

Figure 5.2: Flow diagram

Figure 5.2 illustrates how the system handles information flow. When a user creates a new app, the mobile phone searches and for nearby Arduino boards. RAPT's compiled sketches are uploaded to the discovered Arduino boards. Compilation of sketches happens externally on a server. Sketches are sent to this server and compiled sketches are returned to RAPT. Before uploading a compiled sketch to the Arduino board, RAPT prompts the user for Service Set Identifier (SSID) and password if the created app makes use of a WiFi component. SSID and password are sent to the Arduino board, which returns the given IP address. The messages are set to UDP (User Datagram Protocol) packets due a lower total roundtrip time. The wireless network information is then integrated into the created app (for example, it is needed to receive external messages and communication directly between boards). Thereafter, the compilation server is contacted for each sketch that is to be compiled, and the returned compiled code is uploaded individually to each Arduino board.

The flow sequence when installing an app is described in Figure 4.13.

Figure 5.3 explains in a simplified version how the user interface is connected to other components. ConfigureRules is the activity that handles the main screen where users can drag and drop sensors and actuators. The diagram does not necessary represent the exact coupling, however, it gives the idea of how main components are connected. ConfigureRules instantiates Bluetooth, RulesIntaller and several listeners that handles touch effects on the screen. In addition, Con-

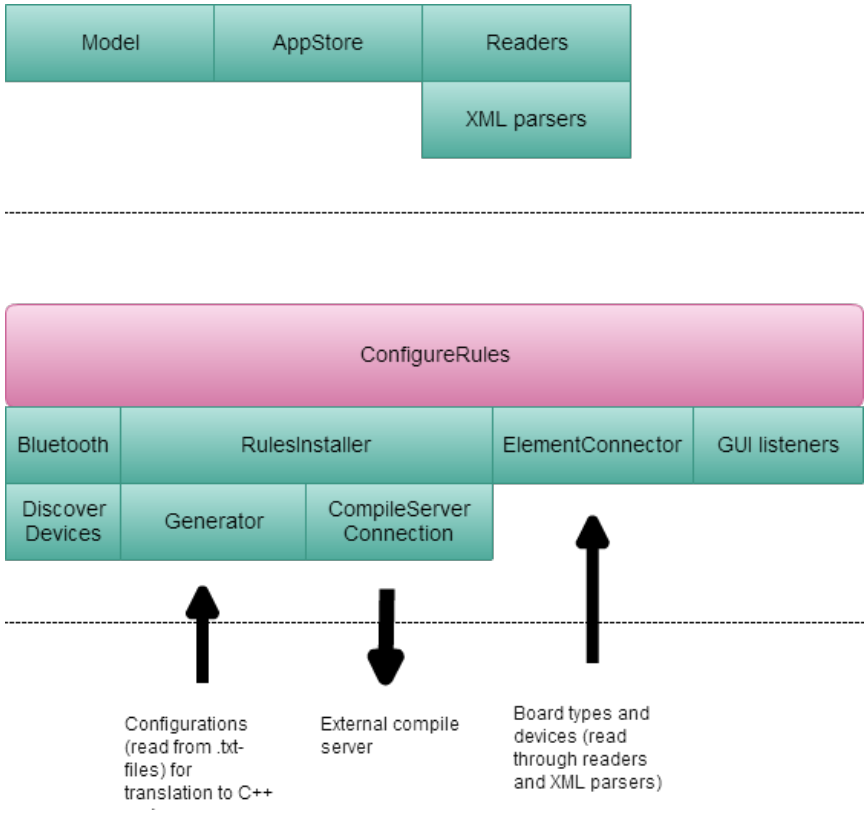

Figure 5.3: Components and their connections

figureRules make use of the help class ElementConnector to extract information from data objects, and to save changes to these objects. The following sections will cover the most important about these classes.

5.2.3 Graphical User Interface

The main functionality of the graphical user interface is to handle user input (especially drag and drop functionality). When a user drag at least one condition, one action and connect them together, it creates a rule. These rules are constructed in order for the generator to generate cpp-code that can run on an Arduino compiler. The graphical user interface, mainly ConfigureRules, is the junction for most of the functionality of RAPT, as well as to control that valid user input is received, and give the user a proper feedback on his or her actions.

5.2.4 Generator

The generator generates cpp-code based on rules. These rules are created according to state of sensors, actuators, their configurations and their connections in the user interface. When building the cpp-code, the generator reads text files that contains predefined cpp-code. These text files are located in the folders actions and conditions in the path: android-toolkit/assets/configurations. An example of output from the generator can be found in Appendix G.2.

5.2.5 Compilation Server

The compilation server is a remote server that receives cpp-code (Sketches), and returns the compiled code (Compiled code) as hexadecimal characters. Appendix F holds the code that executes the compilation. Our implemented compilation server uses AVR [55], which is the same as the Arduino IDE compiler [56].

The server is partially outside the scope of this master dissertation, however, a necessity to hide data streams from the user and improve user experience. Therefore, we have not implemented a scalable version with extra functions such as response messages or support of other operating systems than Windows.

The server is a Python server that execute a batch script (See Appendix F). The server also writes received data to a text file, which for example could look like the code in Appendix G.2.

5.2.6 Bluetooth Connection

Bluetooth connections are used when uploading, getting IP addresses, and when searching for nearby Arduino boards. For uploading to an Arduino board via Bluetooth, the system utilizes the stk500 for uploading compiled sketches to Arduino boards.

A precondition, in order to make the upload work, the ComputerSerial library must already be imported to the Arduino boards' memory (that is, a running sketch with a ComputerSerial import), due to use of STK500. STK500 has set this as a requirement.

5.2.7 Class Diagrams

This section presents RAPT's class diagram, divided in five different diagrams. The two first, Figure 5.4 and Figure 5.5, show the main data objects. Thereafter, Figure 5.6 shows how readers and parsers are implemented, followed by help classes in Figure 5.7 and Appstore classes in Figure 5.8. Graphics and Android activities (which also mainly represent user interfaces) are excludes. These are hardly relevant, in addition to the huge amount of space it would require. Furthermore, most of the classes implements the Java interface Serializable, however, the extent of extra arrows lead to exclude them.

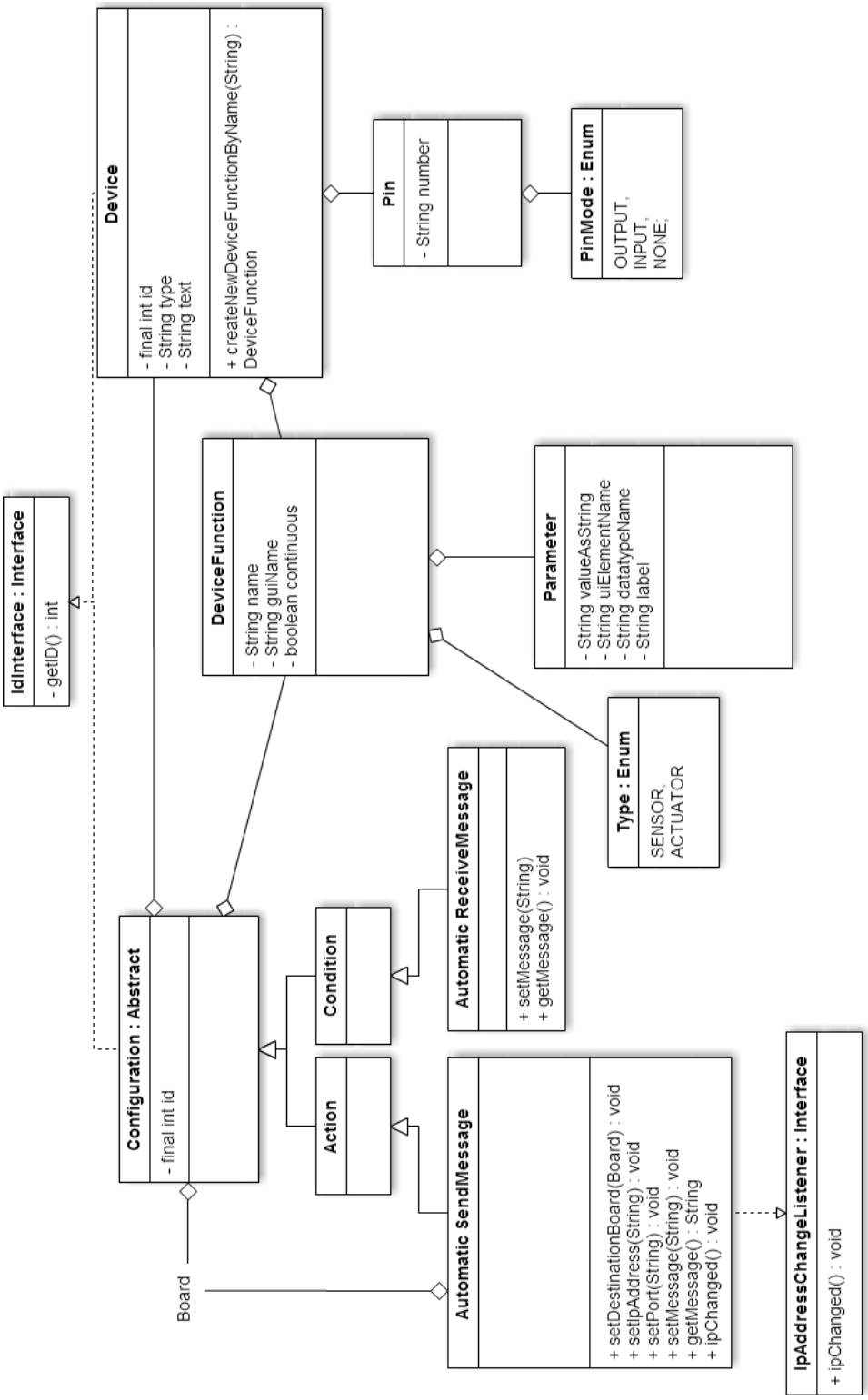

Figure 5.4: Class diagram: Objects, part 1

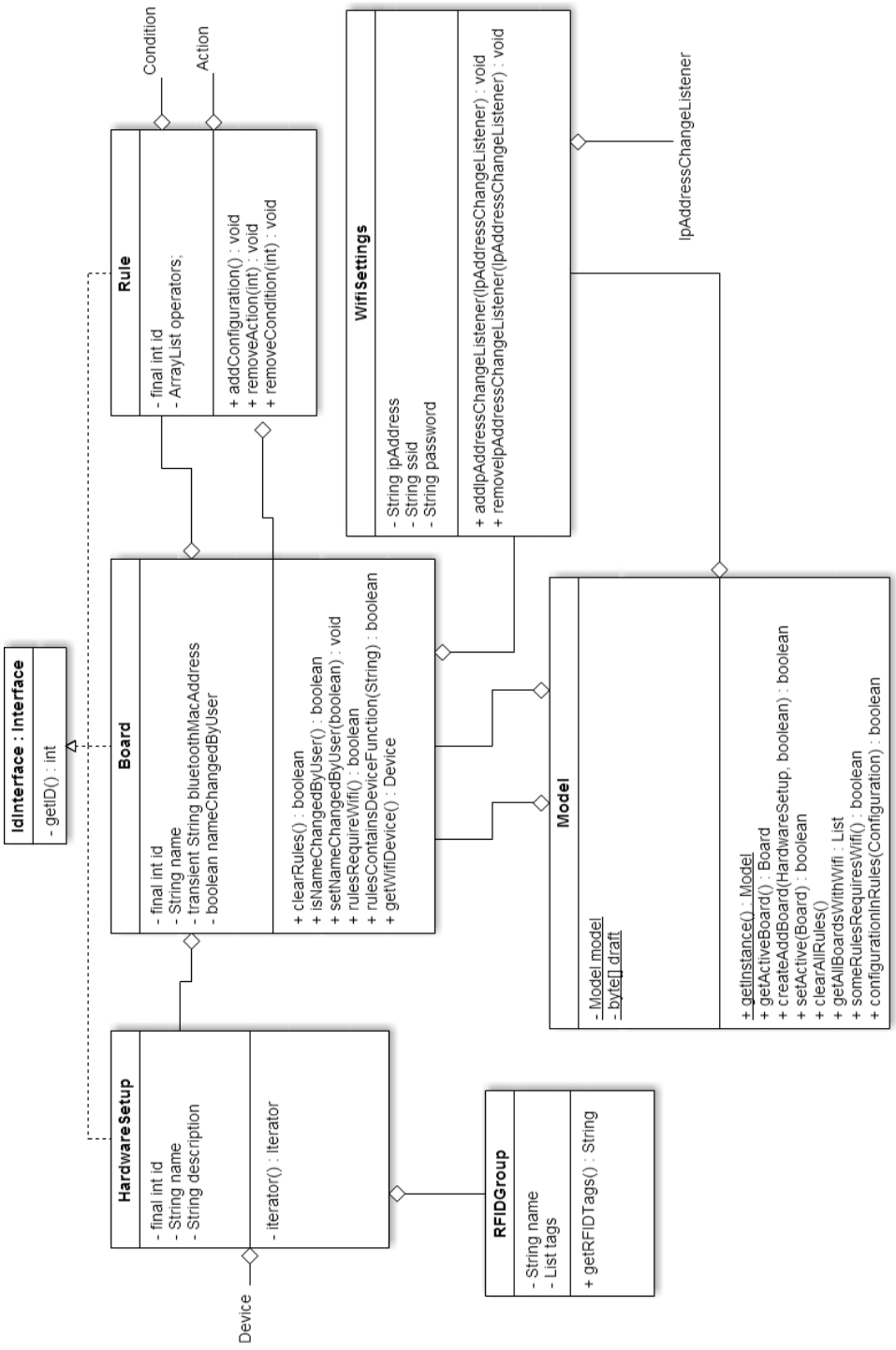

Figure 5.5: Class diagram: Objects, part 2

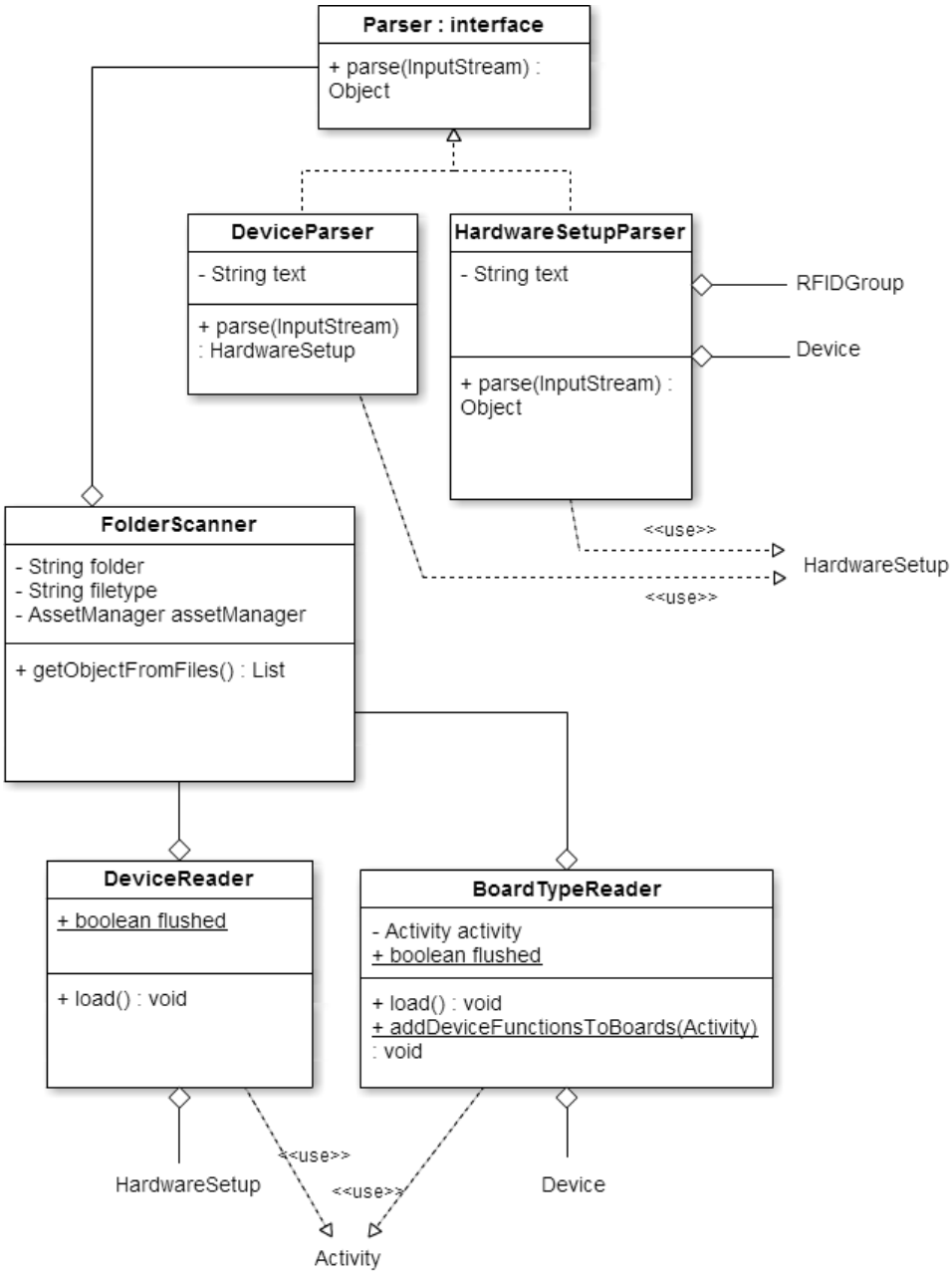

Figure 5.6: Class diagram: Parses and Readers

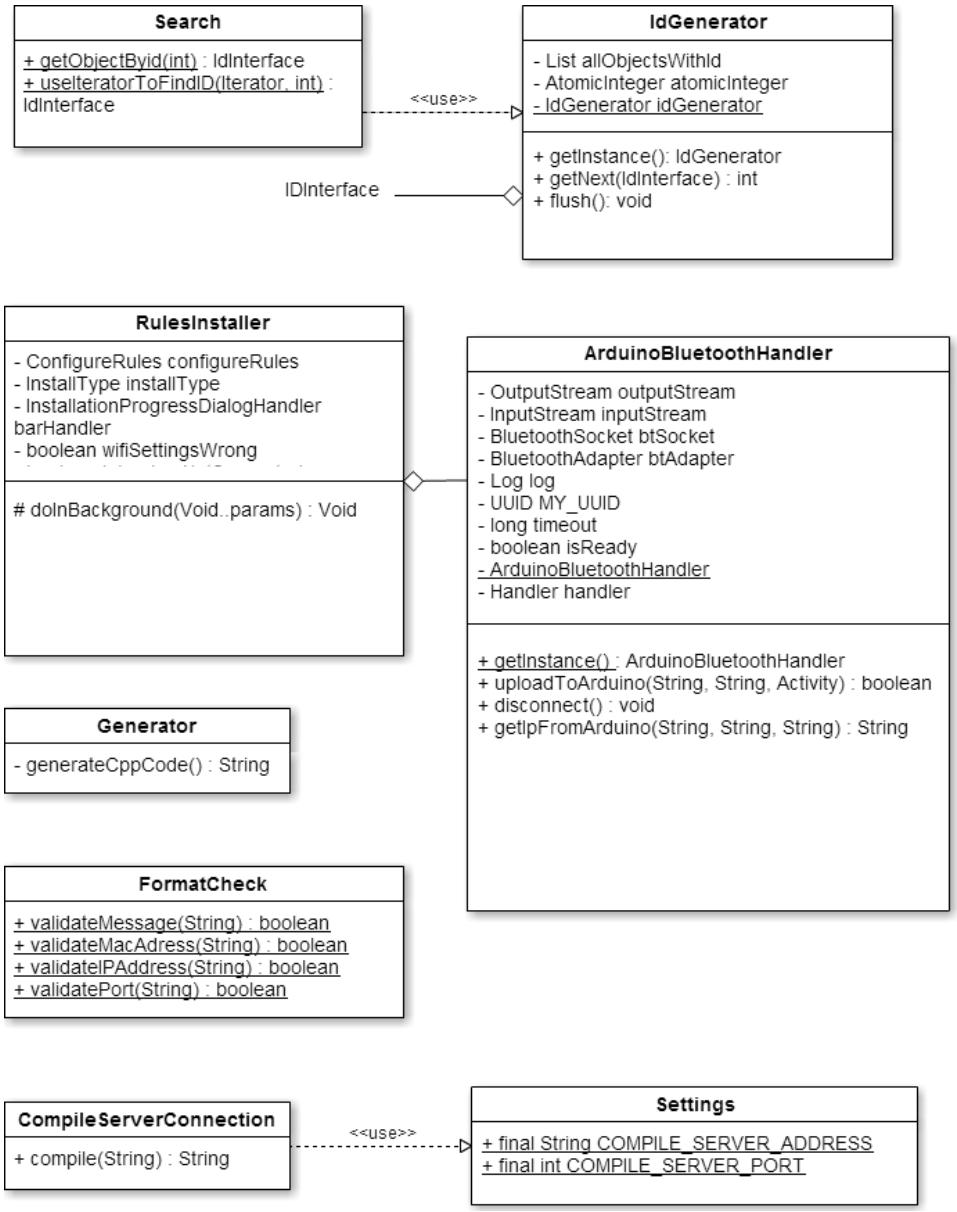

Figure 5.7: Class diagram: Help Classes

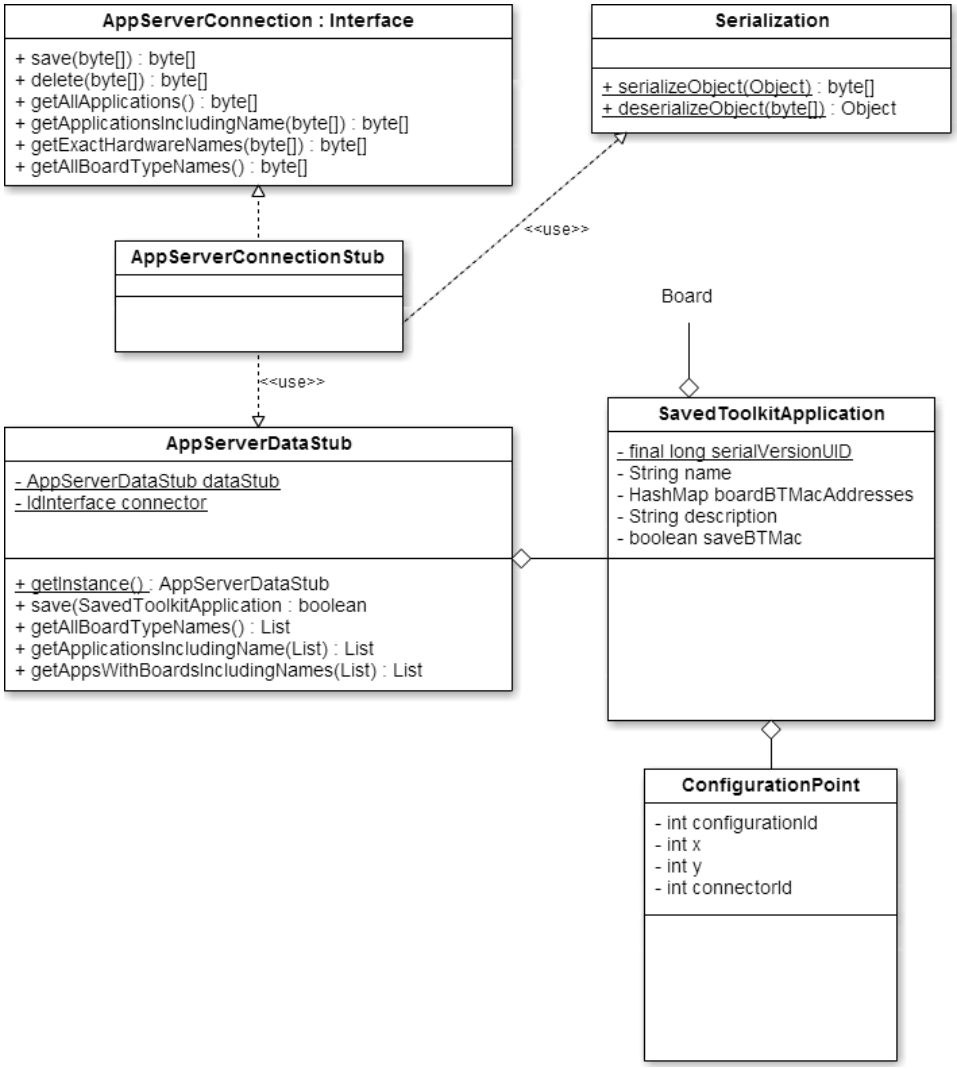

Figure 5.8: Class diagram: Appstore Classes

5.3 Design

5.3.1 STK500 and ComputerSerial

RAPT uses the java implementation of the STK500 [47] protocol to deploy compiled sketches, generated by the cpp-code generator and compiled by the compile server, to Arduino boards. This java implementation uses the ComputerSerial library [57] to trigger a soft reset of the Arduino board. The process of uploading the compiled sketch with STK500 is abstracted into a method call which returns a result based on the success of the upload done. We use this result to give feedback to the user, and cancels any further uploads if the result if negative.

