RWTH AACHEN UNIVERSITY

Haptic Keyboard Prototype for Data Entry

Diploma Thesis at the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Alexander G. M. Hoffmann

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Ulrik Schroeder

Registration date: Nov 30th, 2007
Submission date: Mai 29th, 2008

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, May 29th, 2008

Contents

List of Figures

List of Tables

Abstract

Haptic input devices provide the user a tactile feedback. In virtual reality, medical testing scenarios, and games, tactile feedback is used to provide the user a more authentic experience. While the user is operating the system, the tactile feedback provides him additional information so that errors during the interaction with the system can be corrected. In word processing several error correction methods occur. Most methods intervene when the error was already done. Prevention methods limit the users' interaction and are only useful for special applications like the city selection menu in navigation systems. DEKtf (**D**ata **E**ntry on a **K**eypad with **t**actile **f**eedback) bridges the gap between tactile feedback and data entry error prevention methods.

This paper introduces an approach that utilizes tactile feedback to prevent errors during data entry. We have built a fully functional keypad that provides tactile feedback. Magnets control the pressure resistance of each key. The concept of DEKtf is to block the keys that would provoke a typing error or a misspelling. Depending on the system state, several blocking algorithms determine the blocking of a key. While a key is blocked, the user needs to apply more strength to overcome the resistance. That way the user feels the tactile feedback, can reconsider his action, and maybe prevent an error. A graphical user interface that reproduces the input and the blocking algorithms are provided by the DEKtf software, which is programmed with Swing, a widget toolkit for Java.

Aside from designing the prototype and implementing the corresponding software, we have ran several DIA (Design - Implementation - Analyze) cycles to improve our system. Furthermore, this thesis contains conduction and evaluation of three user tests. The results of the user studies showed that DEKtf can prevent about 30% of typing erros made by the participants. In date entry applications DEKtf bisected the task completion time of some users.

DEKtf can support beginners and advanced typists as well as elder and disabled users with an impairment of the motor nerves. The tactile feedback prevents the user from inadvertent pressing of neighboring or wrong keys.

Überblick

Haptische Eingabegeräte stellen dem Benutzer ein fühlbares Feedback zur Verfügung. In der virtuellen Realität, medizinischen Testszenarios und bei Spielen macht fühlbares Feedback diese Anwendungen authentischer. Während der Benutzer mit dem System arbeitet, stellt das fühlbare Feedback zusätzliche Informationen zur Verfügung, so dass Interaktionsfehler vermieden werden können. In der Textverarbeitung werden verschiedenste Korrekturmethoden angewendet. Die meisten Methoden greifen ein, wenn der Fehler bereits begangen wurde. Fehlervorbeugende Methoden schränken die Interaktionsmöglichkeiten des Benutzers ein und sind nur in speziellen Anwendungen, wie beim Stadteingabemenü in Navigationssystemen, brauchbar. DEKtf (**D**aten-**E**ingabe auf einen **K**eyboard mit **t**aktilem **F**eedback) schließt die Lücke zwischen fühlbarem Feedback und Korrekturmethoden bei Dateneingabefehlern.

Diese Arbeit ist ein Ansatz, der taktiles Feedback benutzt, um bei der Dateneingabe Fehler zu verhindern. Wir haben ein voll funktionstüchtiges Keypad konstruiert, dass dem Benutzer fühlbares Feedback bereitstellt. Magnete regulieren den Druckwiderstand der einzelnen Tasten. Das Konzept hinter DEKtf ist, Tasten zu blockieren, welche einen Eingabe- oder Rechtschreibfehler hervorrufen würden. Abhängig vom Systemzustand, bestimmen verschiedene Algorithmen das Blockieren der Tasten. Während eine Taste blockiert ist, muss der Benutzer mehr Kraft aufwenden um den Widerstand der Taste zu überwinden. Auf diese Weise fühlt der Benutzer das taktile Feedback und kann sein Handeln überdenken, vielleicht sogar den Fehler vermeiden. Die DEKtf Software stellt eine graphische Oberfläche bereit, die die Eingabe- und Blockier-Algorithmen reproduziert. Die graphische Oberfläche wurde in Swing, einem Widget-Toolkit für Java, programmiert.

Neben dem Design des Prototypen und der Programmierung der dazugehörigen Software, haben wir mehrfach den DIA (Design - Implementierung - Analyse) Zyklus zur Verbesserung unseres System durchlaufen. Des Weiteren beinhaltet diese Diplomarbeit die Durchführung und Auswertung von drei Benutzertests. Die Ergebnisse der Benutzerstudien zeigen, dass 30% der Tippfehler der Teilnehmer verhindert werden können. Einige Benutzer konnten bei einer Datumseingabean-

wendung mit der Hilfe von DEKtf die Durchführungszeit halbieren.

DEKtf kann Anfänger und fortgeschrittene Schreiber, sowie auch ältere und behinderte Benutzer mit motorischen Störungen unterstützen. Das taktile Feedback verhindert versehentliches Drücken benachbarter oder falscher Tasten.

Acknowledgements

I want to thank my supervisor, Daniel Spelmezan, for his help and advice throughout the development of this work. Many thanks to Prof. Dr. Jan Borchers for the thesis topic itself, the possibility to write my thesis abroad, and his several propositions for improvement.

Thanks to everybody at the Media Computing Group, especially I want to thank Adalbert Schanowski for his support during the implementation phase of this work, and Clarissa de Gavarelli for her help obtaining hardware.

Furthermore, many thanks to all participants of my user tests, especially to those who participated in two studies. Thanks for your time and explicit feedback!

Also, I want to thank Prof. Dr. Ulrik Schroeder for agreeing to be the second examiner.

Many thanks go to Bettina Giemsa, Andrew Marlow, and Stefan Ramroth for reading my thesis and to Markku Korsumäki for sharing his FT java package that saved me a lot of time.

I want to thank my family for their support. Especially I want to thank my father, Bernhard Hoffmann, for his support building the prototype and my mother, Gabriele Hoffmann, for the energy and motivation she gave me. You two allowed me to keep my focus on computer science during my whole studies.

Last but not least I want to thank my girlfriend Eliza for her constant patience and support, especially in the last months. Kocham cie.

Thank you!

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

DEFINITION/EXPLANATION:
Definitions or Explanations give a formal definition or statement information on a certain topic.

Definition:
Definition/Explanation

Source code and implementation symbols are written in typewriter-style text.

```
myClass
```

The plural "we" will be used throughout this thesis instead of the singular "I", even when referring to work that was primarily or solely done by the author. Unidentified third persons are always described in male form. This is only done for purposes of readability.

The whole thesis is written in American English.

Links to project sites or homepages of mentioned products and applications are shown in a footnote at the bottom of the appropriate page.

Chapter 1

Introduction

"You do not really understand something unless you can explain it to your grandmother."

—Albert Einstein

Input devices are all kinds of devices that provide data or control signals to a data processing unit, such as a computer. 20 years ago the standard input devices for human-computer-interaction (HCI) were the keyboard and the mouse. Beside imaging and audio input devices, the joystick and perhaps in special professions, such as graphic design or architecture, a graphic tablet completed the needs in the following years. Nowadays there is a great variety of input devices. To simplify work or to make computer interaction more attractive, more and more input devices for specific requirements have come to the market. A result of the ongoing research is the system:

"Data Entry on a Keypad with Tactile Feedback"
(DEKtf - pronounced "DEK - t - f").

1.1 Human Input Channels

A human working with a computer perceives information from three input channels:

- visual (to see)
- auditory (to hear)
- haptic (to feel)

Smell is still an underused sense in human-computer interaction [Kaye, 2004]. Interfaces that focus on the users' sense of taste are not sufficiently investigated.

Visual and auditory channel

The human vision is a highly complex activity and the primary source of human perception. The sense of hearing is often considered secondary to the sight [Dix et al., 1993]. The ear has more interesting features than only hearing: To localize the direction, to measure the distance, and even to imagine the source of the sound. We are not aware of it, because we are more concentrated on the visual input we perceive.

Haptic perception

The third sense that we will consider is touch or haptic perception. To give just a few examples: We need the touch to evaluate if things are cold or hot, not to break a glass that we are holding in our hands, and to feel the weight of an object. Dix et al. [1993] says that the skin contains three types of sensory receptors : The thermoreceptors that react on heat and cold, the nociceptors that respond to intensive pain and pressure, and the mechanoreceptors that respond to pressure (see figure 1.1). In this paper we exclusively focus on the mechanoreceptors.

Mechanoreceptors respond to pressure

There are two kinds of mechanoreceptors: The rapidly and slowly adapting mechanoreceptors [Goldstein, 1996]. Rapidly adapting mechanoreceptors react to stimuli as soon as the skin is touched or the pressure intensity changes. These receptors do not respond to continuous pressure. Slowly adapting mechanoreceptors respond to

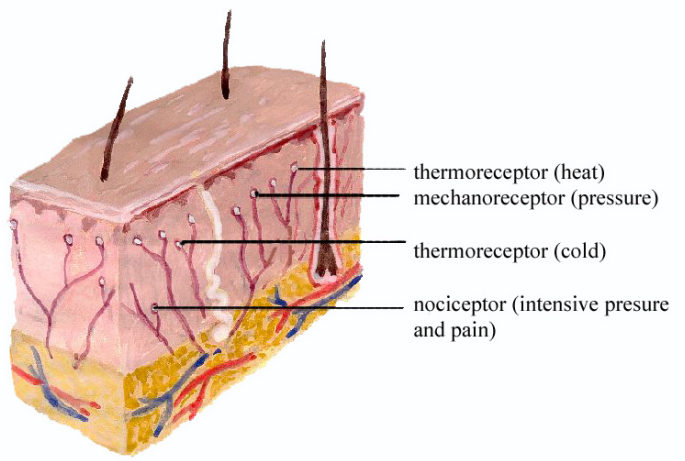

Figure 1.1: The Human Touch Sense - Receptors of the Skin ©www.airflag.com/Hirn/

continuous pressure. The touch sense of thumbs and fingers have the highest acuity compared to the rest of the body. Without receptors we are not be able to perceive tactile feedback from any of the input devices.

Finger coordination

Apart from the receptors the proprioception plays an important role during the interaction with a computer. Proprioception is the ability to sense the position, location, orientation, and movement of the body, in our case more precisely of the fingers. Touch typing on a keyboard needs a high accuracy of proprioception.

1.2 Input Devices

What is a computer that cannot be controlled? To interact with any data processing unit an input device is needed.

Definition:
Input Device

INPUT DEVICE:
An input device is "a method of activating or sending information to a computer or other electronic device. Keyboards, mice and trackballs are common computer input devices" (Write State University - Augmentative Communication Glossary)

Figure 1.2: Chord Key Set, Keyboard, and Mouse

Common input devices

Todays common input devices are microphones (audio), cameras (imaging/video), multi or single touch screens, graphic tablets, joysticks, game pads, scanners, mice and keyboards to name but a few.

Punch cards and tapes

The first computers were controlled by punch cards. Then punch tapes ("endless" punch card) were invented, where a keyboard wrote code on a tape. With the help of a tape reader the information was read out directly by the computer. After the introduction of keyboards the first new innovation was the mouse as shown in figure 1.2. This first mouse was presented by Douglas Engelbart on December 9th, 1968 together with a keyboard, and a chord key set with five piano-like keys [Engelbart, 1970].

Today's input devices include game controllers like the Wii Remote, 3D Mouse[1] (shown in figure 1.3) and, for example, bar code scanners in counter systems.

[1] http://www.3dconnexion.com/3dmouse/spacenavigator.php

Figure 1.3: Logitech 3dconnexion Product Family (©3Dconnexion)

1.2.1 Input Devices for Text/Data Entry

Alphanumeric Keyboard

Dix et al. [1993] mentions that the alphanumeric keyboard is still one of the most common input devices in the Western World. It is used for entering textual data, commands, and numerical input. Different kinds of keyboard layouts exist. QWERTY is the most prevalent keyboard layout in use (Christopher Sholes, 1868). It is named by the first six letters of the top row of alphabetical letters. A disadvantage for touch typer's using the QWERTY layout is that the fingers have to move over long distances, which is tiring for the operator and takes time. In addition, the most frequently used letters in the German and English language are typed by the left hand (located on the left half of the keyboard), though most people are right-handed (in Germany about 70% [Sattler, 1995]). Because of customs and the problems of incompatibility with software the QWERTY layout is still the most used and common layout [Buzing, 2003].

QWERTY keyboard layout

An alternative design is the Dvorak Simplified Keyboard (Dr. August Dvorak, 1936, figure 1.4), also known as "Simplified Keyboard". Dvorak's keyboard layout is sometimes used by computer programmers or other user's who do intensive writing. It is more ergonomic than the QWERTY layout, and the most used letters in the English language are placed on the home row, which are the certain keys

Dvorak keyboard layout

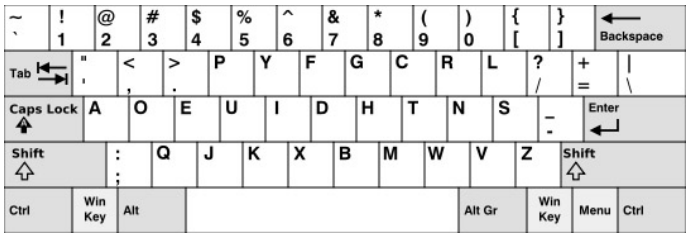

Figure 1.4: Dvorak Layout Keyboard

of the center row of alphabetical letters on a typewriter or computer keyboard. The possible writing speed is 4% faster compared to the QWERTY layout [West, 1998].

Chord Keyboards

High learning effort

Small in size

Chord keyboards require a higher learning effort than full size keyboards. They consist of 4–5 keys. Letters are generated by pressing one or more keys at the same time. The advantage is that chord keyboards can be very small, but are still good to handle. Additionally, they are essential for users, who have only one hand free to interact with a computer. A chord keyboard is shown on the left hand side in figure 1.2.

Phone Pads with T9

26 letters on 12 keys

T9 dictionary support

Today, mobile phones are used for much more than just calling. SMS and Internet are a big part and parcel of mobile device users. The problem is that normally only 12 keys (see figure 1.5) are available. The question is how to arrange the 26 letters of the alphabet plus additional characters on a phone pad in the best way. The first approach was to press a button multiple times in a short period to get the different letters, for example, 1x press = a, 2x press = b (Multi-Tap mode). Modes to change between numerical and alphabetical digits were introduced. This text input method was still too slow and needed improvement. One of the new ideas was T9[2] . It uses a dictionary to disambiguate words

[2] http://www.t9.com/

Figure 1.5: Nokia Phone Pad

by simply typing the relevant letter once (hello = 43556). In case there are words with the same letter code, for example, On = No = 66, the phone normally offers several options to choose from.

