Using tablet devices to control complex home appliances

Aleksi Lintuvuori

University of Tampere
School of Information Sciences
Computer Science
Master of Science Thesis
Supervisor: Zheying Zhang
December 2015

University of Tampere
School of Information Sciences
Computer Science
Aleksi Lintuvuori: Using tablet devices to control complex home appliances
M.Sc. Thesis, 58 pages and 4 index and appendix pages
December 2015

Internet of things has made connected devices and appliances widely available and tablet devices are common household items. This study focuses on technical user interface design challenges and requirements for user interface design of controlling complex home appliances with tablet devices.

There is a literature review about available controlling technologies and usability heuristics related to tablet and mobile devices. An Android test application was created and tested with four test users to find out how well those heuristics work and are covered. That application was tested against the regular user interface of a dishwasher and task completion times and errors were noted down. Test users were asked to answer a questionnaire regarding the heuristics and how well the implementation performed.

Tablet devices should be evaluated using regular usability heuristics, but besides them they require mobile specific heuristics, such as easy of input, screen readability and glancability, physical interaction and ergonomics and privacy and social convention taken into account.

The results showed that a tablet user interface was able to outperform its regular counterpart in task completion times and in number of errors. The implementation also covered those heuristics in a more comprehensive way. But among test persons the most benefit was with users who were familiar with tablets and not with dishwashers. A test user who wasn’t familiar with tablets but was with dishwashers performed tasks faster and with fewer errors with regular user interface.

In conclusion a tablet user interface enabled users who were familiar with tablets to perform tasks faster and with less errors. Those users were also more satisfied with a tablet user interface than a regular one. On the other hand a test user with little experience of tablets and familiarity with dishwashers was able to perform tasks faster an with less errors with the regular user interface. A tablet user interface was able to offer extra benefits and efficiency to users, but regular user interface should be also available to satisfy users who are not familiar with mobile devices.

Author keywords: Tablet devices, complex home appliances, remote control, mobile user interfaces, Internet of things

Table of contents

1. Introduction

Internet of things (IoT) as a concept means pervasive presence of smart, connected things and appliances around us [Chaouchi, 2010]. This includes devices and technologies such as *RFID tags, sensors, actuators, mobile phones* etc. interacting with each other to *offer advantage to their users* [Atzori *et al.*, 2010]. There are a lot of potential and possible uses available for this technology.

Gubbi *et al.* [2013] divided applications into four domains: 1) *personal and home,* 2) *enterprise,* 3) *utilities* and 4) *mobile*. In this classification personal and home domain is at a scale of an individual person or an home, enterprise is at a scale of a community, utilities at a national or regional level and mobile spreads across these domains [Gubbi *et al.*, 2013].

Personal and home domain means using sensor information with security, entertainment, health and utilities and appliances in personal use and home [Gubbi *et al.*, 2013]. This domain includes ubiquitous healthcare, such as gathering information about patient by sensors. Control of home equipment is another big field included in this domain.

Enterprise domain means using sensor information in a work environment. This includes environmental monitoring and managing of utilities of buildings, such as *heating, ventilation, and air conditioning* (*HVAC)* and lightning. Sensors have been always part of factory and building security and automation and thus clearly a part of this IoT domain. Smart environment is a large and growing field of enterprise IoT.

Utilities domain means using sensor information for services. This domain is more service than consumer oriented and includes smart grids, video and media surveillance and water network monitoring.

Mobile domain means *smart transportation* and *smart logistics*. Besides increasing efficiency, applications in these domains can also reduce noise and pollution caused by traffic and prevents traffic jams. This domain includes vehicle tracking and monitoring and enables efficient logistics management [Gubbi *et al.*, 2013].

In Figure 1 Borgia [2014] divided application domains into smaller sections by functions and presents related applications connected to these sections. Smart home domain is highlighted in Figure 1.

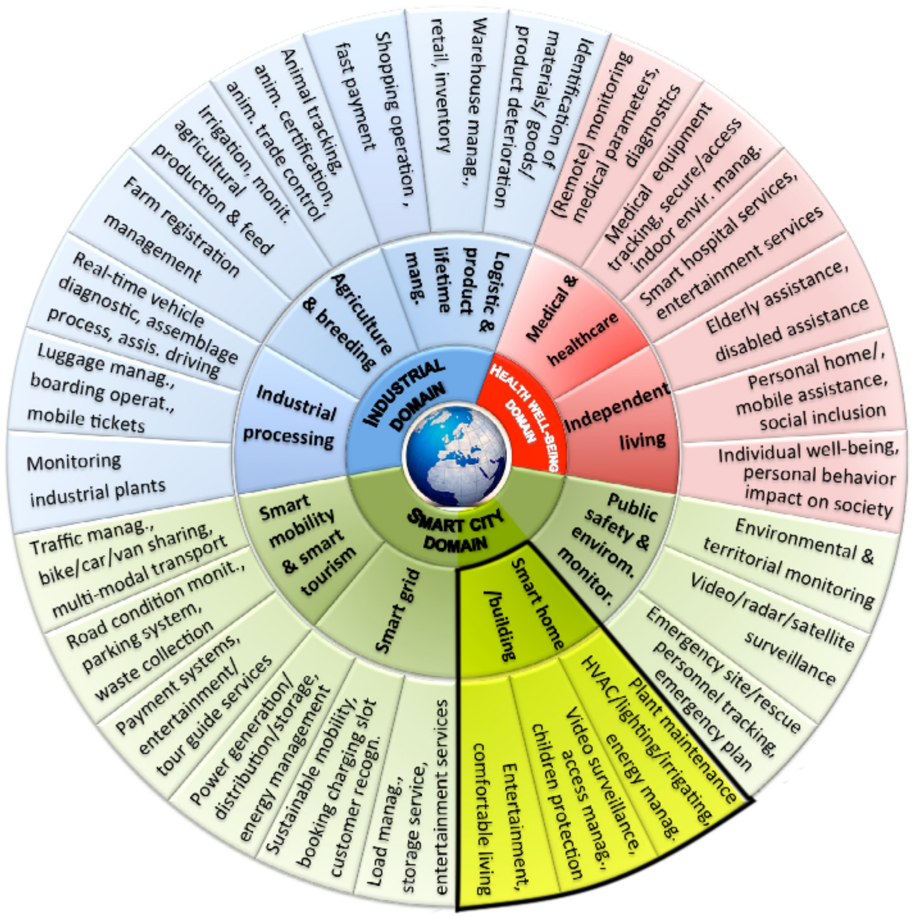

INDUSTRIAL DOMAIN
HEALTH WELL-BEING DOMAIN
SMART CITY DOMAIN
Logistic & product lifetime mang.
Agriculture & breeding
Industrial processing
Medical & healthcare
Independent living
Public safety & envirom. monitor.
Smart home /building
Smart grid
Smart mobility & smart tourism
Identification of materials/ goods/ product deterioration
Warehouse manag., retail, inventory
Shopping operation, fast payment
Animal tracking, anim. certification, anim. trade control
Irrigation, monit. agricultural production & feed
Farm registration management
Real-time vehicle diagnostic, assemblage process, assis. driving
Luggage manag., boarding operat., mobile tickets
Monitoring industrial plants
(Remote) monitoring medical parameters, diagnostics
Medical equipment tracking, secure/access indoor envir. manag.
Smart hospital services, entertainment services
Elderly assistance, disabled assistance
Personal home/, mobile assistance, social inclusion
Individual well-being, personal behavior impact on society
Environmental & territorial monitoring
Video/radar/satellite surveillance
Emergency site/rescue personnel tracking, emergency plan
Plant maintenance
HVAC/lighting/irrigating, energy manag.
Video surveillance, access manag., children protection
Entertainment, comfortable living
Load manag., storage service, entertainment services
Sustainable mobility, booking charging slot customer recogn.
Power generation/ distribution/storage, energy management
Payment systems, entertainment/ tour guide services
Road condition monit., parking system, waste collection
Traffic manag., bike/car/van sharing, multi-modal transport

Figure 1 Internet of things application domains and related applications [Borgia, 2014]

The scope of controlling home appliances is mostly in the home domain. In this domain IoT can mean controlling of appliances, such as air conditioners, washing machines and other household utilities, entertainment systems, security systems and surveillance [Gubbi *et al.*, 2013]. Besides being controlled by mobile devices, appliances can also send notifications back to user via email or social media, for example washing machine can send a notification when the program is ready.

Tablet computers and mobile devices have become affordable everyday gadgets and are widely used. These devices are mostly used for leisure purposes, but also commercial applications exist in industry. For example, many smart TVs can be controlled by using a smart phone or a tablet computer. But how about controlling home appliances with mobile devices? Mobile devices, embedded hardware and sensors are inexpensive and common technology so implementation of remote controlling is plausible.

Using tablet devices to control home appliances should give enough advantage over direct control of the device or over other means of controlling the device. Remote controlling will add additional costs to the appliance in terms of extra hardware and software development. So the benefits should overcome these added costs.

This study is eliciting and defining requirements for this kind of remote controlling using tablet devices. It focuses on these research questions:

1) What technical challenges are there for user interface design of complex home appliance control application?
2) What kind of user-interface requirements do tablet devices require to beat their regular counterparts?

This study tackles the above research questions by first doing a literature review on topic for background knowledge and requirements elicitation. Then a proof of concept demo application is developed to redefine the requirements and test feasibility of this application in practice. The expected outcome of the demo is to have practical knowledge of this field by conducting few user tests to the application. This should give good requirements as an output for such an application and answer the research question whether this application is feasible or not.

2. Tablet devices and communication technologies

2.1. Tablet computers, handheld computers and mobile devices

What is a tablet computer? A definition from Macmillan states that a *tablet computer,* or a *tab,* is a *mobile computing device,* larger than a mobile phone, that can browse the internet, handle email, play music and video, and display e-books. The user controls it by touching the screen [Macmillan Publishers Limited, 2015]. This definition gives a good idea what kinds of devices are in question.

Handheld computers are a subcategory of mobile devices. *Goodwill Community Foundation* divided them into tablet computers, e-readers and smartphones [Goodwill Community Foundation, 2015].

On the other hand, *Systems Audit and Control Association* [ISACA, 2010] used a broader classification. They divided mobile devices into groups of 1) *smartphones*, 2) *laptops and netbooks*, 3) *tablet computers*, 4) *portable digital assistants* (PDAs), 5) *USB storage devices and wireless dongles*, 6) *digital cameras*, 7) *radio frequency identification* (RFID) *devices* and 8) *infrared-enabled (IrDA) devices* such as *printers* and *smart cards* [ISACA, 2010].

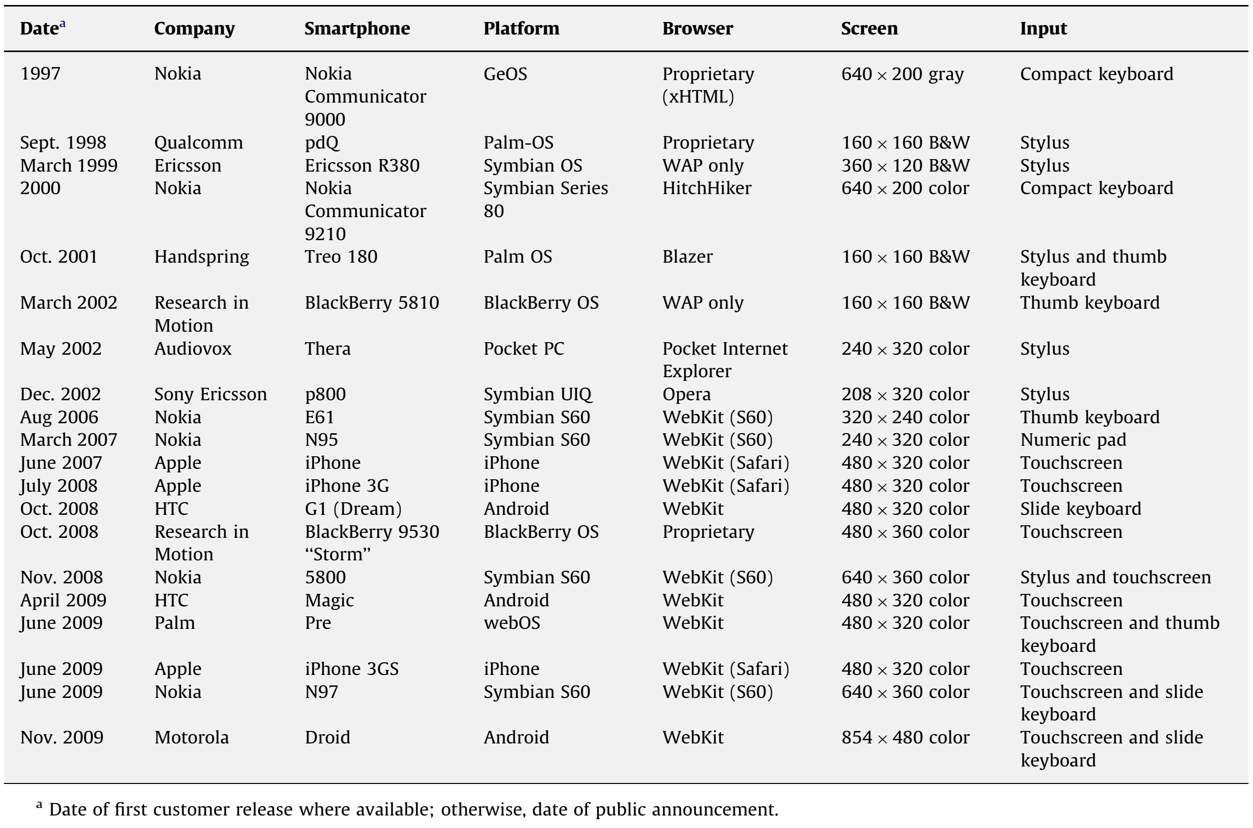
Definitions for handheld computers and mobile devices are arguable. Common attributes for handheld devices are that they are portable and they allow you to do similar things as what you would do with regular computers.

As this study focuses on tablet computers as base for controlling complex home appliances, then tablet computers and other similar devices, like *smartphones* and e-*book readers* are going to be characterized a bit more in-depth. But first history of mobile devices is going to be covered briefly.

2.2. History of mobile devices

Mobile phones first appeared in 1950's as car-bound devices [Agar, 2003]. These devices were heavy and consisted of several parts, but yet they could be easily installed in a car. In the 80's hand-holdable mobile phones became available [Farley, 2005].

In the 90's first devices with computer functionalities appeared, such as *Nokia 9000 Communicator* [Farley, 2005]. In the year 2007 *Apple* announced *iPhone* [Cohen, 2007], a touch-screen featured smartphone, which is the design basis for most current smartphones. Table 1 summarises different smart phone products from various years and their key features.



Date[a]	Company	Smartphone	Platform	Browser	Screen	Input
1997	Nokia	Nokia Communicator 9000	GeOS	Proprietary (xHTML)	640 × 200 gray	Compact keyboard
Sept. 1998	Qualcomm	pdQ	Palm-OS	Proprietary	160 × 160 B&W	Stylus
March 1999	Ericsson	Ericsson R380	Symbian OS	WAP only	360 × 120 B&W	Stylus
2000	Nokia	Nokia Communicator 9210	Symbian Series 80	HitchHiker	640 × 200 color	Compact keyboard
Oct. 2001	Handspring	Treo 180	Palm OS	Blazer	160 × 160 B&W	Stylus and thumb keyboard
March 2002	Research in Motion	BlackBerry 5810	BlackBerry OS	WAP only	160 × 160 B&W	Thumb keyboard
May 2002	Audiovox	Thera	Pocket PC	Pocket Internet Explorer	240 × 320 color	Stylus
Dec. 2002	Sony Ericsson	p800	Symbian UIQ	Opera	208 × 320 color	Stylus
Aug 2006	Nokia	E61	Symbian S60	WebKit (S60)	320 × 240 color	Thumb keyboard
March 2007	Nokia	N95	Symbian S60	WebKit (S60)	240 × 320 color	Numeric pad
June 2007	Apple	iPhone	iPhone	WebKit (Safari)	480 × 320 color	Touchscreen
July 2008	Apple	iPhone 3G	iPhone	WebKit (Safari)	480 × 320 color	Touchscreen
Oct. 2008	HTC	G1 (Dream)	Android	WebKit	480 × 320 color	Slide keyboard
Oct. 2008	Research in Motion	BlackBerry 9530 "Storm"	BlackBerry OS	Proprietary	480 × 360 color	Touchscreen
Nov. 2008	Nokia	5800	Symbian S60	WebKit (S60)	640 × 360 color	Stylus and touchscreen
April 2009	HTC	Magic	Android	WebKit	480 × 320 color	Touchscreen
June 2009	Palm	Pre	webOS	WebKit	480 × 320 color	Touchscreen and thumb keyboard
June 2009	Apple	iPhone 3GS	iPhone	WebKit (Safari)	480 × 320 color	Touchscreen
June 2009	Nokia	N97	Symbian S60	WebKit (S60)	640 × 360 color	Touchscreen and slide keyboard
Nov. 2009	Motorola	Droid	Android	WebKit	854 × 480 color	Touchscreen and slide keyboard

[a] Date of first customer release where available; otherwise, date of public announcement.

Table 1 Key smartphone product milestones, 1997–2009 [West and Mace, 2010]

Mean while first types of handheld computers were pocket sized scientific calculators in the 70's [Polsson, 2015]. After that, during the 80's, various handheld and pocket computers became available. In the 90's various *PDAs* (Personal Digital Assistants) were introduced, such as *Apple Newton MessagePAD* and *Palm Pilot* [Polsson, 2015]. Those were pocket size computers with large touch screen (related to their physical size) and they included various applications.

Starting from the 70's lots of pocket size game machines became available. In the 70's and 80's they usually featured a single game, but in 1989 Nintendo introduces *Game Boy*, a pocket size console [Dillon, 2011]. It used interchangeable cartridges and was bundled with *Tetris* game. Various other pocket size game consoles with interchangeable media became available after that [Dillon, 2011].