5.3.2 XML Parsers

There are two different types of XML content in RAPT. They both use the FolderScanner-class (which scans a selected folder for files), and implements the XmlPullParser library [58]. XmlPullParser reads XML files, and the parsers, DeviceParser and HardwareSeupParser, create Java-objects from XML files.

There are several XML-parser libraries for Java-based environment such as Android. XmlPullParser is not the easiest nor cleanest. However, XmlPullParser is small (118 KB) and at the same time fast, which is possibly two of the most important criteria when working on mobile applications.

The reasons behind use of XML files are reusability and to give future developers an easy way to add representations of Arduino boards, in addition to add new functionality to a device. Figure X presents a snippet of how to add a new board representation as an XML file. This file is in the folder: android-toolkit/assets/boardTypes.

A led, green, is connected to pin 11 at a board called Alfa as is described in the following snippet:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <board>
3      <name>Alfa</name>
4      <description>Has wireless network, 2 buttons, 1 RFID
           reader and 1 yellow led</description>
5      <devices>
6          <device>
7              <type>LED</type>
8              <text>Green</text>
9              <pins>
10                 <pinmode>OUTPUT</pinmode>
11                 <pinnumber>11</pinnumber>
12             </pins>
13         </device>
14     </devices>
15 </board>
```

The next snipped shows how an XML file representing functionality. This file is located in the folder: android-toolkit/assets/devices.

```
<?xml version="1.0" encoding="UTF-8"?>
<devices>
    <device>
      <type>led</type>
      <action>
        <name>switchf</name>
        <guiName>Switch led</guiName>
      </action>
    </device>
</devices>
```

Text files with code used by the generator are needed as well when adding new functionality. is needed as well when adding new functionality. This is located in the folders "actions" or "conditions" in android-toolkit/assets/configurations.

The code below is the cpp-code for the switch function declared in the XML above, written in a text file that is read by the generator.

```
void switchf(int pin) {
    if (digitalRead(pin) == LOW) {
        digitalWrite(pin, HIGH);
    } else {
        digitalWrite(pin, LOW);
    }
}
```

In the current version, one must open the unpacked version of RAPT and redeploy the application to the phone in order to add new board representations or add or change functionality. In Section 7.4 there is more about how make the end user able to further develop RAPT's functions without the need for redeploying the application.

5.3.3 Generator

The generator is designed to generate sketches based on rules created by the RAPT user. Logic in RAPT's GUI forces rules to only have one condition, still, the generator is designed to handle multiple conditions and actions in one rule. The generator produces sketches by combining pre-defined cpp-code and the rules created through GUI.

Generated sketches can be split into seven parts; imports, class declarations, variable declarations, the setup function, the loop function, device functions, and the serial event function. Imports and the serial event function are defined within the generator class itself while the rest are defined in text files. There are two serial event functions, and depending on whether an Arduino board has a WiFi device or not, one of the versions is added. If the board representation has a WiFi

device, the WiFlyHQ library [59] is imported. In addition, the serial event function of the sketch will support looking up IP addresses. More about the IP address handling can be found in Section 5.3.4. The ComputerSerial library [57] and the SoftwareSerial library [60] are always imported, as these libraries are required by all generated sketches.

The setup function is executed each time the Arduino board restarts. If the board representation contains a WiFi device, the generated setup function includes code for connecting to a WiFi network.

The class declarations, variable declarations, the setup function, the loop function and device functions, are fully or partially generated by combining cpp-code from text files. Each configuration (a condition or an action) in a rule can have up to five text files with code, one text file for each part of the sketch.

Examples of device functions are blink (function of led device) and buttonPressed (function of button device). The code below show the content of the text file for a led's blink function.

```
1  void blink(int pin, int timeon, int timeoff, class Long& time, class
       Continuous& continuous) {
2    if (digitalRead(pin) == HIGH && time.t + timeon < millis()) {
3      time.t = millis();
4      digitalWrite(pin, LOW);
5    }
6    else if (digitalRead(pin) == LOW && time.t + timeoff < millis()) {
7      time.t = millis();
8      digitalWrite(pin, HIGH);
9    }
10 }
```

The main part of each sketch is the loop function. The loop function loops consecutively when an Arduino board is powered. The loop function primarily consists of if-sentences. An example of a loop including one if-sentence is shown below.

```
1  void loop(){
2      previousState18 = state18;
3      state18 = digitalRead(buttonPressed18_10);
4
5      if (buttonPressed(buttonPressed18_10, state18, previousState18)){
6          switchf(switchf15_9);
7          led(led16_8, HIGH);
8      }
9  }
```

Each if-sentence in the loop represents one rule. The code above does the following: When a button is pressed, a led (led device with ID 15, connected to pin 9 on the Arduino board) should switch (turn on or off depending on its current state). In addition to switching a led, the button pressed should also turn on another led (led device with ID 16, connected to pin 8 on the Arduino board).

The switchf() and led() functions are only called once, when the buttonPressed() function returns true. Each time the user presses the button, true is returned only once.

An additional feature is that a device function can be declared as continuous in the XML files. As opposed to the standard non-existing or "false", when continuous is set to "true", it ensures that the function is called several times: A new method call in every loop before the variable is set to false. For example, when a button is pressed, a led should turn on for five seconds before turning off. By declaring a function continuous, the rules will be represented by two if-sentences instead of one.

To end the loop method call, the variable is set to false by the clearLED3() function, or from within the flash function after five seconds have passed. The complete sketch generated by this app can be found in Appendix G. The code below are taken from the loop function of a sketch generated based on this app.

```
if (flash37Continuous.b == true) {
    flash(flash3_11, 5000, time37, flash37Continuous);
}
if (buttonPressed(buttonPressed4_10, state4, previousState4)){
    clearLED3();
    flash37Continuous.b = true;
}
```

The generator ensures that no conflict occurs when rules include several usages of the same device function. For example the generator needs to handle two leds blinking simultaneously in the same sketch. This is handled by parameterisation, in addition to include markers in text file code. All variables are declared in the variable declaration text file belonging to each device function and they are parameters of the corresponding device function. The markers are used to generate unique variables for each configuration or device. An example code using markers are shown below. The code is taken from the variable declaration text file for the buttonPressed function.

```
int #d[state]d#;
int #d[previousState]d# = LOW;
```

The tags for creating unique variables are "#d[" and "]d#", and "#c[" and "]c#". The "#d[" and the "#c[" tags are start tags, and the "]d#" and "]c#" are the end tags. The text between the start and end tag is the variable name. The generator searches for these tags, and creates sketch variables based on each variable combined with a unique number. The "d" notation represent that a variable should be unique for the device, while the "c" notation is a variable that is unique to the configuration. The code below presents an example of generated variables.

```
int state18;
int previousState18 = LOW;
```

To exclude a variable from being a parameter, use the special character (^). A variable that will be included in the sketch, but not given as a parameter to the device function, is shown in the code below.

```
Long #d[^startTime]d# = Long(0);
```

5.3.4 WiFi and IP Address Handling

For connections between Arduino boards, there need to be an identifier and a transfer protocol. The team has implemented functions for sending messages over wireless network, thus, enabling conditions and actions on different Arduino boards. At the same time, this solution makes it possible to contact other devices with Internet connection. The implementation is explained in the following paragraphs.

RAPT sends, via Bluetooth, a flag to the Arduino board to indicate that the next messages will contain a Service Set Identifier (SSID) and a password. For the Arduino board to be able to proper handle this information, our overridden method of SerialEvent() must be present in the running sketch. This method is attached in Appendix G.3.

On receiving SSID and password, the Arduino board, will try to connect to the wireless network. Upon success the given IP address is return to RAPT via Bluetooth. If the network settings are wrong, or the Arduino board is not able to connect for some other reason, the Arduino board returns the string “null”.

The returned IP address is used as a destination address. For example, “board1” has a button and when pressed a led should be turned on at “board2”. The destination address for the button pressed at “board1” is the IP address to “board2”. This IP address is a part of the compiled sketch. The port of WiFly components used by the Arduino boards is defaulted to 2000. The following code snippet represent an RFID tag reading, which leads to a message being sent to the IP address 10.0.0.8.

```
if (rfidread(" 010B9900D546,", readTag6)){
    sendMessage("10.0.0.8", 2000,
        "AutomaticSendMessage-Source:28-target:20-con:7");
}
```

However, there are some issues with this approach. For example, if an Arduino board, connected to a wireless network, receives a new IP address from the access point, the sketch needs to be recompiled with the new IP address. Usually access points try to give the same network device the same IP address, but there is no guarantee, especially if the access point is reset.

As a temporally solution the access point can be set to give out static IP addresses instead of dynamically.

A possible long term solution could be to implement a separate server with an IP routing table.

For one Arduino board to be able to communicate with another Arduino board,

the Arduino boards needs to have an IP in order to send and receive messages. To be able to send messages directly to the Arduino boards, it also requires knowledge of the other Arduino board's IP address. To increase usability, the team removed the need for a user to set IP addresses manually. This would cause unwanted complexity.

When users wants to create a rule that involves devices from different boards, the only action necessary is to connect the devices with an arrow. In the background, not visible for the user, two rules are created. Rule one is for the selected sensor device's board. The rule consists of selected sensor device with a send message as an actuator. Rule two is for the second board. Here, a receive message is the sensor and the action is the selected actuator device the user selected. Send message in rule one contains an ID that maps to the receive message in rule two. This ensures correct action being executed. When installing the configuration, rule one is in board one's sketch and rule two is in the sketch installed on board two.

5.3.5 App Store

RAPT has a stubbed implementation of an app store where data is stored locally. Saved data (in RAPT app store) is only stored in memory, thus, deleted when the memory is cleared or overridden. A full implementation of a complete app store is outside the scope of this master dissertation, however, the benefits of an app store can be extremely valuable for the end user as described in 4.9.

To override the current connection to the stubbed application store, locate AppServerConnection class, and make an implementation of this interface. Furthermore, in AppStore's and AppStoreConfiguration's onCreate methods, replace the instantiation of AppServerConnectionStub with your implementation. Frist, make an implementation of AppServerConnection.

```
1 public interface AppServerConnection {
2    public byte[] save(byte[] savedToolkitApplication);
3    public byte[] delete(byte[] savedToolkitApplication);
4    public byte[] getAllApplications();
5    public byte[] getApplicationsIncludingName(byte[] names);
6    public byte[] getExactHardwareNames(byte[] names);
7    public byte[] getAllBoardTypeNames();
8 }
```

To for transformation from and to byte array, use Serialization.serializeObject() and Serialization.deserialize() methods. Note: It is essential to use the same underlying methods for serialization at server side as well as client side. These methods are:

```
1 // Serialization
2 ObjectOutputStream out = new ObjectOutputStream(byteStream);
3 out.writeObject(object);
4
5 // Deserialization
```

```
ObjectInputStream in = new ObjectInputStream(new
    ByteArrayInputStream(byteArray));
object = in.readObject();
```

However, if there is no need to extract the data server side, the data can be stored binary without the need to use the specific serialization methods.

Thereafter, replace AppServerConnectionStub with your implementation:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   //Some other code
   appServerConnection = new AppServerConnectionStub();
   //Some other code
}
```

5.4 Presumptions

RAPT is designed for non-technical users, and therefore, we cannot expect users to be able to configure their own device (wiring) or create corresponding XML files. RAPT comes with a set of predefined board types. RAPT identifies the board type of an Arduino board by looking at the name of the Arduino board's Bluetooth device. The user must have acquired the configuration file (XML file) belonging to the Arduino board (for example from an external vendor). See Section 7.4.3 on how optimise this process.

The user can only use Arduino boards in RAPT if the Bluetooth device on the Arduino board is paired with the user's mobile phone. Therefore, the user needs to pair Arduino boards with the phone before creating or installing apps.

To be able to upload a sketch via Bluetooth, there must be a running app on the Arduino board that imports the ComputerSerial library [60]. Section 5.5.1.1 describes the needs and benefits to this library. A guide on how to install a sketch with the ComputerSerial library is attached within Appendix C.2.2.

5.5 Issues

5.5.1 Compatibility issues

5.5.1.1 Serial Ports

Arduino Uno has only one hardware serial port. Yet, several components such as Bluetooth components, WiFi components and RFID readers, need a serial port to function. To solve the issue of having too few serial ports, the SoftwareSerial library [60] allows us to add several serial ports. However, Arduino Uno, can only listen to a total of two serial ports at the same time. In comparison, Arduino Mega has 4 hardware ports. This limitation of Arduino Uno affect RAPT by limiting

the number of components that make use of a serial port to two. The Bluetooth transfer claims one serial port, which means that only one more serial port is available. The WiFi device and the RFID device both need a serial port. The result is, apps can only have one RFID reader or one WiFi sensor (WiFi device that receives messages) at each Arduino Uno board in addition to the Bluetooth device.

5.5.1.2 WiFly

When WiFly components (WiFi device) are requesting an IP address from the local WiFi network, it sometimes fails to connect to the WiFi network even though the SSID and passphrase are correct. Furthermore, the WiFly device seems not to be able to reset and try again. A restart of the Arduino board, or a new sketch upload are needed to fix the problem.

5.5.2 Software issues

5.5.2.1 WiFi device without IP lookup

This issue is regarding Arduino boards that contain a WiFi device, but does not have the WiFi component specified in the configuration (defined in the corresponding XML file). If the configuration is changed to a configuration with the WiFi component, there will not be possible to get the IP address.

This is somewhat a rare issue, however, still an issue when encountered.

An example of how to encounter the issue: Select "Advanced mode", choose a board type without a WiFi device. Create an app and select upload. Since the selected board type does not contain a WiFi device, the sketch will not include WiFi lookup (as defined in Appendix G.3). In this case, we have an Arduino board with a WiFi device that does not support IP lookup. Next time the user selects a board type that includes the WiFi device, the user will get the following message when trying to install the app: "Error: The app you are trying to install require a board with WiFi device. If the board you are trying to install on have a WiFi device, please install an app that does not require WiFi to fix this problem." In order to solve this issue, the user must upload an app that does not make use of the WiFi device, but still have the WiFi device explicit defined in the configuration (XML file). This new sketch will include the "IP lookup" method (found in Appendix G.3). RAPT needs this method to be available to retrieve the given IP address, to handle wireless network connections.

Chapter 6

Evaluation and Validation

6.1 Introduction

This chapter presents iterative design which describes iterations during the project, in addition to give a short insight to early versions of RAPT. Thereafter, the usability test is described with a background section, an execution section, a result section, and findings and recommendations in the last section. At the end, this chapter gives a conceptual validation of the scenarios, presented with screenshots.

Figure 6.1: Report structure: Evaluation and Validation

6.2 Iterative Design

The team has applied Lean and its principles during the project period. As a result, several prototype demonstrations have been held for the supervisor. The supervisor provided the team with feedback and thoughts on how to continue develop RAPT in the upcoming iteration. The focus on the demonstrations has always been the graphical user interface and how RAPT appears from a user perspective. Each release of a new version has always been based on feedback from previous iterations, and any comments the supervisor has given.

In this paragraph four versions of the graphical user interface in RAPT are presented. Note that only the rule creation space is displayed. The first working graphical user interface is displayed in Figure D.1. The focus in this version was to display sensors and actuators and implement a simplified drag and drop functionality.

The second version, displayed in Figure D.2, emphasized that one sensor triggers an action event.

A complete redesign was decided for the third version, displayed in Figure D.3. The sensor and actuator containers have been placed vertically on each side of the rule container and the devices can now be connected by users drawing arrows.

The final version has been updated with a new colour theme, redesign of the toolbar and new navigation. A figure of the final version is presented in Figure D.4.

6.3 Final Evaluation

6.3.1 Background Summary

This section describes a usability test of RAPT conducted on 24 and 25 April 2014 at St. Olavs Hospital in Trondheim, Norway. The test consisted of four parts; first, an interview with the test subject focusing on collecting demographic data, second, the test subjects were given tasks to complete, third, another interview focused on the tasks and how the test subjects completed them, and finally, the test subjects was given a simplified System Usability Scale (SUS) questionnaire [61]. SUS consists of ten statements with five response options. The test team simplified the questionnaire by removing three statements that was considered not applicable to this usability test of RAPT. The response option for each statement represents the test subject's agreement with the statement, from strongly agree to strongly disagree. The tasks and SUS questionnaire can be found in Appendix E. Declaration of consent was signed at the beginning of the first interview and can be found in Appendix E.4. The usability test and its tasks were primarily focused on the test subject's understanding of the main functionality of RAPT: to create apps. The test was conducted in a usability test lab, with both video and sound recorded. The equipment used during the test was three Arduino boards and two Samsung S4 mobile phones. One of the mobile phones was running RAPT, the other was used a WiFi hotspot. The test team conducting the test consisted of Anders Palfi and Haakon Svendsen Sønsteby.

6.3.2 Test Execution

The test location consisted of the two rooms, one room that the tasks were executed and the interviews were done, another room for cameras, recording, and audio control. The test subjects were first presented the camera room, and the test team showed the test subjects where the cameras were mounted. The first camera was placed straight above the table where the test subjects were seated zooming onto the mobile phone's screen. The second camera was placed so it could record an overview of the scene.

The test layout consisted of two mobile phones, three SINTEF RFID access cards, and three Arduino boards. The mobile phone running RAPT, was laying on the table in front of a chair where the test subject was seated. Three Arduino boards were placed in front of the mobile phone within arm reach of the test subject.

The other mobile phone was placed on the other side of the table. This phone was working as WiFi hotspot.

When the test started one of test team members went to the camera room. The other test team member explained the test structure, what an Arduino board is, what a usability test is, and what are being tested. In addition, the test team member answered any questions the test subject had. The test subject was also given a small introduction to this master dissertation, problem description and how the team is trying to solve the problem by creating RAPT. This introduction was not technical as it is important not to overwhelm the test subject. The test member told the test subject to explain what they were thinking during the execution of the tasks, both positive and negative thoughts. The test team had estimated 45 minutes for each test subject.

During the execution of the test tasks one of the test team members was seated next to the test subject. The test subject was told that this person could not help, but could answer questions if the test subject did not understand the tasks given. The test team member could also guide the test subject if the test subject was completely stuck on a task. The other member of the team was in the camera room taking notes, recording, and controlling cameras. Pictures displaying the usability test is presented in Figure 6.2 and Figure 6.3.

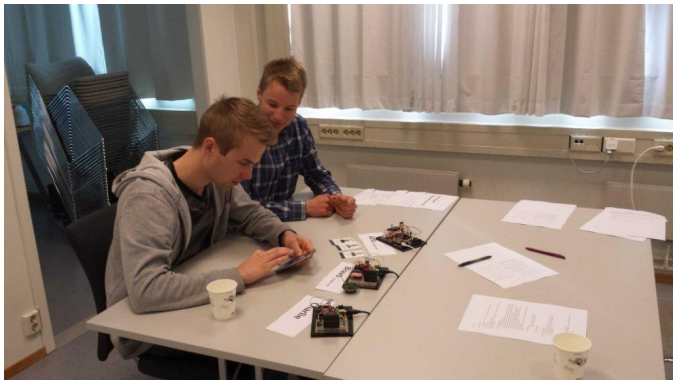

Figure 6.2: Test subject and team member

The test started with a short introduction interview where the test team collected demographic data. Ten people, six female and four male, were the test subject group of the usability test. A wide spread of professions were chosen because one of the goals of RAPT is that everyone, that knows how to use a mobile phone with touch screen, should be able to create apps. The test subject group primarily consisted of students, due to them being most easy to recruit. Majority of the test subjects were students, therefore, the average age were early- or mid-twenties. All test subjects characterised their mobile usage as normal. There was both people with experience with Android and iOS in the test subject group. Only one test subject had previous experience with programming. An overview of the test subjects are presented in Table 6.1.