Handwritten Recognition

Handwritten recognition is the ability of a computer to receive and interpret handwritten text. The most known system using this technique is the palmtop (PDA, personal digital assistant). One text entry method is by recognizing drawn strokes. The users learn how to write strokes for all possible letters that are understood by the system. These are written using a stylus or pen on a small graphic tablet or the screen of the system. One idea is to write a text like on paper, and the computer interprets the text and digitize it. The current techniques of text recognition are fairly inaccurate, so there is still no satisfactory text entry method for small mobile devices available today.

Recognizing drawn strokes

Speech

Speech recognition is a promising domain of text entry. The first trials were undertaken by Bolt [1980] at MIT with the system "put that there". The system had a vocabulary of around 120 words that could be combined to change loca-

Limited vocabulary

tions and colors of shapes on a big screen. We can imagine that an extended version would lead to be really natural such as the Knowledge Navigator [Sculley and Byrne, 1987]. Indeed such a system got partly realized, for example, Microsoft Surface, but the impressive speech input feature is still missing. The best known commercial end user product for speech to text recognition was IBM via Voice but it never established itself on the market, since the error rate was too high.

Research in progress

1.3 Tactile Feedback on Input Devices

Interacting with a computer provides the user with various kinds of feedback. The visual feedback is the most frequently utilized one. Every input can be followed on the screen. Warnings and reminders can pop up to attract the attention of the user. Problems occur when the user's locus of attention [Raskin, 2000] gets distracted by a different area or is even away from the screen. Beside the visual feedback given by the monitor, LEDs are another way to provide feedback. The caps lock key is a well known example. A LED informs the user if the caps lock mode is on or off. In case the caps lock key is pressed accidentally, the first feedback will be perceived from the screen and not by the LED, because capital letters appear. The advantage of LEDs is controlling a system state or a mode. To give the user an alert or advice, the feedback of LEDs is not strong enough and can be ignored too easily.

Visual feedback

Auditory feedback has the advantage that no visual contact is necessary. It gets the attention of the user, even if he is distracted. Furthermore, it needs no screen space, and can reach even an entire group of people. In alarm and security systems, audio signals are the first choice. The disadvantage of auditory feedback is that it is undirected. The user knows something is wrong but not where. Additionally, it can bother people within earshot and annoy the user.

Auditory feedback

The developers have to economize using sounds. Exceeding auditory feedback leads the user to ignore it and turn the sound of the computer off. Another fact is that auditory

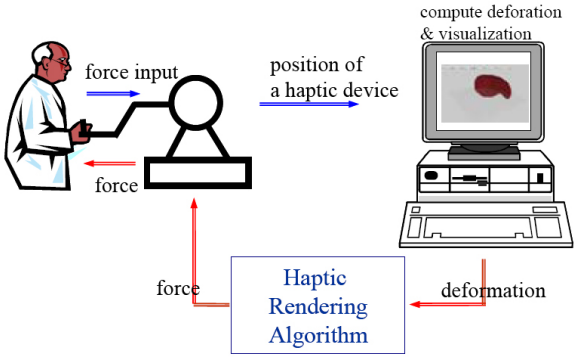

Figure 1.6: Diagram of a Simulator with Haptic Information [Kim et al., 2004]

feedback leads to confusion when the user receives two or more audio signals at once. Moreover, he might be not able to distinguishes between them. A good example is the ringing of two phones in a bus at the same time. Often the user is not able to recognize his own ring tone.

The third and last of the feedbacks we will consider is the haptic feedback. Many different terms with many different definitions are used throughout the literature to describe haptic interaction [Oakley et al., 2000]. Oakley defines:

Haptic feedback

Haptic	Relating to the sense of touch
Tactile	Pertaining to the cutaneous sense but more specifically the sensation of pressure rather than temperature or pain.

Table 1.1: Oakley et al. [2000]: Definition of Haptic and Tactile

The "feel" is often seen as less important although for the most Virtual Reality (VR) and medical applications it is essential. For human controlled medical robots it is fundamental that they pass the feedback they perceive to the operator. Without tactile feedback a surgeon cannot operate precisely and successfully with a robot (figure 1.6).

Haptic feedback applications

Tactile feedback combines some of the advantages of visual and auditory feedback. It can reach the user, even when not concentrating on the screen, and it does not bother people

Haptic for authentic experience

within earshot. In VR environments and prototyping applications, haptic feedback is often used to provide a more authentic experience.

Avoiding spelling errors with tactile feedback

Tactile feedback could probably avoid spelling errors. A key, which does not make sense for a dictionary, could change his resistance to maximum. This could avoid errors even before they occur.

Changing the key resistance

Another application, where tactile feedback probably could support the user, is to avoid pressing keys that do not make sense as input in the current state. The most current systems provide auditory feedback on wrong data input to the user. Making a key, which does not make sense in the current situation, harder to press, probably could avoid this mistake. On the question if the user wants to print something—a receipt, for example—he should only have the choice between "Yes" or "No". Other possibilities such as "Large" or "Small" would not make sense in the context (figure 1.7). This approach works similar to graying out menu items of GUIs. To keep the menu consistent, items that make no sense in the current system state become transparent and remain in the same position.

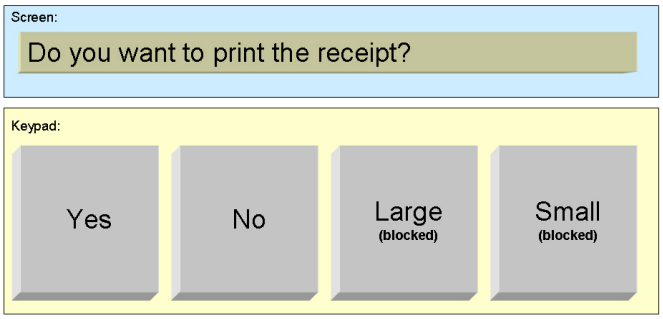

Figure 1.7: Only Useful Keys can be Pressed

Tactile feedback supports user to work correctly

Tactile feedback can provide a more realistic feeling to the user. Like in real life, the user could react on perceived information. It could support the user to work more correctly and efficiently. Tactile solutions are imaginable for keyboards. Tactile feedback means to feel, for example, a vibration or a change in pressure during typing. This is were this thesis takes the idea on.

Figure 1.8: Novint Falcon ©Novint Technologies 2007

1.3.1 Tactile Input Device: The Novint Falcon

Force Feedback joystick

Some recent input devices already offer tactile feedback, such as the Force Feedback joystick (Immersion, 1997). The joystick tries to make the controlled situation more realistic. Vibration motors give the user a more authentic feeling. Another approach providing tactile feedback to feel surfaces of different materials is the Novint Falcon[3] (figure 1.8). The Falcon provides feedback like a force feedback joystick, but additionally the weight of an item and the surface of materials can be felt.

Feel the weight and the surface of items

1.3.2 Tactile Input Device: The Feel Mouse

This mouse is a standard two button mouse. The anchor of a magnet is mechanically attached to a mouse button as illustrated in figure 1.9. With the help of this magnet, the user can "feel" the surface of the desktop. Generally each pixel of the desktop could get a feel value. The mouse button for example can move up or down depending on the 2D vector of the surface. Above an icon the button goes up, on the background the icon stays down. The goal is to give the 2D display an additional dimension of touch [Penz and Tscheligi, 1993].

Monitor pixel gets feel value

[3] http://home.novint.com/products/novint_falcon.php

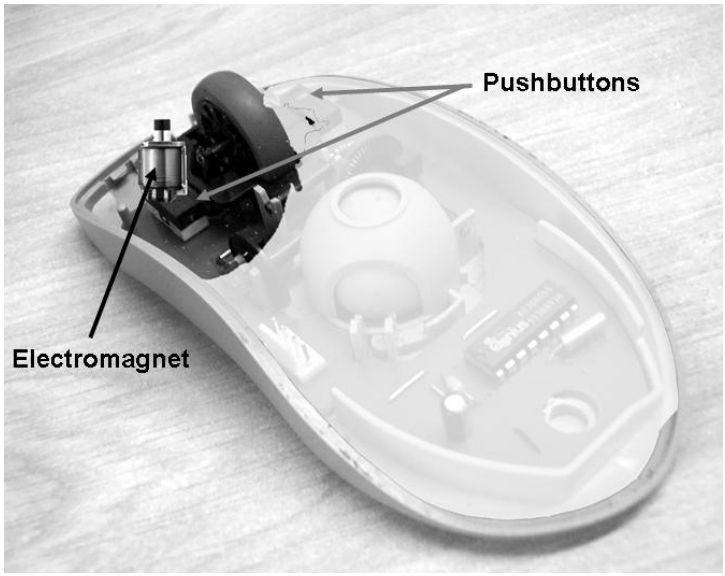

Figure 1.9: Feelmouse Sketch

1.3.3 Tactile Input Device: The Force Mouse

Münch and Dillmann [1997] built a specialized 'multimodal mouse' with a movable pin in the left button and two electromagnets in its base. On an iron mouse pad the electromagnets do not only provide a more difficult movement of the mouse, the second one has enough force to make the mouse irremovable. This would make sense, for example, when the user reaches the monitor limitations. Similar to the feel mouse (Section 1.3.2) the mouse button gives the user an impression of the surface of the desktop.

1.4 Error Detection and Live Correction

Classification

To handle typing errors we define three classifications. Prevention, live correction, and aftercare. We can inform the user before making an error, correct him directly during writing, or mark the possible error to allow the user to correct it afterwards.

Aftercare

Aftercare is the most known and recognized error correction method. Like in common text editors or e-mail programs, possible errors are underlined or marked. They can be corrected manually or with the help of a spell checker.

Live correction

Live correction is getting more and more popular. Long established editors like Microsoft Word, as well as novel systems like the Apple iPhone and the XT9 [Nuance, 2007] adopt live correction algorithms. The common spelling correction algorithms appear to cope reasonably well with a number of common sources of misspellings, including transposed letters ("teh" instead of "the"); missing letters ("tomorrw" instead of "tomorrow"); the 'wrong' letter with a particular phonetic value ("espana" instead of "españa"), or a combination of these mistakes. Even the so called "Sloppy Type regional error correction" allows correction of wrong typing caused by the position of the keys ('o' is next to 'p' and 'u' next to 'i': "ouzza" instead of "pizza").

Error prevention

The prevention method is the strictest one. It does not allow the user to make any errors. However, this also means that the system is always right and the user has only limited options for interaction. This method clearly avoids errors but the accuracy of the system depends on the completeness of the database. The Look-Ahead-Function, like the city select list in navigation systems (see Fig. 1.10), supports the user to avoid errors and to accelerate the data entry. The algorithm hides all letters that make no sense, because in the database there is no city name starting with this letter combination.

1.5 Goals

Inspiration for my thesis

The inspiration to write this paper came from a professor who always accidentally pressed the caps lock key on his keyboard. There upon we asked ourself how we can prevent from pressing unwanted keys during data entry. The idea was to design a keyboard with variable key resistance. For example, keys like the caps lock key could have a higher resistance than other keys.

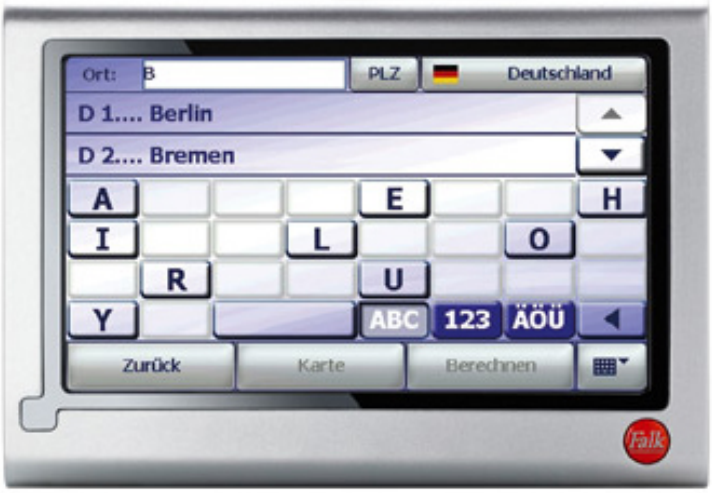

Figure 1.10: Falk Navigation Systems - Look Ahead Function ©Falk

Research questions

The main goal of this thesis is to find out how usable is a keypad prototype that provides this kind of tactile feedback. We want to identify problems users may have during entering data and find out if they can avoid them with the assistance of DEKtf. Questions we would like to answer by means of this research project are:

What possibilities do we have to provide tactile feedback with a keyboard?

Is the build prototype a good approach?

Does a system like DEKtf avoid data entry errors?

Do we consider the user's needs?

Did we choose a good application domain?

Does this facilitate the user's work?

Is it easy to learn and use the system?

What does the user think about the system - what is the level of user acceptance?

1.6 Thesis Structure

To explore a new system like DEKtf it was necessary to familiarize ourselves with physics and electromagnetism, haptics, and error prevention methods. We introduced ourself to the basics of these areas and developed a new input device. Research that had already been done in these fields were compared to our approach. We designed a hardware prototype and the associated software, implemented it and analyzed the discovered results. This DIA cycle (Design - Implement - Analyze) we passed several times to improve our system. We ran three user tests during the analyzing phases and evaluated them. In the final stages the work is outlined and the results are summarized.

The thesis has the following structure:

- In *Related work*, we describe research on existing keyboard modification, tactile feedback and text error prevention.
- In *DEKtf - Haptic Keyboard Prototype for Data Entry*, design decisions for the keyboard prototype, the software, and their reasoning are presented. We describe the way we realized the prototype, and implementation details describe the code of the interface.
- *Evaluation* describes user observations designed to the user tests, and the evaluation of this results. We chart the questionnaires and controlled if we accomplished our design goal of creating a representative user study about the ease of use of tactile feedback in data entry.
- In *Summary and Future Work*, we summarize our conclusions, and present ideas for further improvement of the work.

Chapter 2

Related Work

"I don't understand the technology. But you don't have to. You have to understand what it can do for you."

—Rupert Murdoch

In the first chapter we introduced general terms connected to input devices, tactile feedback, and error detection methods. In this chapter we present workings, which are related to these topics. Also, we discuss how we could apply some of these research findings for the tactile feedback keypad. We could not discover a similar system like DEKtf so we had to concentrate on papers related to the different fields where DEKtf is involved (input devices, tactile feedback, and error prevention methods).

Keyboard and mouse

Most PC (personal computer) users interact with keyboard and mouse. No major changes have been made in the standard keyboard layout for the past 25 years (compare figures 2.1 and 2.2). The computer mouse improved ergonomically since the first presentation by Engelbart in 1968. From a simple wooden box it turned into a high technology, ergonomic, hand-shaped input device. Apple applied the idea of a mouse with just one button, but it never really established itself on the whole market.

Figure 2.1: Keyboard: IBM PC Keyboard 1981 ©vintage-computer.com

Figure 2.2: Keyboard: Cherry Keyboard 2008

Haptic mice

Studies, which deal with the improve of the mouse by using tactile feedback, have already been mentioned in subsection 1.3.2 (Feel Mouse) and 1.3.3 (Force Mouse). Logitech's iFeel Mouse was another approach. In cooperation with Immersion's Touchware[1] a vibration motor was build in a standard Logitech mouse to provide tactile feedback.