Most cameras are hand-holdable, and Sony started portable audio device boom with its *Sony Walkman* cassette player [Laugesen and Yuan, 2010; Kakihara and Sørensen, 2002]. Besides offering audio playback on the go, Walkman also supported the mobile life style of people going to places without physical limitations [Kakihara and Sørensen, 2002].

In the 2000's digital technology was taking its place in mobile technology. Already in the 90's cassette players were superseded by digital *CD* and *MiniDisc* players, and later on during 2000's with *MP3* players [Wee, 2003]. It also happened with film cameras, which were gradually replaced by digital cameras during 2000's [Lucas Jr. and Goh, 2009]. In the era of digitalization, technology doesn’t just add value to something, but it also is a value itself [Yoo, 2010].

In the end of 2010 decade smart phones got a new form when Apple released the *iPhone*. It wasn’t the technology itself, which made it popular and a wanted device, but the way features were offered to users with good usability [Laugesen and Yuan, 2010; Levin, 2014]. Apple also listened to customers and equipped new versions of iPhone with requested features and fixed shortcomings of previous models [Laugesen and Yuan, 2010].

The device also combines features of many separate mobile devices in a single device with good usability: it offered a web browser, applications, music playback, navigation, camera functionality and a decent amount of data storage. [Laugesen and Yuan, 2010]

In beginning of 2010-decade tablet computers made it to the public as Apple released *iPad* and Google made tablet device targeted version of its *Android* operating system [Levin, 2014]. Good amount of their success was caused by a good and increasing selection of dedicated applications.

Tablet devices didn’t just allow users to use computing in a new way, but also changed the way in which existing devices or services were being used [Levin, 2014]. For example people are reading less printed magazines and books, but on the other hand people tend to use their tablets while doing other activities, like watching TV. Figure 2 illustrates how tablet usage time is divided between different activities in the US.

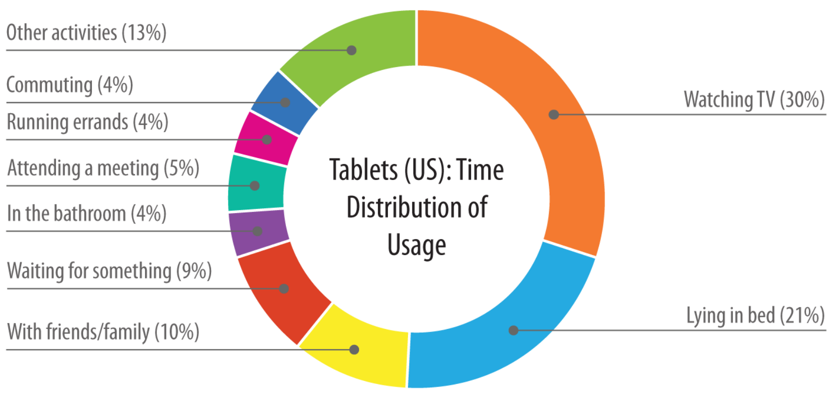


Figure 2 Time distribution of tablet usage in the US market [Nielsen.com, 2014; Levin, 2014]

Tablet devices have experienced an enormous growth in market share [Wauters, 2012] and have became a part of our everyday life. As Levin [2014] pointed out, tablet devices are widely used at home in various activities. This forms motivation factors to choose tablet devices as platforms for complex appliance control. Tablet devices are already quite common and strongly present in many modern homes in every day activities but yet rather new technology to keep it interesting as a research field.

After this brief history of mobile devices this study is going to continue into features of tablet computers, e-book readers and smartphones and compare these devices between each other.

2.2.1. Tablet computers

Tablet computers are portable devices like laptops, except they don't have fixed keyboards or touchpads like laptops do. They have touchscreens and software keyboards instead. A tablet computer has typically a screen diameter between 7 and 12 inches (18-30 cm) [Levin, 2014].

They have *WiFi* connectivity and optionally *3G/4G* modem and *Bluetooth* connectivity technology for accessories. They are equipped with solid state drives and have stand-by time of couple of days and at best can handle a full working day of intensive use.

2.2.2. E-book readers

E-book readers are, as the name suggests, designed for reading e-books. They are similar in size compared to tablet computers, but they are equipped with much less features.

Many of them use electronic paper screen technology, which behaves like normal paper and doesn't require a backlight. These displays cause less eye train, [Siegenthaler *et al.*, 2011] and remain readable in bright sunny conditions. E-readers with LCD displays are quite similar to smartphones and tablets in screen technology.

2.2.3. Smartphones

Smartphones are mobile phones with relatively large touch screens, ability to run applications and considerable amount of calculation and graphics processing power. Compared to tablets, they all have call and *3G/4G* data communication possibilities. Smartphones have similar technology as tablets, except they are smaller devices with screen diameters usually ranging from 3 to 6 inches (8-15 cm) [Levin, 2014].

In common these mobile devices are extremely portable [Myers, 2005] and all of these are suitable for reading e-books. Besides *e-books*, tablets and smartphones can be used for browsing web, watching videos, playing games and using applications [Goodwill Community Foundation, 2015].

After introducing different mobile devices and their history this study continues to different communication technologies.

2.3. Introduction to communication technologies

Appliances can use different data transmission technologies based on functional requirements and costs aspects. A thermal sensor can utilize simple and slow wireless technology. On the other hand, high quality video streaming will require high performance and stable transmission link. Suitable technologies are chosen based on their suitability, cost and previous knowledge.

Most home appliances with lots of features are equipped with rather powerful microprocessors [Can Filibeli *et al.*, 2007]. An *electronic control unit* (ECU) controls device functions and this unit processes commands of the user and monitors machine status. As devices already contain embedded computer to control them, adding a possibility to control the device remotely becomes trivial.

Appliances can have their own embedded web servers, connect to a separate server in home or to a cloud service. Desired security level restricts accessibility of a single appliance. Securing communication and data storage requires strong cryptographic techniques, which are often computationally intensive [Bradley *et al.*, 2015]. Even though powerful embedded hardware is getting more popular and affordable, this still limits use of strong security to more valuable and technical appliances.

2.3.1. Wi-Fi

WiFi connectivity is affordable, widely available and supports high transfer speeds. An appliance can act as an access point or connect to an existing network [Derthick *et al.*, 2013]. Smart phones and tablet computers are usually equipped with WiFi connectivity. Wired local area network connection can be extended to a WiFi network via a WiFi router or wireless network adapters. Embedded hardware might have an integrated WiFi connectivity or can be equipped with an adaptor.

A typical WiFi network has a good range inside and can be used in an entire apartment or small house with a single access point. If required, multiple access points can be added to extent range [ElShafee and Hamed, 2012]. On the downside a WiFi connection requires moderate power to operate and thus its use with sensors is limited. This also affects battery-based operation.

A WiFi connection can offer a good security when using the WPA2 standard [Lackner, 2013]. Using this protocol with a strong key will offer good security for home use. In general WiFi is a good candidate for remote controlling because of its availability and ease of use.

2.3.2. Bluetooth

Bluetooth connection is a good choice for *machine-to-machine* (M2M) communication. This means for example creating a direct connection to a sensor via a tablet computer. Bluetooth has range of approximately 10 m [Lee *et al.*, 2006]. This is enough to cover a room but not for example a complete apartment [Sohag and Ahamed, 2015]. On the positive side power consumption is lower than for example WiFi [Pering *et al.*, 2006] and thus this technology can be utilized with wireless sensors.

Bluetooth supports various predefined connection profiles. This is especially useful for devices in which Bluetooth profiles could be used directly, for example audio playback and text input.

Forming a Bluetooth connection to control an appliance requires a custom transport layer [Pieterse and Olivier, 2014]. This is a point-to-point connection, which will require some sort of interface for communication, for example mobile device requests a value and an appliance will send it back, or an appliance will submit its values periodically.

Bluetooth offers simultaneous connections, but the support is limited to few connections [ElShafee and Hamed, 2012]. Bluetooth is still a good way of connecting to sensors, where number of simultaneous connections is not important. For example an embedded web server would have Bluetooth connectivity to connect to a wireless camera. Server could also act as a Bluetooth master with up to 7 simultaneous connections [ElShafee and Hamed, 2012].

2.3.3. Other data communication technologies

Several different sensor types are available for data gathering about the physical environment, including temperature, humidity and brightness [Borgia, 2014]. Sensors can also monitor state information of objects, for example is a door is open.

A *wireless sensor network (*WSN) technologies support wireless data transmission between these sensors and connected devices [Akyildiz and Vuran, 2010]. *ZigBee* is an example of this kind of technology [Robles and Kim, 2010].

Near field communication (*NFC*) supports data communication by touching a tag or a device or bringing them to a close proximity [Borgia, 2014]. This technology is useful for sharing small portions of data, for example an *URL* of a device. So when for example users touches an NFC tag with a phone it points the browser to a specific web site. NFC communication is based on *RFID* technology standards [Coskun *et al.*, 2013]

Internet of things implementations could utilize *Ethernet, WiMAX, xDSL, cellural* or *satellite* data communication technologies [Borgia, 2014]. WiMAX and cellural technologies enable data transmission between a device and a base station. Ethernet could be used for in-house wirings and xDSL variants for connecting a device or home to internet via a service provider. *Ultra-wideband* (UWB) wireless transmission also exists for short range but high-speed communication [Jin-Shyan Lee *et al.*, 2007], but this technology hasn't gained much commercial success.

2.4. Related studies

As summarized in Chapter 2, there are different sorts of mobile devices available. This study concentrates on smartphones and tablet computers. Applications for controlling appliances can either run on web technology or as native applications. The former enables wider compatibility for the same application and thus requires less effort from developers.

The latter needs different implementations for different mobile platforms, but offers greater functionality and integration on that specific platform.

Nowadays there are lots of varieties in mobile devices. A user can choose an appropriate device depending on preferred price range, size and mobile platform. For appliance controlling at least a decent quality device is recommended to ensure acceptable battery life and to prevent possible problems related to overheating, processing power, low quality screens etc. Both smartphones and tablets should be suitable for remote controlling and the choice can depend on other uses of the device. Tablets offer bigger screen sizes and smartphones more portability.

Smartphones and tablets offer a good amount of mobile technologies that supports remotely controlling appliances. For example, even smartphones from few years back usually have wireless data transfer technologies like *WiFi*, *Bluetooth* and *mobile data* functionality[Riikonen *et al.*, 2013; Cecere *et al.*, 2015]. So most smartphones and tablets are able to connect to web services on the go, local WIFI or directly to a device by utilizing Bluetooth connection.

Security aspect should be taken into account when planning to control appliances via mobile devices. The mobile device itself can be stolen or fall into the wrong hands. If an Internet connection is used a sufficient authentication and security layer should be implemented. A WiFi network can also be compromised. These potential security risks should be considered when designing systems to control appliances with mobile devices. These risks are discussed in Section 6.2.

There are lots of studies about technical aspects of remote controlling and appliance control available. In this section couple of relevant ones are discussed and their findings are summarised.

Embedded web servers were mentioned in three studies [Can Filibeli *et al.*, 2007; Kumar and Lee, 2014; Li and Wang, 2015]. Embedded hardware is suitable for home use, such as controlling home appliances [Can Filibeli *et al.*, 2007]. Modern web technologies, such as *Representational State Transfer* (REST) were possible [Kumar, 2014]. Embedding a web server into an appliance seems to be a plausible option.

Android was popular choice for implementations in the fields of home automation [Ramlee *et al.*, 2013; Anwaarullah and Altaf, 2013; Panth and Jivani, 2013; Mowad *et al.*, 2014; Kumar and Lee, 2014; Kumar, 2014; Sohag and Ahamed, 2015] and appliance control and sensors [Mayer *et al.*, 2014; Liu and Su, 2015]. Reasons for Android being so popular choice include wide availability, costless development environment, affordable devices and easy learning curve.

Smart homes with remote controlling technology were mentioned in seven studies [Hsu *et al.*, 2010; Dixon *et al.*, 2010; Ramlee *et al.*, 2013; Panth and Jivani, 2013; Mowad *et al.*, 2014; Sohag and Ahamed, 2015; Liu and Su, 2015]. This field is well studied but not really commonly utilized by households. Remote controlling, surveillance and security, sensor monitoring and practical help for elderly people were popular interests in studies. Embedded web servers and Android implementations were popular among these mentioned studies.

There were many implementations for remote controlling and smart home solutions in previous studies. It seems that regardless of the chosen technical implementation these studies were able to get results. In general by choosing a common approach and doing a good design allows getting successful results. Different technical implementations do have a bit different emphasises and suitable applications.

3. Mobile user interface requirements

External interface requirements tackle issues of communicating thoroughly with users and with external hardware or software elements [Wiegers and Beatty, 2013]. These requirements were divided into four interface categories: 1) *user*, 2) *software*, 3) *hardware* and 4) *communication*. This study focuses most on the user interface side.

User interface requirements include for example taking into account standard style guides, fonts and other design related style attributes, screen parameters, message display and phrasing conventions and data validation.

Mobile user interface requirements take mobile usage into account. Buttons should be large enough so user is able to press them, interface should scale into different screen sizes and orientations.

Software requirements describe how this product should communicate between other products. Mobility aspect adds emphasis on performance, security and usability [Wiegers and Beatty, 2013].

Hardware interfaces describe how the product should communicate with different hardware parts. This includes supported hardware types and how to interact with them.

Communication interfaces include methods for communication, for example protocols and applications that utilize them. This should include possible encryption mechanisms, transfer rates and how handshaking is handled.

Mobile devices have slightly different user interfaces depending on their operating systems. Currently the most common ones are *Windows Phone/8*, *iOs* and *Android*. [Chien *et al.*, 2014; Tseng *et al.*, 2014; Rusko, 2014]. Besides the main ones, there are many other operating systems available, but they are outside the scope of the discussion.

Tablet and mobile phone versions of each platform are quite similar, but there are significant differences when comparing software platforms of different software companies. Android-based tablet and phone manufacturers also have their own customised front-end touch interfaces, for example *TouchWiz* developed by Samsung. These graphical user-interfaces enable mobile device manufacturers to customise the look and feel of home screen, menus and such UI components [Park *et al.*, 2011]. This helps different devices to stand out from each other, but adds challenges for making applications for multiple devices and maintaining uniform user experience at the same time.

Mobile devices and their user interfaces require different design patterns than for example personal computers. Mobile devices have varying requirements for screen space, have to support adaptable user interfaces and different types of input [Nilsson, 2009]. Thus, different design guidelines should be used for mobile devices.

Mobile devices are commonly used running one application per time: applications utilize the full screen of the device and a task switch is required to use an another application. Operating systems have restrictions how multitasking can work: can two full scale applications run simultaneously, or is multitasking limited to background services, for example to playing music.

Different types of devices have different screen sizes and resolutions. Especially with mobile phones careful design is needed to take advantage of the small screen space. Modern devices are equipped with high-resolution screens, which allow squeezing many items to the small screen space. However, this easily leads to layouts that are loaded with items, which results in unreadable content and for example physically too small buttons.

Besides variation between smart phones and tablets, there is also lot of variation between devices in the same device group. Newer device models are equipped with physically larger screens, which also have higher resolutions. This evolution has limits in sight: mobile phones with too large physical dimensions are not handy to use anymore and benefits for really high over 400 ppi pixel densities are arguable [Lischke *et al.*, 2015].

Mobile devices can be held in either portrait or landscape mode. Software can be designed to work only with either of those modes, but a best design practice would be to allow user to use both of those orientations. This would enhance usability as the user can use the device in a preferred way. It can also eliminate the need for horizontal scrolling, which is considered as bad design and usability [Nilsson, 2009].

Most mobile devices use software keyboards for text input. Those keyboards show up at the bottom of the screen and require considerable amount of screen space for them. User interfaces should be able to take the presence of software keyboard into account in layout. The virtual keyboard should not block the current text field or important user interface elements.

Usability can be evaluated in different ways. Nielsen and Molich [1990] divide user interface evaluation tools into four methods: 1) *formally* by some *analysis technique*, 2) *automatically* by a *computerised procedure*, 3) *empirically* by *user tests* and 4) *heuristically*[De Kock *et al.*, 2009]. This study is going to focus on formal part by analysing different *heuristics* as *usability inspection tool* and *empirical user test* as a *usability evaluation tool*.

A *style guide* summarizes common recommendations (guidelines) to improve consistency in designing and promotes *good UI practices* [Stewart and Travis, 2003; Park *et al.*, 2011]. They can improve *visual* and *functional consistency* between applications or within an application, and thus make applications *easier to use* [Park *et al.*, 2011; Gale, 1996; Gelb and Gardiner, 1997; Quesenbery, 2001; Reed *et al.*, 1999].

Lots of features are available in varying devices, which makes designing uniform user interfaces difficult. It is reasonable to have style guides for creating mobile user interfaces to enhance uniformity and thus increase usability. But it should be noted that style guides alone do not guarantee a good usability. All in all it is important to tackle these possible shortcomings and discuss design related matters in order to create successful designs for mobile user-interfaces [Park *et al.*, 2011].

3.1. Introduction to usability principles

When designing software and user interfaces, it is good to have some guidelines to prevent usability problems and to ensure good user experience. This is where usability principles come in.

Gould and Lewis [1985] found out that software development in the past lacked *good usability design practices,* even though *user focus* and *usability* have been already in knowledge before. They introduced three usability principles: *early focus on users and tasks*, *empirical measurement* and *iterative design* to have their go at tackling usability problems.

They seem to be good principles, but Cockton [2008] pointed out that in fact there are hardly any successful real world examples of the first two principles used in applications available. So those three were mostly a good first attempt in the right direction to solve usability problems caused by design process flaws. But yet their impact was in getting research in this field forward, not actually having beneficial results.

Jakob Nielsen introduced usability principles called *heuristics* in his book *Usability Engineering* [Nielsen, 1993]. The idea is to have a small set of just *10 rules* to cover most *usability problems* and thus *keeping things simple*. He introduced those heuristics with Molich [Molich and Nielsen, 1990] and revised them later by a *factor-based analysis* of usability problems [Nielsen, 1994]. These revised 10 heuristics are listed in Table 2. Nielsen's heuristics are synthetized from a number of guidelines [Hvannberg *et al.*, 2007].