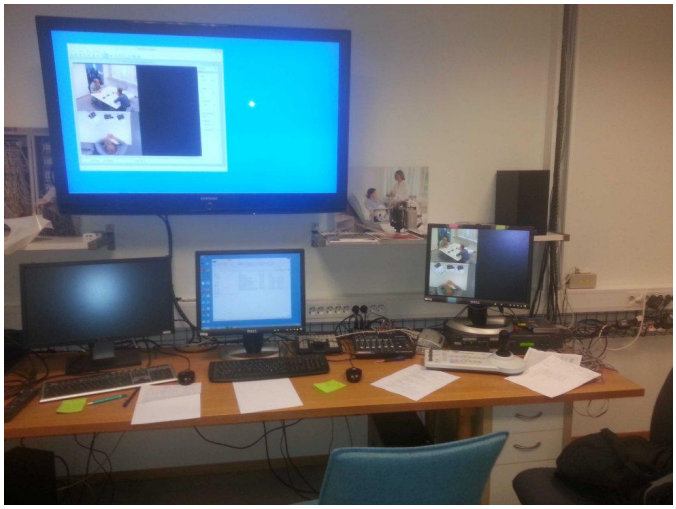

Figure 6.3: Camera control room

After the introduction, the main part of the test began. One task at a time was given to the test subject. This was done to not confuse the test subjects with a paper with too much text, as this could lead the test subjects to feel overwhelmed. The test subject was not told the total number of tasks because it could have stressed the test subject to complete tasks quick. The test subject had to complete their current task to get the next task. The least complex task was given first. Each new task handed to the test subject increase the complexity. The first task was to complete the tutorial. The test team decided to start with the tutorial, because the tutorial is an introduction to how to create apps. If the test subject was on a task that needed WiFi authentication information (SSID and password), they were given the information needed on a paper note. If the test subject was stuck a task, the team member guided him/her on how to complete it. If the test subject needed guidance on a task, the task was deemed not completed. Even though the test subject used several tries, or felt confused, before finishing a task, the task was still considered completed. If the test subject completed the task on the first try, and without being confused, the task execution was considered successful.

The third part of the test was an interview focusing on the tasks and the difficulties that had occurred during the test. The test member that had been sitting in the camera room now joined the other test member and the test subject to discuss findings. During the task execution, the team member in the camera room had been observing and taking notes, besides controlling the cameras and audio. These notes were used during the discussion with the test subject. The test subject was asked what was inconsistent and difficult, and what was easy and intuitive. In addition, the test subject had the opportunity give suggestions that could solve the difficulties that he/she experienced.

After the interview, the test subject answered a SUS questionnaire. The ques-

tionnaire can be found in Appendix E.2. The test team explained the test subject that the questionnaire was anonymous, and honest answers were both preferred and a necessity for further development of RAPT.

Test subject number	**Age**	**Gender**	**Profession**	**Mobile operating system**
1	23	Female	Studying master's degree in Industrial Design	Android
2	24	Male	Studying master's degree in Geology	iOS
3	24	Female	Studying master's degree in Natural Science with Teacher Educationn	Android
4	19	Male	Studying bachelor's degree in Informatics	Android
5	24	Female	Studying master's degree in Natural Science with Teacher Education	Android
6	24	Male	Finished a master's degree in Civil and Environmental Engineering. Currently a studying a year of pedagogy	Android
7	25	Female	Studying master's degree in Special Pedagogy	Android
8	23	Female	Taking preliminary courses to higher education	iOS
9	49	Male	Working as a doctor at St. Olavs Hospital in Trondheim	iOS
10	24	Female	Studying master's degree in Natural Science with Teacher Education	iOS

Table 6.1: Test subject demographics

6.3.3 Test Results

Test results are based on an interview done after the test subject had executed all the tasks, and notes taken by test team members during task execution, In addition, the SUS questionnaire, and recordings of the test is used in the test results. The first task have been ignored due to the task only focusing on the tutorial giving the user an introduction to RAPT.

There are limitations of the test results. The majority of the test subject group were students. They were recruited from a broad spectre of field of studies, but very with very little variation regarding age. Ideally, the test group should have a broad variation of both age, profession, and smartphone usage.

The usability test was conducted in a lab. This is an artificial context for the test subjects and might place them outside their comfort zone. The idea of

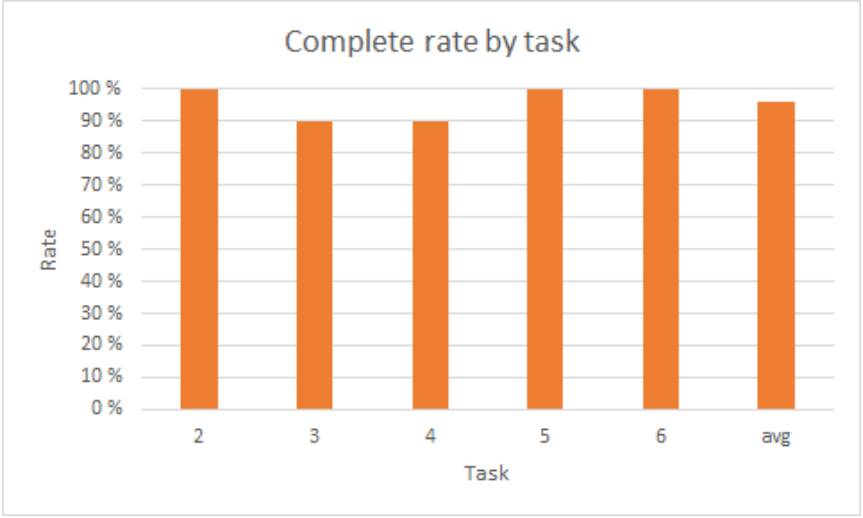

Figure 6.4: Complete rate by task

cameras recording every movement does not help the test subject to feel more comfortable. In addition, people often have a pattern on how and when they use their smartphone. Sitting in a silenced lab with curtains closed does not make them feel at home.

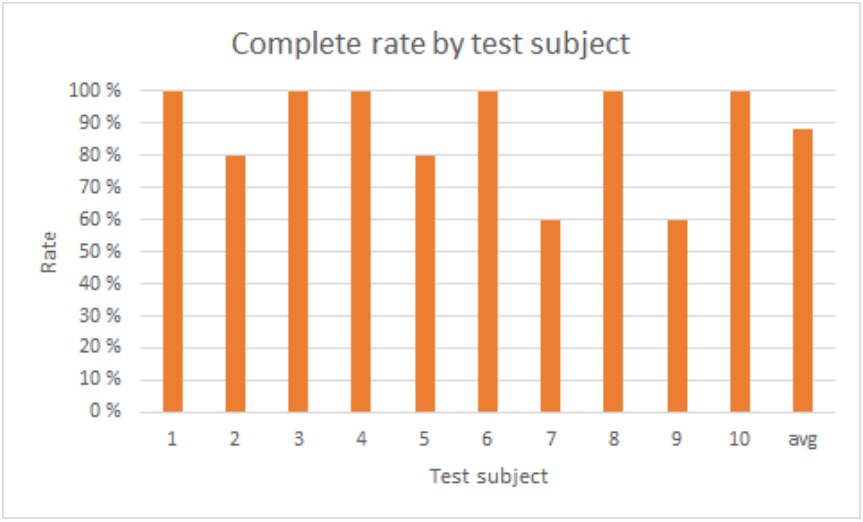

Figure 6.5: Complete rate by test subject

By a few occasions, RAPT crashed during the usability test. The test team member sitting next to the test subject paused the test and restarted RAPT.

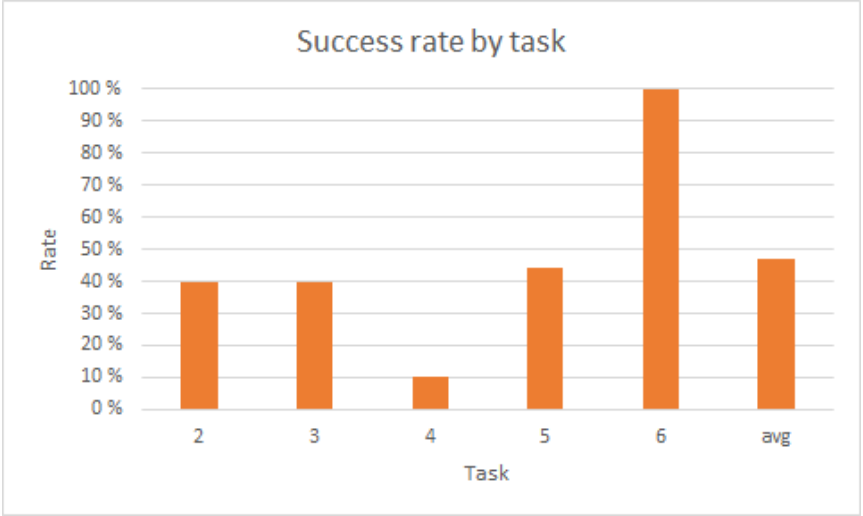

Figure 6.6: Success rate by task

Although, this do not affect the rest results directly, it forces the user to start over and might have affected the results indirectly.

The requirements for a task to be considered completed is that the test subject managed to solve the task, even after multiple tries. Task not completed due to time constraint have been excluded when calculating complete rate by task. Complete rate are illustrated in Figure 6.4 and Figure 6.5.

The usability test has shown that some test subjects managed to complete all tasks within the given time constraint. Test subject five and seven failed one tasks each because they did not manage to solve the task without guidance from the test team. Test subject two and seven did not finish task six, in addition to test subject 9 who did not complete both task five and six. The reason the three test subjects did not finishing the tasks was the 45 minutes time constraints given to each test subject.

For a task execution to be included in the success rate, the test subject must complete the task on the first try without being confused. Tasks not completed due to time constraints have not been included when calculating success rates by task. Success rates are presented in Figure 6.6 and Figure 6.7.

The task with the highest success rate are task six. This task was considered the most difficult of all the tasks by the test team. During this task the user had to create two rules on the same board representation, both rules using the same RFID reader, but depending on the RFID card read different led should flash. All seven test subjects that was given this task completed it successfully. The least successfully completed task was task four. This task was the only task that forced the user to create an app including multiple board representations. Only one out of ten test subjects managed to complete task on the first try without being confused.

The results of the SUS questionnaire can be found in Appendix E.3.

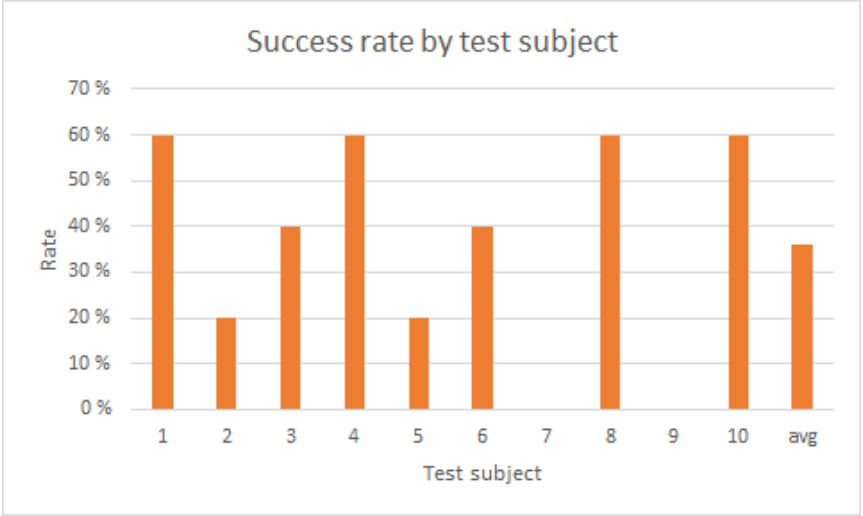

Figure 6.7: Success rate by test subject

6.3.4 Findings and Recommendations

The task most test subjects had difficulties solving was task four. The problem test subjects had during this task was creating rules including multiple board representations. The test subjects did not find the speaker device belonging to the second board representation when they had dragged in a button device from the first board representation. Some test subjects said in the second interview, done after the tasks, that they did not manage to navigate to the second board representation. Others said they were certain the button they had already dragged from the first board representation would be removed when switching between board representations. Even though many found switching between board representations difficult at first, they thought the current solution is a good solution after they first had learnt it. The test team was recommended to implement a button on the task bar that should be used for switching between board representations, as some test subjects felt the need for button with this functionality.

Almost all test subjects were confused during the installation process after loading an app from RAPT app store. The test subjects that had chosen an app that required only one Arduino board managed to pair board representation with the correct Arduino board. Test subjects that had chosen an app that required two Arduino boards was confused with the board settings screen, where they had to pair both boards representations with the correct Arduino boards. The board settings screen are shown in Figure 6.8. Many test subjects thought they had to tell RAPT which Arduino boards should communicate with each other. Therefore, the two sketches generated during installation was installed on the wrong Arduino boards. The test team did not manage to get feedback on how to solve this problem, but some test subjects mentioned they were surprised that the installation process

differed from the installation process on previous tasks.

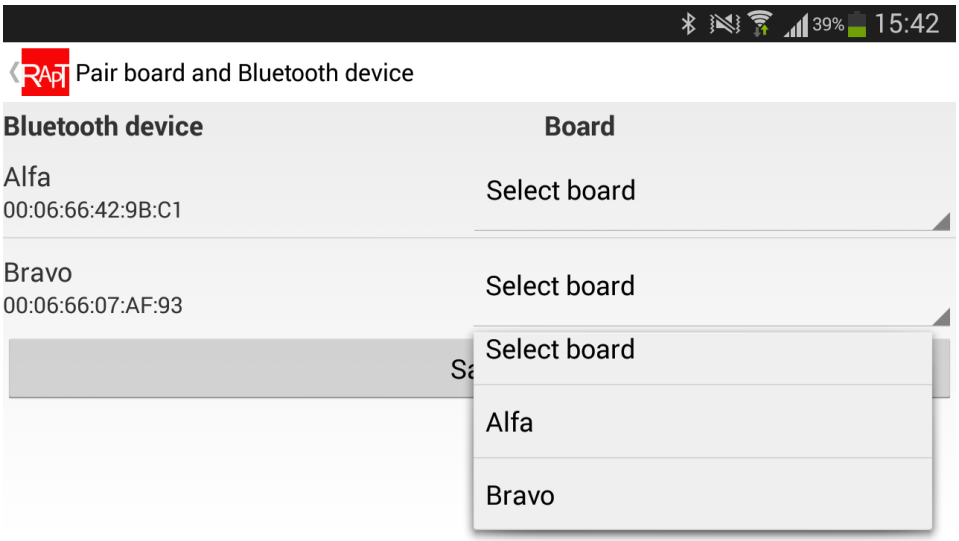

Figure 6.8: Board settings when installing app that requires multiple board representations

The test team observed that the test subjects had an easy time creating apps. Dragging devices from the sensor and actuator containers into the rule container to create rules, was intuitive as all members of the test subject group managed this. Connecting devices together was also easy for the majority of test subjects, as well as removing devices from the rule container. A few test subjects were confused when they were given the task to rearrange the layout of a rule. The test subjects did not understand what the different editor modes were, and how to switch between them. A few test subjects clicked the connect devices mode button between every action. From observation it seemed plausible that there is three possible reasons for these problems. First, test subjects did not understand which buttons switched between modes. Second, test subjects did not understand which mode were currently active. Finally, test subjects did not know that there were any modes at all. The mode buttons are presented with a red outline in Figure 6.9. Recommended improvement to solve the difficulties surrounding editor modes are making the mode buttons more intuitive, or focusing more on the different modes in the tutorial.

Multiple test subjects told the test team during the second interview that they thought RAPT was easy to use once learnt. This is reflected in the success rate of task six where every test subject handed this task completed it successfully, even though they had problems on previous, easier tasks. The success rate of task six can indicate that the previous tasks where harder than task six even though the app created in task six where more complex compared apps created in previous tasks. Another indication, test subjects learnt the system quick and found it easy

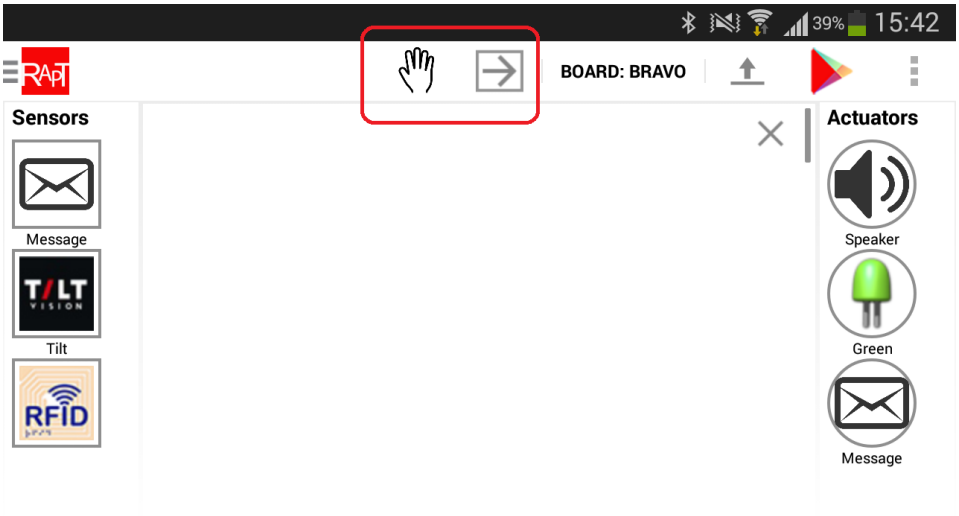

Figure 6.9: Outlined mode buttons

to use. In addition, the results from the SUS questionnaire found in Appendix E.3 indicates that the test subjects found RAPT easy to learn.

6.4 Conceptual Validation

To validate that RAPT has implemented necessary requirements, this section will present apps created for all three scenarios. With RAPT's current device support it is not possible to fully implement the apps for home assistance scenario or restaurant scenario, as some of the devices needed for these scenarios are not implemented. Apps for these scenarios will be presented, with the missing devices added as GUI elements. No underlying functionality for these devices is implemented.

Restaurant

To implement the restaurant scenario, the following devices are needed; an RFID reader, one RFID tag for each dish, a green led, a screen, a button, a yellow led and two Arduino boards with WiFi. Two Arduino board is required because there is a limit of how many devices that can be wired to an Arduino board. The first Arduino board contains all devices, except the screen. The screen is wired to the second Arduino board. The restaurant needs to make use of minimum five rules to realise the scenario. The app is shown in Figure 6.10.

First rule is with RFID reader as a sensor and send message as an actuator. The RFID reader is configured to read the RFID tag of a dish. When the tag is read a message will be sent to the external kitchen server with the dish's number, in addition to the table number. For each dish, a new rule is required. Second

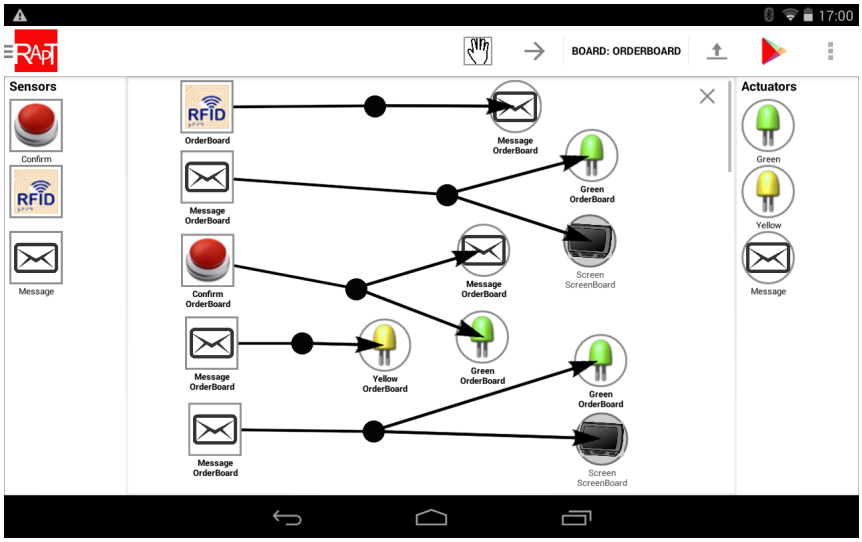

Figure 6.10: Screenshot of restaurant app

rule states that when the first Arduino board receives a specific message from the kitchen server, the green led will start blinking. In addition, the screen on the second Arduino board will display the name of the dish ordered. Third rule consists of a button as a sensor and send message and green led as actuators. The functionality of this rule is that when the button is pressed for three seconds, the Arduino board sends a message to the kitchen telling the kitchen that the order is complete. In addition, turns the green led on. Forth rule is created by using receive message and yellow led. This rule is configured to turn on the yellow led for five seconds when the kitchen server sends a message to notify that the food is ready. The last rule resets the green led and screen to default state. Note that the current version of RAPT does not support screen device. Screen is a device that should be supported in later versions. Figure 6.11 displays the screen the user is presented when configuring a receive message. The text written in the receive message must be mapped with the messages sent by the kitchen server. Example of the message sent by the kitchen server when an order has been received could be “orderReceivedNumber16”. Then the dish with number 16 will be displayed on the screen when the Arduino board receives this message. Figure 6.12 presents the interface for configuring a send message. Port, IP and a message is required. Example of the message text for the first rule can be “orderNumber16Table1”. When the kitchen server receives a message with the text “orderNumber16Table1”, it registers this dish as ordered by table 1. After the dish is registered, it sends the message “orderReceivedNumber16” back to the Arduino board.

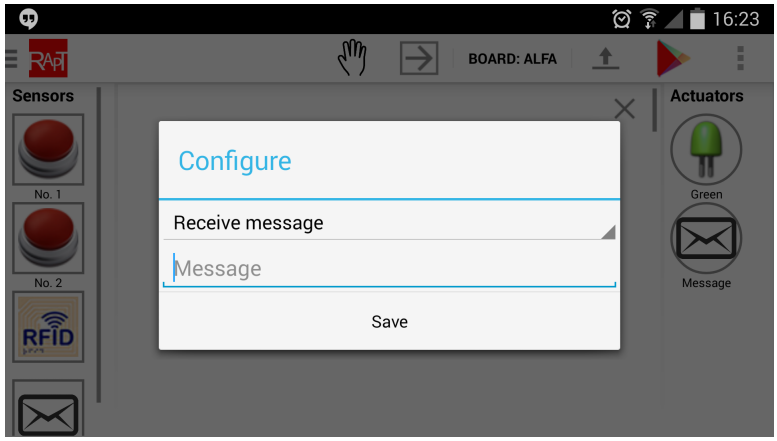

Figure 6.11: Screenshot of receive message

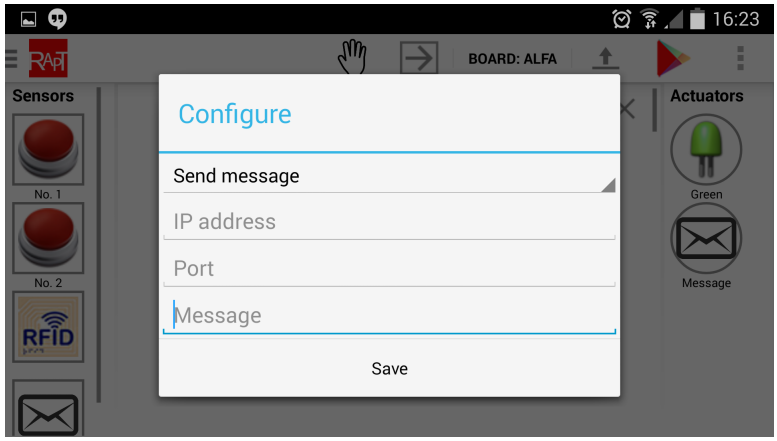

Figure 6.12: Screenshot of send message

Buddy Notifier

To implement the buddy notifier scenario, the following devices are needed; a button, a speaker and two Arduino boards with WiFi. As these two boards are not in the same local network, the messages sent between the Arduino boards are needed to be specified explicit in the apps. Two apps are needed in this scenario, one for each board. The first Arduino board needs an app with the rule: When a button is clicked, send a message to the second Arduino board.