Tactile feedback keyboard

A tactile feedback keyboard has not been realized yet. A patent by Goodwin et al. [2001] exists that describes a similar hardware approach. However, we could not find any reference that the keyboard has been built yet.

[1] http://www.immersion.com/

Figure 2.3: DAS Keyboard ©DASKeyboard.com

2.1 Existing Keyboard Modifications

Changing positions of the function keys and specially added buttons by companies, for example, the Apple and the Windows keys, are common keyboard variations. More fundamental modifications are presented in this section.

2.1.1 DAS Keyboard

The idea of the "DAS Keyboard" (figure 2.3) based on the observation that users, even when they are proficient typists, spend some time looking at the keys. On this keyboard the keys are blank, and the user learns quite fast to adjust to this circumstance. The manufacturer claims that some users, forced to memorize key positions, can type twice as fast within a few weeks [Zipern, 2005]. Additionally, to be more responsive the auditive feedback (i.e. the click) is louder than from other membrane keyboards. The key resistance is depending on the finger the key is typed by. The ring finger has to afford less force than, for example, the index finger.

Blanc keyboard

2.1.2 Optimus Maximus Keyboard

The Optimus Maximus Keyboard (figure 2.4) is composed of small displays embedded in every keytop. This screens

Figure 2.4: Optimus Maximus Keyboard ©Art. Lebedev Studio

Small displays as keys

display the letters or the function the key is associated with. The keyboard is arbitrary configurable. All language keyboard layouts that are supported by the operating system can be displayed. Additionally, the use as a "short cut keyboard" for applications like Photoshop, AutoCAD, or games like Half Life is possible. (Demo)[2] .

2.2 Tactile Feedback

2.2.1 User Evaluation with a Key-Press Simulator

Virtual push-buttons with tactile feedback

Doerrer and Werthschuetzky [2002] stated in their paper that touch screens replace more and more common input devices. Virtual push-buttons are freely configurable in number, size, and appearance. One disadvantage is the missing tactile feedback that push-buttons naturally would provide. The idea is a new approach that combines the flexibility of screens with the behavior of push-buttons. An arbitrary number of push-buttons is arranged like a mosaic. Each push-button can be pressed down with a

[2] http://www.artlebedev.com/everything/optimus/demo/

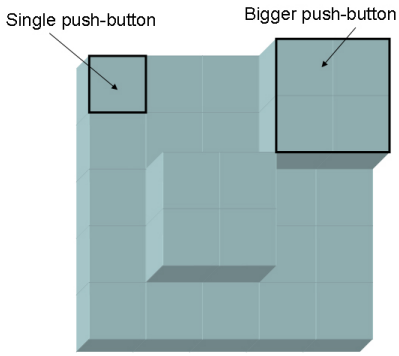

Figure 2.5: Sketch of a Haptic Display with buttons variable in number and size

programmable resistance. In case bigger or fewer push-buttons are needed, smaller ones can be grouped together to form a new, bigger button (see figure 2.5).

Interesting about this approach is the variable resistance of the buttons. Doerrer et al. found out that a force of at least 1,5N is required for an unique tactile feedback push-button to satisfy most users' expectations. Theses ideas, and the use of actuators to manipulate the press-resistance, is relevant for our work.

1,5N push-button resistance

2.2.2 Studies about Stiffness and Dumping Characteristics of a Computer Keyboard

Nagurka et al. [1999] developed a more complete understanding of the tactile "feel" of the computer keyboard. With the help of a computer-controlled test installation they measured and analyzed the key displacement, typing speed, and contact force. They compared spring-loaded and rubber-dome computer keyboards. The results showed that some floating, so-called dumping, is desirable to avoid oscillations of the keys. Too much floating handicaps the advanced typist because too much pressure has to be applied. Interesting for our approach is the fact that the stiffness of a keyboard should be between 2.00mm and 4.00mm, and the force point between 0,25N and 1,5N.

Stiffness and dumping desirable

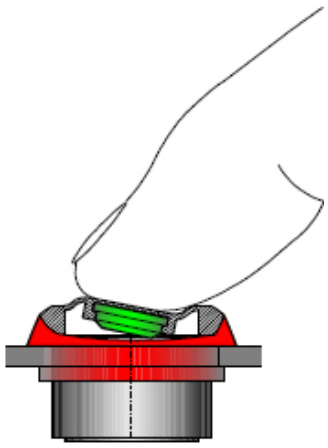

Figure 2.6: Haptic Solution for Disabled - Tilt of the Motor Surface

2.2.3 Missing Feedback for Keystrokes

Increase typing errors

Rabin and Gordon [April 2004] found out that the number of typing errors will increase if the users do not receive tactile feedback while typing. They tested professional typists on a keyboard. Those participants had no visual contact to the keys. First they were typing under normal conditions, then later one fingertip was anesthetized. Typing errors of that finger increased sevenfold. On this account we increased the common tactile feedback of keys through intelligent handling, to see if we even might decrease typing errors.

2.3 Tactile Feedback Devices

2.3.1 HSD - Haptic Solution for Disabled

Motors that tilt in all directions

CompuTouchAS [2002] idea works with a small tactile motor like the one shown in figure 2.6. Because of its small size and weight it is usable in various applications such as the integration in a mouse button, a keyboard key, or any other input device that is controlled by the fingertips. The

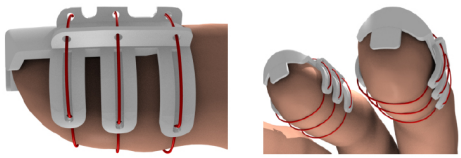

Figure 2.7: Schematic View of a Thimble ©ART

motor receives input from a driver electronic unit. The upper part can tilt in all directions. A fingertip on the motor allows the user to feel different surfaces or system states. CompuTouch believes a "haptic alphabet", which contains a set of unique movement patterns, can be constructed. Different patterns like for text, figures, areas without useful information, and point-and-click areas can be included in the alphabet. Thus, the user could feel the areas shown on the screen. Interacting with a computer in such a manner opens up a range of new possibilities.

Haptic alphabet

A problem mentioned by the author is that a fingertip has only 100 nerve-ends per square centimeter. This means that two stimuli that are only separated by 1mm still can be distinguished [Johnson and Phillips, 1981]. Compared to the eye, which can differ 10 to 100 times more accurate, the "resolution" of the fingertips is lower. Zooming is one way to cope with the lower resolution of the fingertip. Splitting the screen in 9 segments, each one again splittable in 9 new segments, gets the resolution more accurate. That way the tactile system is able to provide a more detailed output to the user. After the blind people studied the alphabet (patterns), HSD would allow them to read text, numbers, and even to interact with the PC.

Resolution of the eye is 10 times higher than of the finger

2.3.2 Tactile Feedback at the Fingertips

Scheibe et al. [2007] present a new tactile feedback system for finger-based interactions in virtual reality applications. The system consists of tracked thimbles for the fingers with thin shape memory alloy wires wound around each thimble (see figure 2.7). To realize the tactile feedback, the wires

Wire around thimble provide tactile feedback

can be shortened by slightly heating them up. The users preferred to work with the tactile system compared to a VR system without any feedback. They claimed that the feeling was quite similar to what is felt when touching objects in reality. To use this system to prevent typing errors, the system would need to localize the finger position. In case the user wants to press the wrong key, tactile feedback would have to appear shortly before pressing the key. One idea to locate the finger position is using RFID Sensors in the key, and the finger thimble.

Flexible in shape and touch

In their work Michelitsch et al. [2004] described the ultimate version of the Haptic Chameleon - an input device flexible in shape and touch (shown in figure 2.8). The changing shape can mimic, for example, real world objects, selection devices, or system states. Users can interact with the device without looking at it. The tactile feedback provides an additional communication channel to the user. However, Michelitsch et al. are more focused on the shape changing functionality.

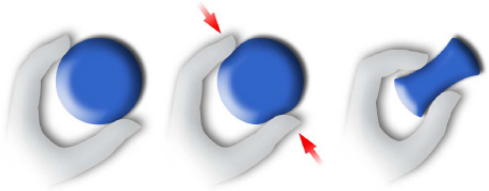

Figure 2.8: Concept of the Haptic Chameleon ©Dr. Stefan Rapp

2.3.3 Tactile Feedback in Mobile Devices

Change the shape of buttons

Hemmert et al. [2008] developed a mobile hardware prototype using a dynamic knob as an interaction device for the user. The knob changes the shape depending on a system event or a system state. The user can feel, for example, incoming or missed calls just by touching the phone. To realize the feedback Hemmert et al. used a mini servo motor controlled by the Arduino board [Banzi et al., 2005]. Keeping its state without spending energy is one advantage of a servo motor.

Hoggan et al. [2008] deals with the problem of missing tactile feedback of touchscreen keyboards. They ran tests with a physical keyboard, a touchscreen, and a touchscreen with tactile feedback. The Samsung i718 phone, which contains a Samsung Electro-Mechanics Linear Resonant Actuator and Immersion VibeTonz technology (www.immersion.com) to controll the actuator and to provide tactile feedback, was chosen for those tests. The comparison showed that tactile feedback improves fingertip interaction and performance on mobile devices with touchscreen.

Tactile feedback on touchscreens

Nokia's Haptikos (release date unknown) and iPhone-Haptics [Computing Science Department of the University of Glasgow, 2008] are research projects that apply tactile feedback to the touchscreen keyboards of the Nokia S60 and the Apple iPhone.

2.4 Data Entry Error Prevention

There are different error prevention methods on the market today. An overview of the different methods was given in section 1.4.

After T9, Nuance [2007] introduced an expanded version: XT9. Here, the input devices can change between soft and full hard qwerty keyboard, handwriting, and 12-key phone pad. Included features related to error prevention are: sloppy type regional error correction, spelling correction, and spell checking. The system is available in more than 65 languages.

XT9 with automatic error correction method

Apple had some other ideas to prevent errors. The new Apple Keyboard[3] has a hardware side programmed delay on the caps lock key. To enable the caps lock mode the caps lock key has to be pressed a little longer than other keys. Besides, if the caps lock is already engaged, the keystroke will be registered immediately, even before the upstroke. One disadvantage of this innovation is that this behavior is not changeable. For example, when remapping the caps

Delay of keystrokes

Not modifiable

[3] http://www.apple.com/keyboard/

Figure 2.9: iPhone - Catchment Area

lock key for another function this behavior is not always desirable by the user. Furthermore, the delay is programmed on the hardware of the keyboard (firmware). A better approach could be a solution like the one that has been applied on the eject key. A delay is programmed by software and can be modified if needed.

Enlarged catchment area

Another novelty is the keyboard of the Apple iPhone[4] . The iPhone checks the existing words in its dictionary, and when a key makes no sense for the dictionary but the key next to it does, then the this key has a bigger virtual catchment area (see figure 2.9). Furthermore, Sloppy Type regional error correction, spelling correction, and spell checking are included.

[4]http://www.apple.com/iphone/gettingstarted/keyboard.html

Microsoft Word[5] applies error correction in the following way. It has a list of common errors. These kinds of errors, for example, "teh" instead of "the", are corrected automatically during writing. Detecting that the caps lock key is activated by accident, is corrected by another changing function ("pETER" instead of "Peter"). Other auto correction mechanisms are: correcting double capital letter at the beginning of words, new phrases have to start with a capital letter, and weekdays have to be written with a capital letter first.

List of common errors

Auto correction mechanisms

Apart from the "Caps-Lock-Delay" of the Apple Keyboard, which is maybe too limited and unchangeable, none of the presented system uses hardware-based error correction. *DEKtf* creates a new field of typing-error detection.

2.5 Review

The research we presented started with basic experiments about the tactile recognition of the human. These papers provided us useful informations that we were able to build an ergonomic keypad. The presented tactile feedback devices are systems that give the desktop an additional dimension, or provide a more authentic experience for the user. However, it was not the aim of these systems to avoid typing errors. The data entry error prevention methods are software based. They work out for the most input methods and support the user to prevent typing errors. In contrary, they do not use the input devices actively to prevent errors, and they only offer visual feedback to the user. With DEKtf we want to close the gap between tactile devices and data entry error correction. We provide a high-fidelity prototype that fulfill these requirements.

DEKtf combines tactile feedback with error prevention

[5] http://office.microsoft.com/en-us/word/

System	Relation to our work
User Evaluation with a Key-Press Simulator	To perceive tactile feedback the required pressure has to change for more than 1,5N
Studies about Stiffness and Dumping Characteristics of a Computer Keyboard	Stiffness (2-4mm) and standard key resistance (0,25-1,5N) of the key.
Missing Feedback for Keystrokes	Studies about the increasing error rate when no tactile feedback is available. The intensity of the tactile feedback could be varied during typing to decrease errors.
HSD - Haptic Solution for Disabled	Feedback not strong enough to prevent errors.
Tactile Feedback at the Fingertips	Interesting approach if finger position can be localized (RFID)
Tactile Feedback in Mobile Devices	Idea how to apply DEKtf to touchscreen devices. Error prevention again difficult because the key is already pressed when user receives feedback.

Table 2.1: Tactile Feedback Systems

System	How does it work	Missing function
XT9	Software based	DEKtf could prevent errors hardware-based
Apple Keyboard	Firmware	DEKtf uses Hardware, but controlled by software (reprogrammable)
Apple iPhone	Software based	Missing tactile feedback
Microsoft Word	Software based	Missing tactile feedback

Table 2.2: Error Prevention Methods

Chapter 3

DEKtf - Data Entry on a Keypad with Tactile Feedback

"I find out what the world needs. Then I go ahead and try to invent it."

—Thomas Edison

The last chapter gave us an overview of research, which has already been done in the area of input devices, input devices with tactile feedback, and error prevention methods during text typing. It is out of question that tactile feedback provides great support for the user and that already a lot of research has been done in the field of haptics. Since more and more paperwork is written digitally, text entry support and automatic error prevention methods are essential. The same applies to mobile phones, where text typing, for example, writing e-mails, creating calendar entries, and sending SMS is common theses days.

Changeable key resistance

The aim of this paper is to build a keypad prototype with keys that are able to change their resistance. Furthermore, we evaluate the pros and cons of using this tactile feedback as an error prevention method.

3.1 Alternative Ideas for the Prototype Construction

Fully functional keypad

Starting this thesis made it necessary to build a prototype that allows us to conduct a high quality user study. We constructed a fully functional keypad with the specifications of variable tactile feedback. We wanted to be able to increase the resistance of the keys to a certain level, which blocks them. We started to think about the possibilities to realize tactile feedback for a keyboard. Implementing controllable pressure strength of the different keys was a priority.

Bi-metallic strips

The first idea was to work with bi-metallic strips. Two metals with different coefficients of linear thermal expansion are connected. When heating the system the strip will bend, when cooling it below its normal temperature it will bend in the opposite direction. We could put this strip under a push-button to provide different pressure points. The difficulty of this system is to heat or cool the system without spending too much energy. Additionally, the user should not feel the temperature change.