1	**Visibility of system status** The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2	**Match between system and the real world** The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
3	**User control and freedom** Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
4	**Consistency and standards** Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5	**Error prevention** Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
6	**Recognition rather than recall** Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
7	**Flexibility and efficiency of use** Accelerators – unseen by the novice use – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
8	**Aesthetic and minimalist design** Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
9	**Help users recognize, diagnose, and recover from errors** Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
10	**Help and documentation** Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Table 2 Usability Heuristics for User Interface Design [Nielsen, 1994]

These heuristics are proven to be very good at explaining previous usability problems [Nielsen, 1994] and using them in design helps to reduce potential shortcomings in usability.

Gerhardt-Powals [1996] presented another set of principles named *cognitive engineering principles* (Table 3). These principles are based on situation awareness and cognitive theory [Hvannberg *et al.*, 2007].

1	**Automate unwanted workload** Free cognitive resources for high-level tasks. Eliminate mental calculations, estimations, comparisons, and unnecessary thinking.
2	**Reduce uncertainty** Display data in a manner that is clear and obvious.
3	**Fuse data** Reduce cognitive load by bringing together lower level data into a higher level summation.
4	**Present new information with meaningful aids to interpretation** Use a familiar framework, making it easier to absorb. Use everyday terms, metaphors, etc.
5	**Use names that are conceptually related to function** This is context-dependent. Attempt to improve recall and recognition.
6	**Group data in consistently meaningful ways to decrease search time**
7	**Limit data-driven tasks** Reduce the time spent assimilating raw data. Make appropriate use of color and graphics.
8	**Include in the displays only that information needed by the user at a given time** Allow users to remain focused on critical data. Exclude extraneous information that is not relevant to current tasks.
9	**Provide multiple coding of data when appropriate**
10	**Practice judicious redundancy** To resolve the possible conflict between heuristics 6 and 8.

Table 3 Cognitive engineering principles [Gerhardt-Powals, 1996]

Law and Hvannberg [2004] analysed strategies for improving and estimating effectiveness of heuristic evaluation. In their study they compared Nielsen's heuristics and Gerhardt-Powals' principles to see which one is more effective.

Nielsen's heuristics seemed to yield into better results [Law and Hvannberg, 2004]. The authors discussed that Nielsen's heuristics covered broader range of usability principles, in comparison to the narrower focus set of Gerhardt-Powals' principles. Familiarity of Nielsen's heuristics might also help them to get better results. They also noted that Nielsen's heuristics used a plain language where as Gerhardt-Powals' principles used more technical language, for example *fuse data*, which wasn't referred by any participants of that study.

The most notable difference between Gerhard-Powals cognitive engineering principles and Nielsen's heuristics was about displaying information. Nielsen [1994] suggests that the whole system status should be always visible, where as Gerhardt-Powals [1996] suggests that only information needed a given time should be displayed.

Nielsen explains that user should be informed about what is going on and get appropriate feedback about it. Gerhardt-Powals on the other hand justifies her opinion that users should be focused on critical data and extraneous information should not be shown. This actually is in line with Nielsen's 8th heuristic about minimalistic design, which says irrelevant or rarely needed information should not be shown. So in fact these different principles don't have a contradiction there, they just have names that seem to create a contradiction.

Heuristic evaluation and usability testing find different types of problems. In a study by De Kock *et al.* [2009] heuristic evaluation found 53 usability problems, where as usability testing found 25. On the other hand a heuristic evaluation don't include *efficiency* (time) or *user satisfaction* in its scope. Thus these methods are additive and complete each other, and it should be noted to remember different natures of their results.

3.2. Usability principles in mobile user interfaces

There are also heuristics available that are specifically made for mobile devices. Bertini *et al.* [2006] presented a set of heuristics especially made for mobile devices. They first were analysing usability issues in mobile computing by going through research papers independently, then joining in a group and combining their findings. They categorized and discussed heuristics, and then eliminated redundant ones and clarified the remaining ones. These heuristics are represented in Table 4.

1	**Visibility of system status and losability/findability of the mobile device** Through the mo- bile device, the system should always keep users in formed about what is going on.
2	**Match between system and the real world** Enable the mobile user to interpret correctly the information provided, by making it appear in a natural and logical order; whenever possible, the system should have the capability to sense its environment and adapt the presentation of information accordingly.
3	**Consistency and mapping** The user's conceptual model of the possible function/interaction with the mobile device or system should be consistent with the context.
4	**Good ergonomics and minimalist design** Mobile devices should be easy and comfortable to hold/ carry along as well as robust to damage.
5	**Ease of input, screen readability and glancability** Mobile systems should provide easy ways to input data, possibly reducing or avoiding the need for the user to use both hands.
6	**Flexibility, efficiency of use and personalization** Allow mobile users to tailor/personalize frequent actions, as well as to dynamically configure the system according to contextual needs.
7	**Aesthetic, privacy and social conventions** Take aesthetic and emotional aspects of the mobile device and system use into account. Make sure that user's data are kept private and safe. Mobile interaction with the system should be comfortable respectful of social conventions.
8	**Realistic error management** Shield mobile users from errors. When an error occurs, help users to recognize, to diagnose, if possible to recover from the error.

Table 4 Mobile usability heuristics [Bertini *et al.*, 2006]

A study by Machado Neto and Pimentel [2013] expands Nielsen's heuristics by conducting *usability simulation tests,* in which experts simulated the behaviour of novice users, with popular Android applications. They used Nielsen's heuristics to detect problems and expanded those heuristics with problems that could not with in any of Nielsen's heuristics. They presented 11 heuristics, which are presented in Table 5.

1	**Use of screen space** The interface should be designed so that the items are neither too distant, nor too stuck. Margin spaces may not be large in small screens to improve information visibility. The more related the components are, the closer they must appear on the screen. Interfaces must not be overwhelmed with a large number of items.
2	**Consistency and standards** The application must maintain the components in the same place and look throughout the interaction, to facilitate learning and to stimulate the user's short-term memory. Similar functionalities must be performed by similar interactions. The metaphor of each component or feature must be unique throughout the application, to avoid misunderstanding.
3	**Visibility and easy access to all information** All information must be visible and legible, both in portrait and in landscape. This also applies to media, which must be fully exhibited, unless the user opts to hide them. The elements on the screen must be adequately aligned and contrasted.
4	**Adequacy of the component to its functionality** The user should know exactly which information to input in a component, without any ambiguities or doubts. Metaphors of features must be understood without difficulty.
5	**Adequacy of the message to the functionality and to the user** The application must speak the user's language in a natural and non-invasive manner, so that the user does not feel under pressure. Instructions for performing the functionalities must be clear and objective.
6	**Error prevention and rapid recovery to the last stable state** The system must be able to anticipate a situation that leads to an error by the user based on some activity already performed by the user. When an error occurs, the application should quickly warn the user and return to the last stable state of the application. In cases in which a return to the last stable state is difficult, the system must transfer the control to the user, so that he decides what to do or where to go.
7	**Ease of input** The way the user provides the data can be based on assistive technologies, but the application should always display the input data with readability, so that the user has full control of the situation. The user should be able to provide the required data in a practical way.
8	**Ease of access to all functionalities** The main features of the application must be easily found by the user, preferably in a single interaction. Most-frequently-used functionalities may be performed by using shortcuts or alternative interactions. No functionality should be hard to find in the application interface. All input components should be easily assimilated.
9	**Immediate and observable feedback** Feedback must be easily identified and understood, so that the user is aware of the system status. Local refreshments on the screen must be preferred over global ones, because those ones maintain the status of the interaction. The interface must give the user the choice to hide messages that appear repeatedly. Long tasks must provide the user a way to do other tasks concurrently to the task being processed. The feedback must have good tone and be positive and may not be redundant or obvious.
10	**Help and documentation** The application must have a help option where common problems and ways to solve them are specified. The issues considered in this option should be easy to find.
11	**Reduction of the user's memory load** The user must not have to remember information from one screen to another to complete a task. The information of the interface must be clear and sufficient for the user to complete the current task.

Table 5 Heuristics for evaluating the usability of mobile device interfaces [Machado Neto and Pimentel, 2013]

They tested their heuristics against the original ones from Nielsen and found out that these new heuristics were able to find more usability problems. There was in total 75 distinct usability problems. These problems consisted of 20 problems which were only found by Nielsen's heuristics and 38 ones which were only found by their mobile heuristics. As there were problems that were only found by the traditional heuristics, authors modified the heuristics in a way that they also cover those 20 problems.

Even though these new heuristics are quite similar compared to Nielsen's heuristics, they still were able to find considerably more usability problems. It should be noted these new heuristics were based on the ones by Nielsen, and the test group of specialists were trained to use them. Yet it seems sensible to use these heuristics, as they are especially tailored for mobile use and offered better efficiency.

Inostroza *et al.* [2012] conducted a study about usability heuristics for touchscreen-based mobile devices. They continued on their previous work and as a result created 12 usability heuristics for touchscreen-based mobile devices based on Nielsen's heuristics. As opposed to Machado Neto and Pimentel [2013], their heuristics didn't expand and rewrite Nielsen's heuristics but used them as such for base of their heuristics. They have *physical interaction and ergonomics* as a new heuristic and *flexibility and efficiency of use* is split into *customization and shortcuts* and *efficiency of use and performance*. *Recognition rather than recall* was renamed to *minimize the user's memory load* and it's description was slightly changed.

As results their heuristics found more usability problems than Nielsen's heuristics: 23 vs. 14. But usability problems found by Nielsen's heuristics were more severe: Nielsen's average severity was 2,02 compared to 1,67 of Inostroza *et al*. They did a statistical comparison of their results and found out that there was not a significant difference between these methods: both are able to identify usability problems in similar way. The difference between severities was not significant. But as there was a advantage in number of found usability problems heuristics of Inostroza *et al* seem to be a bit better for touchscreen-based mobile devices.

Android operating system also has its own design principles for developers. They focus on giving users a good experience and in keeping users' interests in mind [Android.com, 2015]. They introduced 17 principles for Android developers and designers (Table 6).

1	**Delight me in surprising ways** Encouraging designers to use beautilful design, a carefully-placed animations, or a well-timed sound effets to enhance experience
2	**Real objects are more fun than buttons and menus** Allow people to directly touch and manipulate objects in the application.
3	**Let me make it mine** Provide sensible defaults, but also consider optional customizations that don't hinder primary tasks.
4	**Get to know me** Learn peoples' preferences over time.
5	**Keep it brief** Use short phrases with simple words.
6	**Pictures are faster than words** Consider using pictures to explain ideas. They get people's attention and can be much more efficient than words.
7	**Decide for me but let me have the final say** Take your best guess and act rather than asking first. Just in case you get it wrong, allow for 'undo'.
8	**Only show what I need when I need it** People get overwhelmed when they see too much at once. Break tasks and information into small, digestible chunks. Hide options that aren't essential at the moment, and teach people as they go.
9	**I should always know where I am** Make places in your app look distinct and use transitions to show relationships among screens. Provide feedback on tasks in progress.
10	**Never lose my stuff** Save what people took time to create and let them access it from anywhere. Remember settings, personal touches, and creations across phones, tablets, and computers.
11	**If it looks the same, it should act the same** Avoid modes, which are places that look similar but act differently on the same input.
12	**Only interrupt me if it's important** Like a good personal assistant, shield people from unimportant minutiae.
13	**Give me tricks that work everywhere** Make your app easier to learn by leveraging visual patterns and muscle memory from other Android apps.
14	**It's not my fault** If something goes wrong, give clear recovery instructions but spare them the technical details
15	**Sprinkle encouragement** Break complex tasks into smaller steps that can be easily accomplished. Give feedback on actions, even if it's just a subtle glow.
16	**Do the heavy lifting for me** Allow people to directly touch and manipulate objects in the application.
17	**Make important things fast** Decide what's most important action in your app and make it easy to find and fast to use, like the shutter button in a camera, or the pause button in a music player.

Table 6 Android design principles [Android.com, 2015]

These android design principles are quite different than those other mentioned research based principles. First instead of abstract and broad terms, they use concrete examples and informal language with lots of first person form usage. This makes them easy to understand, but also limits the depth of these principles.

Second point is that they use experience related words like *delight* and *fun.* Besides in terms of user experience, these principles might aim to be attractive business-wise. Having attractive software in Android market will make the whole ecosystem more compelling. The problem is that these kinds of principles are hard to measure. Especially fun is really subjective measure and it's hard to evaluate. In addition to this developers probably are not designing "boring" software on purpose.

Third problem with those principles is they have not been evaluated. So even if they sound sensible it would be beneficial to have some proof of their efficiency. However, it should be noted that even the page states: "consider these principles as you apply your own creativity and design thinking" [Android.com, 2015]. This suggests that those principles are guidelines to keep in mind and should not be taken as strict rules for Android developers.

3.3. Summary of different mobile usability principles

Usability principles for mobile devices and usability principles in general mentioned in this study were quite similar when compared against each other. This is understandable as they were mostly based on Nielsen's heuristics. Even the Android design principles have many similarities to Nielsen's heuristics. Those similarities remain a bit hidden as a result of informal wording of those Android design principles.

Mobile usability principles are quite similar to each other and the research based ones by Bertini *et al.* [2006], Park *et al.* [2011], Inostroza *et al.* [2012] and Machado Neto and Pimentel [2013] had successful results in finding more usability problems than Nielsen's heuristics. But what do these principles mean in practice when designing mobile user interfaces? Let's have a deeper look in how they relate to mobile user interface design.

In this study these heuristics are distinguished into two categories: first the ones that were included in Nielsen's heuristics and then later on the heuristics that were only found in mobile specific heuristics sets. Let's summarize the former ones first.

3.3.1. Nielsen's heuristics

Visibility of system status means the application shall keep user informed about what is going on. [Bertini *et al.*, 2006; Park *et al.*, 2011; Inostroza *et al.*, 2012; Machado Neto and Pimentel, 2013; Android.com, 2015]. In a mobile application user should be aware of the current location, and also going back from there should have a consistence and predictable result.

Match between systems and the real world [Bertini *et al.*, 2006; Inostroza *et al.*, 2012; Android.com, 2015] means that objects in the user interface should be familiar to the user and use language that has a connection to the real world. Symbols and graphics should be chosen carefully to make sure that they represent their real life counterparts. This is especially important for controls replacing real life ones, like sliders and knobs.

User control and freedom [Bertini *et al.*, 2006; Park *et al.*, 2011; Inostroza *et al.*, 2012; Machado Neto and Pimentel, 2013] ensures that user should be able to the application in a preferred way, for example assuring easy navigation from a function to another. Allowing user to customize the application for better efficiency and user experience is highly recommended.

Consistency and standards [Bertini *et al.*, 2006; Park *et al.*, 2011; Inostroza *et al.*, 2012; Machado Neto and Pimentel, 2013; Android.com, 2015] are especially important for controls and gestures, which should work in a consistent way between screens of application and different applications. When users find a new function they expect it to work in a same way in other places and applications. Keeping standards and common design practices in mind would be a good thing.

Error prevention [Bertini *et al.*, 2006; Park *et al.*, 2011; Inostroza *et al.*, 2012; Machado Neto and Pimentel, 2013] is noted in the studies: if an error would occur it should be explained in an understandable plain language and help the user to recover from that error state. This was also noted in the Android design principles.

Recognition rather than recall [Park *et al.*, 2011; Inostroza *et al.*, 2012; Machado Neto and Pimentel, 2013] could be explained through couple of examples. First user shouldn't need to remember needed information from another part of the application, for example from a previous screen. Another example could be associating familiar icons with actions so user will recognize them quickly instead of trying to remember where a specific action was located in a menu.

Flexibility and efficiency of use [Bertini *et al.*, 2006; Park *et al.*, 2011; Inostroza *et al.*, 2012; Android.com, 2015] means besides making applications easy to use for most people they should also enable expert users to use them in an effective way.

Aesthetic and minimalist design [Bertini *et al.*, 2006; Inostroza *et al.*, 2012; Android.com, 2015] suits well mobile devices with limited screen space. Thus it's important to utilize this screen space well and logical way, but also keeping applications' aesthetic aspect in mind.

Error recovery [Bertini *et al.*, 2006; Park *et al.*, 2011; Inostroza *et al.*, 2012; Machado Neto and Pimentel, 2013; Android.com, 2015] means for example that error messages should be written in understandable language and they should be specific. They should also suggest a solution how to fix the problem.

Help and documentation [Park *et al.*, 2011; Inostroza *et al.*, 2012; Machado Neto and Pimentel, 2013] is an important topic as ways how to give help and documentation is not really standardized yet in mobile applications. Google [2015] suggests that contextual help should be provided for Android in a way it's easily available. Especially gestures, keyboard shortcuts and other such "hidden" features should be documented that users would know these options are available and how to use them.

3.3.2. Mobile specific heuristics

*Ease of inpu*t [Bertini *et al.*, 2006; Machado Neto and Pimentel, 2013] is obvious feature of desktop computers and laptops as they use well-established input methods so special remark is not needed there. However, for mobile devices ease of input is crucial. *On-screen keyboard* is provided by the operating system, but the application developer should make sure it's not blocking UI elements and numerical only version is shown when input dialog only accepts numbers.

Screen readability and glancability [Bertini *et al.*, 2006] means the ability to see crucial information by a glance. Screen content should be readable in different environments with various light conditions. Crucial information should be possible to fetch with a glance in ideal conditions. These are also mobile-specific principles and take different use cases for mobile devices into account.

Physical interaction and ergonomics [Inostroza *et al.*, 2012] means that main functionalities, for example volume setting and power button, should have their own physical buttons located in recognizable places. They should also fit the model how user would normally hold the phone in hand [Inostroza *et al.*, 2012].