The second board needs an app with the rule: When a specific message (message with the same text as the message sent by the first board) is received, play a melody on the speaker. The app for the first board needs the IP address of the second Arduino board. This can be found by connecting the second board to its WiFi

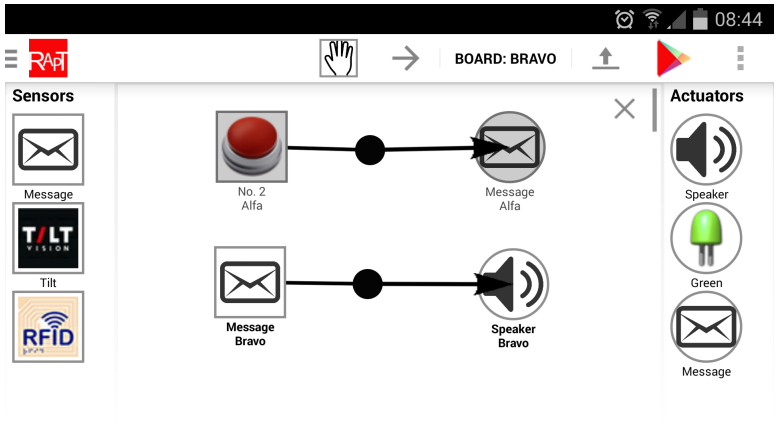

Figure 6.13: Screenshot of buddy notifier app

network. The IP address of the Arduino board is then displayed in network settings. Figure 6.13 presents the two rules. Board representation "Alfa" corresponds to the first Arduino board and board representation "Bravo" corresponds to the second Arduino board. Note that the rules are displayed in 6.13 should be split into two apps, and not just the one app presented.

Home Assistance

To implement the home assistance scenario, following devices are needed; a power device, a light bulb, a pressure sensor and two Arduino boards with WiFi. The first Arduino board, with the power device attached, is placed in the kitchen by the stove. The second Arduino board, with the light bulb and pressure sensor, is placed in the bedroom. Figure 6.14 presents the app created for this scenario. The first rule needed is a rule with the power device as a sensor and a light bulb. The power device is configured to sense if the stove is turned on. If it is turned on a message will be sent from the kitchen Arduino board, to the Arduino board placed in the bedroom. When the bedroom Arduino board receives this message the light will be turned on. Additionally, a second rule is required that configures the pressure sensor to feel that trigger a send message when it feels pressure. Furthermore, the light is turned off. This message will be sent to the Arduino board in the kitchen. When this message is received the power device will turn the stove off.

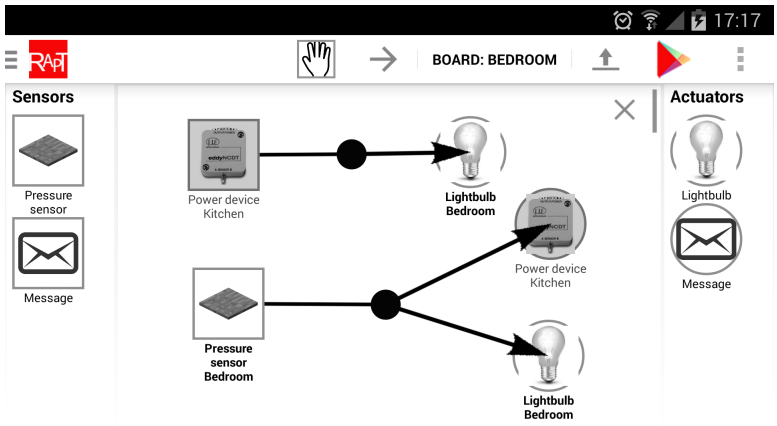

Figure 6.14: Screenshot of home assistance app

Chapter 7

Conclusion

7.1 Introduction

This chapter summarises the dissertation, and discusses results of the usability test while also adding reflections in hindsight. Moreover, recommendations for further work and development are presented.

Figure 7.1: Report structure: Conclusion

7.2 Summary

We used systematic mapping study to research and find the state of the art within tangible user interfaces and end user development. Thereafter, we used the gained knowledge of the domains as a foundation to create a toolkit to program Arduino boards through a mobile phone.

Drag and drop became the selected end user development technique after discussing several techniques discovered in the pre-study. In general, important benefits of drag and drop technique are: It is intuitive, Android or mobile phones in general have touch screen, as well as the importance to use the most of a limited screen space.

Findings in the physical user interface domain were among others the importance of using cognitive advantages and that it often is easy for users to relate to objects. These are aspects to be aware of and simultaneously exploit. We manually searched for toolkits that made use of both tangible user interfaces and end user development to further discover existing implementations within the domains. The relevant toolkits helped to give insight to both drawbacks and advantages which contributed to enhanced decision making.

RAPT (Rapid Arduino-Prototyping Toolkit) is our technical implementation. It is designed for non-technical users to create apps for Arduino boards. RAPT maps hardware devices on the physical Arduino board with icons representing these within the graphical user interface. Furthermore, the user can configure the behaviour on each device. To create an app, the user drag and drops devices and connect them with arrows to make a sensor active an actuator. Thereafter, the user is able to install the app on his/her Arduino board via Bluetooth. Alternatively, the user can search within an app store to find existing apps to further develop or directly install.

As the final evaluation, a usability test was conducted. The test gave positive results, indicating that the concept of configuring an Arduino board through a mobile phone is easy to understand. At the same time, user feedback from the test provides us with several possible future expansions of RAPT.

7.3 Discussion

This dissertation has performed a usability test on RAPT. The test group consisted of ten people. They were mostly students with no programming experience. The age of the people in the test group varied from 19 to 49 years. They were recruited from a broad spectre of fields of studies. Due to the majority of student in the test group, the composition was not optimal regarding age and occupation. The optimal test group would consist of a wider spread of age. Additionally, having both students and people with regular jobs would be desirable. We believe that the test results would not be as different from the test results collected. This is because the test group already had large variations of technical experience. Increasing the age variety would not significant increase the variations of technical experience. Therefore, we believe that the parts of the app the test subjects had difficulties with would not change by increasing the age variety of the test group.

The usability test revealed some areas with room for improvements. There were especially two design decisions that did not appear intuitive for the test users at first sight. First issue was switching between board representations, which allow users to create communication between Arduino boards. Second issue was mapping board representations to Arduino boards. We were not surprised by test subjects having these issues, since these parts of RAPT were considered by us to be least intuitive.

The usability test was expected to find issues and room for improvements. However, looking at the large picture, a majority of the test persons explicit claimed that RAPT's concept was easy to use once learned. No test user had any major issues with using sensors and actuators, and connects them by an arrow. In addition, results from the system usability scale questionnaire in Appendix E.2 were overall positive and they indicated that RAPT is developing in the correct direction.

The hardware prototyping platform selection of this master dissertation was specified to be Arduino (see Section 1.1). There are several microcontrollers on the market, for instance Raspberry Pi [62] and Beagle Bone Black [63]. Adding support creating apps using other hardware prototyping platforms (microcontrollers) would

both be a realistic and a useful addition to RAPT.

As presented in [64], adding communication with existing social media and other social applications would be important in further development of RAPT. Sending emails, updating status on Facebook, or tweeting by clicking a button or tilting a physical object would be a great addition to RAPT. These external systems could also work as sensor devices. For example, a led could start blinking when an email is received.

7.4 Further Work

7.4.1 Compilation

The current implementation of RAPT uses an external compilation server to compile the generated sketch. Internet connection is therefore required before an app can be uploaded to an Arduino board. If the compilation is accomplished directly on the phone, rather than on a server, the Internet requirement would be excluded. Compiling cpp-code on an Android phone is possible to do as proven by ArduinoDroid [40]. However, there are also issues such as the compiler size. Section 3.4.3 elaborates the issues and advantages.

7.4.2 Local Storage

As for now, RAPT does not save data to the file system or other persistent storage. Saved data, for example drafts and apps in apps store, are saved to the phones memory. To be able to access saved data later, RAPT's memory allocation must be unaltered (for example the garbage collector cannot not to free up this space, the memory cannot be cleaned up manually, and phone restart cannot happen). However, this is not a way to save data for longer sessions or persist saved data over multiple sessions. A useful feature would be to create files on Android's file system or use a database.

7.4.3 Devices

RAPT has added some devices (sensors and actuators) with functions that each device can perform. Pressure sensors, temperature sensors, light sensors, GPS and motors are examples of devices that should be added as these would be useful when creating tangible user interfaces with RAPT. In addition, coffee makers, TVs, ovens and more can with adjustments be future devices.

One of the design choices in RAPT is separation of concerns, with focus on further development at a later stage. However, there is still need for some further development to optimize adding new functionality.

All device specifications and their possible functions are separated in XML files, and the translation code to cpp-code is defined in text files. However, these lies within the RAPT APK (the application package). To be able to change these files one must decode the APK-file (or get hands on source code) and redeploy

the application. For new devices, a corresponding picture must be added as well (currently in the res/drawable folder). This is too cumbersome for any user. The goal, the further work, is to save these files outside the application package in order to use the same deployed version of RAPT. It would be beneficial to create a download function for the XML- and text files, alternatively read them directly from an URL. Another solution to get configuration files, for example when RAPT is launched for the first time, is to make use of QR codes. On the Arduino board's box, a QR code can redirect the user to download or read configuration files.

7.4.4 Store Sensor States

To greatly increase complexity of apps created within RAPT, a user could be able to create apps that can store information about sensors. For example, a triggered sensor could set a flag, that later could be used to trigger actions. A scenario could be somebody lying down in the bed and the stove is turned on. These two conditions then trigger an alarm. More specific, when the stove in the kitchen is turned on a flag is set, and similar when bedroom's pressure sensor receives an input of someone lying there. If both flags are set an alarm will trigger.

7.4.5 Ease Usability

In the current version of RAPT, installing an app to multiple Arduino boards simultaneously is done in advanced mode. During the user evaluation, see Section 6.3.3, it was clear that manually connecting boards (software) to a corresponding Arduino board was confusing. There is no doubt that this operation can be made easier and more intuitive in further improvements of RAPT.

Results from the user evaluation, see Section 6.3.3, also reveal that some users found it difficult to navigate between different boards in RAPT. Rather than having a list of available boards in the Android slide in menu, images of the actual boards and how they are connected might increase users overall understanding. For example, one could add a snapshot of nearby boards as a network diagram, with updated connections as the users draw arrows between board representations.

RFID tags must be specified in the XML file of the boards' specification, and new tags cannot be added from within RAPT. For this to be possible. A possible solution is to create a standard Arduino sketch that reads RFID input and stores them. This sketch could be uploaded to the Arduino board on user command. Another possibility is to add the functionality in every generated sketch, as the IP lookup utility (see Section 5.3.4).

7.4.6 App Store

The app store in RAPT is only a stubbed implementation of what the app store could be. Having a complete app store with search functions, categories, user rating and user submitted content would be a logical next step in further development of RAPT.

7.4.7 IP mapping

To avoid problems if an Arduino board gets an new IP address from the network's access point, a solution could be to implement an address resolution protocol (ARP) [65]. In the current version of RAPT, the IP addresses are set in the compiled code and the sketch needs to be recompiled if IP addresses change. ARP's function is to map MAC addresses with IP addresses, and thereby not be dependent on a static IP address. This solution is only possible when all the connected devices are within the same local network.

When the Arduino boards are connected to different network, it is the job of each network router (if existing) to keep track of the connected devices IP addresses. However, if the ISP (Internet Service Provider) not offer static IP addresses, a similar problem as above can occur. A possible solution could be to implement an own mapping of MAC addresses or another unique name to IP addresses on a server. The sketch would then be compiled with the unique key (MAC address or name), and use the server to lookup the current belonging IP address.

Appendix A

Systematic Mapping Study

NTNU
Norwegian University of
Science and Technology

Systematic Mapping Study of Tangible User Interfaces in Social Computing, and End User Development

May 21, 2014

Authors:
Daniel T. Abrahamsen
Anders Palfi
Haakon S. Sønsteby

Supervisor:
Babak A. Farshchian

Abstract

The progressive research and use of tangible user interfaces, also called physical user interaction, have revealed major opportunities within the field of human computer interaction. Types of interfaces are continuing to emerge and combining these interfaces with the social domain to support collaboration. Demanding users and a vast increase of data availability have made it more important than ever to adapt the application to the users and not the other way around. End user development aims to allow the user to create, customize and tailor applications, in contrast of hiring a professional software developer.

This study is a preliminary study for a master dissertation regarding tangible user interfaces and end user development within the domain of social computing. This paper conducts a systematic mapping study in three main fields: End user development, tangible user interfaces and social computing. The goal of the research was to identify solutions that are using tangible user interfaces in the domain of social computing, and gather knowledge regarding characteristics, advantages and challenges of these different solutions. Furthermore, the paper presents an overview of various end user development interfaces found in existing solutions, and strength and weaknesses of these.

45 studies of tangible interfaces and 23 studies of end user development were identified. The outcome of this study is a visual map giving an overview of the research area, classifications of the studies found, in addition to a literature base to be used for further development.

Keywords: Tangible user interfaces, physical user interfaces, social computing, social software, end user development, end user customization

Introduction

The progressive research and use of tangible user interfaces, also called physical user interaction, have revealed major opportunities within the field of human computer interaction. Types of interfaces are continuing to emerge and combining this technology with the social domain to support collaboration and learning is a challenge [S41], [S43]. Demanding users and a vast increase of data availability have made it more important than ever to adapt the application to users and not the other way around. End user development aims to allow the user to create, customize and tailor applications, in contrast of hiring a professional software developer [S80].

Software engineering studies are affected by a huge number of empirical studies, which leads to the need for a systematic research when studying particular topics within computer science [3]. Most commonly method used is the systematic review with the goal of Identify Best and Typical Practices [4]. In the growing field of tangible user interfaces and end user development, a detailed research such as systematic review could be too much to comprehend. By focusing on classification and thematic analysis a larger number of articles and documents can be analysed and processed. Systematic mapping is a method that make use of classification, thorough overview, and visual mapping. It is worth noticing that one methods does not exclude the other [4].

This paper will present an execution of a systematic mapping study focusing on the characteristics, advantages and challenges, of tangible user interface solutions within the domain of social computing. Additionally, the paper identifies different types of interfaces in end user development and strength and weaknesses of these. This paper is written as a preliminary study for a master dissertation. This paper is devided into sections. In chapter Research method the research method used, systematic mapping, will be described along with the steps of implementation. Furthermore, in the chapter Result, the result of the study will be presented, before the Discussion chapter discuss the method as well as the findings, limitations and conclusions of the findings.

Research method

The objectives of this study are to find studies presenting tangible user interfaces in social computing, or studies presenting different kinds of end user development, as a preliminary study for a master dissertation. The relevant studies found are going to be used as a literature base that will be used to guide and direct further research within the area. A systematic mapping study has been selected because the main goal of a systematic mapping studies is according to [4] not to provide specific details, but to get an overview of a research area. This mapping study follows the steps described in [5].

In order to address the wide scope of a systematic mapping study, the research

Table 1: Research questions

ID	
RQ1	In which fields of social computing are tangible user interfaces developed or suggested, and what characterises these tangible user interfaces?
RQ2	What are the characteristics, challenges and advantages of tangible user interfaces in social computing?
RQ3	What types of user interfaces in end user development exist and what are their weaknesses and strengths?

questions are likely to be much broader than in a normal systematic literature review [5]. By following the mentioned guideline and our goal of finding relevant studies of both tangible user interfaces in social computing and end user development, three research questions were created. These questions are presented in Table 1.

According to Budgen [5] a mapping study is very time-consuming. Budgen suggests to constrain searches to a set of digital databases to reduce the amount of search results. The limited set of electronic databases for this study consists of EngineeringVillage[1] and Scopus[2] online databases. Only studies found in these databases are used in this systematic mapping study. To further limit the amount of results only the title and abstract of each study will be searched.

The process of building the search string consisted of three stages. First using our research questions to find important terms, second finding terms from already known relevant literature, and third finding synonyms for these terms. The search string for each database can be found in the Appendix A.

Relevant studies where found by filtering the search results with the use of both inclusion- and exclusion criteria. Inclusion criteria used: (1) Only studies written in English; (2) Only journal articles, conference papers, books, reports, book sections, master dissertations and doctorate thesis; (3) Only studies which of the abstract or title must give the impression of being relevant to find answers to one or more of this mapping study's research questions. The exclusion criteria were: (1) Studies not accessible; (2) Duplicate studies. In this case, just one study will be considered; (3) Studies reporting similar results; (4) Collection of studies. If this is the case, only the most complete study will be considered. After applying the inclusion and exclusion criteria we were left with a subset of the search results consisting of relevant studies.

The method of keywording the papers' abstract described in [5] was used to help later classification and characterisation of each relevant study. To answer the research questions all relevant studies found were read. Studies that were found not relevant were removed from the set of relevant studies.

Results

By applying the search string to EngineeringVillage[1] and Scopus[2] 415 studies were found, 197 from Scopus and 218 from EngineeringVillage. 405 studies were downloaded, as Scopus marked 10 of the 197 studies as unavailable for download. We started considering the inclusion and exclusion criteria by removing 201 studies because of duplication. 87 studies were deemed relevant after all studies were filtered by inclusion and exclusion criteria. Each study were given a study ID from S1 to S87. All relevant studies were read and a spreadsheet was made to record data about how each relevant study would answer one or more research question. After reading all studies, 65 studies were still considered relevant, as 22 studies were removed because of not answering the research questions. The main steps of the process is presented in Figure 1. All 65 studies judged relevant are listed in the Appendix B. The answers found for each research question are presented in the following paragraphs.

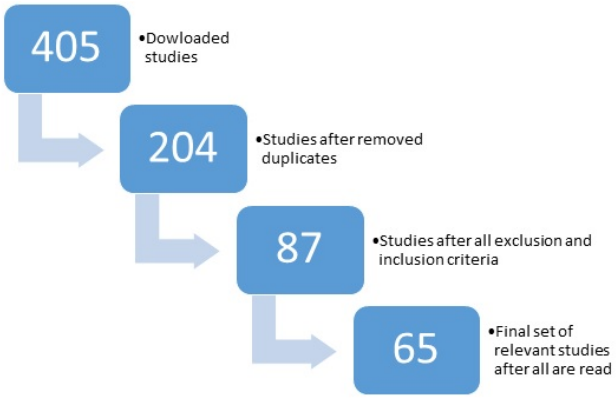

Figure 1: Steps of finding relevant studies

Figure 2 presents the publication date of the studies, sorted by research question.

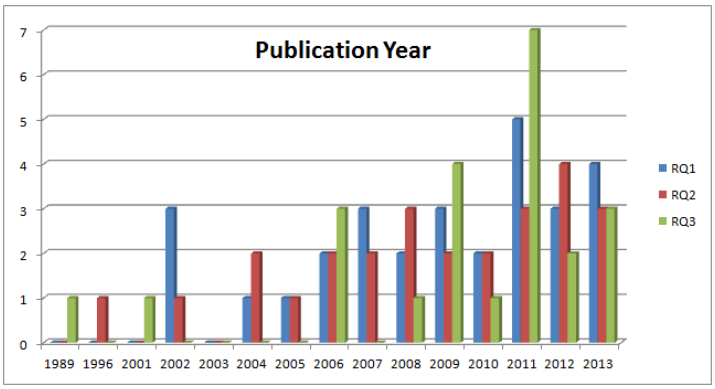

Figure 2: Publication year

RQ1: In which fields of social computing are tangible user interfaces developed or suggested, and what characterises these tangible user interfaces?

Table 2 presents fields of social computing in which tangible user interface solutions are developed or suggested. Figure 3 visualizes that the findings indicate tangible user interfaces focused on children are the most common based on the relevant studies, along with solutions trying to improve both remote and face-to-face collaboration.

Table 2: Relevant studies in which field of social computing

Field	**Study ID**
Elderly	S21, S67
Children	S1, S2, S14, S20, S31, S32, S36, S48, S78
Disabled	S31, S32, S67
Remote collaboration	S4, S34, S47
Face-to-face collaboration	S6, S15, S28, S70, S82
Augmented/Mixed reality	S4, S6, S16, S70, S74
Entertainment	S12, S16, S20, S64, S74
Travel	S26
Shopping	S84
Cooking	S63
Home	S50, S63, S69
Friends	S69
Healthcare	S53

A notable observation is that all relevant studies found in the field of elderly, children or disabled are mainly focused on one user at one location. [S1], [S2],

[S14], [S20], [S78] are focused on children learning, creating or solving problem with the use of some kind of interactive tabletop. There have been less to none focus on remote collaboration or communication between remote users, comparable to the communication between remote friends in [S69], in these fields.