Hydraulic

The car industry gave the next impulse with its use of hydraulic or pneumatic shock absorbers. Nowadays even magnetorheological fluid [Chouvardas et al., 2005] is used for this field of activity. The problem about this development is that miniature hydraulic shock absorbers are not available on the market, and the production cost would exceed the available budget for this project. We ran some tests with magnetorheological fluid. Unfortunately, a magnet with sufficient strength to change the consistency of the fluid consumes too much energy.

Magnetorheological fluid

Adjustable springs

The next idea was inspired by the automotive industry as well. We were thinking about using adjustable springs. A motor could change the base of the spring to make it harder/easier to press. However, we doubted that the motors would react fast enough and think they would break too easily.

We decided to work with magnetism. The first challenge was that we did not find inductors that were strong enough

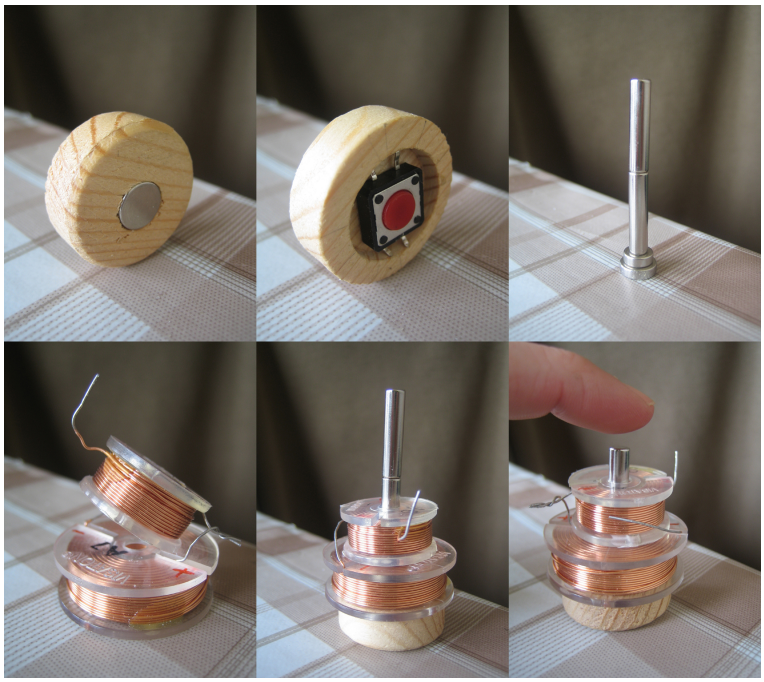

Figure 3.1: First Test with Two Inductors and Four Magnets

Inductors

for our requirements. Therefore, we created our own one. However, this inductor was unusable, because the distances between the turns of the wire were too big. Thus, we could not achieve enough turns with the small size we required, to get the magnetism strong enough. In addition, the inductor got too hot after short time of operation. We came across with inductors made for loud speakers. These can provide sufficient power.

High operating temperature

First experiments

For the first test we used two of these inductors and four magnets as shown in figure 3.1. This system worked out however, the test showed us that a full activation of all keys of the keypad would consume too much energy (approximately 20A). In permanent operation the inductor got immoderately hot. An advantage is that this system could provide a constant resistance from the first contact up to the end of the key press. A complete blocking of the key would still be hard to realize.

Constant resistance, complete blocking not realizable

Finally we decided to work with small electromagnets such as Doerrer and Werthschuetzky [2002] used, who built a

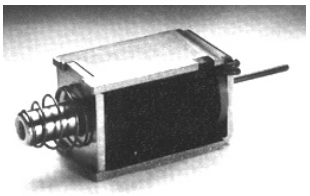

Figure 3.2: Solenoid Series 44A

Solenoids fulfill requirements

key-press simulator. We used existing lifting magnets or so-called solenoids (Solenoid Series 44A, figure 3.2).

Definition: *Solenoid actuator*

SOLENOID ACTUATOR:
The solenoid usually refers to a tube like coil only. When current passes through the solenoid, it generates a magnetic field around it. The magnetic field inside is much larger than it is outside, and as a result considerable magnetic energy is stored in the interior. If a bar of permeable material (plunger) is placed at one end of the solenoid, it will be drawn into the solenoid as the magnetic circuit will try to reduce the reluctance, which is mostly made up by the air [Rashedin and Meydan, 2005].

3.2 Building the Prototype

12 button keypad

The aim of this work was to check if DEKtf can avoid errors. For this reason and because of the limited available time we decided to construct only a 12 button prototype keypad with 10 number keys and 2 function keys (DEKtf keypad in figure 3.7, section 3.5). These function keys can be assigned functions such as deletion or mode changes.

Wooden box

We built a box made of wood (30cm x 21cm x 7cm). Inside we designed a matrix of switches. Depressing a key switch connects one row line to one column line. For example, depressing the '3' key connects row 1 and column 3. The prototype and a schematic view of the matrix keypad are shown in illustration 3.3.

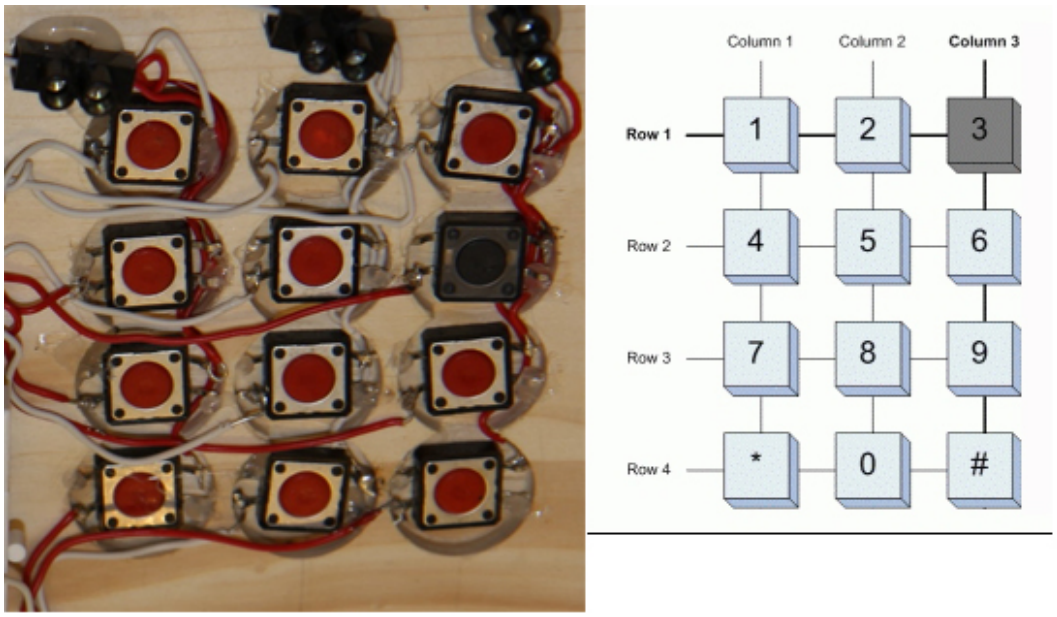

Figure 3.3: Matrix Keypad of the Prototype (left), Functioning of the Matrix (right)

Prototype composition

On top on the switches we positioned the solenoids, which are screwed to the lid (DEKtf keypad lid in figure 3.9, section 3.5). The pin extension of the plunger pokes out of the box. We covered the end of the plunger with key caps from a common keyboard (cross section of a key illustrated in figure 3.4). The key resistance was chosen with about 1,5N [Doerrer and Werthschuetzky, 2002]. To block a key we increased this value to the maximum. When pressing between two keys, of which one key is blocked, we achieve the same effect like the Apple iPhone (see section 2.4). By this way we enlarged the catchment area of the unblocked key.

Arduino connects prototype with computer

Together with the solenoids, the circuits, and the switches, one Arduino board [Banzi et al., 2005] was also placed in the box (DEKtf keypad interior view in figure 3.10, section 3.5). The Arduino connected the switches and the solenoids with the computer we used. Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. The Arduino uses its own programming language and can work independently of any computer. Furthermore, it can communicate via USB or Bluetooth with software running on a PC (Flash, Java, etc.).

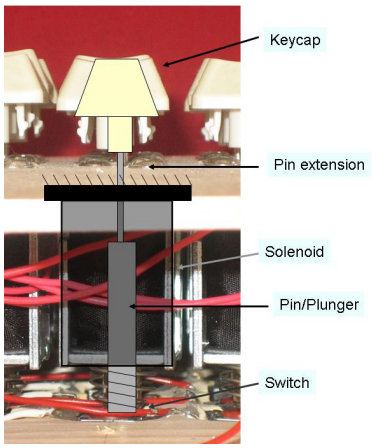

Figure 3.4: DEKtf - Cross Section of a Key

Electric circuit

The output pins of the Arduino can only provide a current of 40mA to control circuits. However, this current is not sufficient to actuate motors so that we used an external power supply to provide the sufficient electricity for the magnets (approximately 300mA per magnet). A MOS-FET connected the two circuits. MOS-FETs receive a control signal and proportionally adjust the current of a second circuit. Similarly to the output pins, the input pins of the Arduino can only receive currents up to 40mA. We used 1K ohm resistors to reduce the arriving currents. The schematic diagram of wiring the prototype is illustrated in appendix C.

GUI

The next step after wiring and fine adjustment was to create a Graphical User Interface (GUI) for the user test and to program the Arduino. We intended to keep the interface simple and decided to use Java Swing as our preferred programming language for the GUI. Sun's Javax.comm package implements the data communication between software and Arduino.

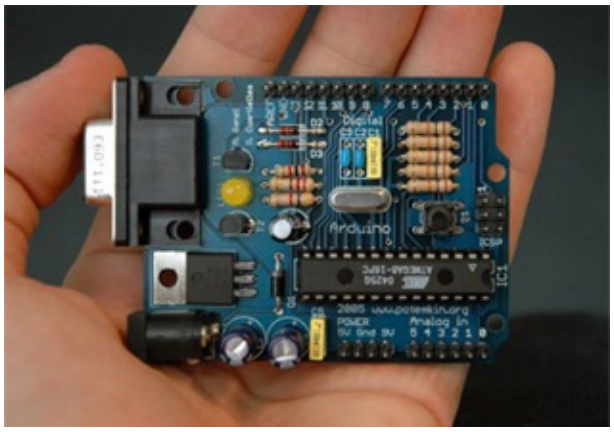

Figure 3.5: Arduino - Photo by Nicholas Zambetti ©Arduino

3.3 System Design of the Arduino

The Arduino, shown in figure 3.5, has 14 digital output pins to send, and 6 analog input pins to receive signals. Each pin can provide a voltage of 5V. To program the Arduino we had to assign these pins as follows. The digital output pins 0 and 1 are reserved for the serial communication between the Arduino and the Computer (send and receive). Three output pins are required for the switch keypad as well as additional 10 pins for controlling the magnets. Therefore we use 11 digital output pins and the analog input pins 0 and 1, which were reconfigured to function as digital output pins. The analog input pins 2 to 5 are connected to the rows of the matrix keypad.

Assigning Arduino pins

Wiring the Arduino

With the information about which output pin is sending and which input pin is receiving the signal, the program can infer which button was pressed. Therefore the Arduino continuously sends a 5V signals to each of the three output pins connected to the three columns of the matrix keypad. In case a push-button is pressed, one of the analog input pins 2-5, in each case connected to one row of the matrix keypad, receives the signal. The information, which key is pressed, is transmitted as a character to the computer. A tutorial how to operate with a matrix keypad is given at the

Connecting the matrix keypad

Sending data

Arduino Playground[1] .

Receiving data

On the other hand, if the computer sends a character to the Arduino to block or unblock a key, the Arduino sends an amplified 5V signal to the appropriate magnet. A variable is set to remember if a magnet is activated or not. Similar to a reset, a '#' sent to the Arduino releases all magnets. A C library, which we adjusted to our requirements, provides the keypad control. The communication between Arduino and PC was programmed directly on the Arduino board in its own programming language. We connected only 10 magnets because of the limited number of pins that the Arduino provides. However the blocking of the two function keys is unessential for our application.

10 magnets

3.4 Software Development

Text entry method

Entering text was realized similar to the T9 system as applied in mobile phone devices. We decided to be consistent with the Nokia keyboard layout (figure 1.5), and located the button to change between words with similar key combination in the bottom left ('*' key) of the keypad. The 'delete' button was placed on the head left corner of the keypad, because the '1' key was not reserved in the dictionary mode.

Keypad layout

Data and Multi-Tap mode

In the further development we added a data mode to enter numbers, and a Multi-Tap mode to enter words that were not included in the dictionary. To change between the modes the 'mode' button in the bottom right ('#' key) corner of the keypad had to be pressed. In the data mode we had to change the delete function to the '*' key, because here key '1' is required to enter numbers.

Definition:
dictionary mode

DICTIONARY MODE:
In the dictionary mode the user has only the possibility to enter words that are in the dictionary. The input method is similar to the T9 system of mobile phones.

[1] http://www.arduino.cc/playground/Main/KeypadTutorial

DATA MODE:
In the data mode the user can enter the numbers from 0 to 9.

Definition: *data mode*

MULTI-TAP MODE:
In the Multi-Tap mode the user can enter every letter through multiple pressing of the corresponding key. After a short delay, or by means of pressing another key the next letter can be typed. This mode was the standard text input method of mobile phones before T9 was established. The keyboard layout is similar to mobile phones (Key 2 = ABC, Key 3 = DEF, etc.).

Definition: *Multi-Tap mode*

We intended to implement a system that is simple and intuitive. On the GUI we displayed a visual keypad with labeled keys, a text field with a small task description and a panel that displays the user's input. To demonstrate the user the blocking of single keys during the introduction of the user test, we provided a panel that sends commands entered by the supervisor, directly to the prototype. In order to progress to the next task or jump back to the former we provided a "next" and a "back" button on the bottom right corner of the interface (interface illustrated in figure 3.6).

Designing the GUI

The system can easily be operated by the visual keypad and the hardware keypad prototype. The received signals from the Arduino (characters) simulate directly a button press on the corresponding visual push-button. An ActionEvent, triggered by this button, calls a procedure with the specified parameters depending on the key that was pressed.

Visual and hardware keypad

We implemented several blocking algorithms to test the system. While a user types on the keypad, the software controls the resistance of the keys depending on the applied algorithm. To have an actual setup for the current state of the system, the algorithm decides after every key press again, which keys become directly blocked or not. The different blocking algorithms for each task type are explained in the belonging user test sections.