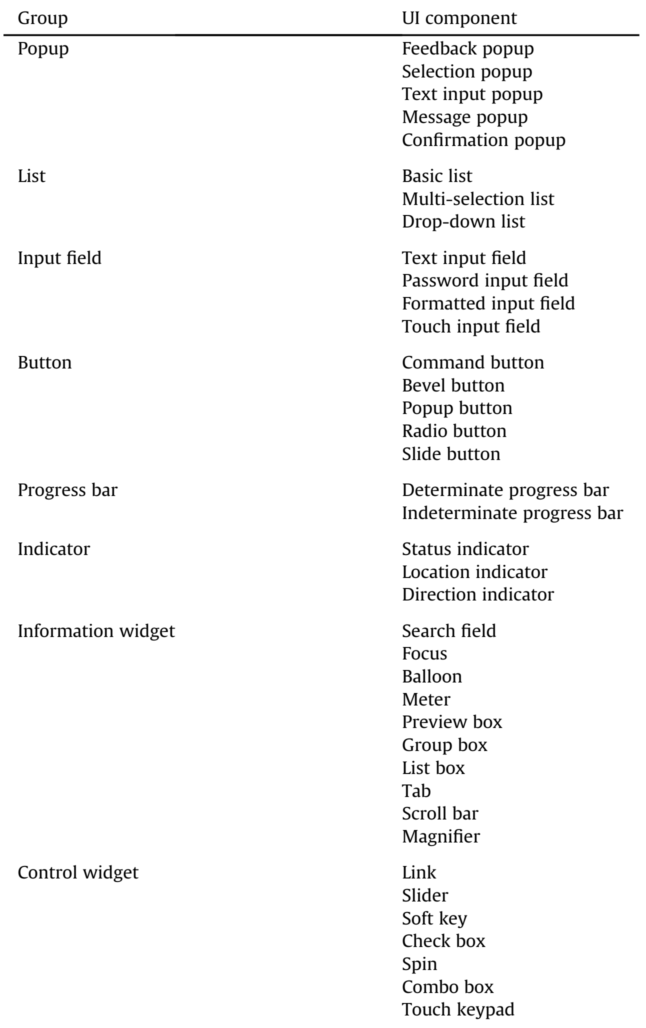
Privacy and social convention [Bertini *et al.*, 2006] are important with mobile device s as they pose a higher risk for being lost or stolen than a computer. Thus user's data should be kept private and safe. The application should also take social conventions into account, for example sound and vibration should be easily turned off if the application uses them.

As a result of summarizing different mobile usability principles, 14 principles were separately discussed. *Customization and shortcuts* was merged into *user control and freedom* and *privacy and social conventions* were separated from *aesthetic, privacy and social conventions* as aesthetics were separately discussed.

In general mobile usability principles are quite similar to regular principles: they just have a few new principles regarding mobility aspect of their use. This set keeps Nielsen's heuristics as they are proven to be good at finding usability problems [Law and Hvannberg, 2004; De Kock *et al.*, 2009], but also takes into account new aspects required by mobile environment from studies by Bertini *et al.* [2006], Park *et al.* [2011], Inostroza *et al.* [2012] and Machado Neto and Pimentel [2013].

3.4. Usability principles among UI components

Park *et al.* [2011] conducted a study in which they tested factor combination method for developing style guides. In this study they developed guidelines for 39 mobile UI components. They browsed literature for different mobile UI components. Then they classified those components based on their purpose of use. So components with the same roles and interaction methods were grouped together, even though if they appeared different [Park *et al.*, 2011]. As a result 39 components were identified (Table 7).



Group	UI component
Popup	Feedback popup
	Selection popup
	Text input popup
	Message popup
	Confirmation popup
List	Basic list
	Multi-selection list
	Drop-down list
Input field	Text input field
	Password input field
	Formatted input field
	Touch input field
Button	Command button
	Bevel button
	Popup button
	Radio button
	Slide button
Progress bar	Determinate progress bar
	Indeterminate progress bar
Indicator	Status indicator
	Location indicator
	Direction indicator
Information widget	Search field
	Focus
	Balloon
	Meter
	Preview box
	Group box
	List box
	Tab
	Scroll bar
	Magnifier
Control widget	Link
	Slider
	Soft key
	Check box
	Spin
	Combo box
	Touch keypad

Table 7 UI components of mobile phones [Park *et al.*, 2011]

Park *et al.* looked for general usability principles in 17 different studies. They collected 65 usability principles. They compared those principles against each other and then reorganized based on criteria such as similarity, inclusiveness, and relevancy [Park *et al.*, 2011]. They were able to reduce the number of principles down to 20 relevant ones.

Figure 3 demonstrates how many of the UI components represent these usability principles. Park *et al.* [2011] found out that factor combination method tackled these limitations of literature and brainstorming methods in creating usability guidelines for mobile devices.

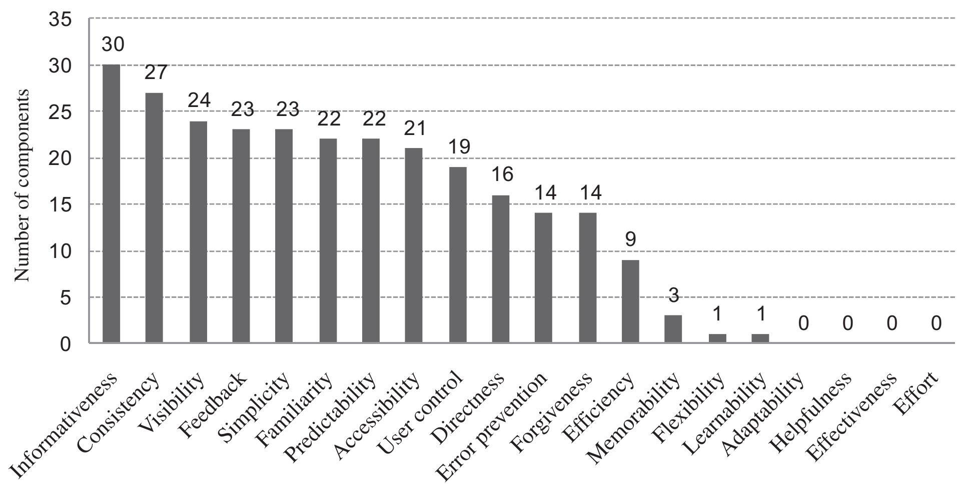


Figure 3 Number of UI components that are relevant to usability principle [Park *et al.*, 2011]

They found out that on average, each UI component was relevant to 7 UI principles. Or in other words each UI principle was applicable to 13 UI components [Park *et al.*, 2011] There were 8 UI principles, which were relevant to over 20 UI components, and on the other hand also 8 UI principles that were applicable to less than 10 UI components. Out of those 8 UI principles four weren't applicable to any UI component.

They proposed two causes for this. First one is that these principles are out of scope of the target product of their study. Their task was to study mobile UI based on conservative usability principles, so flexibility and adaptability were not important matters [Park *et al.*, 2011]. Another reason is that a specific UI principle might not be important to an individual UI component. Learnability might not be important for simple UI components, which should be rather easy to use anyway.

The main point of their study was to test factor combination method, but it also produced results about different usability principles and how many of their chosen UI components were applicable to a specific principle. Figure 3 presents quite well those UI principles that have an important role when thinking about suitable UI components for a given task.

When using mobile devices to control appliances it's important to keep the connection to real world controls. Mayer *et al.* [2014] studied mobile user interfaces for smart things. It was a model-based interface description scheme for automatically generating interfaces.

Their study included different abstractions for different interactions. This means for example that temperature is presented by ordered domain with a gauge as example symbol. These abstractions were designed for many domains, but in this study home and building automation as well as home and office appliance domains are concerned. Data presentation abstractions for sensors are shown in Figure 4. They are used for presenting measured data.

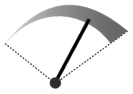
Name	Example Symbol	State Description	Example
`get data`	"General Data"	General data.	Display current song.
`get value`		Ordered domain.	Display temperature.
`get proportion`		Ordered domain with fixed range.	Display load of a server.

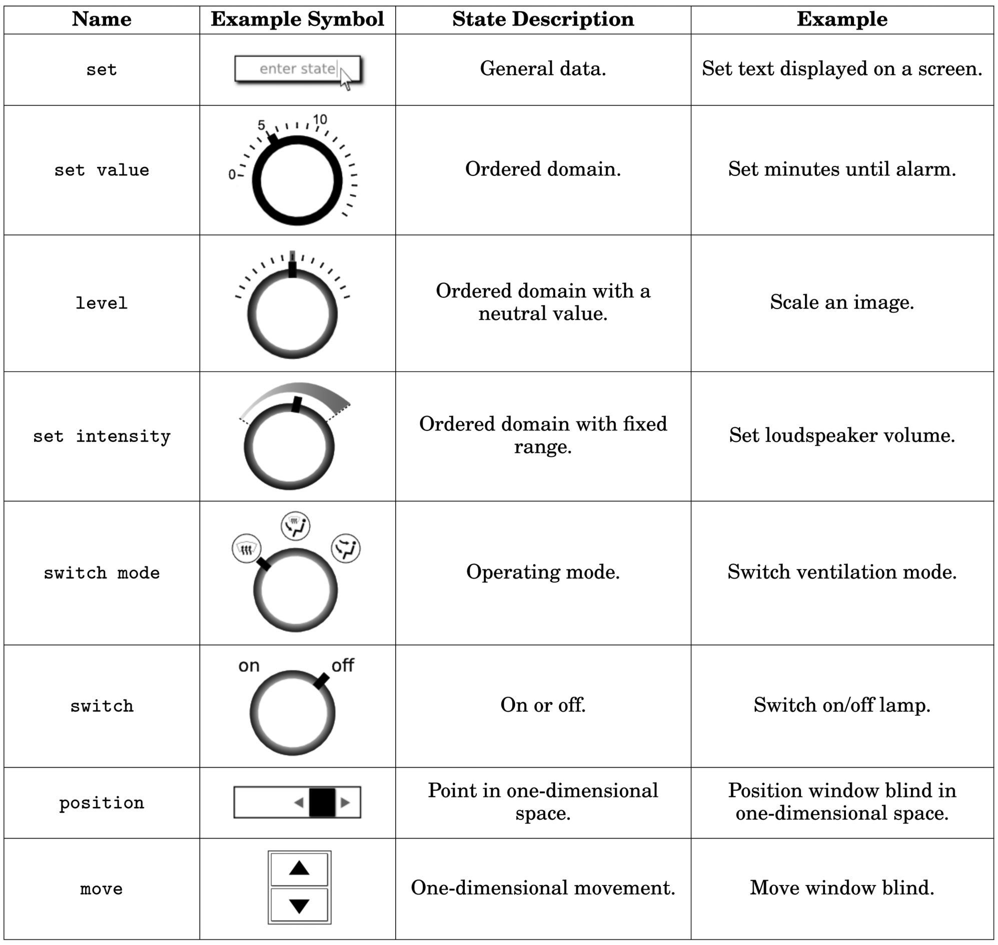
Figure 4 Interaction Abstractions for Sensors [Mayer *et al.*, 2014]

Stateless actuators accept commands as user input, but those commands won't change their state (Figure 5).

Name	Example Symbol	Description	Example
`trigger`		Trigger an action.	Reset button.
`goto`	⏮ ⏭	Adjust one-dimensional state.	Change track on hi-fi unit.

Figure 5 Interaction Abstractions for Stateless Actuators [Mayer *et al.*, 2014]

Stateful actuators allow user input, which will change state of the actuator. These are represented in Figure 6. "*Goto*" and "*move*" abstractions seem to be similar, but they are meant for different interaction patterns. Move has a direct relation to something, for example a blind curtain that would go up and down when buttons are pressed. "Goto" on the other hand relates for example to a track changing of a CD player: buttons will change the track, but it would require a CD to be present, playback going on and a track to be available in that direction. So in this analogy the action of abstraction button doesn't have direct connection to real life consequence.



Name	Example Symbol	State Description	Example
`set`	enter state	General data.	Set text displayed on a screen.
`set value`	5 10 0	Ordered domain.	Set minutes until alarm.
`level`		Ordered domain with a neutral value.	Scale an image.
`set intensity`		Ordered domain with fixed range.	Set loudspeaker volume.
`switch mode`		Operating mode.	Switch ventilation mode.
`switch`	on off	On or off.	Switch on/off lamp.
`position`		Point in one-dimensional space.	Position window blind in one-dimensional space.
`move`	▲ ▼	One-dimensional movement.	Move window blind.

Figure 6 Interaction Abstractions for Stateful Actuators [Mayer *et al.*, 2014]

They also did a user study to determine how well those abstractions presented in performed. Participants (N = 780) were asked to pick a correct abstraction type for 19 different scenarios. Different scenarios and their results are presented in Figure 7. Exact abstraction shows how many participants chose the abstraction of choice named by researchers and correct abstraction how many participants choose a suitable abstraction for the given scenario, if not the best one. Exact abstractions chosen by the researchers are shown in brackets. Average time for each scenario was also measured.

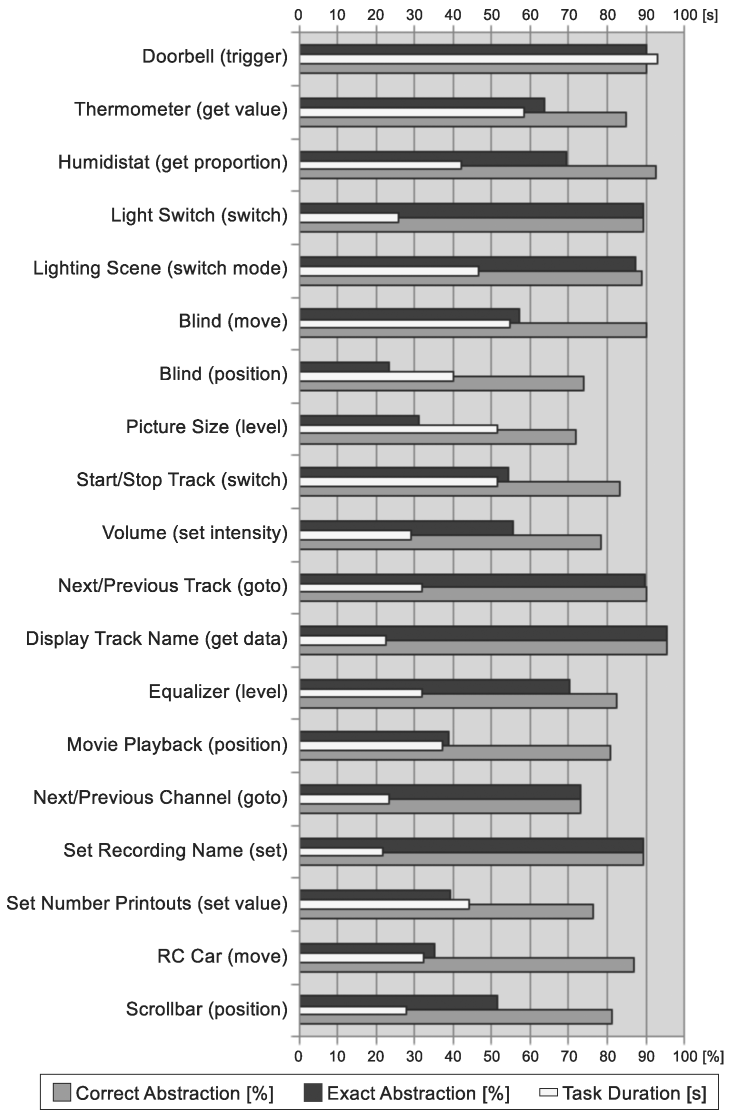


Figure 7 Performance and timing values for each of the 19 scenarios [Mayer *et al.*, 2014]

Participants were able to perform given tasks quickly and the accuracy and agreement of abstractions surprised even the researchers [Mayer *et al.*, 2014]. Their abstractions represented closely their "natural" counterparts, and this supports the idea of controlling complex home appliances with tablet devices. Authors of that study were also pleased with their results in general, and would like to see future research on how their tested control interfaces could be further improved by tailoring them to the user's situation.

4. Complex home appliances

4.1. Introduction to complex home appliances

Referring to a home appliance as a complex one doesn't only mean it should be difficult one to use or loaded with features and functions. It is about defining appliances, which have enough functions to be remotely controlled, or ones that can benefit from added features like monitoring. An air conditioning unit can be identified as a complex home appliance as it has many functions that can be controlled, such as desired temperature, fan speed and air oscillation. They might also have a possibility to be timed.

On the other hand appliance like a refrigerator itself doesn't have many functions to be controlled, but there is extra potential functionality that could be implemented to a refrigerator and to its remote user interface. A refrigerator could have an embedded web camera to show it's content. RFID tags could also be used to keep track of stored items and their expiration dates [Xie *et al.*, 2013].

Some devices might benefit from augmented control options offered by remote user interfaces. A dishwasher is relatively easy to control in every day use, but for example disabling notification sound when the program is ready might be quite tricky. The device itself usually has limited set of buttons, designed for direct control functions like choose a different wash program, timer and start. Recent devices can offer a settings menu where settings can be changed by using key combinations with the help of manual. Going through such a menu hierarchy and finally being able to change a desired settings could take up to 20 steps.

Chan *et al.* [2008] studied current state of smart homes and their future challenges. Even thought existing technology would enable it, smart homes are far behind from what was anticipated. All the smart homes they reviewed were targeted to support elder people to live autonomously. Challenges for home automation were concluded as following goals: good integration with surrounding environment, better standard of living, better knowledge of habits and intentions of users. There is potential for further research in legal and ethnical problems.

Household appliances can be capable and expensive but still contain very few UI buttons. In a study by Derthick et al. [2013] there was a cappuccino machine, which contained 7 physical buttons enabling users to control 20 distinct functions. Now that makes it quite hard to implement a functioning UI to access other than everyday functions.

There are several ways to offer an access to these advanced or seldom used functions. A menu system in a graphical user interface is the usual approach. However, that would require a display screen, hardware for running the UI, buttons and some implementation effort. This would add extra cost and it would also require physical space from the appliance itself.