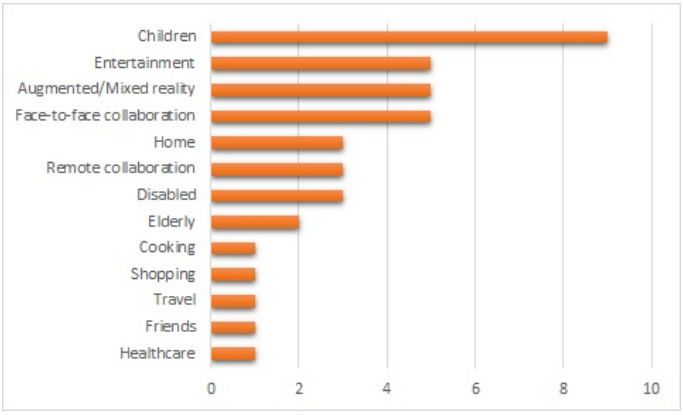

Figure 3: Amount of solutions within fields of social computing

Table 3: Characteristics of tangible user interface solutions in relevant studies

Characterstic	**Study ID**
Tabletop	S1, S2, S14, S15, S20, S28, S47, S53, S63, S64, S78, S82
Head-mounted display	S4, S6, S16, S70, S74
Infrared communication	S12
Wireless communication	S16, S53, S64
RFID	S26, S31, S32, S36, S48, S67
NFC	S21, S67, S84
Bluetooth	S26, S34, S48, S53, S69
Kinect	S15
Sensors	S16, S50, S53, S64, S69
Mobile phone	S50, S67
Personal digital assistant	S70
Wearable	S16

Table 3 lists what characterises the solutions found in this study. Tabletop is used as a wide term: Solutions involving interaction with a horizontal surface have been included in the characteristic named tabletop. The solutions which have "Sensors" as a characteristic are using sensors which is not presented as

another characteristic, e.g. pressure sensors [S69], accelerometer [S64] and home automation sensors [S50]. A notable result is that head-mounted displays from Table 2 and augmented/mixed reality are heavily coupled. This is because every solution found with augmented/mixed reality uses a head-mounted display to create illusions of virtual objects, as seen by the user, into the physical world.

RQ2: What are the characteristics, challenges and advantages of tangible user interfaces in social computing?

Tangible user interfaces can appear in several forms (for example as tabletop or physical blocks), and the characteristics will vary depending on the implemented solution. However, there are several similar characteristics, listed in Table 4, that can be seen as almost universal in tangible user interfaces, and in contrast to normal graphical user interfaces. It is important to notice that this is not a complete nor a definitive list.

Table 4: Characteristics of tangible user interfaces

Characteristic	**Description**	**Study ID**
Real-time feedback	Users receive feedback from physical objects as they grasp and manipulate them.	S43, S44, S49
Persistency of tangibles	Physical objects hold an informative state, even when a connected device is turned off.	S43
Input/output	There is no separation of input and output channels.	S43, S44
Space-multiplexed input	In contrast to GUI (time-multiplexed), TUI can provide space-multiplexed input.	S43, S49
Support IUUI (Intuitive Use of User Interfaces)	The affordances of physical objects are often highly apparent.	S46

Table 5 contains advantages of tangible user interfaces. Easy-to-use, collaborative benefits and the reinforcement of creativeness are generally the most emphasized. In addition, study [S83] mentions that tangible user interfaces mimic the F2F environment, which is stated in [7] that is the best kind of communication, which can be seen as an indirectly benefit.

Even though it is not directly mentioned that co-location is important in several articles, the studies [S33], [S38], [S14], [S41], [S49], [S54], [S55] and [S79], all make use of tabletops that require users to be located around the same tabletop.

Another aspect observed in [S54] and [S55]: When graphical user interfaces are used, with users co-located, communication and cooperation are often verbally, while when use of tangible user interfaces, the communication is object oriented (users try by doing, moving and gesturing).

Table 5: Advantages of tangible user interfaces

Advantage	**Description**	**Study ID**
Reinforce creativeness	Especially in modelling, designing, and visual testing.	S33, S37, S44, S49, S54, S55, S83
Enhance co-creativeness	The creative process enhanced when co-located.	S37, S49, S55
Easy to use	Usually little training needed to incorporate TUI: Users explore, and are often able to use TUI with minimum effort (even if the interface is not optimal).	S12, S21, S22, S26, S33, S35, S49
Cognitive advantages	When handing objects that exist in real life, TUI is proven more effective and useable than GUI, due to that objects are more graspable. Effective TUI design can result in pragmatic, exploratory, collaborative and cognitive benefits.	S2, S14, S46, S54, S55, S79, S83
Collaborative benefits	Physical objects, e.g. passed around between users, can make collaboration effective. It could also be easier to understand each other even with different backgrounds.	S2, S26, S44, S54
Easy to manipulate objects found in real life (Free-form modelling)	Gesturing and grasping with mechanical systems (e.g. 3D modelling, CAD Systems)	S9, S44, S55
Benefits impaired users	For example, impaired children and elderly can benefit highly from TUI compared to normal GUI. Cognitive aspects play a role.	S21, S31

Table 6: Challenges of tangible user interfaces

Challenge	**Description**	**Study ID**
Lack of TUI modelling capability	Currently, TUI often uses pre-defined models (that does not support real-time free form modelling)	S33
Remote collaboration must be implemented carefully	Users often benefit more when they are located together (especially in a collaborate setting). Presence disparity (the awareness of others) are important.	S33
Finger gestures might not be the best alternative	Using physical objects for controlling digital information improves performance compared to using finger gestures.	S38, S79
Object selection	Studies have shown that in object selection, a mouse may be the best input device.	S2
Cognitive challenges	When the handled information object is abstract (e.g. as query tool for database) or unnatural action must be taken on an object, it could apply a cognitive load.	S12, S14
Hardware limitations	Different sensors must be in place.	S26, S28
Standardization of common functions	E.g. functions such as save, write, exit, delete are standard for GUI-experienced users. The question is how to make a standard that is efficient and possible to all TUIs.	S28
Commercial viability	It is challenging to turn a TUI-based product into a commercial product.	S28
Specific/concrete vs. generic/abstract	Tailored TUI applications cannot be reused for most other applications (due to the physical objects)	S43
Tangible elements could distract	If mixed (tangible and non-tangible), the tangible could distract from intangible elements.	S50
Seamlessness	Need to be seamless and without overhead to be usable.	S79

Other studies, [S4] and [S50], suggest that mixed reality is one of the best solutions within the domain of collaborative tangible user interfaces due to both users' backgrounds and skills, and that tasks differ in how they are easiest executed.

RQ3: What types of user interfaces in end user development exist and what are their weaknesses and strengths?

Looking at the variety of the papers found relevant, the terminology used to describe the interfaces is often ambiguous. To overcome the difference in terminology, similar types of interfaces were combined and categorized. Table 7 is an explanation of the different terms.

Table 7: Explanation of end user interface terms

Type of interface	**Explanation**
Natural language	Eliminate the syntax difficulties and turn the programming language into more natural language.
Block structured/ drag and drop/component based/ rule based	One component, possible to customize and tailor, containing functionality, usually integrated into a bigger piece of software.
Wizard driven/ step-by-step/form-filling	A sequence allows the user to incrementally insert and configure the component.
Application customization/ component tailoring/ unwitted programmers	An application/component developed by an expert user, possible to tailor in order to adapt the application/component to the preferred context.
Model driven development	A model, presumably using abstraction/metaphors, to transform a problem into objects, structure and create a relationship between them. The result is a well-proven solution given it's context. The model is created and visualized, before the programming begins.
Programing by example	An example problem is inserted and explained by the user to the computer. Next time the computer is presented a similar task, it tries to use the previous example to solve the new problem.
Incremental programming	Extending the current applications functionality by gradually increasing the level of functionality when needed.

Table 8 presents the studies that identified different types of end user interfaces.

As Table 8 and Figure 4 present, application customization/ component tailoring / unwitted programmers along with block structured/ drag and drop/ component based/rule based have the highest amount of studies related to them. Furthermore, it is worth noticing that all of the papers in wizard driven/ step-by-step design pattern/ form-filling, with [S17] as an exception, are also

Table 8: Types of interfaces found in the relevant studies

Type of interface	**Study ID**
Natural language	S13
Block structured/ drag and drop/ component based/rule based	S17, S21, S32, S42, S50, S51, S61, S62, S77, S86
Wizard driven/ step-by-step design pattern/ form-filling	S17, S29, S21, S31
Application customization/ component tailoring / unwitted programmers	S5, S18, S42, S75, S21, S31, S32, S86, S19, S29
Model driven development	S27, S32, S31, S75
Programming by example	S66
Incremental programming	S42, S86

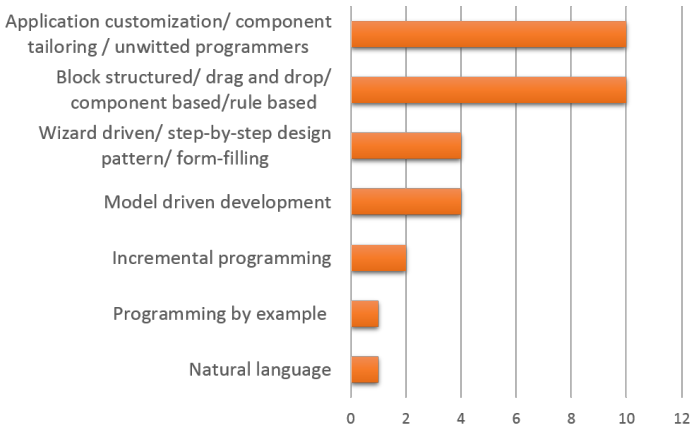

Figure 4: Amount of studies in the different categories

present in application customization/ component tailoring / unwitted programmers.

RQ3 questions the weaknesses and strengths of the different interfaces. Many studies explicit state the strengths of the technique(s) discussed in the study. This is not necessarily general for entire category used in this paper, but for one or more subcategories. Very few studies mention the weaknesses of their solution. For this paper to be as objective as possible, the weaknesses need to be stated explicit in the study, in order to make it a general disadvantage for a sub-

category. Table 9 presents a summary of the strengths and weaknesses.

Table 9: Overview of strengths and weaknesses in end user development interfaces

Type of interface	**Strengths**	**Weaknesses**
Natural language	Overcomes the barrier with the end user needing to learn and understand a new language in order to create the desired application [S13].	
Block based/ component based/ drag and drop/ rule based	- Mitigating the risk of user errors by having predefined components [S62]. - Very often visual, with color references and intuitive interface [S86]. - Tailoring flexibility [S86]. - No programming knowledge needed [S77]. - Eases work distribution of the actual development of components, maintainability, parallel development and extensibility [S62].	- Communication overhead, development timing, difficult integration when the development of the components takes place [S62].
Wizard driven/ step-by-step/ design pattern/ form-filling	- Supports easy interaction by elderly people [S21]. - Ensures validation of the data and that all the necessary data are inserted [S29]. - No direct manipulation of source code or underlying structure [S31], [S29]. - No programming expertise needed [S31]. - Visual, instead of textual [S21], [S29].	

Table 9 – continued from previous page

Type of interface	Strengths	Weaknesses
Application customization/ component tailoring/ unwitted programmers	- Mitigates the possibility for user errors [S18]. - Users are not conscious about customization or overcoming difficulties [S19]. - Use of tacit and cognitive skills [S29]. - No direct manipulation of source code [S31]. - Flexible customization for the end user [S5]. - Visual elements improve the overview of the application [S5, S21]. - No programming expertise needed [S31].	- Domain expertise is sometimes acquired [S5].
Model driven development	- Flexible and solid: It enables developers to analyze the structure and behavior of application to support future extensions [S32], [S75]. - Proven to solve a problem in a given context [S32]. - Facilitates the capturing and sharing of design expertise [S31], [S32], [S75].	- Needs guidance on issues that the users should carefully observe after they have put a specific pattern in place [S75]
Programming by Example	[S66] does not explicit mention any strengths using Programming by Example. However, nesting out references in [S66], [6] is referenced. The title of the book is "Programming by Example" and the book claims that the challenge is to train the computer. Once this barrier is overcome, it tends to work very well.	
Incremental programming	- Suspend important design decisions to the point in the design and development process when the decision really needs to be made [S42]. - Gradual design and problem solving process [S42]. - Tailoring flexibility [S86]. - The level of expertise is increasing proportionate with the level of functionality added [S86].	- Security issues [S86].

Discussion

The purpose of this study was to find relevant studies of tangible user interfaces in social computing and end user development, which will later be used as a literature base for further study. The main limitation of this mapping study is the decision of not doing a manual search and to only use two digital databases. This limits the amount of relevant studies found, and there are probably many relevant studies of both tangible user interfaces and end user development not found because of this limitation. However, we believe enough studies were found with only the two databases to defend our choice of limiting the search. By using a systematic mapping study, even with the limit of two databases, the goal of finding enough relevant articles was achieved.

Our research questions were not specifically addressed in several studies, thus making data extraction from these studies difficult. These studies were either deemed irrelevant or a conclusion was drawn even though it was not explicitly stated in the reviewed study.

Table 6 describes some of the possible challenges of tangible user interfaces. None of them is especially repeated throughout the study, however, there are several pitfalls to be aware of. The use of tangible interfaces have a huge potential with the mentioned advantages in Table 5, though "Realizing the context dependence of the potential of tangible interaction is not only important in terms of user experience, but also important to understand the products learning, usability, or collaboration benefits." [S85].

As Table 5 and Table 6 presents, tangible user interfaces have both cognitive advantages and cognitive challenges. How users grasp an idea, an object or any other physical or abstract concept, has shown to be connected to how concrete or "real life"-like the tangible interface was. The cognitive payload plays an essential role, and is one of the more important aspects of tangible user interfaces.

Regarding RQ3, a number of studies have selected an interface to their solution based on previous studies, target groups and/or environment. This has affected in a large number of the studies, not mentioning the specific weaknesses explicit. Therefore, answering RQ3 completely with the current set of studies is a challenge. Creating a new, more specific study, should be considered in order to fully answer the weaknesses of the question.

Many of the studies have very specific interfaces, with an overall goal to support end user interaction in the best possible way. To ease readability of this paper, the terminology were categorized. Similar types of interfaces were placed in the same category. Categorization was selected since systematic mapping focuses on width, rather than depth. Arguments can be stated that this is not optimal regarding to the overview of strengths and weaknesses. It is important to notice that one strength may not necessarily be a general strength for the entire categorization, but at least for one subcategory.

Acknowledgements

We appreciate the guidance from Researcher and Research Manager at SINTEF and Adjunct Associate Professor at Norwegian University of Science and Technology Babak A. Farshchian by being our supervisor during this systematic mapping study.

References

[1] Engineering Village, http://www.engineeringvillage.com/ (Accessed: 21 October 2013)

[2] Scopus, http://www.scopus.com/ (Accessed: 21 October 2013)

[3] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," Journal of Systems and Software, vol. 80, no. 4, pp. 57183, Apr. 2007.

[4] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, "Systematic Mapping Studies in Software Engineering". School of Engineering, Blekinge Institute of Technology.

[5] D. Budgen, M. Turner, P. Brereton, B. Kitchenham, "Using Mapping Studies in Software Engineering". Department of Computer Science, Durham University.

[6] H. Lieberman, "Your Wish is My Command: Programming by Example". Academic Press, 2001, pp. 21 - 43.

[7] C. Lee, Face-to-face Versus Computer-mediated Communication: Exploring Employees Preference of Effective Employee Communication Channel. INTERNATIONAL JOURNAL FOR THE ADVANCEMENT OF SCIENCE & ARTS, Vol 1, No 2, 2010, pp. 38 - 48.

Appendix A

Search string:

((”tangible user interface*” OR ”tangible interaction” OR ”physical user interface*” OR ”graspable user interface*” OR ”tangible ui” OR ”tabletop computing”) AND (”social computing” OR ”social informatics” OR ”social software” OR ”social app*” OR collaboration OR communication OR cooperation OR co-operation))

OR

((”end user development” OR ”end user customization” OR ”end user customisation” OR ”end user programming”) AND ((type* OR method* OR approach* OR category OR categories) AND (limitation* OR benefit* OR weakness* OR strength* OR advantage* OR disadvantage* OR drawback* OR characteristic* OR aspect* OR tendency OR tendencies OR attribute*)))

Appendix B

The selected studies:

[S1] A. Alves, R. Lopes, P. Matos, L. Velho, and D. Silva, "Reactoon: Storytelling in a Tangible Environment," in Third IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL 2010), 12-16 April 2010, 2010, pp. 1615.

[S2] A. N. Antle, "Exploring how children use their hands to think: An embodied interactional analysis," Behaviour and Information Technology, vol. 32, no. 9, pp. 938954, 2013.

[S3] P. G. Austrem, "Using EUREQA for End-user UML Model Development through Design Patterns," Journal of Software, vol. 6, no. 4, pp. 690704, Apr. 2011.

[S4] Y. Bannai, H. Tamaki, Y. Suzuki, H. Shigeno, and K. Okada, "A tangible user interface for remote collaboration system using mixed reality," in 16th International Conference on Artificial Reality and Telexistence, ICAT 2006, November 29, 2006 - December 1, 2006, 2006, vol. 4282 LNCS, pp. 143154.

[S5] M. Baron and P. Girard, "Bringing robustness to end-user programming," in Proceedings: IEEE Symposia on Human-Centric Computing Languages and Environments, September 5, 2001 - September 7, 2001, 2001, pp. 142149.

[S6] D. Belcher and B. Johnson, "MxR A physical model-based mixed reality interface for design collaboration, simulation,visualization and form generation," in 28th Annual Conference of the Association for Computer Aided Design in Architecture: Silicon + Skin: Biological Processes and Computation, ACADIA 08, October 16, 2008 - October 19, 2008, 2008, pp. 464471.

[S9] W. Bruns, "Grasping, communicating, understanding: Connecting reality and virtuality," AI and Society, vol. 10, no. 1, pp. 614, 1996.

[S12] K. Camarata, E. Yi-Luen Do, B. R. Johnson, and M. D. Gross, "Navigational blocks navigating information space with tangible media," in 2002 International Conference on intelligent User Interfaces (IUI 02), January 13, 2002 - January 16, 2002, 2002, pp. 3138.

[S13] J. Cao, "An idea garden for end-user programmers," in Conference on Human Factors in Computing Systems - Proceedings, 2012, pp. 915918.

[S14] S. Cecilia, G. Andrea, D. G. Armando, B. Sandra, M. Javier, and C. Eva, "Games as educational strategy: A case of tangible interaction for users of Alternative and Augmentative Communication," in 2013 International Conference on Collaboration Technologies and Systems, CTS 2013, May 20, 2013 - May 24, 2013, 2013, pp. 377381.

[S15] T. Chen and A. Kratky, "Touching buildings - a tangible interface for architecture visualization," in Universal Access in Human-Computer Interac-

tion. Design Methods, Tools, and Interaction Techniques for eInclusion. 7th International Conference, UAHCI 2013, 21-26 July 2013, 2013, vol. pt. I, pp. 31322.

[S16] A. D. Cheok, Wang Weihua, Xubo Yang, S. Prince, Fong Siew Wan, M. Billinghurst, and H. Kato, "Interactive theatre experience in embodied + wearable mixed reality space," in Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality, 30 Sept.-1 Oct. 2002, 2002, pp. 59317.

[S17] S. Chimalakonda and K. V. Nori, "GURU: An Experimental Interactive Environment for Teachers/Learners," in 2013 IEEE 13th International Conference on Advanced Learning Technologies (ICALT), 15-18 July 2013, 2013, pp. 2489.

[S18] M. F. Costabile, D. Fogli, R. Lanzilotti, P. Mussio, and A. Piccinno, "Supporting work practice through end-user development environments," Journal of Organizational and End User Computing, vol. 18, no. 4, pp. 4365, 2006.

[S19] M. F. Costabile, P. Mussio, L. P. Provenza, and A. Piccinno, "End users as unwitting software developers," in 30th International Conference on Software Engineering, ICSE 2008 - 4th International Workshop on End-user Software Engineering, WEUSE IV, May 12, 2008 - May 12, 2008, 2008, pp. 610.

[S20] A. Crevoisier and C. Picard-Limpens, "INSIDE: Intuitive Sonic Interaction Design for Education and Entertainment," in 4th International ICST Conference on Intelligent Technologies for Interactive Entertainment, INTETAIN 2011, May 25, 2011 - May 27, 2011, 2012, vol. 78 LNICST, pp. 236239.

[S21] J. Criel, M. Geerts, L. Claeys, and F. Kawsar, "Empowering elderly end-users for ambient programming: The tangible way," in 6th International Conference on Grid and Pervasive Computing, GPC 2011, May 11, 2011 - May 13, 2011, 2011, vol. 6646 LNCS, pp. 94104.

[S22] N. Dalton, G. MacKay, and S. Holland, "Kolab: Appropriation improvisation in mobile tangible collaborative interaction," in Designing Interactive Systems Conference, DIS 12, June 11, 2012 - June 15, 2012, 2012, pp. 2124.

[S23] B. De Silva and A. Ginige, "Solving design issues in web meta-model approach to support End-User Development," in 2nd International Conference on Software and Data Technologies, ICSOFT 2007, July 22, 2007 - July 25, 2007, 2007, vol. PL, pp. 298304.

[S24] D. Edge, A. Blackwell, and L. Dubuc, "The physical world as an abstract interface," in Contemporary Ergonomics 2006, 2006, pp. 224228.

[S25] L. Ehrenstrasser and W. Spreicer, "kommTUi - A design process for a tangible communication technology with seniors," in 1st International Conference on Human Factors in Computing and Informatics, SouthCHI 2013, July 1, 2013 - July 3, 2013, 2013, vol. 7946 LNCS, pp. 625632.

[S26] A. Esteves and I. Oakley, "Mementos: A tangible interface supporting travel," in 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries, NordiCHI 2010, October 16, 2010 - October 20, 2010, 2010, pp. 643646.

[S27] R. Farmer and P. Gruba, "Towards model-driven end-user development in CALL," Computer Assisted Language Learning, vol. 19, no. 23, pp. 149191, 2006.

[S28] M. Fjeld, M. Morf, and H. Krueger, "Activity theory and the practice of design: evaluation of a collaborative tangible user interface," International Journal of Human Resources Development and Management, vol. 4, no. 1, pp. 94116, 2004.

[S29] D. Fogli, "End-user development for E-government website content creation," in 2nd International Symposium on End-User Development, IS-EUD 2009, March 2, 2009 - March 4, 2009, 2009, vol. 5435 LNCS, pp. 126145.

[S30] F. G. Furtmller and S. Oppl, "A tuple-space based middleware for collaborative tangible user interfaces," in Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, 2007, pp. 418423.

[S31] F. Garzotto and M. Bordogna, "Paper-based multimedia interaction and disabled children: from experience to learning-for-all," International Journal of Arts and Technology, vol. 5, no. 24, pp. 12650, 2012.

[S32] F. Garzotto and R. Gonella, "An open-ended tangible environment for disabled childrens learning," in 10th International Conference on Interaction Design and Children, IDC 2011, June 20, 2011 - June 23, 2011, 2011, pp. 5261.

[S33] N. Gu, M. J. Kim, and M. L. Maher, "Technological advancements in synchronous collaboration: The effect of 3D virtual worlds and tangible user interfaces on architectural design," Automation in Construction, vol. 20, no. 3, pp. 270278, 2011.

[S34] F. Guo, C. Zhang, and L. Cui, "Sketching interfaces for remote collaboration," in Proceedings of the 2007 11th International Conference on Computer Supported Cooperative Work in Design, CSCWD, 2007, pp. 6368.