Blocking algorithm

For the software we used two existing packages: Markku

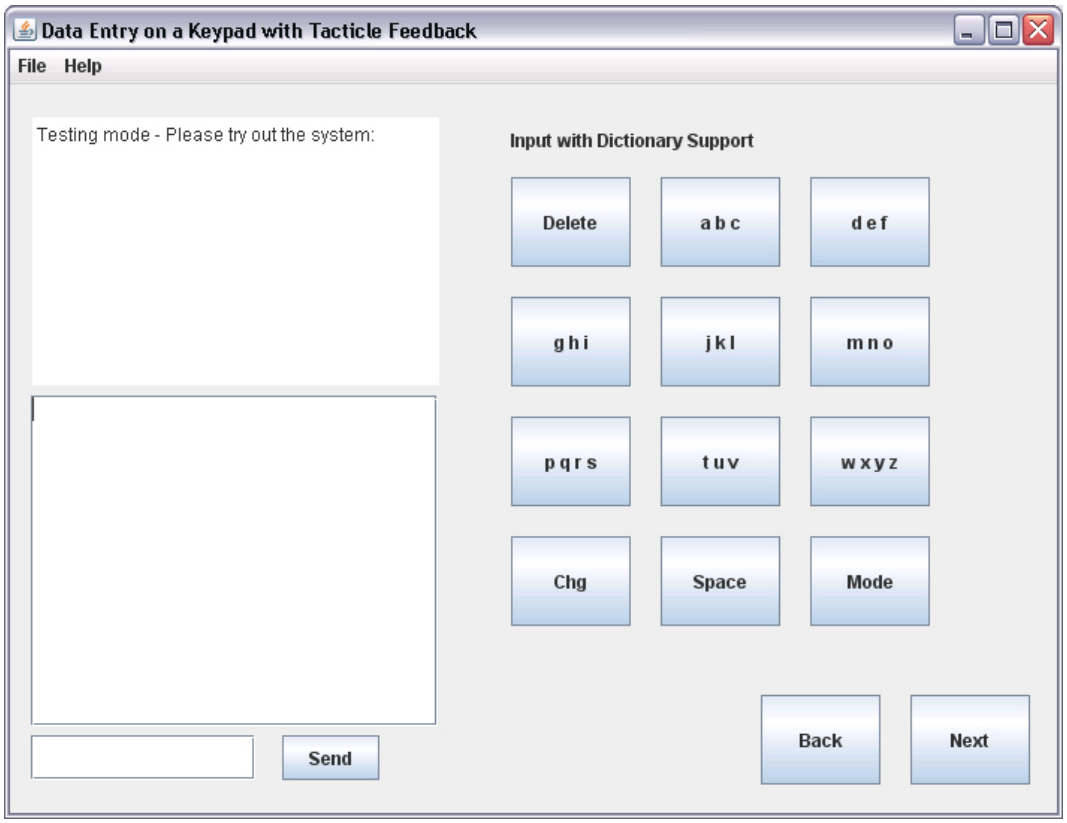

Figure 3.6: DEKtf - Graphical User Interface Using Letter-labeled Keys as Found on a Mobile Phone

Java packages

Korsumäki made the dictionary file and the FT[2] package available, which spared us the implementation of a T9-like basic system. The second package is Sun's javax.comm[3] package. It permits us to exchange serial data with the keypad prototype.

Step-by-step task description

The software offers a GUI, communication between soft- and hardware, and a step-by-step task description for the user. After a small introduction the user would not need further help for running the user test on his own.

The different functions and classes we developed are explained in the code example in appendix B.

[2] http://koti.mbnet.fi/korsu/ft.html
[3] http://java.sun.com/products/javacomm/

3.5 Photos of the Hardware Prototype

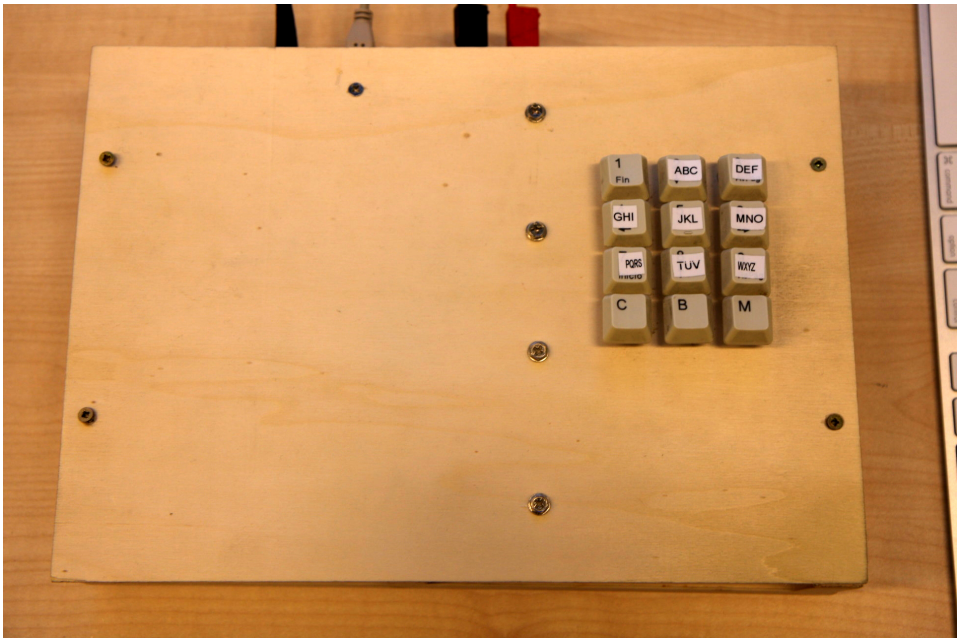

Figure 3.7: DEKtf Keypad - Bird's Eye View

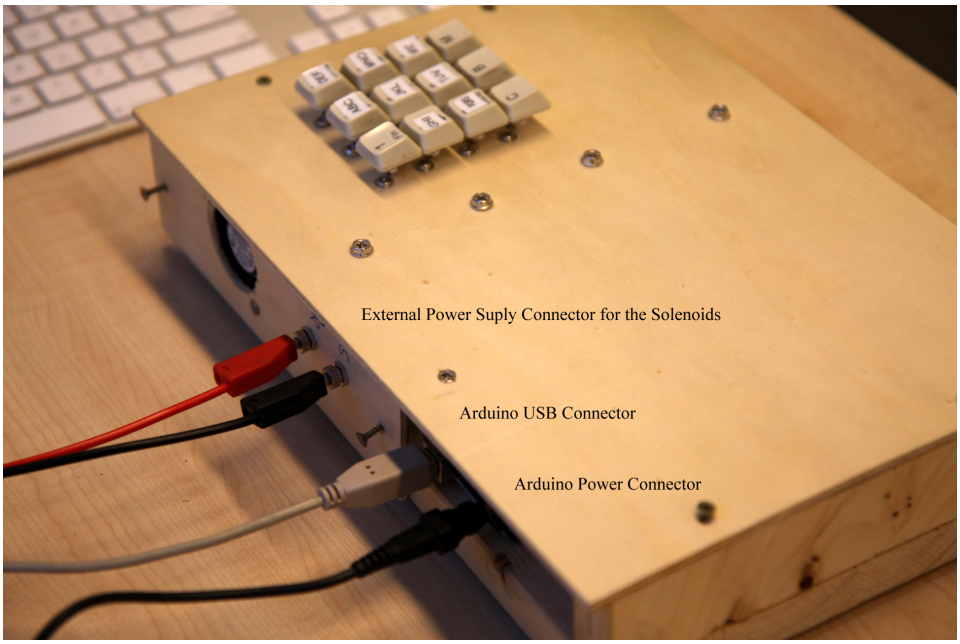

Figure 3.8: DEKtf Keypad - Back View

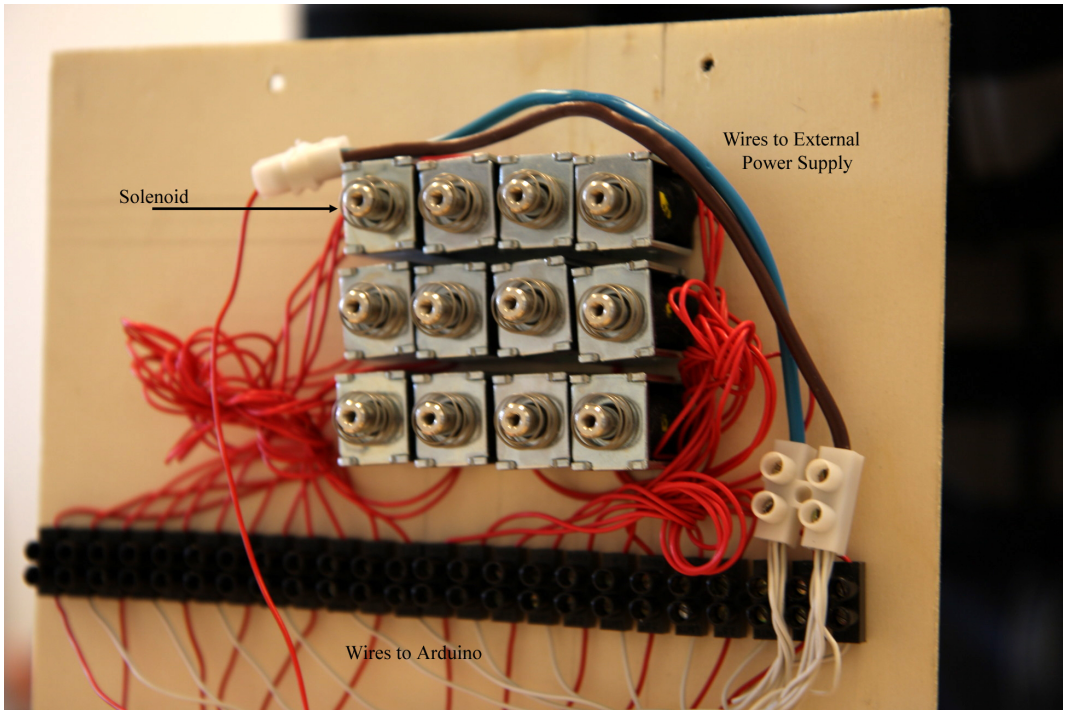

Figure 3.9: DEKtf Keypad - Back View of the Lid

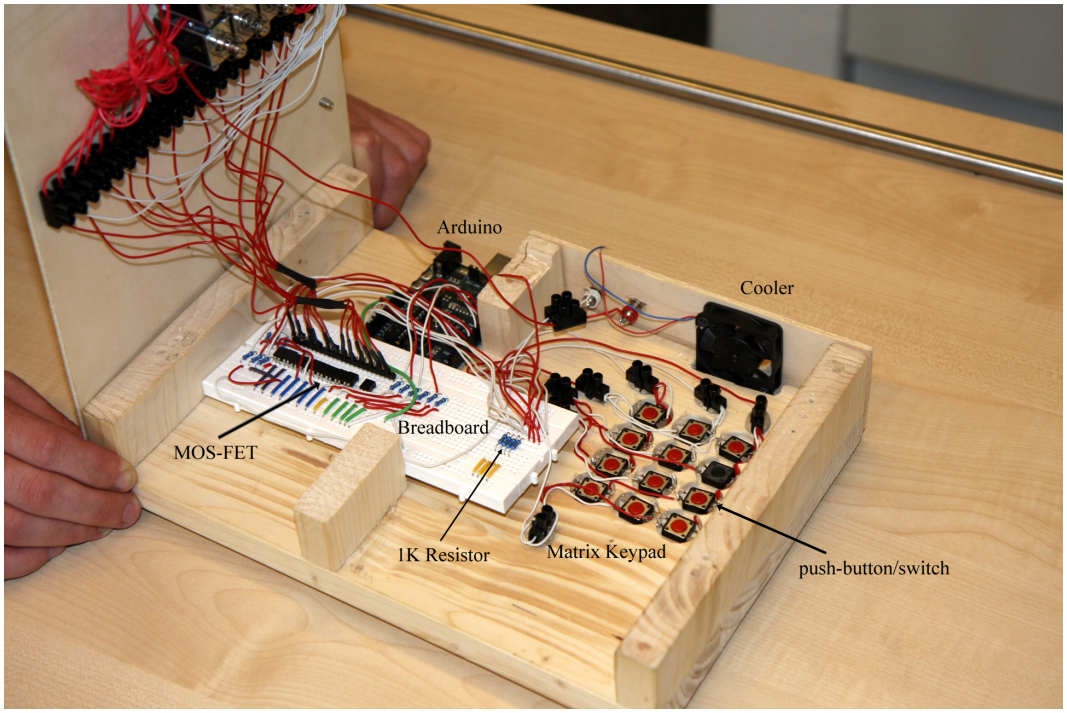

Figure 3.10: DEKtf Keypad - Interior View

3.6 Preliminary User Test

During the first DIA cycle, we used the technique known as *Cognitive Walkthrough* to analyze our system. We chose possible tasks, described goals, and determined actions to find possible errors a user could encounter.

DIA cycle

COGNITIVE WALKTHROUGH:
Cognitive Walkthroughs are performed at any stage of design using a prototype, a conceptual design document, or the final product. This is a more specific version of a design walkthrough, focusing on cognitive principles.
Based on a user's goals, a group of evaluators steps through tasks, evaluating at each step how difficult it is for the user to identify and operate the interface element most relevant to their current subgoal and how clearly the system provides feedback to that action. Cognitive walkthroughs take into consideration the user's thought processes that contribute to decision making, such as memory load and ability to reason. [...]
This approach is intended especially to help understand the usability of a system for first-time or infrequent users, that is, for users in an exploratory learning mode. (http://www.usabilityfirst.com/)

Definition: *Cognitive Walkthrough*

After three months of designing, analyzing, and implementing the prototype, it was time to run the first user study. We used our high-fidelity hardware prototype and a preliminary software that reproduced the input received by the tactile feedback keypad. The software already included a T9 system based on a 6000 word English dictionary.

High-fidelity hardware prototype

We developed a limited system: The entry was just based on the dictionary (dictionary mode), words that were not in the dictionary were impossible to type. In case the dictionary could not interpret a key press, this key was previously blocked by the system. That's why we implemented a blocking algorithm that simulates the pressing of all keys to determine which key had to be blocked before the next entry was done. Every time a key was pressed the algorithm starts again to provide an updated setup. At this development point the deletion of already written words as well as

Limited software

the entry of numbers was impossible for the user.

The goals of this first user test were to test the hardware, to get feedback from the user, and to find out where improvements would be required.

3.6.1 Set-Up and Participants

The keypad was placed on a table in front of a 40″ NEC LCD screen. The Graphical User Interface (GUI) was set in the middle of the screen. During the first test the supervisor made a note of the user's reactions and comments to improve the system.

Four computer science students

Four computer science students between the age of 20 and 30 participated in the user test. The participants were given a short introduction explaining the functionality and the idea behind the system. After the introduction the users were asked to write three phrases. Two students started with and the other two without tactile feedback support to avoid a learning effect. We alternately activated and deactivated the tactile feedback from task to task. We ran two cycles that every user wrote each task both with and without activated tactile feedback.