As Dey *et al*. [2011] and Levin [2014] have pointed out many users have mobile devices such as smart phones and tablets widely available at their disposal nowadays. So offering a controlling option by a mobile device would be a feasible option to offer better usability to users. The appliance itself should be usable, and a mobile user interface should be able to add a lot of extra value over that [Nielsen, 1994].

4.2. Remote controlling versus direct controls at appliances

When evaluating tablet devices and other mobile devices as control interfaces, it's important to compare them with direct controls of controller devices. Nichols and Myers [2003] have conducted a study in which a home stereo system and an answering machine were used by a simulated mobile user interface and times to perform given tasks were measured.

In that study it took the participants a fraction of time needed to complete those tasks by remote mobile user interface rather than performing the tasks by using user interface of the device. Besides that, the number of missteps and help requests was also significantly smaller. The study consisted of two experiments; the first one was conducted by using paper prototypes tested by computer science students, and the second one used a real mobile device prototype user interface. They used again diverse group of university students as participants.

One shortcoming of that discussed study was that the devices in question, a stereo system and a phone answering machine, had rather poor user interfaces [Roduner *et al.*, 2007].

Roduner *et al.* [2007] carried out a similar experiment in which mobile devices were used to control appliances, but the appliances were simpler to use and had well designed direct user interfaces at the machines.

This experiment had four categories of tasks: *control*, *problem solving*, *everyday* and *repeated control*. Control tasks included for example a task to change a special device parameter or a setting, problem solving tasks dealt with abnormal situations, everyday tasks were typical and easy regularly performed tasks and repeated control tasks were ones which were performed very recently by the user and thus were still familiar by the test participant. These results are shown in Figure 8.

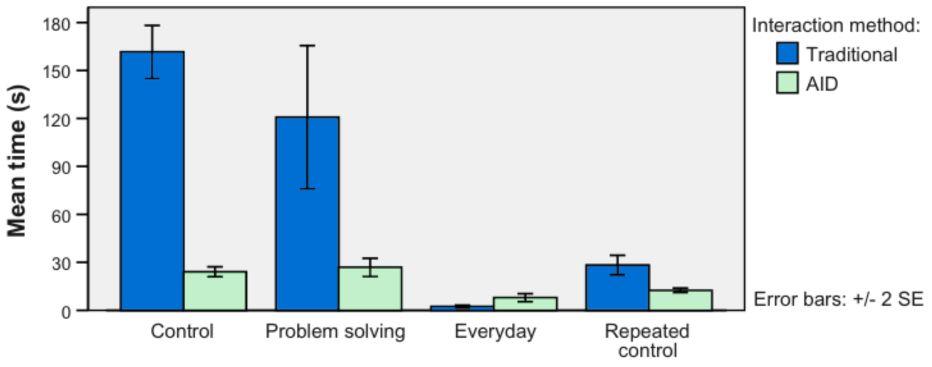


Figure 8 Mean time of task completion [Roduner *et al.*, 2007]

Now the results were quite different. In control and problem solving tasks mobile user interfaces were still much faster, but in repeated control tasks the difference was already *a lot smaller*. Everyday tasks were performed in a way shorter time with traditional user interface than with the *AID* (*Appliance Interaction Device*) corresponding one.

This finding is important for requirements elicitation when planning remote control interfaces. For easy everyday kind of tasks good traditional user interfaces offer better performances, so those kinds of controls should not be replaced completely with remote user interfaces. These controls in remote user interfaces could be still offered to the user as an option.

Complex tasks and the ones requiring troubleshooting seemed to benefit the most from remote user interfaces. There seems to be a good demand for remote user interfaces with advanced graphics and a help support.

Many different machines and appliances could utilize the same user interface. The user interface could be uniform among different types of devices and appliances, which should make it easy to adopt for new users. If there is a separate remote user interface, for example one designed for a tablet device, the regular user interface should be able to be in use simultaneously, when the remote one is used for troubleshooting. This enables troubleshooting of an error while the machine is still operating, of course if that error is not blocking the operation of the machine completely.

There are several other research papers that are related to this field or contain parts that are related to this study. Nichols *et al.* [2002] and Nichols and Myers [2003] did a study about controlling appliances with handheld computers in the beginning of 2000's. They found out that the remote user interface would outperform the regular one in almost every way.

Research of remote controlling with handhelds continued as Myers [2005] did a study of capabilities of handheld remote system. He concluded that this technology could provide a consistent user interface for different devices.

4.3. User interface

User interfaces for devices can be implemented as standalone application directly to a given platform (like Android), or as a multipurpose web page implementation, which is accessible with many mobile devices as well as computers. As for the server side there can be a dedicated server computer or the server functionality can be integrated into embedded hardware.

Neira *et al.* [2013] created a builder for adaptable human machine interfaces for mobile devices as a study. They went through a selection of *HTML5* and *JavaScript* based frameworks for creating application like user interfaces for smart phones. Those solutions offer native mobile operating system like feel and look UI based on web components. They used JavaScript implementation to offer activities of the UI. Mobile interfaces included *LWIUT 1.1*, *jQT*, *Jo*, *Titanium Mobile* and *The M Project*.

Their project used two different approaches: a web based UI and an Android application with offline capabilities. Even though the Android application requires extra effort and a layer for communicating between it and the web server, it still has the considerable benefit of being able to tackle offline situations. In those cases the UI doesn't update but it is still able to show the last information available before the connection was lost. A purely web-based solution would have problems with connection loses: it might lose all or some of the shown information after a lost connection. Such a web page is not available without a connection, where an application could be always launched.

When designing an application to control complex home appliances by using mobile devices it should taken into account that the system should be able to recover from a lost connection. Mobile devices can be used in different places where wireless connection is bad or sometimes lost by electrical interference. Implementation of such a user interface should have a requirement for coping with such situations. JavaScript technology can be utilized here to provide a dynamic web page, which should be able to show the last information even though the connection is lost, and preferably inform user about that situation. A specific Android application should also have the same capability.

4.4. Summary

To offer extra value to users and to beat their regular counterparts mobile user interface for controlling appliances should offer benefits the regular user interface cannot provide. Relation to usability heuristics is taken into account.

User should be able to check the status of the machine and do it everywhere in the house, for example check which of program phases is running from a bedroom. This relates to visibility of system status and user control and freedom. The mobile user interface should have a match between system and the real world, have consistency and follow standards and take the principle "recognition rather than recall" into account.

User control and freedom is important for having an extra value over the regular counterpart UI. The design should done in a way which prevents errors, but if an error would occur, the system should help users recognize, diagnose, and recover from errors. Thus sufficient help and documentation is essential.

The user interface should have aesthetic and minimalist design. Form and functionality should be a top priority over gimmicks. Recognition rather than recall is also important: users should know their way around and don’t have to try to remember how to use the system. Flexibility and efficiency of use can make the difference to top the regular user interface.

Ease of input is important if there is a need to change settings or write some text. Screen readability and glancability is important as mobile devices are often not used in ideal office conditions. There might be bright sunlight coming from a window or other such disturbances. User interface should take physical interaction and ergonomics into account. Buttons and other UI components should be big enough and associable to their real life counterparts.

Privacy and social conventions are also important with mobile devices. For example tablet devices might be shared along family members, so there will be multiple users. Appliances could have access rights for only certain family members.

Challenges are related to offer better usability than a regular user interface. There should be more efficiency, less errors, good user experience and some added value, for example a functionality to check status of the device. User should not feel bothered to use the interface, so it should feel natural. It should also be able to handle error situations when there is no connection or there is a problem with the machine.

There are also technical challenges, but as shown in related studies there is a good number of successful attempts to control appliances. Possible technical problems, privacy issues and security matters should be considered with a good thought.

5. Dishwasher machine controlling tasks

These tasks try to represent typical use cases for each appliance, but yet having aspects that could benefit from tablet remote controlling. These tasks also try to take into account mobile usability principles.

A dishwasher machine was chosen to represent complex home appliances in tasks. It's a quite common household device; it is widely available and has potential use for mobile device remote control.

Tasks should be executed both with direct controls available at the device and with the virtual user interface on the tablet computer. Execution time, number of errors and user satisfaction via a feedback form should be gathered.

Results of those user tests should give feedback for requirements of a remote control user-interface to control complex appliances. The proof of concept implementation should be able to point out possible problems and also benefits that such a user interface can offer. These experiments should be conducted with at least a few participants to find as many as possible of potential usability problems, and to give good feedback for requirements evaluation.

5.1. Start a wash program

Set a 50°C eco program. First task is to set a program on a dishwasher. User should choose a program with 50°C washing temperature. Here is the proposed task script:

1. Add the dishes in order
2. Add detergent
3. Check that machine is not blocked
4. Close the lid
5. Turn the machine on
6. Choose correct wash program option (Eco 50°C)
7. Check if water valve needs to be opened
8. Start the machine

5.2. Program start time

Program the dishwasher to start in two hours. In this scenario washing machine should be programmed to start automatically during the coming night.

1. Follow the steps of task 5.1 (every day task)
2. Start programming the timer
3. Choose time after two hours
4. Confirm setting
5. Start the machine

5.3. Change the program on the fly

Change the chosen program to a different one after the machine has already started. A wrong wash program was accidentally chosen and user should switch it to the correct one.

1. Follow the steps of task 5.1 (every day task)
2. Remember that there were some plates with serious grime, so a more comprehensive wash program is needed
3. Cancel the current program
4. Choose a wash program with 65°C temperature and enable *Vario Speed* option
5. Start the machine again

5.4. Stop the dishwasher during a wash

Stop the dishwasher program after it has already started. User has just started the machine, but then the user notices that one spoon was left on the table. User should stop the program, open the lid, place the spoon inside the machine, close the lid and start the program again.

1. Machine has just started
2. Stop the wash program
3. Open the machine lid
4. Place the spoon inside the machine
5. Close the lid
6. Start the program again

5.5. Choose water hardness level

Get instructions how to change water hardness level. Machine has a setting for water hardness level. User should find instructions how to change it. This task doesn't need to be performed in practice; finding correct solution is sufficient.

1. Get user manual
2. Find instructions how to change water hardness level

5.6. Implementation architecture

User interface was designed taking mobile heuristics and principles discussed in Section 3.3 into account. To summarize this current status of the machine should be visible and the user interface should match between the real worlds one. User should have control and freedom how to use the application and the design should be consistent and follow standards. The design should prevent errors and controls should be recognizable so user knows what do they do. The interface should be flexible, efficient to use and with aesthetic and minimalistic design. In case of an error, the application should provide error recovery and provide help and documentation. Interface should offer easy input of text and good screen readability and glancability. It should also allow user to have privacy and to follow social conventions.

Even though those principles are being considered when designing the user interface it should also be tested how well they are covered in practise. Test users will be asked how well those principles were covered in the implementation after they have participated in testing it.

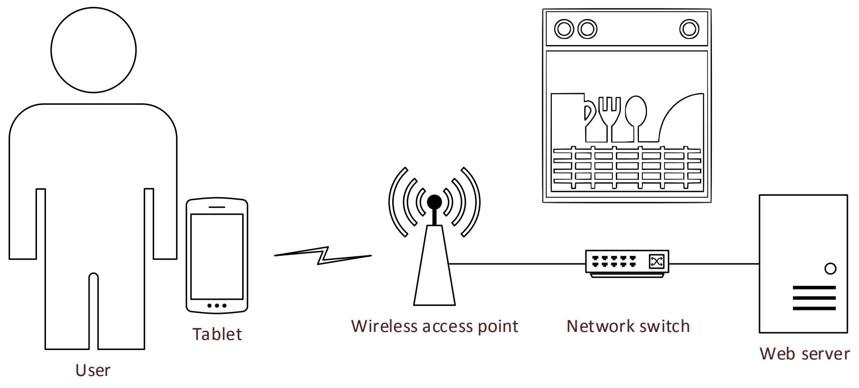


Figure 9 Application structure

To test specified scenarios and feasibility of such an application a proof on concept test environment is created. The system will consist of an *Android 5.0 tablet* and *Raspberry Pi 2 B* embedded Linux board. *Raspbian* was chosen as embedded Linux distribution because of author's previous experience with Debian-based systems.

For web server *Dropwizard* was chosen after a discussion with person with knowledge about different web frameworks [Nummila, 2015]. This framework would enable building of *RESTful web services*, can handle *JSON* and should be easy and fast to deploy. A basic web interface is going to be implemented in addition to the Android application. Android application will communicate with the server with JSON objects.

User will use Android based tablet to connect to an appliance server via wireless network. Tablet should be connected to the same wireless network where the server is located, so *network discovery service* can detect server address and connect automatically. Wireless network should cover well that area where the appliance is located. For convenience broader range is recommended, so user can for example check the appliance status in living room when the appliance is in the kitchen.

5.7. Hardware

Sony Xperia™ Z2 tablet was chosen as it has good feature set for an affordable price. It features *Android 5.1.1 Lollipop* [Google, 2015] *Qualcomm Snapdragon 801 (MSM8974AB) system on a chip (SOC)* with *Quad-core 2.3 GHz CPU* and *Adreno* 330 *GPU*, *3 GB of RAM*, *32 GB flash storage* and *10.1" touch screen* with 1920x1200 resolution [Sony Mobile, 2015]. The device in question has *only* WiFi connectivity, but mobile data operation can be simulated with a phone based WiFi hot spot if needed. Android-based tablet was chosen as it is proven to easy and costless platform for development and the author has personal experience on it.

Raspberry Pi 2 Model B is used as an embedded Linux board and it will act as a server. It has a *900MHz quad-core ARM Cortex-A7 CPU*, *1 GB of RAM*, *four USB 2.0* ports, *HDMI* and *Ethernet* ports [Raspberry Pi Foundation, 2015]. A *32 GB microSDHC* memory card is going be its storage. The Pi was chosen as it's affordable, there are various embedded Linux distributions available for it, it has low energy consumption [Yoneki, 2014] and it is powerful enough to run Java based web services. It will be connected to a WiFi router via its Ethernet port.

5.7.1. Web server

Web server will be running *Raspbian Wheezy May 2015 Linux distribution*. It has *modest hardware requirements* and there was *ready to run image file* available for Raspberry Pi 2 [Raspbian, 2015]. It offers graphical desktop environment and package management for installing software.

Dropwizard will be used for web server framework. It is a *Java based web framework*, which should *simple* and *lightweight*, but yet *powerful enough* to enable *sophisticated access control* and *good functionality* [Dropwizard Team, 2015].

As Dropwizard is a framework, there is a need for other software to form a complete web server. As the developers of Dropwizard [2015] suggest, *Jetty* is going to be used as

HTTP daemon, *Jersey* for *REST support*, *Jackson* for *JSON*, *MySQL* as *database engine*. *Apache Maven* was chosen as project management and deployment tool.

Based on experiences of Guinard *et al*. [2011] and Fielding [2000], *a RESTful web architecture* [RestApiTutorial.com, 2015] is going to be used. Each appliance will be modeled as a resource and will have its own *uniform resource identifier* (URI), for example a dishwasher would have address like:

http://server/appliance/1

The root path would return a basic information page for web browser of dishwasher. There would be a specific JSON resource to retrieve status of the dishwasher in JSON format for the Android application. Objects or functions of dishwasher could be accessed through their own resources to get status information or to change values. User should be able to assign how much is going to be shown without authentication, if nothing at all.

Appliance list can be fetched with address http://server/appliances. This will return an array of available appliances in JSON format. Each entity will contain dishwasher id, name, power state and status information. Each dishwasher has a separate settings entity, which has option for prewash, timer and its set time, time left and notification sound.

The server will return the appliance information with a GET request. An appliance can be updated with a PUT request, deleted by DELETE one and created with a POST request. Settings for an appliance can be accessed by http://server/appliance/1/settings. This resource supports GET and PUT requests.

The server will use *basic HTTP authentication* [Fielding and Reschke, 2014] for user identification. Database at web server will contain supported credential combinations for users. Android application has these as settings so users don't have to provide credentials every time.

5.7.2. Android application

Android application would be developed with *Android Studio* [Android Open Source Project, 2015]. It would be preferable to have automatically generated UIs, but for the sake of simplicity it will be left out of scope of this study. Instead this is going to focus on user interface requirements of the application, test various tasks using the proof of concept application and see what kind of requirements this user testing elicits.

The main goal is to redefine requirements defined in tasks (in Chapter 5) and finally achieve requirements that are able to answer research questions about requirements for using tablet devices to control complex home appliances.

The application consists of four features: 1) main screen to connect to a server, 2) appliance list to chose the right appliance, 3) appliance view to control the appliance and 4) settings view to change the settings. Features and related principles are presented in Table 8.

Feature	Related principles
Main screen	User control and freedom Flexibility and efficiency of use Aesthetic and minimalist design Help users recognize, diagnose, and recover from errors
Appliance list	Visibility of system status User control and freedom
Appliance view	Visibility of system status User control and freedom Match between system and the real world
Settings	User control and freedom Flexibility and efficiency of use

Table 8 Features and related principles

Main screen (Figure 10) will have server address and port where to connect. This relates to user control and freedom. A *network service discovery* is used to detect server address if the server is located in the same sub network. In this case the application connects automatically to that server. This connects to flexibility and efficiency of use and aesthetic and minimalist design. If server is offline or cannot be reached the application shows corresponding error message. This relates to principle "help users recognize, diagnose, and recover from errors".