[S35] A. Hang, G. Broll, and A. Wiethoff, "Visual design of physical user interfaces for NFC-based mobile interaction," in DIS 2010 - Proceedings of the 8th ACM Conference on Designing Interactive Systems, 2010, pp. 292301.

[S36] B. Hengeveld, C. Hummels, K. Overbeeke, R. Voort, H. Van Balkom, and J. De Moor, "Tangibles for toddlers learning language," in 3rd International Conference on Tangible and Embedded Interaction, TEI09, February 16, 2009 - February 18, 2009, 2009, pp. 161168.

[S37] J. Herstad and H. Holone, "What we talk about when we talk about co-creative tangibles," in 12th Participatory Design Conference, PDC 2012, August 12, 2012 - August 16, 2012, 2012, vol. 2, pp. 109112.

[S38] C. Holzmann and A. Ferscha, "Tangible interaction in collaborative environments," in 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2007, June 18, 2007 - June 20, 2007, 2007, pp. 409411.

[S39] K. Hook, "Knowing, communication and experiencing through body and emotion," IEEE Transactions on Learning Technologies, vol. 1, no. 4, pp. 248259, 2008.

[S40] E. Hornecker, "A design theme for tangible interaction: Embodied facilitation," in 9th European Conference on Computer-Supported Cooperative Work, ECSCW 2005, September 18, 2005 - September 22, 2005, 2005, pp. 2343.

[S41] E. Hornecker and J. Buur, "Getting a grip on tangible interaction: A framework on physical space and social interaction," in CHI 2006: Conference on Human Factors in Computing Systems, April 22, 2006 - April 27, 2006, 2006, vol. 1, pp. 437446.

[S42] A. Ioannidou, A. Repenning, and D. C. Webb, "AgentCubes: Incremental 3D end-user development," Journal of Visual Languages and Computing, vol. 20, no. 4, pp. 236251, 2009.

[S43] H. Ishii, "Tangible bits: Beyond pixels," in TEI08 - Second International Conference on Tangible and Embedded Interaction - Conference Proceedings, 2008, pp. xvxxv.

[S44] H. Ishii, C. Ratti, B. Piper, Y. Wang, A. Biderman, and E. Ben-Joseph, "Bringing clay and sand into digital design - continous tangible user interfaces," BT Technology Journal, vol. 22, no. 4, pp. 287299, 2004.

[S46] J. H. Israel, J. Hurtienne, A. E. Pohlmeyer, C. Mohs, M. C. Kindsmuller, and A. Naumann, "On intuitive use, physicality and tangible user interfaces," International Journal of Arts and Technology, vol. 2, no. 4, pp. 34866, 2009.

[S47] S. Izadi, A. Agarwal, A. Criminisi, J. Winn, A. Blake, and A. Fitzgibbon, "C-Slate: A multi-touch and object recognition system for remote collaboration using horizontal surfaces," in 2nd Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems, Tabletop 2007, October 10, 2007 - October 12, 2007, 2007, pp. 310.

[S48] A. Karime, M. A. Hossain, A. S. M. M. Rahman, W. Gueaieb, J. M. Aljaam, and A. E. Saddik, "RFID-based interactive multimedia system for the children," Multimedia Tools and Applications, vol. 59, no. 3, pp. 749774, 2012.

[S49] M. J. Kim and M. L. Maher, "The impact of tangible user interfaces on

spatial cognition during collaborative design," Design Studies, vol. 29, no. 3, pp. 222253, 2008.

[S50] J. Lee, L. Garduo, E. Walker, and W. Burleson, "A tangible programming tool for creation of context-aware applications," in UbiComp 2013 - Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2013, pp. 391400.

[S51] G. Leitner, A. J. Fercher, and C. Lassen, "End users programming smart homes - A case study on scenario programming," in 3rd International Workshop on Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data, HCI-KDD 2013, Held at SouthCHI 2013, July 1, 2013 - July 3, 2013, 2013, vol. 7947 LNCS, pp. 217236.

[S52] M. A. Linton, J. M. Vlissides, and P. R. Calder, "Composing user interfaces with InterViews," Computer, vol. 22, no. 2, pp. 822, Feb. 1989.

[S53] Magnus Bang and T. Timpka, "Ubiquitous computing to support co-located clinical teams: Using the semiotics of physical objects in system design," International Journal of Medical Informatics, vol. 76, pp. 5864, Jun. 2007.

[S54] M. L. Maher and M. J. Kim, "Do tangible user interfaces impact spatial cognition in collaborative design?," in 2nd International Conference on Cooperative Design, Visualization, and Engineering, CDVE 2005, September 18, 2005 - September 21, 2005, 2005, vol. 3675 LNCS, pp. 3041.

[S55] M. L. Maher and M. J. Kim, "Studying designers using a tabletop system for 3D design with a focus on the impact on spatial cognition," in First IEEE International Workshop on Horizontal Interactive Human-Computer Systems, TABLETOP06, January 5, 2006 - January 7, 2006, 2006, vol. 2006, pp. 105112.

[S56] I. Mantegh and N. S. Darbandi, "Knowledge-based task planning using natural language processing for robotic manufacturing," in ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC/CIE2010, August 15, 2010 - August 18, 2010, 2010, vol. 3, pp. 13091316.

[S57] N. Marquardt and S. Greenberg, "Distributed physical interfaces with shared phidgets," in 1st International Conference on Tangible and Embedded Interaction, February 15, 2007 - February 17, 2007, 2007, pp. 1320.

[S58] T. McGill and C. Klisc, "End-user perceptions of the benefits and risks of end-user web development," Journal of Organizational and End User Computing, vol. 18, no. 4, pp. 2242, 2006.

[S59] D. Merrill, J. Kalanithi, and P. Maes, "Siftables: Towards sensor network user interfaces," in TEI07: First International Conference on Tangible and Embedded Interaction, 2007, pp. 7578.

[S61] S. N. H. Mohamad, A. Patel, R. Latih, Q. Qassim, L. Na, and Y. Tew, "Block-based programming approach: Challenges and benefits," in 2011 International Conference on Electrical Engineering and Informatics, ICEEI 2011, July 17, 2011 - July 19, 2011, 2011, p. IEEE Indonesia Section; IEEE Eng. Med. Biol. Soc. Indonesia Chapter; IEEE Circuit and Systems Society Indonesia Chapter; IEEE Electron Devices, Educ., Signal Process.,; Power Energy Syst. Soc. Indonesia Jt. Chapter.

[S62] S. N. H. Mohamad, A. Patel, Y. Tew, R. Latih, and Q. Qassim, "Principles and dynamics of block-based programming approach," in 2011 IEEE Symposium on Computers and Informatics, ISCI 2011, March 20, 2011 - March 22, 2011, 2011, pp. 340345.

[S63] T.-Y. Mou, T.-S. Jeng, and C.-H. Ho, "Relationship enhancer: Interactive recipe in kitchen island," in 13th International Conference on Human-Computer Interaction, HCI International 2009, July 19, 2009 - July 24, 2009, 2009, vol. 5612 LNCS, pp. 641650.

[S64] B. Nadeau and A. Williams, "Tactful interaction: Exploring interactive social touch through a collaborative tangible installation," in 3rd International Conference on Tangible and Embedded Interaction, TEI09, February 16, 2009 - February 18, 2009, 2009, pp. 147152.

[S66] A. C. Pena Rios, J. S. Y. Chin, and V. L. Callaghan, "A web based approach to virtual appliance creation, programming and management," in 2010 6th International Conference on Intelligent Environments, IE 2010, July 19, 2010 - July 21, 2010, 2010, pp. 174177.

[S67] K. Peternel, M. Pogacnik, J. Beter, L. Zebec, M. Pustiek, and A. Kos, "Touch to communicate using NGN open interfaces," in 2011 9th Annual Communication Networks and Services Research Conference, CNSR 2011, May 2, 2011 - May 5, 2011, 2011, pp. 130136.

[S69] M. S. Quintanilha, "BuddyWall: A tangible user interface for wireless remote communication," in 28th Annual CHI Conference on Human Factors in Computing Systems, April 5, 2008 - April 10, 2008, 2008, pp. 37113716.

[S70] H. T. Regenbrecht, M. T. Wagner, and G. Baratoff, "Magicmeeting: A collaborative tangible augmented reality system," Virtual Reality, vol. 6, no. 3, pp. 151166, 2002.

[S74] U. Sargaana, H. S. Farahani, J. W. Lee, J. Ryu, and W. Woo, "Collaborative billiARds: Towards the ultimate gaming experience," in 4th International Conference on Entertainment Computing - ICEC 2005, September 19, 2005 - September 21, 2005, 2005, vol. 3711 LNCS, pp. 357367.

[S75] T. Schummer and J. M. Haake, "Shaping collaborative work with protopatterns," in 2nd International Symposium on End-User Development, IS-EUD 2009, March 2, 2009 - March 4, 2009, 2009, vol. 5435 LNCS, pp. 166185.

[S77] M. Spahn and V. Wulf, "End-user development for individualized information management: Analysis of problem domains and solution approaches," in 11th International Conference on Enterprise Information Systems, ICEIS 2009, May 6, 2009 - May 10, 2009, 2009, vol. 24 LNBIP, pp. 843857.

[S78] A. I. Starcic and M. Zajc, "An interactive tangible user interface application for learning addition concepts," British Journal of Educational Technology, vol. 42, no. 6, pp. 1315, Nov. 2011.

[S79] S. Subramanian, D. Pinelle, J. Korst, and V. Buil, "Tabletop collaboration through tangible interactions," in Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, 2007, pp. 412417.

[S80] A.-L. Syrjanen and K. Kuutti, "From technology to domain: The context of work for end-user development," in 6th Annual Conference on 2011 iConference: Inspiration, Integrity, and Intrepidity, iConference 2011, February 8, 2011 - February 11, 2011, 2011, pp. 244251.

[S82] M. Waldner, J. Hauber, J. Zauner, M. Haller, and M. Billinghurst, "Tangible tiles: Design and evaluation of a tangible user interface in a collaborative tabletop setup," in 18th Australia Conference on Computer-Human Interaction, OZCHI 06, November 20, 2006 - November 24, 2006, 2006, vol. 206, pp. 151158.

[S83] W. Wang, X. Wang, and R. Wang, "A spatial faithful cooperative system based on mixed presence groupware model," in 6th International Conference on Cooperative Design, Visualization, and Engineering, CDVE 2009, September 20, 2009 - September 23, 2009, 2009, vol. 5738 LNCS, pp. 269275.

[S84] A. Wiethoff and G. Broll, "SoloFind: Chains of interactions with a mobile retail experience system," in 29th Annual CHI Conference on Human Factors in Computing Systems, CHI 2011, May 7, 2011 - May 12, 2011, 2011, pp. 13031308.

[S85] B. Zaman, V. Vanden Abeele, P. Markopoulos, and P. Marshall, "Editorial: The evolving field of tangible interaction for children: The challenge of empirical validation," Personal and Ubiquitous Computing, vol. 16, no. 4, pp. 367378, 2012.

[S86] L. Zhu, I. Vaghi, and B. R. Barricelli, "A meta-reflective wiki for collaborative design," in 7th Annual International Symposium on Wikis and Open Collaboration, WikiSym 2011, October 3, 2011 - October 5, 2011, 2011, pp. 5362.

Appendix B

Hardware Selection

B.1 Arduino versus Raspberry Pi versus BeagleBone Black

The problem definition clearly specifies the requirement to use the toolkit to bridge the gap between a hardware prototype and a software system. The hardware platform, however, is not specified. There are numerous of hardware prototyping platforms out there, each with their strengths and weaknesses. Common for all the platforms are the low price and the encouragement to get people to learn basic programming and computer knowledge. Below is a summary of three popular platforms the team considered when selecting prototyping platform.

Raspberry Pi is a small computer originally aiming to encourage kids to learn basic programming [62]. The device is a circuit board with CPU, RAM, USB, LAN, HDMI, audio and SD card slot, all the basic elements of a computer. It requires installation of an operation system, and will function as a mini-computer [66]. The operating system is installed on the SD card. Depending on the selected programming language, there are many libraries possible to install. Because of its rich set of features, the many projects created are typically software based with media centers, multimedia and other software tricks (see Figure B.1).

"Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software" [2]. The board itself is contains a USB connection, power jack, a set of digital input and output pins, a set of analog inputs and a ceramic resonator. It has a very flexible solution with a high number of kits, shields and other devices that can be connected to the Arduino by using the pins. The Arduino is a micro controller. Programming an Arduino is done by writing Arduino code. Arduino code is a variation to the programming language C. Arduino offers an easy to use IDE preloaded with examples lowering the entry bar (see Figure B.2).

BeagleBone Black is, like Raspberry Pi, a tiny computer and has many of the same features. Connectivity interfaces with HDMI, Ethernet, USB and 2 x 46 pin headers among with a powerful set of processing power [63]. These pins in one of

Figure B.1: Front of Raspberry Pi [67]

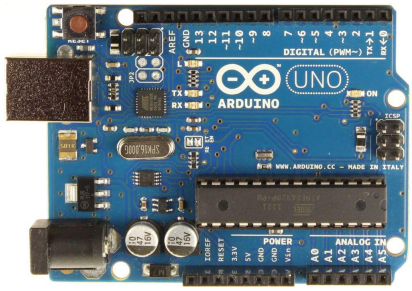

Figure B.2: Front of Arduino Uno R3 [68]

the biggest advantages with the device, as it gives great flexibility when it comes to connecting external sensors or other hardware to your device. It was originally designed to support teaching of open source software used in combination of open source hardware. It is more powerful than Raspberry Pi and has official support for both Android and Ubuntu (Raspberry Pi does not have this). In contrast to Raspberry Pi the BeagleBone Black ships with a Linux operating system already installed, so getting started with the device is very easy [69] (see figure B.3).

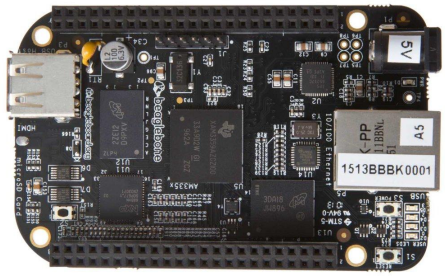

Figure B.3: Front of BeagleBone Black [63]

The team chose the Arduino as the prototyping platform. This is mainly be-

cause of its features as a micro controller. The team had no need for a small sized computer. The Arduino gave us simple and flexible hardware with a low level of complexity. This allowed the team to quickly wire prototypes and connect different kinds of component and sensor to the input and output on the board. The board used is called Arduino Uno, model R3.

B.2 Bluetooth v2.1 Versus Bluetooth v4.0

The selected Bluetooth module, Bluetooth Mate Silver, uses Bluetooth version 2.1+EDR support [70]. Bluetooth 2.1 has a theoretical data rate of 3 Mbit/s and a maximum application throughput larger than 80 Kbit/s. The Mate Silver has a Bluetooth class 2, this means that the device has a rage up to 10 meters [71]. The continuously improvement of technology encourages rapidly releases of new versions. Bluetooth v4.0, marketed as Bluetooth Smart, is a new standard. Beside from a drastically increased application throughput of 24 Mbit/s, the main focus is on the low power consumption. This makes Bluetooth 4.0 very suitable for use in sensors and applications utilizing these sensors. Typical industries are healthcare, security, technology and fitness. A product using Bluetooth Smart can run for months or years on a small coin battery [72] and this makes it ideal for all the scenarios the team has pictured. The reason the team selected the Silver Mate Bluetooth dongle with Bluetooth version 2.0 is because of an advice from a PhD candidate at NTNU. He stated that the few modules out there supporting Bluetooth v4.0, is unreliable and not worth taking the risk. However, if the toolkit were to be published this would be a feature with high importance.

B.3 Electric Imp Versus RN-XV Wifly

There are several Arduino components available offering the functionality to connect the Arduino board to a wireless network. At NTNU the Electrical Imp has previously been used successfully. Due to its availability and previously successful usage, the team decided to start with this device and determine it had the desired functionality. The Electrical Imp consist of the actual Imp, a SD-card shaped device that is inserted into a device called Electric Imp April, that allows the device to be connected to the Arduino board via pins [73]. Using your smartphone, you make the Electrical Imp go online by using a mobile application, which connects it to the Imp Cloud. The Imp Cloud has a web based IDE, allowing the user to configure the Imp in a language called Squirrel [74].

The architecture Electrical Imp provided turned out quickly not to provide the optimal functionality for the team. However, the RN-XV WiFly adapter provided functionality to control the WiFly adapter from the Android application, which means generation of needed Arduino code based on the input from the user [48,59]. This solution makes it possible not only to have two or more Arduino boards communicate, like the Imp, but also having an Arduino communicating with local servers and servers on the Internet.

Appendix C

Guides

C.1 Installation Guide

C.1.1 Introduction

The following sections are a guide to install and setup what is needed in order to install and use RAPT as intended. You will either need the APK-file or the source code, and depending on what you have available you can skip either Section C.1.3 or Section C.1.4. However, in the current version of RAPT, to be able to change the compilation server IP within RAPT, you need the source code (or extract it from the APK).

Note that the compilation server, located in Appendix F (Python script and batch script), is only tested on Microsoft Windows 7.

C.1.2 Compilation Server Setup

To be able to compile sketches, a compilation server must be set up. The server consists of two parts, a Python script F.1 that servers as a server, and a batch script F.2 that the server executes in order to do the compilation from cpp-code to hexadecimals.

There are two necessities for the Python script to function:

- Change the IP address to the local IP address at line 62.
- A folder named "sketch" must be located in the same directory as the script.

Furthermore, log path at line 10 can be changed to wherever desired.
The batch script, however, needs a few more tweaks:

- The avr library [75] needs to be installed and added to the environment path. You can download the Arduino IDE [56] which comes with avr library.

- At line 5, make sure the sketch folder (referring to the Python script list), include the folders ComputerSerial, SoftwareSerial and WiFlyHQ, each containing their corresponding two files, with the endings: .cpp.o and cpp.d. These files are attached, or it is possible to generate them importing the libraries and pressing "compile" or "verify" using the Arduino IDE.
- The batch script must be named compiler.bat and be located in the same folder as the Python script (if you want to change the name and path, make sure to change it respectively at line 35 in the Python script).

Remember to make sure necessary ports are open if the server is behind a router or similar.

C.1.3 Installation of APK

If you seek to install the APK without any changes there are several ways to do so:

- Download the APK on the phone: Click on the downloaded APK to install.
- Download the APK on the computer: Transfer the APK via USB cable to the storage on the phone (turn on USB storage). Locate the APK, then click the APK to install.
- From command line, type: adb install <APK path>, while connected to the phone via an USB cable. You will need the ADB from Android [76] installed.

C.1.4 Installation from Source Code

The project is set up with Maven [77] to help organise dependencies and building the application.

Note: To change the compilation server IP address and port, the Settings class in Figure 5.7 can be changed.

The following is two simple commands to build and deploy the project to a connected phone (through USB cable). You will need the ADB from Android [76] installed. Depending on the version of Android installed, remember to set your phone in USB debugging, Connected as media device and/or check the Media device (MTP) option.

```
1 mvn clean install & mvn deploy
```

C.2 Wire Guide

C.2.1 Introduction

This chapter will describe how to wire a board with a corresponding XML file (the XML file defines the wired Arduino board within RAPT). The board that is presented is called Alpha, and is one of the three board types currently defined in RAPT. A picture taken of Alpha, the Arduino prototype, is presented in Figure C.1.

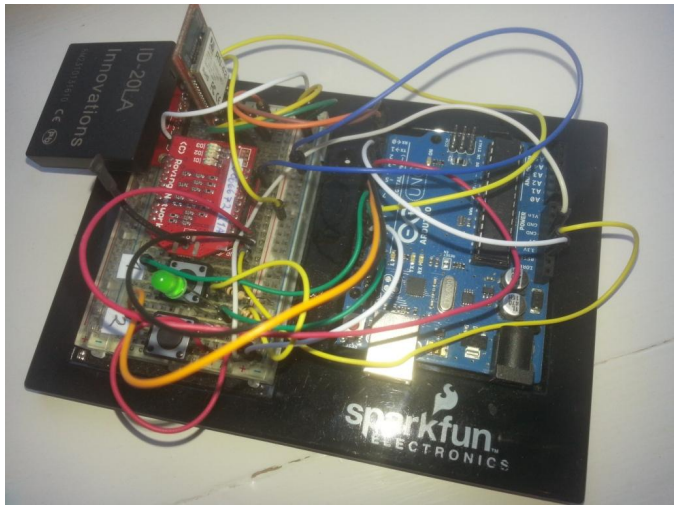

Figure C.1: Alpha

Alpha consists of:

- 1 x Arduino Uno Board
- 1 x Arduino Breadboard
- 1 x Led (green)
- 2 x Button
- 1 x RFID Reader ID-20LA (125 kHz)
- 1 x RFID USB Reader
- 1 x RN-XV WiFly Module - Wire Antenna
- 1 x Breakout Board for XBee Module
- 1 x Bluetooth Mate Silver
- 2 x 10k ohm resistor

- 1 x 220 ohm resistor
- Additionally, in this setup there are used 19 wires, however, changes can be made to reduce the number.

C.2.2 Wiring

To ease readability, this section shows the wiring of Alpha. It is important to remember to connect a wire from pin 4 to reset as shown. This connection enables code to be transferred via Bluetooth without actually pressing the reset button on the Arduino board.

An additionally note is to remember that RX and TX on the modules, here the Bluetooth Mate Silver must be switched when connecting to the Arduino board's TX and RX. Thus, RX (Bluetooth Mate Silver) to TX (Arduino) and TX (Bluetooth Mate Silver) to RX (Arduino).

The red wires are connected to power (vcc), the grey wires are connected to ground (gnd) and the blue wires handles signals.