Learning effect

Tasks Given to the Participants:

1. "Please write: great idea to write a program like dektf"
2. "Please write: 10 x problem guy"
3. "Please write: 4 x good to be at home"

3.6.2 Results

The preliminary user test was conducted as an informal user test to test the hardware. The hardware worked out

well. The participants only criticized the noisy sound produced by the magnets, the missing letter labels on the keys, and the unfamiliar key resistance compared to standard keyboards. The key resistance of our system is comparable to cash drawers or security systems. The participants mainly criticized the limited functionality of the first software prototype.

Software improvements required

Appearing software problems:

- There was no possibility of deleting or modifying previously written words.
- A cursor that showed the current position was missing in the text field.
- While writing, the current word was not highlighted or underlined.
- The user received no visual feedback for a key press.
- The 'Change' button did not always react by the first button press.

3.6.3 Improvements

After the preliminary/first user text we decided to test other applications than just typing in the dictionary mode. We added a date task where participants were required to transcribe numbers/dates from a given list, such as 17.05.1982. Keys were blocked if a key press would lead to an invalid date that did not conform to the rules, for example, 45.24.9798 is not a valid date (*grammatically invalid date*). For example, the system blocks the keys 4–9 in the first position, the other positions are illustrated with the flowchart in appendix D. To evaluate our system we used a simplified blocking algorithm. An error could occur when a month has less than 31 days. The system would not prevent the user from typing 31.02. The range of dates was restricted from 01.01.1000 to 31.12.2999. The system will automatically insert decimal separators (periods) and proceed to the next line (Enter) when required.

Date task

Date blocking algorithm

Task five was the second new task, which we included to evaluate the writing behavior of the user during typing of longer phrases. Furthermore, we deleted the word *week-end* from the dictionary to see how the users react on unexpected errors.

The third new task (task six) asked the user to write three long words (infrastructure, pharmaceutical, and semiconductors). Since the words were longer, the system had more possibilities to intervene to avoid more errors. Besides, no other words were in the dictionary that started with infras..., phar..., and semic... Thus, from this moment all keys except one were blocked.

Additional Tasks:

4. "Please change to the data mode (periods and Enter are set automatically) and write:

   08.08.2008

   16.04.2008

   01.08.1980

   17.05.1982

   31.08.1968

   12.12.1221

   11.09.1995

   24.12.1924"

5. "Please write: I am writing to you in order to check how you and your project are doing and if you have any plans for the weekend."

6. "Please write the following 3 words:

   infrastructure

   pharmaceuticals

   semiconductors"

Expanded Software Version

Because of the results from the preliminary (first) user test we expanded the software by the following functionalities:

- Visual "click"-feedback on the monitor keypad that confirmed visually that a key was pressed.
- The 'change' button reacted always on the first press.
- A function that allowed scrolling through all the possible word combinations by pressing the change button already existed. We added the possibility of pressing the "change" button when the last possible combination was reached, which then takes the user back to the first possibility. This was implemented so that the user can always browse through the possibilities again in case he hit the change button to often at the first time.
- The cursor showed the current position in the input field.
- The current typed word was highlighted (dictionary mode).
- The data mode was introduced to enter numbers (definition on page 37).
- The Multi-Tap mode was added to write words that are not included in the dictionary (definition on page 37).
- Both the deletion of single characters in the data mode and the deleting of whole words in the dictionary mode were added.
- To provide additional visual feedback to the user, the blocked keys were grayed out on the keypad (monitor). However, we observed that the user paid too much attention to the visual feedback and avoided errors only because of this. After three participants, we decided to change the setup and removed this functionality. The keypad on the screen then only showed the labels of the different modes and gave a "click"-feedback if a key was pressed.

Unknown words

To enter words that were not in the dictionary, the user had to change to the Multi-Tap mode. In the dictionary mode the user was not able to enter words that were not included in the dictionary. In case the user overcame the force of the blocked key, nothing happened. That was one reason why we added a number task, where it was possible to type the "wrong" number if the user overcame the force of the button.

Constant resistance in Multi-Tap mode

In the dictionary mode we kept the same blocking algorithm like in the fist user test. Because of no restrictions in the Multi-Tap mode, no key got blocked.

Chapter 4

Evaluation

"You know you have achieved perfection in design, not when you have nothing more to add, but when you have nothing more to take away. "

—Antoine de Saint-Exupery

The last chapter gave us an overview over the development process of the prototype. We build a prototype keypad that provides variable key resistances, designed the corresponding software, and conducted a preliminary user test.

4.1 Second User Test

After conducting the preliminary (first) user test, we modified our system according to the users' feedback collected during the first test. We conducted a second user study to test the modified system. The six tasks that had to be completed by the users were introduced in section 3.6.1 and 3.6.3.

The goals of the second user study were to test the software, to get feedback from the user, to evaluate if the new date task is a good approach, and to find out once more where further improvements would be required.

4.1.1 Set-Up and Participants

Avoids learning effect

Ten students between the age of 20 and 30 participated in the user test. Five students started the first task with, the other five without tactile feedback support. We alternately activated and deactivated the tactile feedback from task to task to avoid the influence of a learning effect. Both groups performed all tasks (two cycles). The users had number-labeled keys on the hardware keypad. We scheduled the tests over two days.

Definition: *Silent Observation*

SILENT OBSERVATION:
The designer watches and records the actions of the user while working on a task. There is no communication during the observation.

Recorded by camera

The user test was conducted with the *Silent Observation* method. To evaluate the user study we recorded the experiment with a video camera, which was placed behind the user. We recorded the screen, the users' finger movements, and the completion time for each task. The participants were given a short introduction explaining the functionality and the idea behind the system. The supervisor of the experiment took notes of the errors made by the user.

4.1.2 Results

Tactile feedback avoids errors

The results of the user test showed that active tactile feedback definitively avoids errors. The tactile feedback attracts the users attention to reconsider his actions. More errors were identified and a fourth of the errors made without tactile feedback would have been avoided if the feedback was activated.

Unavoidable Errors

One problem we discovered was that our system (in the dictionary mode) only interpreted a letter combination as

Unavoidable errors

an error, if a typed word or character combination was not available in the dictionary. Almost all possible letter combinations with up to three letters were available in the dictionary. Therefore, the blocking algorithm could not intervene to words up to three letters. The error prevention in average started with pressing the fourth letter. This fact and the problem of confounding keys made up 75% of all errors. The following errors were frequently made and could not be avoided by DEKtf:

- Starting a word with the wrong letters.
- Confounding of keys like the '0' and 'space' keys (confusion because of different modes), and the 'space' and 'change' keys (neighboring keys).
- Deleting of words because the dictionary showed an alternative word with the same key combination and the user thought he typed wrong.
- Forgetting to use the 'change' key, because the word intended was not the first choice and the users were concentrated on the keypad.

Grammatical correctness of date

On the other hand, DEKtf could avoid typing grammatically invalid dates, for example, 45.24.9798. The users whose attention was more focused on the keypad, avoided more errors with activated tactile feedback because the prototype directly assisted them before an error was made. Due to the received feedback, their attention was diverted to the screen, in order to control their already written data.

Typing Speed

We compared the task completion times of the group that had feedback in the first cycle, to those who had no feedback. For the second cycle we proceeded identically.

From the times taken we could see that text writing in the first cycle was faster with feedback, but after a training period of one cycle the system without feedback was even

Date task confirms DEKtf

faster and less errors were made. Just the date entry task brought an interesting effect. The times in both cycles for typing without feedback were in average the same (1:04 min.). The results showed us that the speed of typing was 20% higher (1:16 min.) without tactile feedback (first run), but in average at least one error was made and not discovered until finishing the task. In the second cycle, the trust in the system was higher, and the group with feedback was 20% faster (0:53 min.). All errors that could be avoided by the blocking algorithm did not happen.

Low error rate

Unknown Words by the Dictionary

At the beginning of the user test we explained to the users that words not known by the dictionary cannot be entered in the dictionary mode. In this case, the tactile feedback got activated and the keys were blocked. The user had to change to the Multi-Tap mode to enter the word manually.

Handling of unknown dictionary words

In task five the users had to type the word *weekend*. However, we deleted this word previously from the dictionary. Until *week* everything worked like expected, then the key '3', which corresponds to the letters 'def', was blocked.

The reaction of the users were different. Some participants noticed the feedback and started to think what was going wrong. Some tried 2–3 times to overcome the feedback, thinking it is an error of the system, until they understood that they were not able to enter the word. With feedback every user understood after a period of time that *weekend* is not present in the dictionary. To finish the word, one user pressed the 'space' key first, then the 'delete' key and wrote *end* in the dictionary mode. That way he avoided to change to the Multi-Tap mode to complete the task.

Unknown words without feedback

The users without feedback in the second cycle were reminded that *weekend* had to be typed in the Multi-Tap mode. However, in the first cycle the users without feedback got lost, did not realize the error because they were concentrated on the keypad, or tried several times to press the 'def' key again until they gave up.

Hard- and Software

During the second user study the hardware worked well again. Only once during one test a switch broke. Therefore, we had to change the push-button inside of the prototype. However, this trouble was solved after 15 minutes, so that the test could be completed without any further interruptions. There were only few incidences where the software crashed. However, we managed to solve this problem in a timely manner. Thus, we could achieve valid results during the user tests.

Hardware worked out

4.1.3 Improvements

The results presented above made us introduce another number task because the success achieved in the date entry task of the last user test seemed promising. We exchanged task five (task: *"I am writing to you in order to check how you and your project are doing and if you have any plans for the weekend."*) because the words were to short for DEKtf to intervene and avoid errors. We added an additional date task. Here, the blocking algorithm blocked all keys except the delete, mode, and digit key that has to be entered next.

New blocking algorithm blocks all number keys except one

Second Date Task:

5. "Please change to the number mode (points and enter are set automatically) and write: 02.05.1957 18.02.2008 04.07.1986 01.09.1945 30.04.1972 16.05.1999 12.11.1345 26.03.1608"

We are not saying that this limited approach (blocking all keys except the next expected key) was better. We wanted to compare this approach with the blocking algorithms of the text (task one to three) and the date (task four) entry tasks. After the introduction of the new task five, we had another blocking algorithm to evaluate.

Limited interaction

4.2 Final User Test

The goal of the third and final user study was to investigate that number tasks with DEKtf were more efficient (faster, less errors) than entering text where the keys have been assigned multiple letters. Furthermore, we wanted to find out if letter-labeled hardware keys workout better than the numbered labels before. As in the latter study (second user study), we wanted to check again if the hard- and software function without problems, receive feedback from the user, and find out once more where further improvements would be desirable.

4.2.1 Set-Up and Participants

Letter-labeled keys

Fourteen volunteers (age: 23-31 years) participated in the third user test. The users were again divided in two groups. One group started with, the other one without tactile feedback support. The users had now letter-labeled keys on the hardware prototype.

Log file

The experimental setup (figure 4.1) was changed. The camera was positioned on the top of the monitor to record the finger movements on the keypad and to observe the focus of view of the user. The users' entries were logged in a text file together with the current time.

Short introduction

The participants were given a short introduction explaining the functionality and the idea behind the system. Additionally to the log file, the supervisor of the experiment took notes of the errors made by the user. Before the actual test with our prototype started, the users had to type one text phrase and a list of four dates with a common keyboard to evaluate the typing behavior of the user.

4.2.2 Results

One issue of the final user test was to see if the users work more efficiently (typing speed and error rate) with num-

Figure 4.1: Experimental Setup of the Third User Test

bered (second user test) or with lettered labels (final user test) on the keypad.

Lettered vs. Numbered Key Labels

Numbered and lettered labels

We found out that both labels (numbered and lettered) on the keypad took the attention of the participants during writing. The group that had the numbered labels (second user test) was concentrated on the screen during writing in the dictionary mode. However, during the task where the users had to enter dates their attention changed to the keypad. In the third user study we observed the opposite result.

Importance of hardware labels

Five of six tasks (83%) required the user to type text. Therefore, the users who had numbered labels said that the use of the labels was not so important, in addition they recommended lettered key labels for further user tests. In the third user test (lettered labels), which consisted only of four out of six tasks requiring text typing, still 71% of the users said that the labels were important for using the system (diagram in figure 4.2).

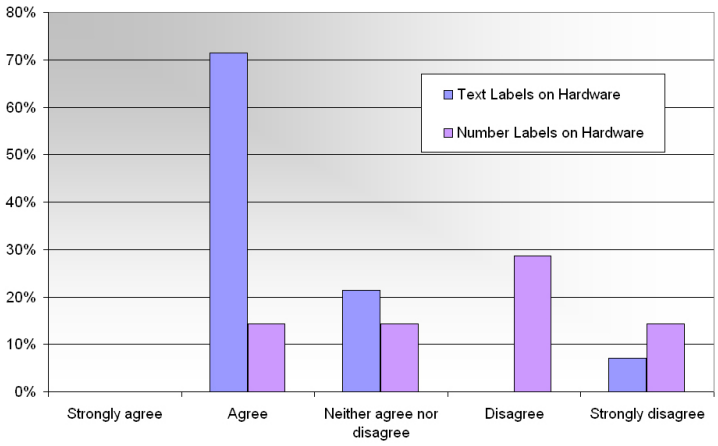

Figure 4.2: The labels on the key of the keypad prototype hardware are very important.

These results showed us that lettered labels were more important compared to numbered labels, even if there were already less text entry tasks to complete. Although the users had lettered labels in the third user test, the task time for text entry tasks did not change exorbitantly compared to the second test. Thus, we think that after some learning period the users would work faster, even if there would be no labels on the keys (similar to DAS Keyboard, section 2.1.1). Or alternatively, we could use double labels to provide the users both information, numbers and letters on each key of the keypad.

Importance of visual keypad

Besides, we asked the users of both studies how important the labels were on the monitor keypad. Because of the great share of text typing tasks, the results were similar to the question about the importance of the hardware key labels. This explains that 64% of the participants of the second user test (numbered labels on the hardware) agreed that the labels on the screen (lettered) were important. For the users of the third usertest (lettered labels) it seemed to be less important. Only half of the participants agreed that monitor labels are important for the use of the system. These users were more focused on the prototype keypad rather than the screen labels. The screen labels were only used from time to time for the date entry task (diagram in figure 4.3).

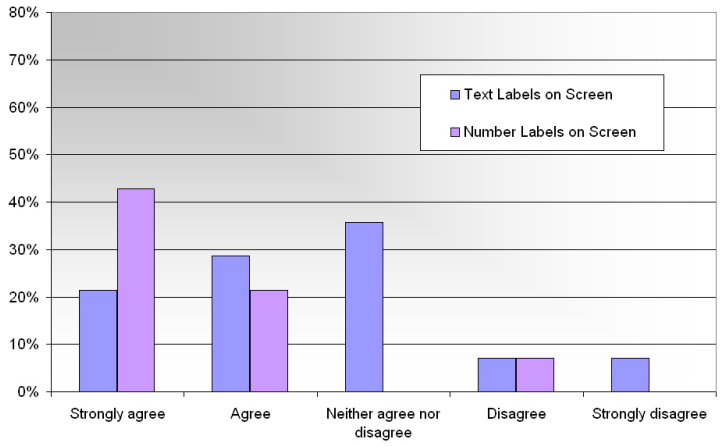

Figure 4.3: The labels on the monitor keypad were important to use the prototype.