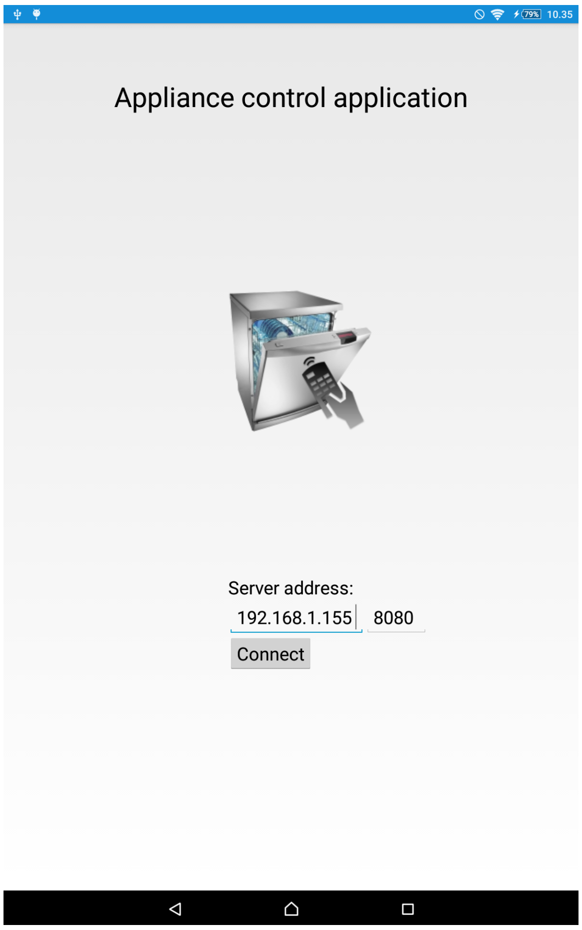


Figure 10 Main screen

Appliance list view shows appliances provided by array of appliances JSON. This list will have their names and pictures (Figure 11). Device status is also shown. This relates to visibility of system status and user control and freedom.

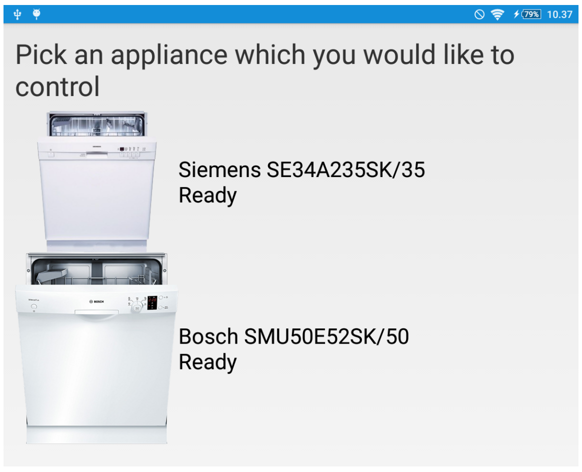
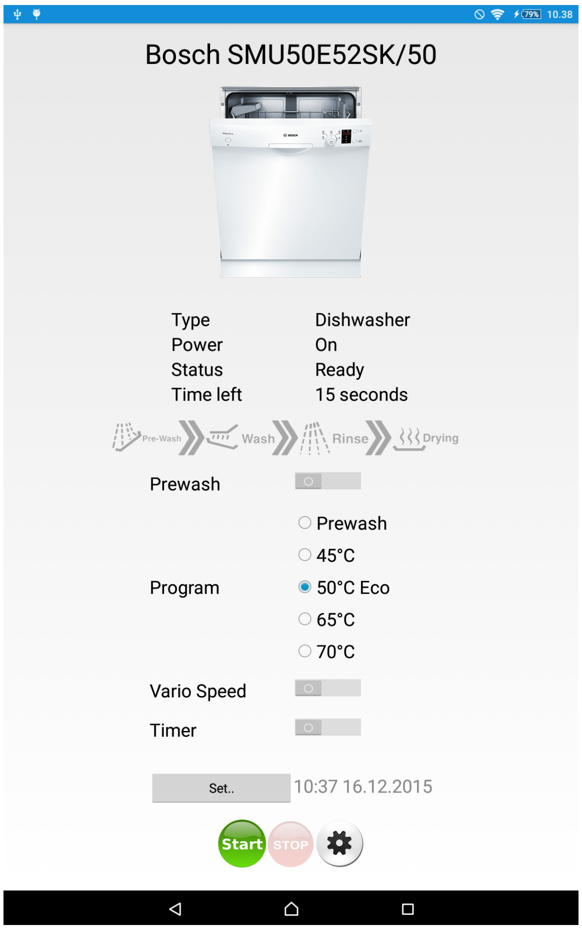


Figure 11 Screenshot of appliance list view

Appliance view (Figure 12) shows status of the dishwasher and options to control the device. This relates to visibility of system status and user control and freedom. User can toggle on the prewash, change the wash program, set a timer and enable *Vario Speed* feature. Clicking set opens a dialog for changing time and date of the timer. User can start and stop the machine. Besides principles mentioned before, this relates to "match between system and the real world". Current phase is shown and can be changed by the user.



Bosch SMU50E52SK/50

Type	Dishwasher
Power	On
Status	Ready
Time left	15 seconds

Prewash

Program: ○ Prewash ○ 45°C [✓ 50°C Eco] ○ 65°C ○ 70°C

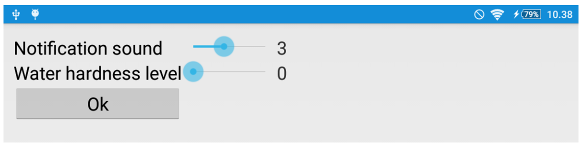
Vario Speed

Timer

10:37 16.12.2015

Figure 12 Screenshot of appliance control view

From this view user can go the settings view (Figure 13) to change settings of the dishwasher. Notification sound, water hardness level and other machine specific settings can be changed.



Notification sound 3
Water hardness level 0

Figure 13 Settings

6. Evaluation and discussion

6.1. Experiment

The implemented application was tested by four test subjects: two students who were familiar with tablets but not with dishwashers, one teacher who was familiar with both dishwasher and tablets and one secretary who was familiar with dishwashers but not with tablets.

Users went through the tasks (in Chapter 5) doing them first with the machine user interface and then with Android application. Time to complete the task and number of errors was measured.

After completing the tasks users filled questionnaire about the application. Chosen set of heuristics was evaluated for both implementations. A Likert-scale from strongly disagree to strongly agree was used for evaluation. How pleasant the Android application was use and usefulness of tablet remote controlling were asked using also Likert-scale. Finally there were questions about which devices users world like to control with tablets and at the end space for comments and feedback.

Task completion times in general with machine user interface (Figure 14) were consistent among test participants. The first task involved many prerequisites, which yielded to longer completion times, compared to other tasks, even though the first task was the simplest one.

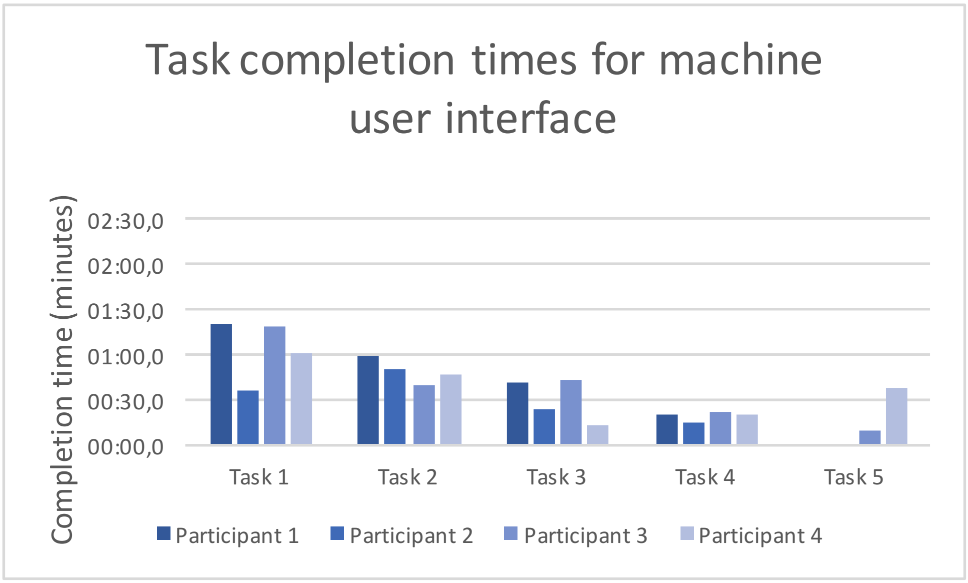


Figure 14 Task completion times for machine user interface

There were two usability problems with the first task at machine user interface: two participants didn’t figure out the main power button needs to be pressed before operating the machine. The button was located on the left side of machine separated from the main control panel area.

Surprisingly completion times were reduced noticeably with the Android user interface (Figure 15). However, test participants were familiar with the task so it was easier for them to do it again and that can reduce completion times. Second factor is usability problem with power state of the machine. Android user interface provided better feedback of machine power state so there was no delay caused by figuring out how to turn on the machine. This was related to visibility of system status.

First test showed a tablet user interface could be competitive against its regular counterpart. However, this indicates that there were usability problems with the regular user interface. A well-designed regular user interface should be able to outperform the Android counterpart, as user doesn't need to pick up the tablet before performing actions.

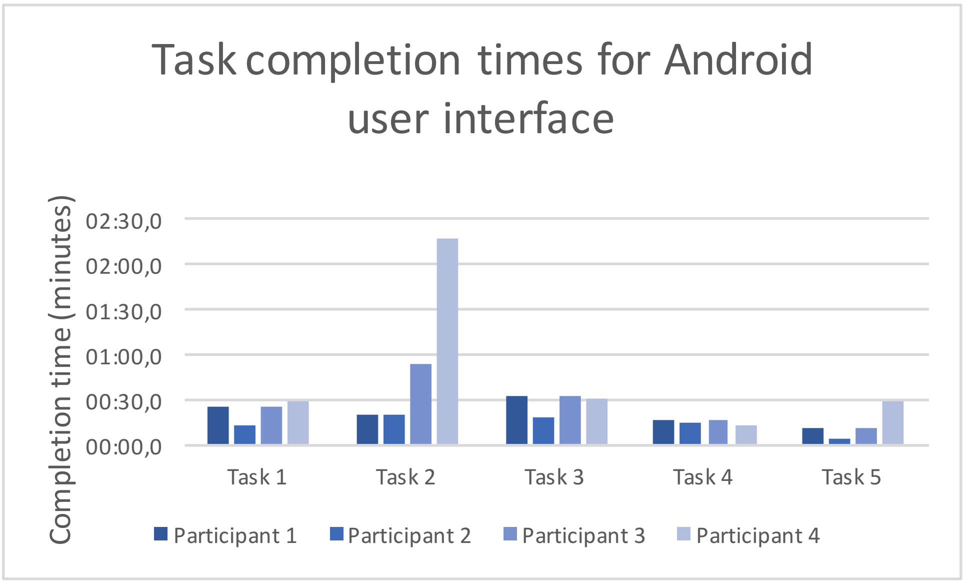


Figure 15 Task completion times for Android user interface

The second task was problematic with machine user interface. Completion times ranged from 40 seconds to one minute, but three participants had usability problem and one had two. All participants had problems with setting the correct time. Timer functionality is shown in the small LCD screen (in Figure 16) in format h:01. The problem is that the hour digit is in the field where normally there are minutes. As a result of that each participant thought they were setting minutes even though the setting was actually changing hours. Those problems were related to heuristics visibility of system status, match between systems and the real world and consistency and standards.

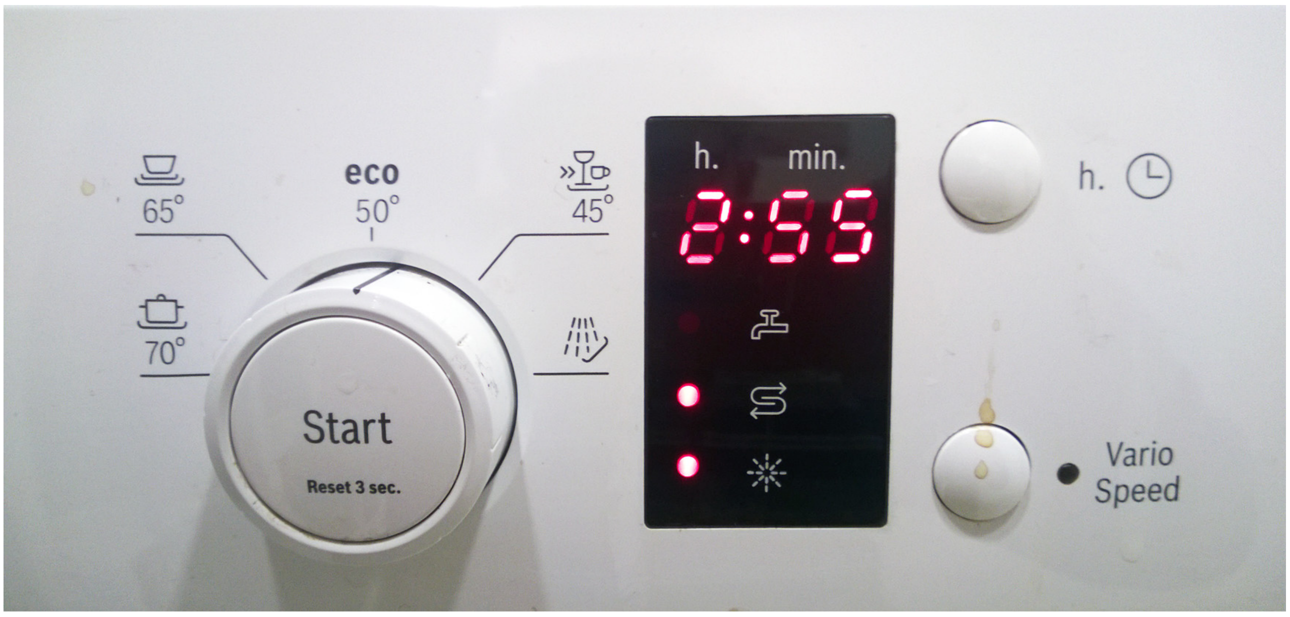


Figure 16 The machine user interface

Another usability problem was confirmation of the set time. One test participant was expecting some kind of button to confirm the setting for timer. As function for confirm was missing, test participant wasn't sure if the setting would apply or not.

Completion times for Android user interface were quite inconsistent. Two test participants were able to perform the task in around 20 seconds but other results were almost a minute and over two minutes.

Both usability problems were with the dialog (Figure 17), which was used to set time and date. One participant had problems with the fact that the task asked to set the machine to start after two hours, but the dialog asked for actual time and date. That required cognitive load to figure out the clock time from two hours of the test moment. Timer had current time as the default value and two participants found out that they could just roll the time down two hours. However, this wasn't really obvious in the Android user interface.

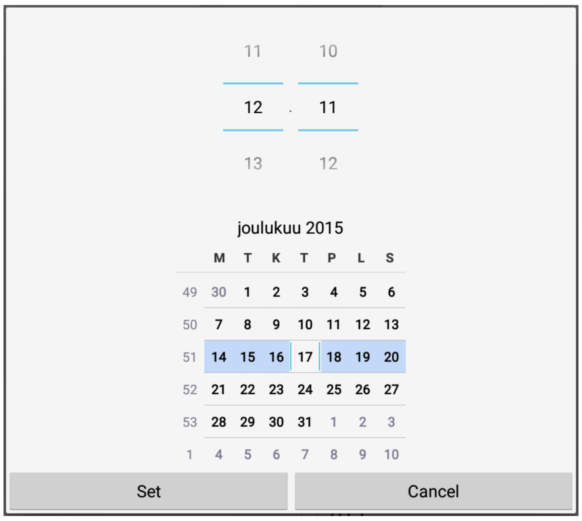


Figure 17 Dialog for setting time and date

The second problem was related to Android user interface. One participant wasn't familiar with default controls in Android (Figure 17) to set time and date. Instead of scrolling down the time, test participant tried to type the number into the control. It also took time to find the button for displaying dialog for setting time and date. As a result completion of task took over 2 minutes. This was related to match between system and the real world.

The third task had one usability problem with machine user interface: participant wasn't sure how to stop a wash program before changing the program. For other two participants it took some time to figure the behaviour out. This explains the inconsistent completion times. The problem relates to Consistency and standards.

In the Android application performing the task went fine for everybody, but two participants had left the timer on and didn't notice that. It took additional time to figure this out. On average Android completion times were lower than machine user interface ones but the fastest time of 13,9 seconds was done with the machine user interface.

In the fourth task there were no usability problems with either machine or Android user interface. Average completion times were 19,5 seconds for machine and 15,4 seconds for Android user interface so on average the Android UI was 21 % quicker.

Fifth task was problematic, as it required reading the manual, which was only available in Nordic languages. One participant could understand Finnish and the other tried to attempt the task anyway. Average times were 24,2 seconds for reading manual and 14,0 seconds for Android user interface. One participant had usability problem finding the settings feature. Unfortunately the results were incomplete, but it left an impression that a tablet user interface could offer quite big advantage in such a task.

Next coverage of how well each user interface supports selected usability heuristics is evaluated. Privacy and social convention was left out from machine user interface as it's not really relevant for a dishwasher. Coverage of heuristics in machine user interface are presented in Figure 18.