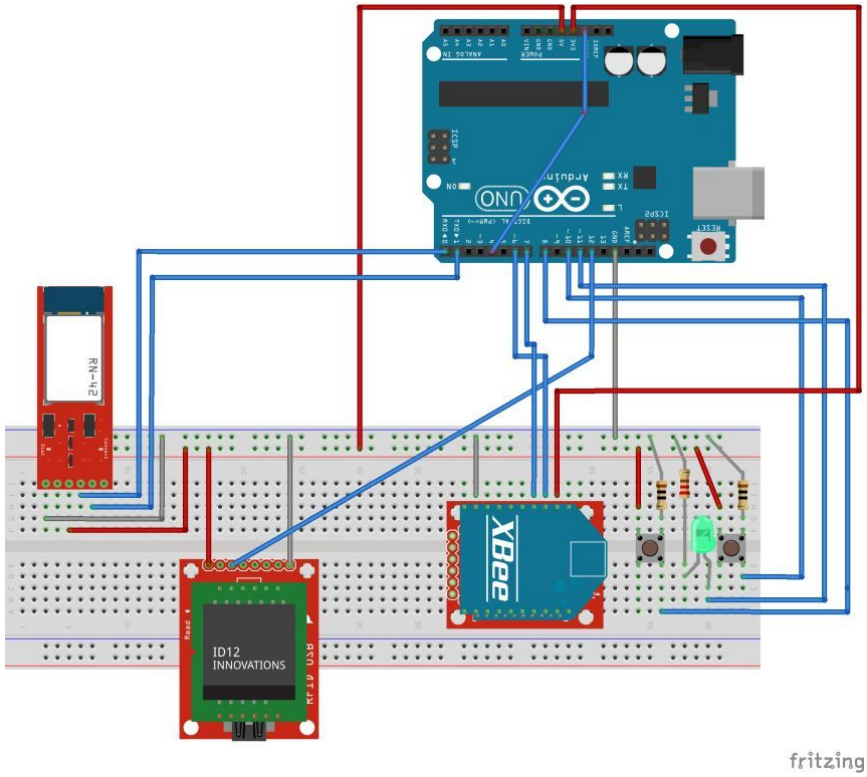

Figure C.2: Alpha Wiring

In order to make your board eligible for RAPT, you need to setup the board

with a running sketch that imports ComputerSerial. This sketch cannot be transferred via Bluetooth, and must be transferred via USB. You can use the sketch from Appendix G.2, and replace the SerialEvent function with the SerialEvent in Appendix G.3. Additionally, you need to disconnect the RX and TX pins while transferring.

C.2.3 Corresponding XML

```
<?xml version="1.0" encoding="UTF-8"?>
<board>
    <name>Alfa</name>
    <description>Has wireless network, 2 buttons, 1 RFID
        reader and 1 yellow led. Perfect for RFID
        apps.</description>
    <devices>
        <device>
            <type>LED</type>
            <text>Green</text>
            <pins>
                <pinmode>OUTPUT</pinmode>
                <pinnumber>11</pinnumber>
            </pins>
        </device>
        <device>
            <type>Button</type>
            <text>No. 1</text>
            <pins>
                <pinmode>INPUT</pinmode>
                <pinnumber>10</pinnumber>
            </pins>
        </device>
        <device>
            <type>Button</type>
            <text>No. 2</text>
            <pins>
                <pinmode>INPUT</pinmode>
                <pinnumber>8</pinnumber>
            </pins>
        </device>
        <device>
            <type>RfidReader</type>
            <pins>
                <pinmode>NONE</pinmode>
                <pinnumber>12</pinnumber>
            </pins>
```

```
        </device>
        <device>
            <type>Wifi</type>
            <text>Message</text>
            <pins>
                <!--rx pin must be first-->
                <pinmode>NONE</pinmode>
                <pinnumber>6</pinnumber>
            </pins>
            <pins>
                <pinmode>NONE</pinmode>
                <pinnumber>7</pinnumber>
            </pins>
        </device>
    </devices>
    <RFIDTags>
        <Groups>
            <Group>
                <RFIDName>Student card</RFIDName>
                <Tags>
                    <Tag>210B38A54C71</Tag>
                    <Tag>21104268A00A</Tag>
                </Tags>
            </Group>
        </Groups>
    </RFIDTags>
</board>
```

Appendix D

Versions of RAPT

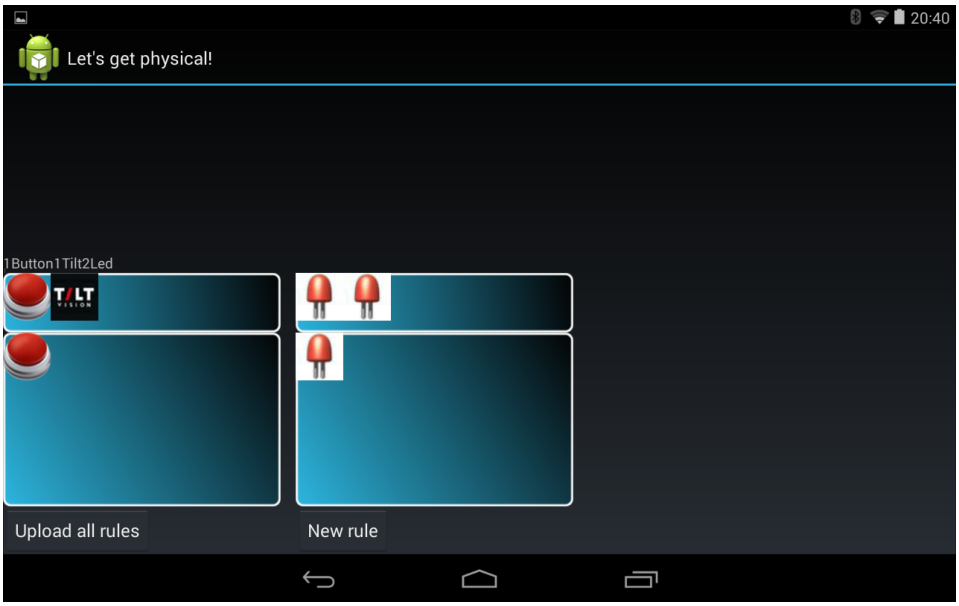

Figure D.1: User interface, draft 1 of RAPT. Screenshot from a tablet

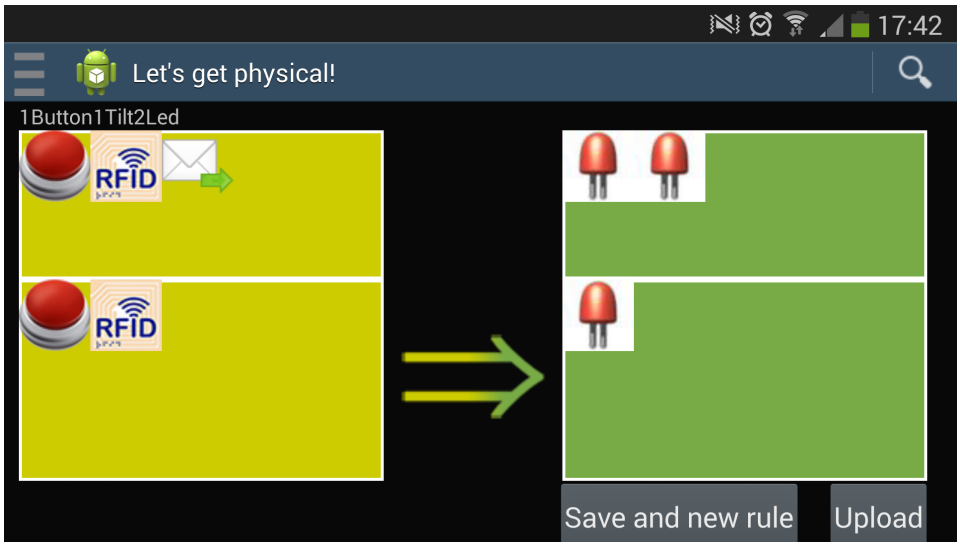

Figure D.2: User interface, draft 2 of RAPT

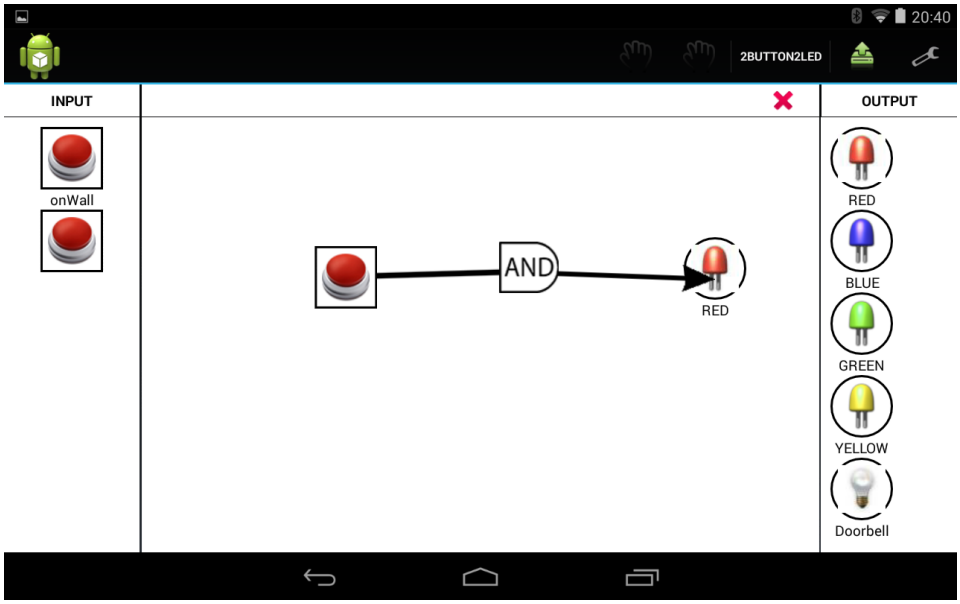

Figure D.3: User interface, draft 3 of RAPT, Screenshot from a tablet

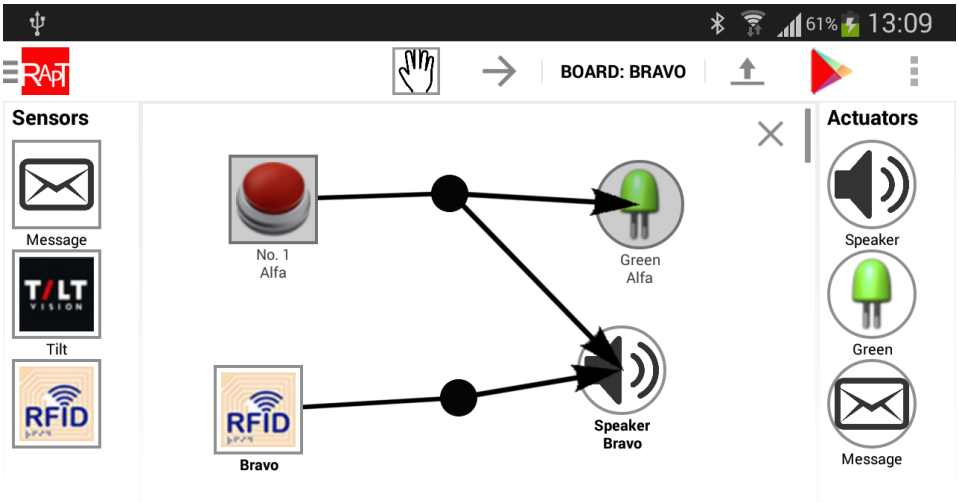

Figure D.4: User interface, final version of RAPT

Appendix E

Usability Test Attachments

E.1 Tasks

E.1.1 English

1. Complete the tutorial
   (a) Navigate back to the main menu by pressing the back button top left or the back button on the phone.
   (b) You can always go back to the tutorial if you need.
2. Create a new app
   (a) Create a new app, where a button is pressed will make a led blink.
   (b) Install the app on the board.
   (c) If everything went ok: Press the button on the board you install the app on to check if it works.
3. Create a new app
   (a) Create a new app for the board "Bravo".
   (b) If Haakon's access card is read should:
      i. The green led be lit for 2 seconds.
      ii. The speaker should play a optional melody.
   (c) Drag the speaker so it is below the RFID reader.
   (d) Install the app on the board.
   (e) If everything went ok: Read Haakon's SINTEF card to check if the led is lit and the melody plays.
4. Create a new app
   (a) Create a app, where

   i. Button on one board should
   ii. Play a melody on another board.
   (b) Install the app.
   (c) If everything went ok: Press the button to check if the melody plays.
   (d) Publish the app to RAPT App Store.
5. Navigate to app store
   (a) Choose a board.
   (b) Choose a app.
   (c) Modify the app so it does what you want.
   (d) Install the app.
   (e) Check if everything is OK by testing the app you just created/modified.
6. Create a access control app on one board. Start by creating a new app.
   (a) Goal:
      i. Anders' student card does not have access. If this card is read a red led should be lit for 2 seconds.
      ii. Daniel's student card does have access. If this card is read a green led should be lit in for 2 seconds.
   (b) Install your app.
   (c) If everything went ok: Read the RFID cards and see if correct accesses are given by checking the green/red leds.

E.1.2 Norwegian

1. Gjennomfør tutorial
   (a) Gå tilbake til hovedmenyen ved å trykke på tilbakeknappen øverst til venstre eller tilbakeknappen på telefonen.
   (b) Du kan alltid gå tilbake til tutorial hvis du ønsker det.
2. Lag en ny app
   (a) Lag en app, der en knapp som trykkes vil få et led til å blinke.
   (b) Installer appen på boardet.
   (c) Hvis alt gikk ok: Trykk på knappen på det boardet du installerte appen på for å sjekke at det fungerer.
3. Lag en ny app
   (a) Lag en ny app til boardet "Bravo".
   (b) Hvis adgangskortet til Haakon blir lest skal:

   i. Den grønne led'en lyse i 2 sekunder.
   ii. Høyttaleren skal spille av en valgfri melodi.
   (c) Dra høytaleren slik at den står under RFID leser.
   (d) Installer appen på boardet.
   (e) Hvis alt gikk ok: Les Haakon sitt SINTEF kort for å sjekke at led lyser og melodien spilles.

4. Lag en ny app
   (a) Lag en app, der
      i. Knapp på et board skal.
      ii. Spille av en melodi på et annet board.
   (b) Installer appen.
   (c) Hvis alt gikk ok: Trykk på knappen for å sjekke at melodien blir avspilt.
   (d) Publiser appen til RAPT App Store.

5. Gå til app store
   (a) Velg et board.
   (b) Velg en app.
   (c) Modifiser appen til å oppfylle eget ønske
   (d) Installer appen på boardet
   (e) Sjekk at alt er OK ved å teste appen du akkurat har laget/endret

6. Lag en adgangskontroll app på ett board. Dette starter du å gjøre ved å lage en ny app.
   (a) Mål:
      i. Anders sitt studentkort ikke har tilgang. Hvis dette leses skal rødt lys lyse i 2 sekunder.
      ii. Daniel sitt studentkort har tilgang. Hvis dette leses skal grønt lys lyse i 2 sekunder.
   (b) Installer din app.
   (c) Hvis alt gikk ok: Les RFID kortene og se om riktige tilganger er satt ved at grønt/rødt lys lyser.

E.2 System Usability Scale Questionnaire

E.2.1 English

Some questions about the application you used

Please check only one square for each question.

		Strongly disagree				Strongly agree
1.	I thought the system was easy to use.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
2.	I think the system was unnecessary complicated.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
3.	I found the various functions in this system were well integrated.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
4.	I think that I would need the support of a technical person to be able to use this system.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
5.	I thought there was too much inconsistency in this system.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
6.	I would imagine that most people would learn to use this system very quickly.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5
7.	I needed to learn a lot of things before I could get going with this system.	☐ 1	☐ 2	☐ 3	☐ 4	☐ 5

E.2.2 Norwegian

Noen spørsmål om applikasjonen du har brukt

Vennligst sett kryss i kun en rute pr. spørsmål.

		Sterkt uenig				Sterkt enig
1.	Jeg synes systemet var lett å bruke.	1	2	3	4	5
2.	Jeg synes systemet var unødvendig komplisert.	1	2	3	4	5
3.	Jeg synes at de forskjellige delene hang godt sammen.	1	2	3	4	5
4.	Jeg tror jeg vil måtte trenge hjelp fra en person med teknisk kunnskap for å kunne bruke dette systemet.	1	2	3	4	5
5.	Jeg synes det var for mye inkonsistens i systemet. (Det virket ulogisk).	1	2	3	4	5
6.	Jeg vil anta at folk flest kan lære seg dette systemet veldig raskt	1	2	3	4	5
7.	Jeg trenger å lære mye før jeg kan komme i gang med å bruke dette systemet på egen hånd.	1	2	3	4	5

E.3 System Usability Scale Questionnaire Results

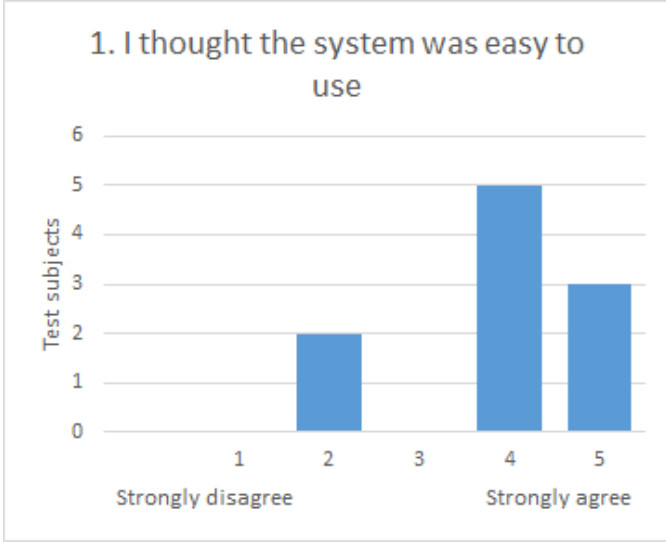

Figure E.1: Questionnaire statement 1

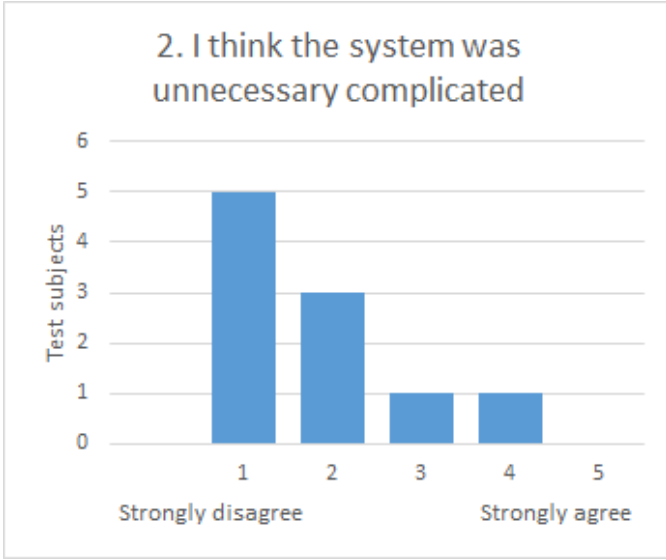

Figure E.2: Questionnaire statement 2

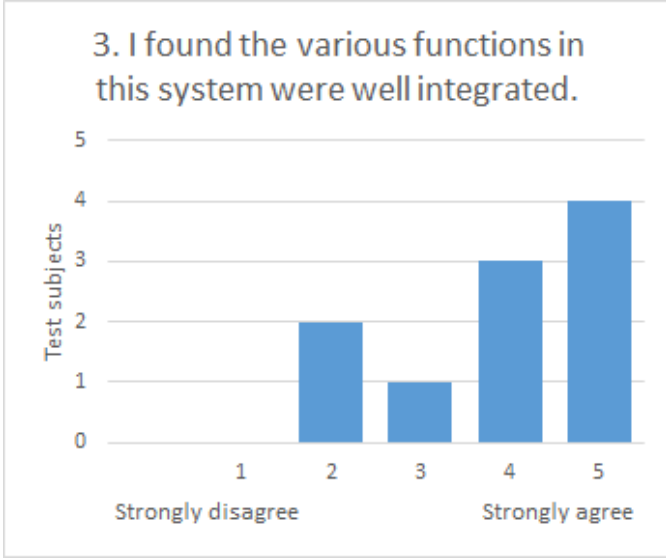

Figure E.3: Questionnaire statement 3

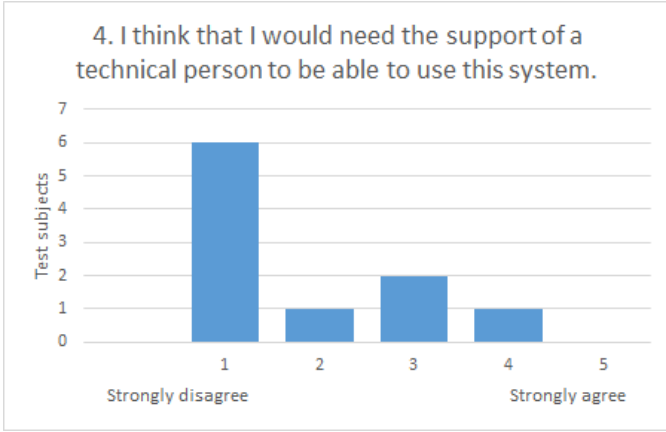

Figure E.4: Questionnaire statement 4

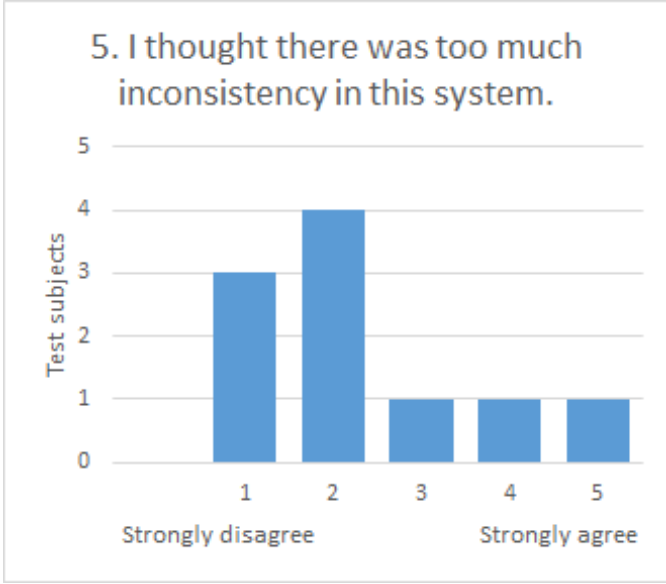

Figure E.5: Questionnaire statement 5

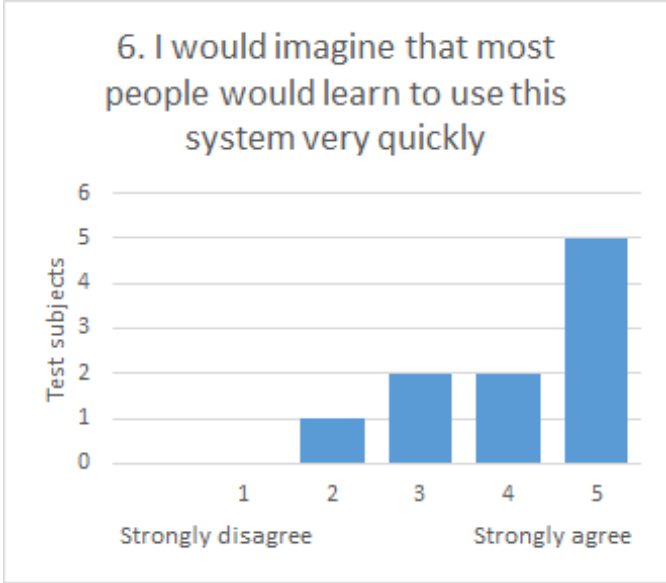

Figure E.6: Questionnaire statement 6

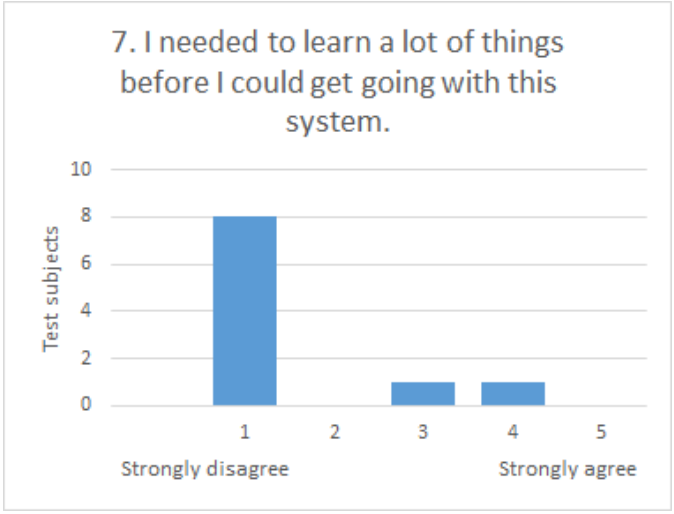

Figure E.7: Questionnaire statement 7

E.4 Declaration of Consent

Samtykke og taushetserklæring

Jeg deltar frivillig i brukervennlighetstesting i forbindelse med master-gruppen «RAPT». Som testbruker har jeg rett til å avbryte testen når som helst uten begrunnelse. Jeg er anonym, og mine personalia og kontaktinformasjon skal ikke offentliggjøres eller brukes i annen sammenheng. Dersom jeg ønsker det, har jeg rett til å få slettet video opptaket. Som kompensasjon for deltakelse mottar jeg et gavekort hos Trondheim Kino på 100 kr.