Text Entry

The errors that occurred during the text entry tasks were similar as in the second user test. Most users of the third user test (letter-labeled keys) wrote a little bit faster than in the second user test (number-labeled keys).

Typing Speed

In average the task completion times of the final user test were for the first task 26%, the second task 1%, and the third task 2% faster compared to the second user test. The biggest time saving was identified in the first task with tactile feedback activated (36%). After a learning period of only one task the time saving, caused by the lettered labels, could be ignored.

In contrast to the second user test, the participants had now lettered labels for the date entry task (task four) and commented that the numbered labels were missing on the keys. The average time needed for this task was approximately 10% more compared to the latter test (second user test). One reason besides the fact that there were users with dif-

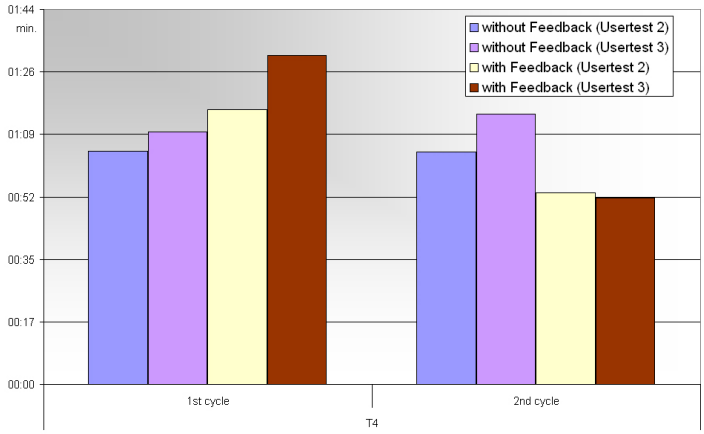

Figure 4.4: Task 4: Comparison of the Task Completion Time of the Second and Third User Test

ferent levels of writing abilities, could be the missing number labels on the keypad.

Too much trust

In the first cycle of task four the users trusted too much in the error correction of the system. The participants overlooked faults that were neither discovered by DEKtf. The users had to go back to the position where the error occurred and correct them. Thus, the task time increased by 30% due to the late error correction. In the second cycle the users who received tactile feedback were 31% faster than the others, ilustrated in figure 4.4. Most common date errors that occurred were transposed digits in the year and copying errors. These kinds of errors cannot be detected by DEKtf, because these are not grammatically invalid dates. However, grammatical errors made without feedback did not happen with feedback. DEKtf assisted the users in avoiding these errors. In case of overcoming the resistance of a key, the system advised the user to control their typing.

Avoiding grammatical errors

DEKtf faster in limited date task

The new date task (task five), which blocked all keys except the next expected digit, gave a precise result. In both cycles the users were faster with tactile feedback support (first cycle 45%, second cycle 10%). The error rate was nearly zero, just one user tried once to overcome the feedback. In the cycle without feedback an average of three typing errors occurred.

Time needed by the user in task six (long words in dictionary mode) was the same with and without feedback (0:35sec). Only in the first cycle the group who started with feedback needed around 0:53 seconds in average. That could have been caused by two users in this group who were writing carefully. In the second cycle they had gained more trust in the system and got faster.

No speed up when typing long words

Types of Mistakes

The first three dictionary tasks exposed that DEKtf does not sufficiently assist the user in the dictionary mode. The errors that were made cannot be prevented by the system. Too much trust in the error correction methods brought the users to type carelessly. For example, the users were not focusing their attention on the monitor so that errors went undetected. The times of cycles with tactile feedback were slower due to the users having to correct themselves afterwards. Because the assignment of multiple letters per key, DEKtf was not discovering a sufficient amount of errors. Too many words start with the same key combination. More success was achieved using single lettered key applications, for example, the date entry tasks as previously illustrated.

Errors went undetected

Inconsistence of the 'Delete' Key Another problem that occurred was the inconsistence of the 'delete' key. In the dictionary mode we decided to place the 'change' key in the bottom left of the keypad, similar to the mobile phone. Therefore, the deletion key was located to the '1' key, the only available key.
In contrary, in the data mode the key number '1' was needed. For this reason the deletion key moved to the bottom left of the keypad. This inconsistency appeared when we added the second date task. 60% of the users of the last user test were confused about the changed postition of the 'delete' key.

Consistent with phone pad layout

Key Confounding The lettered keys were located similar to the mobile phone pad layout. Due to consistency, we decided to design the numbered labels similar to mobile phone, cash dispenser, and security systems.

1/7 problem

Operating with mobile phones, the users do not have to think about where which number key is located. However, while working with DEKtf, because of *look & feel* of a normal keyboard or a calculator, the user changed in mind the configuration. Instead of pressing the keys '1' to '3', over 50% of the users frequently tried to press the keys '7' to '9' and contrary. The '0' was pressed with the thumb like on the keypad of a standard keyboard so the user hit the delete button. In most situations the tactile feedback reminded the user that he changed the keys once more.

List of words

Browsing Words with the Same Key Combination When users changed between words with a similar key combination, for example, good = home = 4663, they often missed the intended word. They had to go through all words again to come back to the right choice. Some users deleted the last letter and retyped it in order to return faster to the first word in the word list. An alternative approach, also interesting for mobile phone devices, could be to display a list with all words with the same key combination. With a number before each word, the user could directly choose the intended word.

With limited keys inevitable

Modes The 'mode' key pressed by mistake could neither be avoided by the system because this key has to be available in every moment. Modes are never a good approach, but with limited keys inevitable. We could avoid this problem by using only data applications that require typing of numbers or, alternatively, developing a full keyboard.

Auditive feedback

Noise of the Magnets A side effect provoked by the noise of the keypad was that one user thought this noise is confirming the key press. He got irritated when typing the first 3–4 letters of words because of no interaction of the magnets, he received no auditive feedback. In the repetition word tasks (task two and three) some users also controlled there writing by memorizing the sound made while typing the same word. Another observation caused by memorizing sounds was that one user said that he was able to hear when the 'change' button had to be pressed.

Deletion Function The deleting function works out in the Multi-Tab and data mode, but in the dictionary mode we tried the new idea to delete entire words by pressing the delete button once. During writing the current word, the user could delete digit by digit. However, when the whole word was deleted the next word was deleted completely by pressing the 'delete' key once. A problem occurred when users discovered an error in a previous word. They accidentally deleted more words than necessary by multiple presses of the 'change' key. This happened because the user still thought that he had to delete digit by digit. They were not used to this new function. One user proposed the idea that the word gets highlighted the first time the 'delete' key is pressed, with the second press it gets deleted.

Deleting letters

Deleting words

Dictionary Problems When the feedback was deactivated the users often made mistakes, but the algorithm of the dictionary mode did not allow the user to write on, because no word like this existed in the dictionary. That means that the user was typing, but nothing was displayed on the screen. The same effect happened when the user pressed the button so hard that he overcame the blocking of the key.

Problems with the dictionary algorithm

Reference Point One user mentioned as positive that the '5' key stuck out a little bit higher than the other keys. It was a reference point for the user who was not looking to the keypad.

Center key higher than others

4.3 Final Implementation Changes

After the last user test we rearranged the GUI. The task description got its own panel above the data panel, and the input field was located vertically, parallel to the data panel. For the user it facilitates to identify the row, which contains the data he needs to copy next.

Rearranged GUI

There is practically no application on the market that provides a visual keyboard, and additionally, the tests showed that the visual keypad distracts the user. For this reason the

Hiding visual keypad

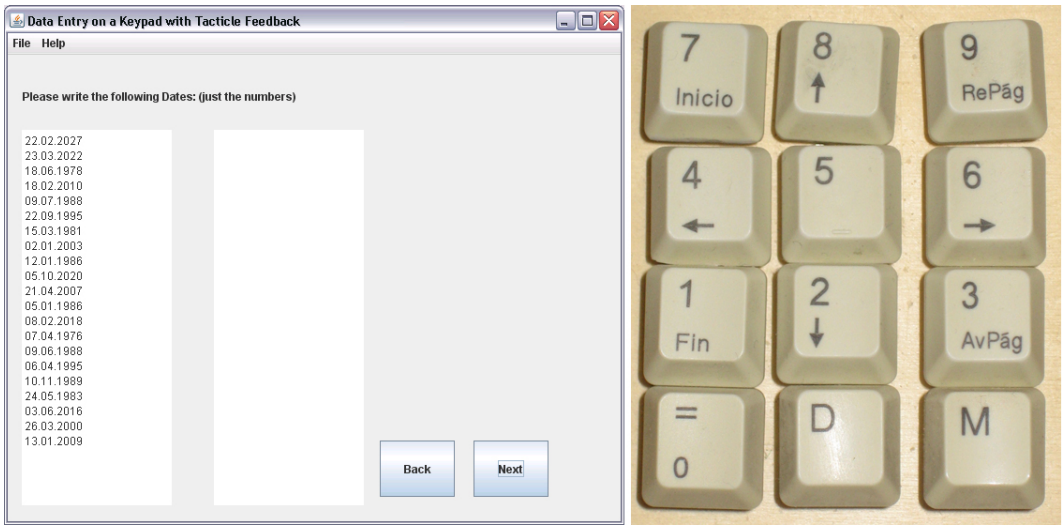

Figure 4.5: Final GUI (left) and Alternative Keyboard Layout (right)

visual keypad is hidden in the new software version (figure 4.5, left).

Keypad layout change

Many users mentioned the problem that they thought the numbers were located like on a standard computer keyboard keypad. The design of the prototype supported this mistake. Therefore, we exchanged the keys from the first row with the keys from the third row. The hardware was reprogrammed to send the correct signal corresponding to the new key position. The zero key was placed in the bottom left of the keypad, according to the keypad of a standard keyboard. This change gives the users the possibility to press the zero key comfortably with the thumb (figure 4.5, right).

No modes

We decided, as long as no full keyboard prototype is developed, to remove the dictionary mode tasks. This change entailed that the system has no modes anymore.

Only data tasks

We designed a new date task that generates randomly in execution time the dates to type. The range of dates is restricted from 01.01.1970 to 31.12.2030. In the new implementation all keys, except the next key the user has to type, are blocked. An additional blocking algorithm, which includes the controlling of the grammatical correctness and the limited temporal range of the dates, should be devel-

Improved blocking algorithm

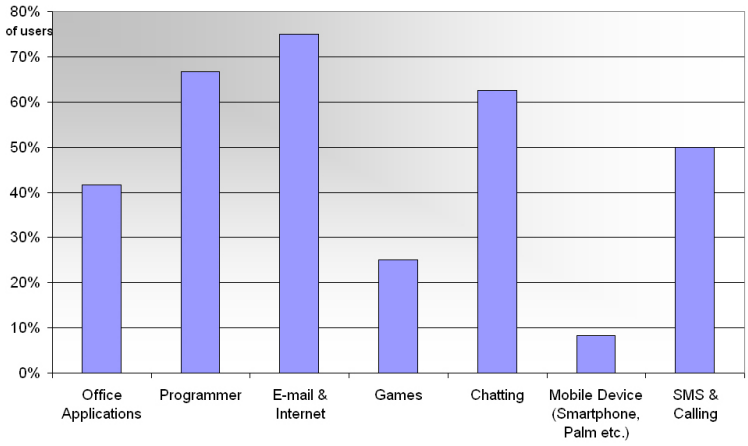

Figure 4.6: The participants daily typing tasks.

oped. With this additional blocking algorithm and the new range of about 60 years, we could reduce the errors made in the *year* field. Many users made these mistakes because of the huge temporal range of the *year* in the third user test (1000-2999).

4.4 Evaluation of the Questionnaires

We asked the users to fill out questionnaires (appendix A) after the completion of the tasks. This section evaluates the users' feedback.

Typing behavior

Participants of the user tests were asked to describe their daily typing tasks. We wanted to find out in which areas DEKtf could be useful, and how much the users type. Programming, Internet, and chatting were most used applications (figure 4.6). 75% of the users said that they type frequently.

Typing on a QWERTY Keyboard

On a QWERTY keyboard 54% are typing with more than 8 fingers and the third of all participants had ex-

Touch typing

perience in touch typing. But in the user test when the users had to type dates with the keypad of a normal keyboard, most were slow and had to check the labels. The users commented that they are using more the numbers above the alphabetic characters of the keyboard because their notebooks do not provide a keypad.

Typing on a Phone Pad

Typing on a mobile device

Because we were using the keypad layout of a mobile phone, we wanted to know how experienced the users were with typing text on a mobile phone device. Almost 80% of the users said they have normal to professional T9 writing experience (scale: none, little, normal, advanced, professional), and more than two thirds always use T9 support when entering text. However, the users had problems to transfer the knowledge using a mobile phone to the DEKtf prototype.

Typing on the DEKtf Prototype

Comfort

After using DEKtf we asked the participants how comfortable they felt using the keyboard (figure 4.7). Two-third were feeling comfortable. One user commented that the activation and deactivation of the magnets were noticeable and uncomfortable for the fingers. Someone else observed that the feedback of the unblocked keys was strange and the keys felt rigid. Another user noticed that he enjoyed using the prototype, and that the use with five fingers should be even faster than the use of a mobile phone, where only one finger per hand is available for typing. One user said that he was worried that the system would constrain his actions too much.

4.4.1 Suggestion for Hardware Improvements quoted on the Questionnaire

Changing the key resistance similar to a normal keyboard, the use of flat keys like used in notebooks, and making the

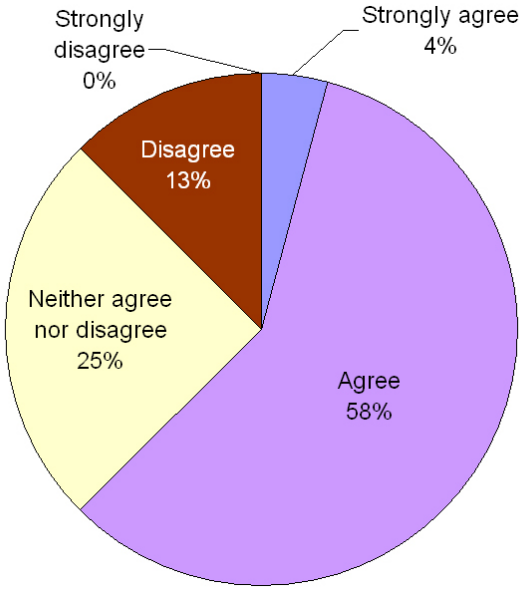

Figure 4.7: I felt Comfortable Using the Keypad.