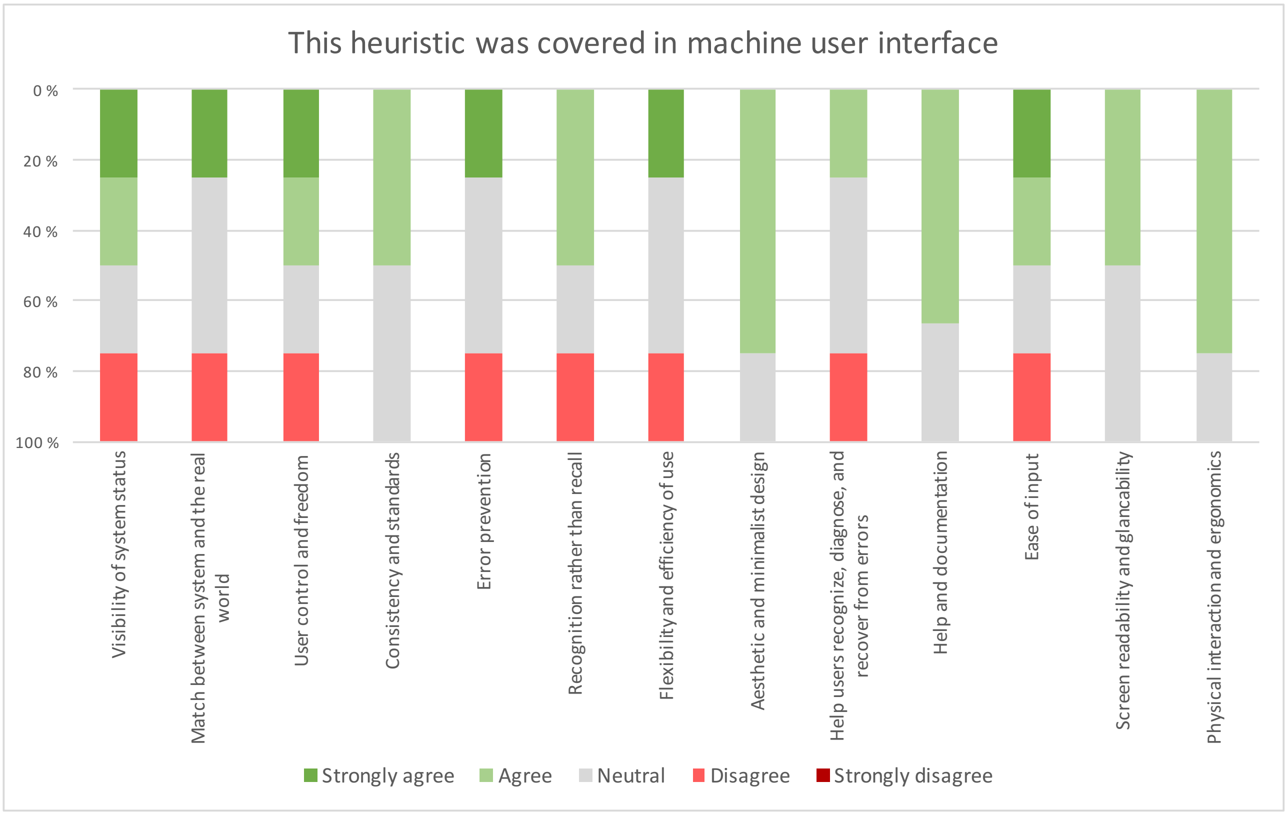


Figure 18 Heuristics covered in machine user interface

On average Android user interface supported these heuristics better but it wasn't perfect either. There were 8 disagrees in machine user interface and 4 in Android one. Help and documentation and error prevention were worse in Android user interface, otherwise Android user interface faired better or the same with heuristics. Results for Android user interface are presented in Figure 19.

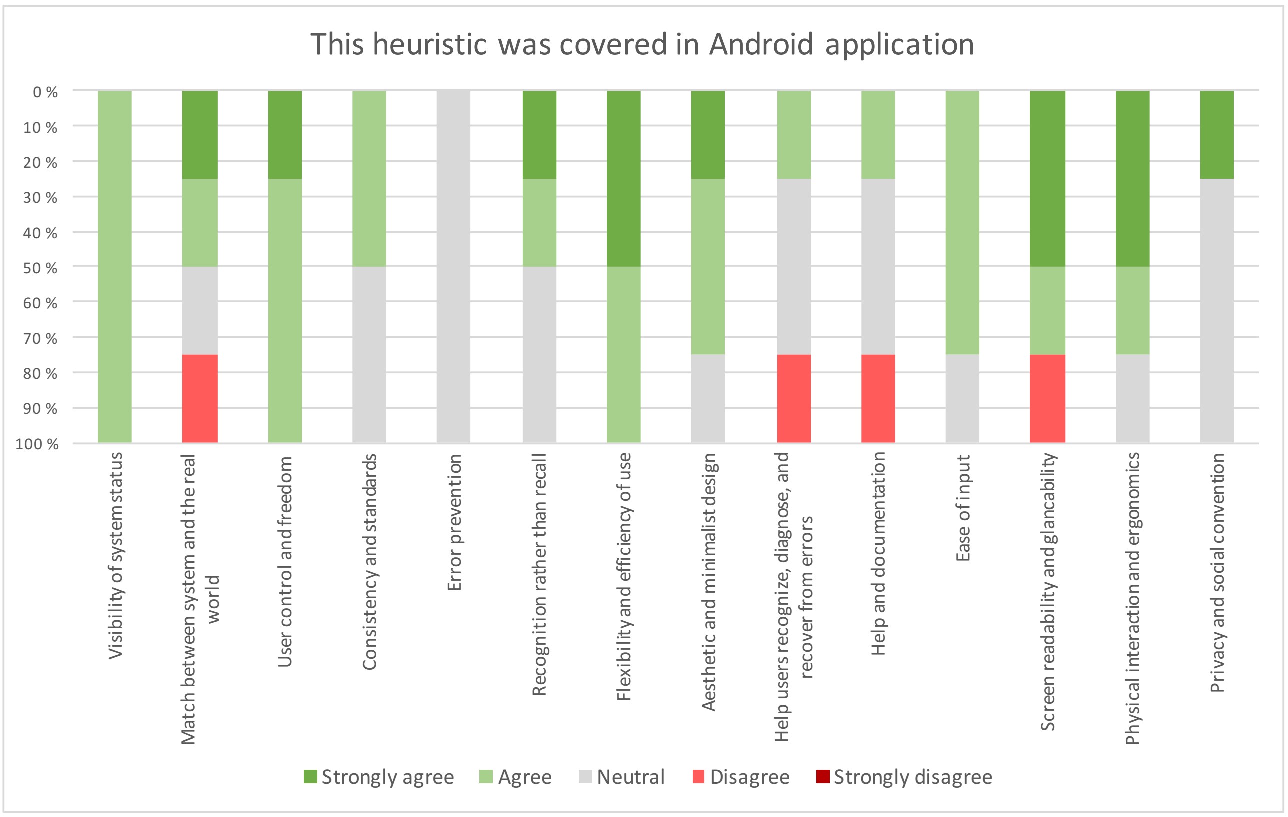


Figure 19 Heuristics covered in Android user interface

User control and freedom and flexibility and efficiency of use heuristics were significantly better in the Android user interface. On average Android application scores were one third of a grade better than machine user interface scores. Android application had an average of 3,7 and machine user interface had 3,4. The closest match of grade for Android application was agree and for machine user interface neutral but the difference was quite small.

All participants said they agreed that the Android application was pleasant to use. Three participants agreed and one strongly agreed that remote controlling of home appliances with tablet computer is useful. This and findings about completion times and slightly better scores with heuristics support usefulness of tablet device remote controlling.

There were 8 usability problems with machine user interface and 6 with the Android user interface. In general participants familiar with tablets but not with dishwashers had very few problems with the Android application and were able to perform the tasks much faster. On the other hand a participant with experience with dishwashers but not with tablets had fewer problems with the machine and most with the Android user interface. This was reflected in completion times, which were worse for Android application than the machine user interface one for that test participant

Test participants listed lights, air conditioners twice, dish washers, robot vacuum cleaners, TVs and washing machines as devices which they would like to control with a tablet user interface. Comments about current implementation were that the system could give more info of devices, like current status. One said receiving notifications when the machine is not functioning properly would be useful. Another one noted that the user interface could be optimized. There were indeed few usability problems with the tablet user interface so there is room for future improvement.

The results show that a tablet user interface can offer significant improvement over the regular counterpart. That was especially noticeable with test participants who were familiar with tablets and mobile devices but didn't have much experience with dishwashers. Opposite to that test participant with little knowledge of tablets but who was familiar with dishwashers was doing worse with the tablet user interface. Even that participant saw potential in such a solution and said it could be useful if that person would have more interest in mobile devices.

Remote controlling with tablet devices can offer benefits to users and also raised several potential devices, which could be used with them. This technology could offer freedom, efficiency of use, speed advantage and consistency between different devices. But it's important to keep the regular user interface as using tablet device as the only controlling option could be tricky to a person who is not familiar with mobile devices.

6.2. Security

Controlling home appliances via internet or trough a wireless network connection exposes a security threat. If an intruder gains access to the system, it can be used for deliberate mischief making or even an identify theft. For this reason, it is important to take security issues seriously and use secure wireless connection and a proper authentication mechanism accessing the system remotely.

6.2.1. Access rights

Obtaining an appropriate security level for remote device controlling in home environment is a complex task. Kim *et al.* [2010] found out in their study that home users have a need for complex rights control. A complex rights control would require administration, but home users often lack the expertise and patience to properly take care of feature rich access control system [Kim *et al.*, 2010]. This emerges a need for sophisticated but yet simple to use access control system.

Many persons usually use the same home appliance, and thus it doesn't have a clear owner. Yet there is need for different levels of access rights for different users or user groups. Besides permanents users, guests have also a need to operate appliances and also require specific level of access [Kim *et al.*, 2010; Mazurek *et al.*, 2010].

In addition to the complexity of user needs, the home devices themselves might have support for rather high dimension of different resource types [Kim *et al.*, 2010]. For example a smart TV might have storage media or cloud storage access to watch recorded programs, access to paid content or different allowed TV programs to watch, for example based on user type and TV program age rating.

According to Kim *et al.* [2010] a single administrator is not sufficient for home use. In case that one person would be away from home there might still be need to change access policies. However, only trusted people should have that power. For example a small child could compromise the security of home device by granting rights to felons.

6.2.2. Threats

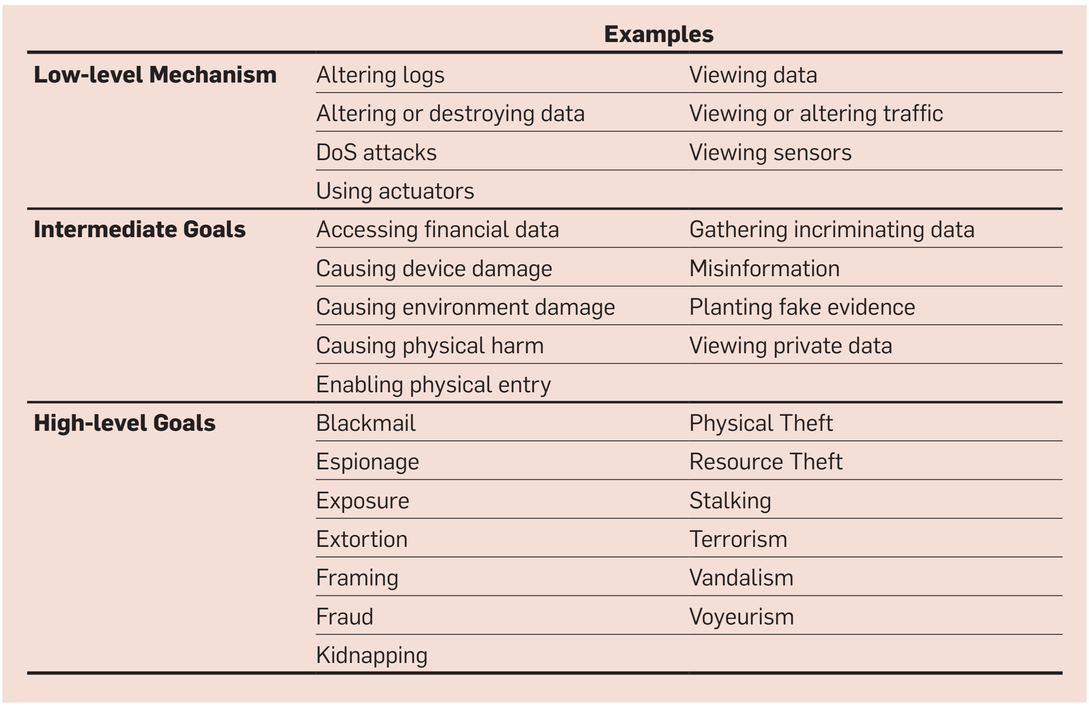
Smart appliances and their remote controlling enable new security threats. The technology itself is vulnerable but there are also indirect risks caused by remote operation of complex home appliances. Denning *et al.* [2013] have conducted an article of smart home computer security risks. Many of them are also applicable to remote control of complex home appliances. This section summarises them.

Smart devices can reveal whether a home is occupied or not. A device might provide status information, or an attacker can monitor communication to a specific device. Even if an attacker cannot break the security key of a wireless network he or she can draw conclusions based on device communication activity whether someone is at home or not.

If an attacker could break into the system, that person could cause distraction or financial loss. Financial loss could be caused by using heating system or air condition at full power when no one is at home in a form of heating or electrical bill. Any form of unauthorized use of appliances would scare residents when they would see appliances look like they would operate by themselves, when felons actually are operating them.

Vandalism could be caused also by denial of service or radio frequency interference. Engineers and designers should draw attention to making an implementation in which jamming the remote controlling part doesn't hang the whole device. So if an attacker performs denial of service attack to an AC unit, it would still accept commands from the device itself.

An attack could be targeted on a particular person or by trying to find any location with known software or hardware flaws by using technique called *wardriving*. If a particular person is chosen as a victim, the motive is usually a high-level goal. Table 9 summarizes risks with different goal levels.



	Examples	
Low-level Mechanism	Altering logs	Viewing data
	Altering or destroying data	Viewing or altering traffic
	DoS attacks	Viewing sensors
	Using actuators	
Intermediate Goals	Accessing financial data	Gathering incriminating data
	Causing device damage	Misinformation
	Causing environment damage	Planting fake evidence
	Causing physical harm	Viewing private data
	Enabling physical entry	
High-level Goals	Blackmail	Physical Theft
	Espionage	Resource Theft
	Exposure	Stalking
	Extortion	Terrorism
	Framing	Vandalism
	Fraud	Voyeurism
	Kidnapping	

Table 9 An overview of the structure of attacks to the home ecosystem [Denning *et al.*, 2013]

Controlling applications remotely allows attacks with physical consequences. In addition to data theft and harm, an attack could also change physical state of a system. In an example heating could be turned off completely during wintertime. This is a good note to take into account when designing requirements for access control, chosen encryption ways and level of control that permitted trough a remote connection.

6.2.3. Security goals

Complex appliance remote controlling system should be designed in a way that it tries to prevent human errors and devices are designed in a way that it guarantees acceptable security level. Denning *et al.* [2013]wrote an article that lists goals for smart devices:

1. Device privacy: a smart device should not broadcast its presence.
2. Device availability: a device should not able to made unavailable by an attack.
3. Device operability: a device should have protection against self-destructive or harmful operation.
4. Command authenticity: a device should only perform legit commands.
5. Execution integrity: a device should not operate in a way it was not specified to do.

[Denning *et al.*, 2013]

Having a security requirements and specifications is an important factor for complex appliance remote control. These goals mentioned earlier don’t cover the whole field, but instead they offer a good guideline and start point for creating requirements for such a system. Security matters in design are important as humans do make errors and there are security flaw exploiters out there.

7. Conclusions

Using tablet devices to control complex home appliances proved to be useful and implementable option for using appliances. There were many implementations in related studies that proved the technological feasibility of remote controlling. The remote controlling is easily implementable with current technology, but taking into account security and privacy is important.

The experiment conducted in this study found out that the tablet user interface could outperform its regular counterpart in many fields. But that was only with test users who had experience with mobile devices and tablet devices in general.

The Android application implementation supported fairly well selected usability heuristics. Most test users agreed it supported those heuristics, but there were also disagreements and in some heuristics the regular user interface performed better. But on average the Android user interface had better coverage of those heuristics.

One limitation of this study was the number of test participants, which was four. They were able to find 14 usability problems, but as Nielsen [1993]p ointed out more participants would give a more complete overview of possible problems. Having more participants would also cover more user types and thus give a broader coverage on the topic.

Another limitation was the implementation, which also had usability problems. Having an improved user interface and doing the tests again with that one would reduce user interface related problems and focus the results on differences between regular and tablet user interface. In similar way the machine user interface had even more usability problems. Improving and polishing both user interfaces would give more focused results of tablet user interfaces compared to regular user interfaces, and not test implementation's user interface usability faults against the ones of a commercial product.

There is potential for future research for doing such an experiment with improved user interfaces. Tests could be run with different appliances and with real functionality, so remote controlling with tablet would actually control an appliance. This study didn't concentrate in security and privacy matters, but if this remote controlling becomes more popular it's important to also study these matters to prevent misuse.

References

[Agar, 2003] Jon Agar, *Constant touch : a global history of the mobile phone.* Icon, 2003.

[Akyildiz and Vuran, 2010] Ian F. Akyildiz and Mehmet Can Vuran, *Wireless sensor networks.* 4, John Wiley & Sons, 2010.

[Android Open Source Project, 2015] Android Open Source Project, *Android Studio.* 23.7.2015, https://developer.android.com/sdk/index.html.

[Android.com, 2015] Android.com, *Android Design Principles.* 13.8.2015, http://developer.android.com/design/get-started/principles.html.

[Anwaarullah and Altaf, 2013] Syed Anwaarullah and S. V. Altaf, *RTOS based Home Automation System using Android.* International Journal of Advanced Trends in Computer Science and Engineering, 2, 480-484, 2013.

[Atzori *et al.*, 2010] Luigi Atzori, Antonio Iera and Giacomo Morabito, *The Internet of Things: A survey.* Computer Networks, 54, 2787-2805, 2010.

[Bertini *et al.*, 2006] Enrico Bertini, Silvia Gabrielli and Stephen Kimani, *Appropriating and assessing heuristics for mobile computing.* Proceedings of the working conference on Advanced visual interfaces, 119-126, 2006.

[Borgia, 2014] Eleonora Borgia, *The Internet of Things vision: Key features, applications and open issues.* Computer Communications, 54, 1-31, 2014.

[Bradley *et al.*, 2015] David Bradley, David Russell, Ian Ferguson, John Isaacs, Allan MacLeod and Roger White, *The Internet of Things – The future or the end of mechatronics.* Mechatronics, 27, 57-74, 2015.

[Can Filibeli *et al.*, 2007] M. Can Filibeli, Oznur Ozkasap and M. Reha Civanlar, *Embedded web server-based home appliance networks.* Journal of Network and Computer Applications, 30, 499-514, 2007.