Samtykke til opptak

Det vil bli gjort opptak av lyd, bilde og skjerminteraksjon. Jeg samtykker til at disse opptakene kan brukes til brukervennlighetsanalyse, og jeg fraskriver med herved alle rettigheter til opptaket.

Taushetserklæring

Den informasjon og kunnskap om systemet som jeg tilegner meg, erklærer jeg herved at jeg ikke skal dele med andre.

Navn	Signatur	Sted/Dato

Appendix F

Compile Server

F.1 Python Server

```
1  import SocketServer
2  import subprocess
3  import os
4  import datetime
5  import codecs
6  import time
7
8  class MyTCPHandler(SocketServer.BaseRequestHandler):
9      def handle(self):
10         log = codecs.open("C:\Users\Daniel\Dropbox\Masteroppgave
               2013-14\ServerOutput\serverlog.txt", 'a+', 'utf-8')
11         # self.request is the TCP socket connected to the client
12         log.write("---------" + str(datetime.datetime.now()) +
               "---------\n")
13         time.sleep(1)
14         self.data = self.recv_all(8192)
15         log.write("Read: " + str(len(self.data)) + " bytes\n")
16         log.write("{} wrote:".format(self.client_address[0])+"\n")
17         log.write(self.data + "\n")
18         file = open("sketch/sketch.cpp", 'w')
19         file.write(self.data)
20         file.close()
21
22         #Remove old hex in case of compile fail
23         try:
24             os.remove("sketch/sketch.cpp.hex")
25             print("Removed old file")
26         except OSError:
27             pass
28
```

```
29          #Operating System
30          if os.name == "posix":
31              #linuxStuff
32              pass
33          elif os.name == "nt":
34              #windows
35              subprocess.call("compiler.bat", shell=True)
36              print subprocess.CalledProcessError
37
38          else:
39              log.write("Only support for Linux/Windows (none detected)\n")
40          log.close()
41
42          if os.path.isfile("sketch/sketch.cpp.hex"):
43              file = open("sketch/sketch.cpp.hex", 'r')
44              hex = file.read();
45              file.close()
46              hex = hex.replace(":", "3A")
47              self.request.sendall(hex)
48          else:
49              self.request.sendall("fail")
50
51      def recv_all(self, readSize):
52          total_data=[]
53          data = self.request.recv(readSize)
54          total_data.append(data)
55          while len(data) == readSize:
56              data = self.request.recv(readSize)
57              total_data.append(data)
58          return ''.join(total_data)
59
60
61  if __name__ == "__main__":
62      HOST, PORT = "192.168.0.102", 8000
63
64      server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)
65      server.serve_forever()
```

F.2 Compile Batch Script

```
1  @ECHO off
2  ECHO %DATE% %TIME%
3  ECHO Compile Starting ...
4  avr-g++ -c -g -Os -Wall -fno-exceptions -ffunction-sections
       -fdata-sections -mmcu=atmega328p -DF_CPU=16000000L -MMD -DUSB_VID=null
       -DUSB_PID=null -DARDUINO=105 -I"C:\Program Files
       (x86)\Arduino\hardware\arduino\cores\arduino" -I"C:\Program Files
```

```
    (x86)\Arduino\hardware\arduino\variants\standard" -I"C:\Program Files
    (x86)\Arduino\libraries\ComputerSerial" -I"C:\Program Files
    (x86)\Arduino\libraries\WiFlyHQ" -I"C:\Program Files
    (x86)\Arduino\libraries\SoftwareSerial" "sketch\sketch.cpp" -o
    "sketch\sketch.cpp.o"
avr-gcc -Os -Wl,--gc-sections -mmcu=atmega328p -o "sketch\sketch.cpp.elf"
    "sketch\sketch.cpp.o" "sketch\ComputerSerial\ComputerSerial.cpp.o"
    "sketch\WiFlyHQ\WiFlyHQ.cpp.o"
    "sketch\SoftwareSerial\SoftwareSerial.cpp.o" "sketch\core.a"
    -L"sketch" -lm
avr-objcopy -O ihex -j .eeprom --set-section-flags=.eeprom=alloc,load
    --no-change-warnings --change-section-lma .eeprom=0
    "sketch\sketch.cpp.elf" "sketch\sketch.cpp.eep"
avr-objcopy -O ihex -R .eeprom "sketch\sketch.cpp.elf"
    "sketch\sketch.cpp.hex"
ECHO Compile finished
```

Appendix G

Generator Example

G.1 Graphical Application

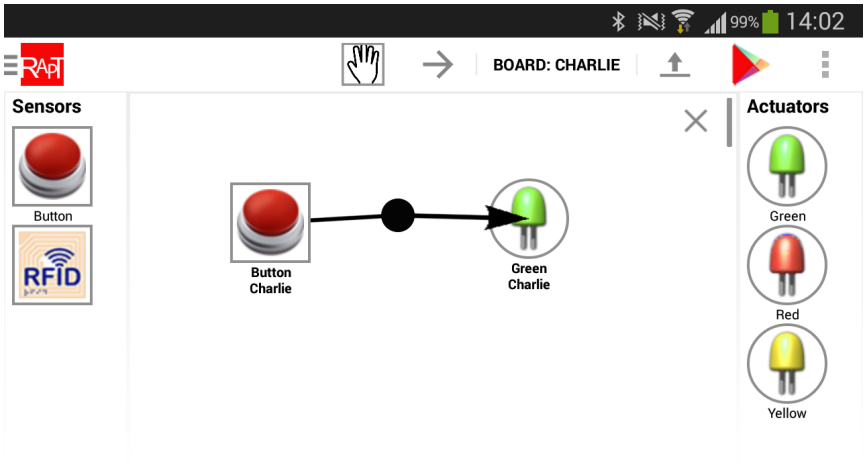

Figure G.1: Simple Application

The sketch generated by the app displayed in Figure G.1 are presented in Appendix G.2. When the button is pressed a led should turn on (flash) for five seconds.

G.2 Cpp code

```
1 #line 1 "Sketch.ino"
2 #include <ComputerSerial.h>
```

```
#include <SoftwareSerial.h>
#include "Arduino.h"
void setup();void loop();void serialEvent();
boolean buttonPressed(int pin, int state, int previousState);
void flash(int pin, int timeon, class Long& startTime, class Continuous&
    continuous );
void clearLED15();
#line 2
class Continuous { public: boolean b; Continuous(boolean a){ b = a;} };
class Long{
  public:
  long t;
  Long(long start) {
    t = start;}
};

int buttonPressed18_10 = 10;int flash15_9 = 9;
int state18;
int previousState18 = LOW;
Long time66 = Long(0);
Continuous flash66Continuous = Continuous(false);static ComputerSerial
    comp;
void setup(){
comp.begin(115200);
pinMode(buttonPressed18_10, INPUT);pinMode(flash15_9, OUTPUT);}
void loop(){

previousState18 = state18;
state18 = digitalRead(buttonPressed18_10);
if (flash66Continuous.b == true) {
flash(flash15_9, 5000, time66, flash66Continuous);
}

if (buttonPressed(buttonPressed18_10, state18, previousState18)){
clearLED15();
flash66Continuous.b = true;
}
}

boolean buttonPressed(int pin, int state, int previousState) {
    if (state == LOW && previousState == HIGH) {
        return true;
      }
      return false;
}

void flash(int pin, int timeon, class Long& startTime, class Continuous&
    continuous ) {
    if(startTime.t == 0 || startTime.t + timeon + 500 < millis()){
        digitalWrite(pin, HIGH);
```

```
        startTime.t = millis();
    }
    else if (startTime.t + timeon < millis()) {
        digitalWrite(pin, LOW);
        startTime.t = 0;
        continuous.b = false;
    }
}
void clearLED15() {
flash66Continuous.b = false;
}
void serialEvent() {if (Serial.peek() == 0xFD) {
Serial.read();
Serial.print("not_supported");
 Serial.write(0xFE);
}
else {comp.serialEvent();}
}
```

G.3 IP lookup code

```
void serialEvent() {
    if (Serial.peek() == 0xFD) {
        delay(1000);
        Serial.read();
        boolean ssidDone = false;
        String ssid = "";
        String pass = "";
        while (Serial.available() > 0) {
            if (ssidDone) {
                pass = pass + (char)Serial.read();
            } else if (((char)Serial.peek()) == ':') {
               Serial.read();
                ssidDone = true;
            } else {
                ssid = ssid + (char)Serial.read();
            }
        }
        wifiSerial.begin(9600);
        wifiSerial.listen();
        if (wifly.begin(&wifiSerial)) {
            if (wifly.isAssociated()) {
                wifly.leave();
            }
            delay(1000);
            char ssidc[64];
            ssid.toCharArray(ssidc, sizeof(ssidc));
```

```
27              wifly.setSSID(ssidc);
28              char passc[64];
29              pass.toCharArray(passc, sizeof(passc));
30              wifly.setPassphrase(passc);
31              wifly.enableDHCP();
32              if (wifly.join()) {
33                  char buff[64];
34                  wifly.getIP(buff, sizeof(buff));
35                  Serial.print(buff);
36                  Serial.write(0xFE);
37              } else {
38                  Serial.print("null");
39                  Serial.write(0xFE);
40              }
41          } else {
42              Serial.print("null");
43              Serial.write(0xFE);
44          }
45      } else {
46          comp.serialEvent();
47      }
48  }
```

Bibliography

[1] "Android," accessed 6 May 2014. [Online]. Available: http://www.techterms.com/definition/android

[2] "What arduino can do," accessed 1 Mars 2014. [Online]. Available: http://arduino.cc/

[3] "Arduino development environment," accessed 01 May 2014. [Online]. Available: http://arduino.cc/en/guide/Environment

[4] M. M. Burnett and C. Scaffidi, *End-User Development.* Aarhus, Denmark: The Interaction Design Foundation, 2013. [Online]. Available: http://www.interaction-design.org/encyclopedia/end-user_development.html

[5] "Definition of rfid," accessed 6 May 2014. [Online]. Available: http://www.centrenational-rfid.com/definition-of-rfid-article-71-gb-ruid-202.html

[6] T. Erickson, *Social Computing.* Aarhus, Denmark: The Interaction Design Foundation, 2013. [Online]. Available: http://www.interaction-design.org/encyclopedia/social_computing.html

[7] A. M. Andrea Bellucci and I. Aedo, "Light on horizontal interactive surfaces: Input space for tabletop computing." *ACM Comput. Surv. 46, 3, Article 32 (January 2014)*, 2014. [Online]. Available: http://dx.doi.org/10.1145/2500467

[8] "Ubiquitouscomputing," accessed 6 May 2014. [Online]. Available: http://www.rcet.org/ubicomp/what.htm

[9] "About ubicollab project," accessed 16 May 2014. [Online]. Available: http://www.ubicollab.org/about/

[10] E. Hornecker and J. Buur, "Getting a grip on tangible interaction: A framework on physical space and social interaction," *on Human Factors in Computing Systems, April 22, 2006 - April 27, 2006, 2006, vol. 1, pp. 437–446.*, 2006.

[11] H. Ishii, "Tangible bits: Beyond pixels," *Second International Conference on Tangible and Embedded Interaction - Conference Proceedings, 2008, pp. xv–xxv.*, 2008.

[12] A.-L. Syrjanen and K. Kuutti, "From technology to domain: The context of work for end-user development," *Inspiration, Integrity, and Intrepidity, iConference 2011, February 8, 2011 - February 11, 2011, 2011, pp. 244–251.*, 2011.

[13] F. Balle and M. Balle, "Lean development," 2005.

[14] Perera and Fernandoe, "Enhanced agile software development – hybrid paradigm with lean practice," *Second International Conference on Industrial and Information Systems*, 2007.

[15] P. MIDDLETON, "Lean software development: Two case studies," *Software Quality Journal*, 2001.

[16] A. A. Inc., "The twelve principles of agile software," accessed 17 January 2014. [Online]. Available: http://www.agilealliance.org/the-alliance/the-agile-manifesto/the-twelve-principles-of-agile-software/

[17] S. Ambler, "Strategies for scaling agile software development," May 2010, accessed 17 January 2014. [Online]. Available: https://www.ibm.com/developerworks/community/blogs/ambler/entry/principles_lean_software_development?lang=en

[18] "Jira - atlassian," accessed 6 May 2014. [Online]. Available: https://www.atlassian.com/software/jira

[19] A. Esteves and I. Oakley, "Mementos: a tangible interface supporting travel," *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, 2010.

[20] H. ISHII, "Tangible user interfaces," *CHI 2006 workshop*, 2006.

[21] E. Hornecker and J. Buur, "Getting a grip on tangible interaction: A framework on physical space and social interaction," *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, 2010.

[22] A. Alves, R. Lopes, and P. Matos, "Reactoon: Storytelling in a tangible environment," *Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), 2010 Third IEEE International*, 2010.

[23] H. Ishii, "Tangible bits: Beyond pixels," *Second International Conference on Tangible and Embedded Interaction*, 2008.

[24] W. Wang, X. Wang, and R. Wangi, "A spatial faithful cooperative system based on mixed presence groupware model," pp. 269–275, 2009.

[25] R. Ballagas, M. Ringel, M. Stone, and J. Borchers, "istuff: A physical user interface toolkit for ubiquitous computing environments," 2003.

[26] D. Fogli, "End-user development for e-government website content creation," *V. Pipek et al. (Eds.): IS-EUD 2009, LNCS 5435, pp. 126–145, 2009*, 2009.

[27] M. Spahn and V. Wulf, "End-user development for individualized information management: Analysis of problem domains and solution approaches," *J. Filipe and J. Cordeiro (Eds.): ICEIS 2009, LNBIP 24, pp. 843–857, 2009*, 2009.

[28] Y. T. R. L. Q. Q. Siti Nor Hafizah Mohamad, Ahmed Patel, "Principles and dynamics of block-based programming approach," *IEEE Symposium on Computers and Informatics*, 2011.

[29] S. M. M. M. Kai Petersen, Robert Feldt, "Systematic mapping studies in software engineering."

[30] C. A. Olavo Barbosa, "A systematic mapping study on software ecosystems," *Proceedings of the Workshop on Software Ecosystems 2011*, 2011.

[31] "Google play," accessed 6 January 2014. [Online]. Available: https://www.play.google.com

[32] "Google scholar," accessed 6 January 2014. [Online]. Available: http://scholar.google.com

[33] A. Bellucci, A. Malizia, I. Aedo, and P. Díaz, "Prototyping device ecologies: Physical to digital and viceversa," 2010.

[34] E. Mugellini, S. Gerardi, D. Baechler, and O. A. Khaled, "Mymemodules: a graphical toolkit for the easy creation of personal tui," 2006, journal =.

[35] J. Lee, L. Garduño, E. Walker, and W. Burleson, "A tangible programming tool for creation of context-aware applications," *UbiComp'13*, 2013.

[36] E. Baafi and A. Millne, "A toolkit for tinkering with tangibles & connecting communities," 2011.

[37] "Scratch," accessed 2 May 2014. [Online]. Available: http://scratch.mit.edu/

[38] F. Garzotto and R. Gonella, "An open-ended tangible environment for disabled children's learning," 2011.

[39] S. R. Klemmer, J. Li, J. Lin, and J. A. Landay, "Papier-mâché: Toolkit support for tangible input," 2004, journal =.

[40] "Arduino droid," accessed 16 April 2014. [Online]. Available: http://arduinodroid.blogspot.co.uk/

[41] "Arduino commander," accessed 16 April 2014. [Online]. Available: https://play.google.com/store/apps/details?id=name.antonsmirnov.android.arduinocommander

[42] "Amarino," accessed 16 April 2014. [Online]. Available: http://www.amarino-toolkit.net/

[43] "Ubicollab," accessed 18 April 2014. [Online]. Available: http://www.ubicollab.org/

[44] "Xml," accessed 15 May 2014. [Online]. Available: http://www.w3schools.com/xml/xml_whatis.asp

[45] "Language reference," accessed 18 April 2014. [Online]. Available: http://arduino.cc/en/Reference/HomePage

[46] "Bluetooth," accessed 9 May 2014. [Online]. Available: http://developer.android.com/guide/topics/connectivity/bluetooth.html

[47] "Github: Java implementation of stk500 protocol," accessed 29 January 2014. [Online]. Available: https://github.com/Prosjekt2-09arduino/STK500-Android

[48] "Rn-xv wifly module - wire antenna," accessed 18 April 2014. [Online]. Available: https://www.sparkfun.com/products/10822

[49] M. S. Quintanilha, "Buddywall: A tangible user interface for wireless remote communication," *CHI 2008*, 2008.

[50] M. Wiethoff and G. Broll, "Solofind: Chains of interactions with a mobile retail experience system," *CHI 2011*, 2011.

[51] "Near field communication," accessed 9 May 2014. [Online]. Available: http://www.nearfieldcommunication.org/

[52] "App store," accessed 11 April 2014. [Online]. Available: http://www.techopedia.com/definition/27519/app-store

[53] "Application store," accessed 11 April 2014. [Online]. Available: http://en.wikipedia.org/wiki/Application_store

[54] "Court dismisses apple false advertising claim over amazon appstore name," accessed 11 April 2014. [Online]. Available: http://appleinsider.com/articles/13/01/02/court-dismisses-apple-trademark-suit-over-amazon-appstore-name

[55] "Using avr libraries," accessed 01 May 2014. [Online]. Available: http://arduino.cc/en/Reference/UsingAVR

[56] "Download the arduino software," accessed 01 May 2014. [Online]. Available: http://arduino.cc/en/main/software

[57] "Github: Computerserial," accessed 3 February 2014. [Online]. Available: https://github.com/Prosjekt2-09arduino/ArduinoStore/tree/master/Arduino/ComputerSerial/

[58] "Xmlpullparser," accessed 01 May 2014. [Online]. Available: http://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html

[59] "Wiflyhq," accessed 3 Mars 2014. [Online]. Available: https://github.com/harlequin-tech/WiFlyHQ

[60] "Softwareserial library," accessed 3 May 2014. [Online]. Available: http://arduino.cc/en/Reference/SoftwareSerial

[61] D. C. D. in the U.S. Department of Health and H. S. H. O. of the Assistant Secretary for Public Affairs, "System usability scale (sus)," accessed 01 May 2014. [Online]. Available: http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

[62] "What is a raspberry pi?" accessed 1 Mars 2014. [Online]. Available: http://www.raspberrypi.org/help/faqs/#introWhatIs

[63] "Beaglebone black," accessed 2 Mars 2014. [Online]. Available: http://beagleboard.org/Products/BeagleBone+Black

[64] A. M. Kaplan and M. Haenleini, "Users of the world, unite! the challenges and opportunities of social media," 2009.

[65] "Rfc 826 - ethernet address resolution protocol." [Online]. Available: http://tools.ietf.org/html/rfc826,note=

[66] "Arduino vs raspberry pi: Which is the mini computer for you?" accessed 1 Mars 2014. [Online]. Available: http://www.makeuseof.com/tag/arduino-vs-raspberry-pi-which-is-the-mini-computer-for-you/

[67] "Raspberry pi," accessed 30 May 2014. [Online]. Available: http://upload.wikimedia.org/wikipedia/commons/3/3d/RaspberryPi.jpg

[68] "Arduino uno r3," accessed 30 May 2014. [Online]. Available: https://www.gorilladistribution.com.au/wp-content/uploads/2012/02/ArduinoUno_R3_Front.jpg

[69] "Everything you need to know about the beaglebone black," accessed 2 Mars 2014. [Online]. Available: http://www.tested.com/art/makers/459278-everything-you-need-know-about-beaglebone-black/

[70] "Rn-42/rn-42-n data sheet," accessed 3 Mars 2014. [Online]. Available: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Wireless/Bluetooth/Bluetooth-RN-42-DS.pdf

[71] "A look at the basics of bluetooth technology," accessed 3 Mars 2014. [Online]. Available: http://www.bluetooth.com/Pages/Basics.aspx

[72] "Welcome to bluetooth technology 101," accessed 3 Mars 2014. [Online]. Available: http://www.bluetooth.com/Pages/Fast-Facts.aspx

[73] "Electric imp," accessed 3 Mars 2014. [Online]. Available: https://electricimp.com/docs/

[74] "Squirrel's syntax," accessed 3 Mars 2014. [Online]. Available: http://www.squirrel-lang.org/#look

[75] “Avr libc home page,” accessed 17 January 2014. [Online]. Available: http://www.nongnu.org/avr-libc/

[76] “Android debug bridge:installing an application,” accessed 19 Mai 2014. [Online]. Available: http://developer.android.com/tools/help/adb.html

[77] “Apache maven,” accessed 19 Mai 2014. [Online]. Available: http://maven.apache.org/