[Cecere *et al.*, 2015] Grazia Cecere, Nicoletta Corrocher and Riccardo David Battaglia, *Innovation and competition in the smartphone industry: Is there a dominant design?* Telecommunications Policy, 39, 162-175, 2015.

[Chan *et al.*, 2008] Marie Chan, Daniel Estève, Christophe Escriba and Eric Campo, *A review of smart homes – Present state and future challenges.* Computer methods and programs in biomedicine, 91, 55-81, 2008.

[Chaouchi, 2010] Hakima Chaouchi, *Introduction to the Internet of Things.* The Internet of Things: Connecting Objects to the Web, 1-33, 2010.

[Chien *et al.*, 2014] Chen-Fu Chien, Kuo-Yi Lin and Annie Pei-I Yu, *User-experience of tablet operating system: An experimental investigation of Windows 8, iOS 6, and Android 4.2.* Computers & Industrial Engineering, 73, 75-84, 2014.

[Cockton, 2008] Gilbert Cockton, *Revisiting Usability's Three Key Principles.* CHI '08 Extended Abstracts on Human Factors in Computing Systems, 2473-2484, 2008.

[Cohen, 2007] Peter Cohen, *Macworld Expo Keynote Live Update.* 15.7.2015, http://www.macworld.com/article/1054764/liveupdate.html.

[Coskun *et al.*, 2013] Vedat Coskun, Busra Ozdenizci and Kerem Ok, *A survey on near field communication (NFC) technology.* Wireless personal communications, 71, 2259-2294, 2013.

[De Kock *et al.*, 2009] Estelle De Kock, Judy Van Biljon and Marco Pretorius, *Usability evaluation methods: Mind the gaps.* Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, 122-131, 2009.

[Denning *et al.*, 2013] Tamara Denning, Tadayoshi Kohno and Henry M. Levy, *Computer Security and the Modern Home.* Communications of the ACM, 56, 94-103, 2013.

[Derthick *et al.*, 2013] Katie Derthick, James Scott, Nicolas Villar and Christian Winkler, *Exploring Smartphone-based Web User Interfaces for Appliances.* Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services, 227-236, 2013.

[Dey *et al.*, 2011] Anind K. Dey, Katarzyna Wac, Denzil Ferreira, Kevin Tassini, Jin-Hyuk Hong and Julian Ramos, *Getting Closer: An Empirical Investigation of the Proximity of User to Their Smart Phones.* Proceedings of the 13th International Conference on Ubiquitous Computing, 163-172, 2011.

[Dillon, 2011] Roberto Dillon, *The golden age of Video Games.* 2011.

[Dixon *et al.*, 2010] Colin Dixon, Ratul Mahajan, Sharad Agarwal, A. J. Brush, Bongshin Lee, Stefan Saroiu and Victor Bahl, *The Home Needs an Operating System (and an App Store).* Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, 18:1-18:6, 2010.

[Dropwizard Team, 2015] Dropwizard Team, *Dropwizard.* 23.7.2015, http://www.dropwizard.io/.

[ElShafee and Hamed, 2012] Ahmed ElShafee and Karim Alaa Hamed, *Design and Implementation of a WiFi Based Home Automation System.* 6, 1852, 2012.

[Farley, 2005] Tom Farley, *Mobile telephone history.* Telektronikk, 101.3/4, 22, 2005.

[Fielding, 2000] Roy T. Fielding, *Architectural styles and the design of network-based software architectures.* University of California, Irvine, 2000.

[Fielding and Reschke, 2014] Roy Fielding and J. Reschke, *Hypertext Transfer Protocol (HTTP/1.1): Authentication.* 2014.

[Gale, 1996] Stephen Gale, *A Collaborative Approach to Developing Style Guides.* Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 362-367, 1996.

[Gelb and Gardiner, 1997] Janice Gelb and Jefferey J. Gardiner, *Developing a company style guide.* Annual Conference-Society For Technical Communication, 44, 469-472, 1997.

[Gerhardt‐Powals, 1996] Jill Gerhardt‐Powals, *Cognitive engineering principles for enhancing human‐computer performance.* International Journal of Human‐Computer Interaction, 8, 189-211, 1996.

[Goodwill Community Foundation, 2015] Goodwill Community Foundation, *Computer Basics: Mobile Devices.* 12.6.2015, http://www.gcflearnfree.org/computerbasics/9/full.

[Google, 2015] Google, *Android 5.0 Lollipop.* 23.7.2015, https://www.android.com/versions/lollipop-5-0/.

[Google design guidelines, 2015] Google design guidelines, *Accessibility - Usability.* 26.8.2015, https://www.google.com/design/spec/usability/accessibility.html#accessibility-guidance-feedback.

[Gould and Lewis, 1985] John D. Gould and Clayton Lewis, *Designing for Usability: Key Principles and What Designers Think.* Communications of the ACM, 28, 300-311, 1985.

[Gubbi *et al.*, 2013] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic and Marimuthu Palaniswami, *Internet of Things (IoT): A vision, architectural elements, and future directions.* Future Generation Computer Systems, 29, 1645-1660, 2013.

[Guinard *et al.*, 2011] Dominique Guinard, Vlad Trifa, Friedemann Mattern and Erik Wilde, *From the internet of things to the web of things: Resource-oriented architecture and best practices.* 97-129, 2011.

[Hsu *et al.*, 2010] Chun-Liang Hsu, Sheng-Yuan Yang and Wei-bin Wu, *3C intelligent home appliance control system – Example with refrigerator.* Expert Systems with Applications, 37, 4337-4349, 2010.

[Hvannberg *et al.*, 2007] Ebba Thora Hvannberg, Effie Lai-Chong Law and Marta Kristín Lérusdóttir, *Heuristic evaluation: Comparing ways of finding and reporting usability problems.* Interacting with Computers, 19, 225-240, 2007.

[Inostroza *et al.*, 2012] Rodolfo Inostroza, Cristian Rusu, Silvana Roncagliolo, Cristhy Jimenez and Virginica Rusu, *Usability Heuristics Validation through Empirical Evidences: A Touchscreen-Based Mobile Devices Proposal.* 31st International Conference of the Chilean Computer Science Society (SCCC), 60-68, 2012.

[ISACA, 2010] ISACA, *Securing Mobile Devices.* 2010.

[Jin-Shyan Lee *et al.*, 2007] Jin-Shyan Lee, Yu-Wei Su and Chung-Chou Shen, *A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi.* Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE, 46-51, 2007.

[Kakihara and Sørensen, 2002] Masao Kakihara and Carsten Sørensen, *Mobility: an extended perspective.* System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on, 1756-1766, 2002.

[Kim *et al.*, 2010] Tiffany H. Kim, Lujo Bauer, James Newsome, Adrian Perrig and Jesse Walker, *Challenges in Access Right Assignment for Secure Home Networks.* Proceedings of the 5th USENIX Conference on Hot Topics in Security, 1, 2010.

[Kumar, 2014] Shiu Kumar, *Ubiquitous smart home system using android application.* International Journal of Computer Networks & Communications, 6, 33-43, 2014.

[Kumar and Lee, 2014] Shiu Kumar and Seong R. Lee, *Android based smart home system with control via Bluetooth and internet connectivity.* Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on, 1-2, 2014.

[Lackner, 2013] Günther Lackner, *A Comparison of Security in Wireless Network Standards with a Focus on Bluetooth, WiFi and WiMAX.* IJ Network Security, 15, 420-436, 2013.

[Laugesen and Yuan, 2010] John Laugesen and Yufei Yuan, *What Factors Contributed to the Success of Apple's iPhone?* Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 91-99, 2010.

[Law and Hvannberg, 2004] Effie L. Law and Ebba T. Hvannberg, *Analysis of strategies for improving and estimating the effectiveness of heuristic evaluation.* Proceedings of the third Nordic conference on Human-computer interaction, 241-250, 2004.

[Lee *et al.*, 2006] Young Seok Lee, Sang W. Hong, Tonya L. Smith-Jackson, Maury A. Nussbaum and Kei Tomioka, *Systematic evaluation methodology for cell phone user interfaces.* Interacting with Computers, 18, 304-325, 2006.

[Levin, 2014] Michal Levin, *Designing Multi-device Experiences: An Ecosystem Approach to User Experiences Across Devices.* O'Reilly Media, 2014.

[Li and Wang, 2015] Yang Li and Yunliang Wang, *Design of control system of Smart Home based on embedded Linux.* 2015.

[Lischke *et al.*, 2015] Lars Lischke, Sven Mayer, Katrin Wolf, Alireza Sahami Shirazi and Niels Henze, *Subjective and Objective Effects of Tablet's Pixel Density.* Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, 2769-2772, 2015.

[Liu and Su, 2015] Chungui Liu and Xin Su, *Research and Design for Mobile Terminal-Based on Smart Home System.* Open Automation and Control Systems Journal, 7, 479-484, 2015.

[Lucas Jr. and Goh, 2009] Henry C. Lucas Jr. and Jie Mein Goh, *Disruptive technology: How Kodak missed the digital photography revolution.* The Journal of Strategic Information Systems, 18, 46-55, 2009.

[Machado Neto and Pimentel, 2013] Olibário Machado Neto and Maria d. G. Pimentel, *Heuristics for the assessment of interfaces of mobile devices.* Proceedings of the 19th Brazilian symposium on Multimedia and the web, 93-96, 2013.

[Macmillan Publishers Limited, 2015] Macmillan Publishers Limited, *Tablet computer – definition and synonyms.* 11.8.2015, http://www.macmillandictionary.com/dictionary/british/tablet#tablet__3.

[Mayer *et al.*, 2014] Simon Mayer, Andreas Tschofen, Anind K. Dey and Friedemann Mattern, *User interfaces for smart things – A generative approach with semantic interaction descriptions.* ACM Transactions on Computer-Human Interaction (TOCHI), 21, 12, 2014.

[Mazurek *et al.*, 2010] Michelle L. Mazurek, J. P. Arsenault, Joanna Bresee, Nitin Gupta, Iulia Ion, Christina Johns, Daniel Lee, Yuan Liang, Jenny Olsen, Brandon Salmon, Richard Shay, Kami Vaniea, Lujo Bauer, Lorrie F. Cranor, Gregory R. Ganger and Michael K. Reiter, *Access Control for Home Data Sharing: Attitudes, Needs and Practices.* Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 645-654, 2010.

[Molich and Nielsen, 1990] Rolf Molich and Jakob Nielsen, *Improving a Human-computer Dialogue.* Communications of the ACM, 33, 338-348, 1990.

[Mowad *et al.*, 2014] Mohamed Abd El-Latif Mowad, Ahmed Fathy and Ahmed Hafez, *Smart Home Automated Control System Using Android Application and Microcontroller.* International Journal of Scientific & Engineering Research, 5, 935-939, 2014.

[Myers, 2005] Brad A. Myers, *Using handhelds for wireless remote control of PCs and appliances.* Interacting with Computers, 17, 251-264, 2005.

[Neira *et al.*, 2013] Oscar Neira, Nieto L. Lee Angelica, Jose L. Martinez Lastra and Roberto S. Camp, *A builder for Adaptable Human Machine Interfaces for mobile devices.* Industrial Informatics (INDIN), 2013 11th IEEE International Conference on, 750-755, 2013.

[Nichols and Myers, 2003] Jeffrey Nichols and Brad A. Myers, *Studying the use of handhelds to control smart appliances.* Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on, 274-279, 2003.

[Nichols *et al.*, 2002] Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld and Mathilde Pignol, *Generating Remote Control Interfaces for Complex Appliances.* Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology, 161-170, 2002.

[Nielsen and Molich, 1990] Jakob Nielsen and Rolf Molich, *Heuristic evaluation of user interfaces.* Proceedings of the SIGCHI conference on Human factors in computing systems, 249-256, 1990.

[Nielsen, 1994] Jakob Nielsen, *Enhancing the Explanatory Power of Usability Heuristics.* Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 152-158, 1994.

[Nielsen, 1993] Jakob Nielsen, *Usability Engineering.* Morgan Kaufmann Publishers Inc, 1993.

[Nielsen.com, 2014] Nielsen.com, *Double Vision – Global Trends in Tablet and Smartphone Use While Watching TV.* 17.7.2015, http://www.nielsen.com/us/en/insights/news/2012/double-vision-global-trends-in-tablet-and-smartphone-use-while-watching-tv.html.

[Nilsson, 2009] Erik G. Nilsson, *Design patterns for user interface for mobile applications.* Advances in Engineering Software, 40, 1318-1328, 2009.

[Nummila, 2015] Ilari Nummila, *Discussion about suitable web frameworks for embedded Linux.* Personal communication, 2015.

[Panth and Jivani, 2013] Sharon Panth and Mahesh Jivani, *Home automation system (HAS) using android for mobile phone.* International Journal of Electronics and Computer Science Engineering ISSN-2277-1956, 2013.

[Park *et al.*, 2011] Wonkyu Park, Sung H. Han, Sungjin Kang, Yong S. Park and Jaemin Chun, *A factor combination approach to developing style guides for mobile phone user interface.* International Journal of Industrial Ergonomics, 41, 536-545, 2011.

[Pering *et al.*, 2006] Trevor Pering, Yuvraj Agarwal, Rajesh Gupta and Roy Want, *CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces.*

Proceedings of the 4th International Conference on Mobile Systems, Applications and Services, 220-232, 2006.

[Pieterse and Olivier, 2014] Heloise Pieterse and Martin S. Olivier, *Bluetooth Command and Control channel.* Computers & Security, 45, 75-83, 2014.

[Polsson, 2015] Ken Polsson, *Chronology of Handheld Computers.* 14.7.2015, http://handheldtimeline.info/.

[Quesenbery, 2001] Whitney Quesenbery, *Building a better style guide.* Proceedings of UPA, 2001.

[Ramlee *et al.*, 2013] Ridza A. Ramlee, Mohd A. Othman, M. H. Leong, Mohd M. Ismail and S. S. S. Ranjit, *Smart home system using android application.* Information and Communication Technology (ICoICT), 2013 International Conference of, 277-280, 2013.

[Raspberry Pi Foundation, 2015] Raspberry Pi Foundation, *The Raspberry Pi 2 Model B.* 23.7.2015, https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[Raspbian, 2015] Raspbian, *Raspbian – A free operating system based on Debian.* 16.9.2015, https://www.raspbian.org/.

[Reed *et al.*, 1999] Paul S. Reed, K. Holdaway, Scott Isensee, Elizabeth Buie, J. Fox, J. Williams and Arnold Lund, *User interface guidelines and standards: progress, issues, and prospects.* Interacting with Computers, 12, 119-142, 1999.

[RestApiTutorial.com, 2015] RestApiTutorial.com, *REST API Tutorial.* 23.7.2015, http://www.restapitutorial.com/.

[Riikonen *et al.*, 2013] Antti Riikonen, Timo Smura, Antero Kivi and Juuso Töyli, *Diffusion of mobile handset features: Analysis of turning points and stages.* Telecommunications Policy, 37, 563-572, 2013.

[Robles and Kim, 2010] Rosslin John Robles and Tai-hoon Kim, *Applications, systems and methods in smart home technology: a review.* 2010.

[Roduner *et al.*, 2007] Christof Roduner, Marc Langheinrich, Christian Floerkemeier and Beat Schwarzentrub, *Operating Appliances with Mobile Phones – Strengths and Limits of a Universal Interaction Device.* Pervasive Computing, 4480, 198-215, 2007.

[Rusko, 2014] Rauno Rusko, *Mapping the perspectives of coopetition and technology-based strategic networks: A case of smartphones.* Industrial Marketing Management, 43, 801-812, 2014.

[Siegenthaler *et al.*, 2011] Eva Siegenthaler, Pascal Wurtz, Per Bergamin and Rudolf Groner, *Comparing reading processes on e-ink displays and print.* Displays, 32, 268-273, 2011.

[Sohag and Ahamed, 2015] Md H. A. Sohag and Md A. Ahamed, *Smart Home Security System Based on Microcontroller Using Internet and Android Smartphone.* International Conference on Materials, Electronics & Information Engineering, ICMEIE-2015 4-8, 2015.

[Sony Mobile, 2015] Sony Mobile, *Sony Xperia™ Z2 tablet.* 23.7.2015, http://www.sonymobile.com/global-en/products/tablets/xperia-z2-tablet/specifications/.

[Stewart and Travis, 2003] Tom Stewart and David Travis, *The Human-computer Interaction Handbook.* 991-1005, 2003.

[Tseng *et al.*, 2014] Fang-Mei Tseng, Ya-Lin Liu and Hsiang-Hsun Wu, *Market penetration among competitive innovation products: The case of the Smartphone Operating System.* Journal of Engineering and Technology Management, 32, 40-59, 2014.

[Wauters, 2012] Robin Wauters, *Android Reaches 39% Tablet OS Market Share (Standing On Amazon's Shoulders).* 17.7.2015, http://techcrunch.com/2012/01/26/android-reaches-39-tablet-os-market-share-standing-on-amazons-shoulders/.

[Wee, 2003] Thomas Tan Tsu Wee, *Factors affecting new product adoption in the consumer electronics industry.* Singapore Management Review, 25, 51, 2003.

[West and Mace, 2010] Joel West and Michael Mace, *Browsing as the killer app: Explaining the rapid success of Apple's iPhone.* Telecommunications Policy, 34, 270-286, 2010.

[Wiegers and Beatty, 2013] Karl Wiegers and Joy Beatty, *Software requirements.* Pearson Education, 2013.

[Xie *et al.*, 2013] Lei Xie, Yafeng Yin, Xiang Lu, Bo Sheng and Sanglu Lu, *iFridge: An Intelligent Fridge for Food Management Based on RFID Technology.* Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication, 291-294, 2013.

[Yoneki, 2014] Eiko Yoneki, *Demo: RasPiNET: decentralised communication and sensing platform with satellite connectivity.* Proceedings of the 9th ACM MobiCom workshop on Challenged networks, 81-84, 2014.

[Yoo, 2010] Youngjin Yoo, *Computing in Everyday Life: A Call for Research on Experiential Computing.* MIS Quarterly, 34, 213-231, 2010.