Changing key resistance

keypad smaller were suggestions given by the users. During the development process we thought about doing this, but the problem was that the existing keypads, which we could connect to the Arduino board, had key resistances similar to security systems and cash-drawer. Additionally, the magnets were too big to find sufficient space above the keypads.

Equal key height

Furthermore, the users advised us to make the keys more even in height. However, due to the design of the keys it was possible that they change their height by pressing them with strong pressure. Responsible for that are the key caps that were only clipped on the extension pin of the magnet (figure 3.4).

Reduction of the noise originated from the magnets

The noise originated from the magnets of the keypad was disturbing. To reduce the noise we tried to use springs and a foam material, but both attempts changed the resistance of the keys too much. The keys got too resistant and were not ergonomic anymore. Other users asked to make the keypad more stylish, but we thought this is a less important factor for a prototype.

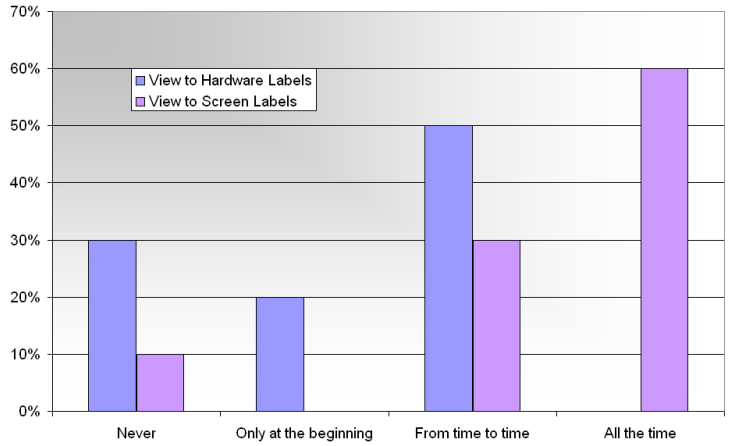

Figure 4.8: Second User Study (Numbered Hardware Labels): How often did you look at the labels of the Hardware and the Software Keys?

LEDs in the keys

One participant mentioned that the addition of visual feedback to the blocked keys (hardware) could help to avoid errors. This expansion could be realized by installing small LEDs in the each key. This approach as well as vibrating keys could be also interesting for mobile phone devices.

Interferences of the magnetic fields

Moreover we have to consider a solution for the problem of induction. Unblocked keys could be effected by neighboring blocked keys. This occurred twice during the user test, especially in task five. Some unblocked keys were harder to press than others.

4.4.2 Monitor vs. Hardware Keypad

Importance of labels depends on application

We asked the users how often they looked to the visual keypad labels (monitor) and how often they looked to the hardware keypad labels. It is interesting that about 60% of the group with numbered keypad labels (second user test) looked all the time to the monitor keypad. No one of this group looked all the time to the hardware keypad, just 50% looked from time to time to the prototype (hardware) labels (diagram in figure 4.8). We want to return to mind the second user test, where four of five tasks were typing

Numbered Labels rarely used

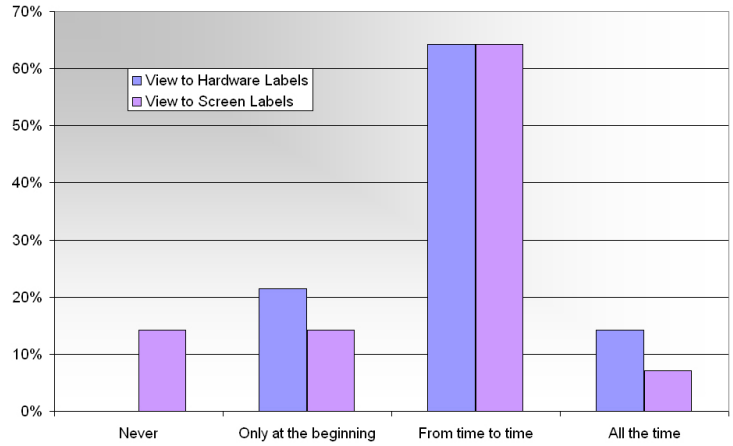

Figure 4.9: Final User Study (Lettered Hardware Labels): How often did you look at the labels of the Hardware and the Software Keys?

tasks. Thus, these results showed us that the lettered labels on the monitor keypad were fundamental for solving the tasks. The numbered hardware labels were rarely used.

Lettered labels more important

On the other hand, over 60% of the group with lettered prototype labels (third user test) looked from time to time on both, the screen and the keypad. In this study the user had to fulfill about 40% date tasks. So they changed continously the view between monitor (numbered) labels for date entry tasks and keypad labels (lettered) in the text entry tasks. These results are illustrated in figure 4.9.

Number positions easier to remember

In average the users with lettered key labels used the prototype labels more often, the other group (numbered labels) more often the visual keypad. The users could remember more easily the number position than the position where they can find the respective letter.

4.4.3 Suggestion for Software Improvements quoted on the Questionnaire

GUI improvements

One user asked for more space for the input field and the task panel on the GUI, with a smaller visual keypad below.

In our final implementation we hide the keypad completely and the input field and task panel are presented as the central point of the GUI. Adding visual feedback when a word is not in the dictionary, and to gray out keys that are not useful in the dictionary context were mentioned as well.

Visual block feedback

The idea of graying out the blocked keys on the visual keypad was applied during the first three participants of the second user test. In case the user was concentrated on the screen, the visual feedback avoided errors even before touching the wrong key. However, most applications do not even have a visual keypad, for this reason we decided to leave the visual feedback out.

Use of forms

In tasks four and five the automatic included dots and line skips were irritating some user. Forms could be used for the visualization of the entered dates. Three fields per row, two with two positions for day and month, and one with four positions for the year.

Users getting lost during copying text

A regular occurring problem was that the users got lost during copying text or dates. They lost the row where they had been, or they skipped a row and did not type it at all. To avoid this problem in the future, we implemented in our final version that the task description is separated from the data panel and we located this panel vertically parallel from the input field. It is then easier for the user to compare the entered data with the template.

Highlighting text

Furthermore, we could mark or highlight the text that the user currently has to copy. Alternatively we might only display one date or phrase, or strike out already written lines when typed by the user.

Blocking 'space' key

In DEKtf it would be possible to block the 'space' key when the text was copied wrongly, but in current systems, where it is unknown what the user wants to write, this is not applicable.

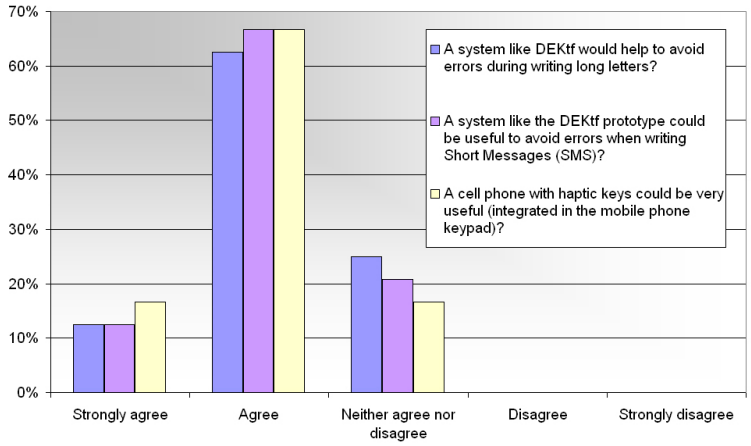

Figure 4.10: Would DEKtf help to avoid errors during writing long letters and SMS? A Cellphone with haptic keyboard could be very useful?

4.4.4 The Users' Opinion about the Application of DEKtf to other Fields

DEKtf Keypad

We were curious if the user could imagine typing with DEKtf in different application domains. Over 80% of the users agreed that Dektf avoids errors even for writing long letters and SMS messages. The same percentage agreed that a system like DEKtf applied inside a mobile phone could be usefull (figure 4.10).

DEKTtf avoids errors

DEKtf Keyboard

Interesting is, compared to the opinion about DEKtf as a keypad, that applied on a full keyboard also about 80% agreed that it would avoid errors. However, they doubt that as a consequence the typing speed would slow down (figure 4.11) and the interruption of the typing flow would cost more time than the standard correction. For the use in a full keyboard an extensive dictionary is necessary. The

DEKtf applied on a full keyboard

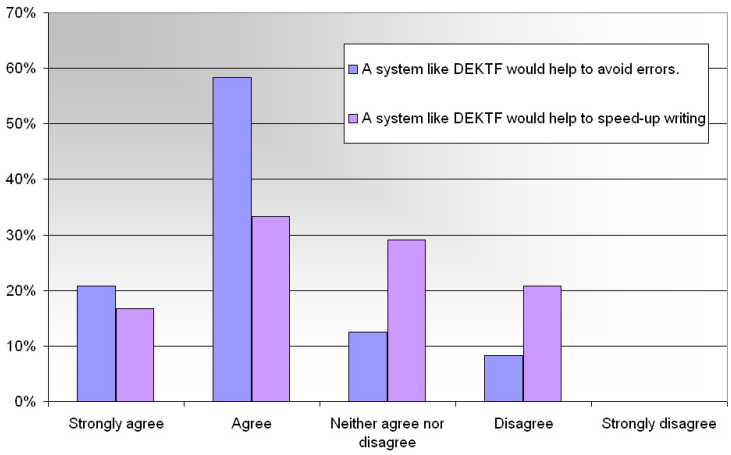

Figure 4.11: Imagine applying these principles to a full keyboard. All keys without sense in the context would be blocked.

Learning algorithm

possibility of overcoming blocked keys should be still available. Thus, it is possible to write unknown words. A learning algorithm could remember these words and in turn expand the dictionary. This means, next time writing this *new* word, the user would not have to overcome the resistance of the keys because the word is included in the dictionary.

The Users' Feedback

Positive feedback

The feedback we received was positive. Users said the use of the system was interesting and enjoyable. Further they said that the prototype and the GUI were easy to understand and that it was possible to work fast and without disturbing interruptions. Furthermore, the participants typed carelessly, which gave the feeling of faster typing compared to the cycles with deactivated feedback.

Beginners that are concentrated on the key labels as well as unpracticed fingers can profit from this invention. DEKtf giving confidence during typing and the complete blocking of keys, in case a word was not in the dictionary, was mentioned as positive. The prototype fulfills the requirements to conduct high quality user studies, however, it is

Use in mobile phones

still to be tested as an end-user product. The small size required for the integration in mobile phones and the costs for the realization have to be considered.

Multiple assigned keys

DEKtf avoided small mistakes that could went unnoticed and helped to discover errors in time. However, one user mentioned that our approach would be more interesting for a full keyboard. A full keyboard would work better than our 12-key prototype because applications where each key was only assigned one function could detect more errors. In our system a key kept unblocked when another word with the same key combination existed in the dictionary, even if the typed key did not belong to the word that had to be written. For example: When an 'e' made sense but 'd' and 'f' not, the 'def' key was not blocked.

game approach

DEKtf as a game approach was considered because during completing the long words task (task six), it was more like a game to find the only unblocked key, without thinking what had to be typed.

After the user test 62% of the participants agreed and 17% strongly agreed that using tactile feedback is a usefull approach (figure 4.12).

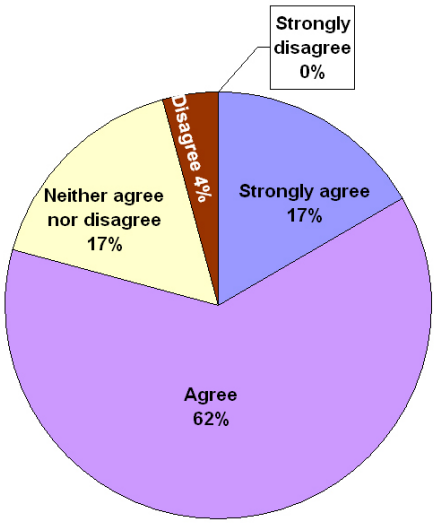

Figure 4.12: Using tactile feedback is generally a useful approach.

Chapter 5

Summary and Future Work

"If we knew what it was we were doing, it would not be called research, would it?"

—Albert Einstein

This chapter summarizes the work performed as part of this thesis. The last section describes concepts and ideas that could not be realized in the available time. Thus, suggesting several ideas for further development of DEKtf.

5.1 Summary and Contributions

Development Idea DEKtf is a new kind of keypad input device that implements the idea of interactive tactile feedback. The system is a hardware-based error prevention method. It can warn of typing errors before they are actually typed in. Additionally, because of the possibility of overcoming the feedback, it is more flexible than the already existing limited prevention methods.

Technical Realization Each key of the keypad has a variable key resistance. The resistance change is controlled by different tasks depending blocking algo-

rithms. Therefore, if a key makes no sense in the context, the algorithm sends a signal to block the key. For instance, the user tries to press one of the blocked keys, he feels the feedback, and can reconsider his action. That way DEKtf can avoid errors even before they are made.

DEKtf - A New Approach We compared our idea to existing approaches of keyboard modifications and software error prevention methods. An interacting keyboard like DEKtf did not exist. As a result we decided to build a new input device that combines tactile feedback controlled with error prevention algorithms.

Evaluation Finally, we conducted user evaluations in form of video analysis and questionnaires to define whether we reached our design goals. The studies showed that 30% of mistakes happening without tactile feedback can be avoided with the aid of DEKtf. Only one participant did not agree that DEKtf helps avoiding errors. These results show that we were sucessful in creating a new kind of typing error prevention device.

Visual Keypad As a side effect of our research, we also found out that an additional interactive visual keypad on the screen could be interesting for game approaches. During typing longer words, only one or two keys kept unblocked. All other keys were grayed out. The participants had a lot of fun, trying to press as fast as possible the available key, even without thinking about the word they had to write. It was more like a reaction test where only the visual feedback advised the user how to write the last part of a longer word. Furthermore we found out that hiding the visual keys, which are blocked on the prototype, can prevent errors when users are focused on the screen.

Target Group DEKtf can support beginners, who are learning to type, as well as advanced writers who are concentrating too much on the keyboard. Accessibility is another field of application. Especially for elder and disabled users that have problems to use a normal keyboard because of an impairment of the mo-

tor nerves, DEKtf is an useful approach. The tactile feedback of DEKtf prevents the user from inadvertent pressing of neighboring keys. That way DEKtf can support the user typing the intended key and prevent errors.

5.2 Future Work

Although DEKtf is already a complete system, it is just the beginning of the research in a wide area of interactive input devices for typing-error prevention. Here, we will present some ideas for further development based on DEKtf.

5.2.1 Keypad

A second Arduino board could be added to the keypad to control the locking of the '*' and '#' keys. For other applications it could be useful to block all available keys. It is still necessary to reduce the noise originated from the magnets of the keypad. The tests we made with springs and foam material failed because the resistance of the keys without activated feedback increased too much.

Second Arduino

5.2.2 Dictionary Mode

For the further development of the dictionary mode the integration of already existing live correcting methods (see section 1.4) of mobile phones should be researched in more detail. Furthermore, the user should be able to type unknown words by overcoming the tactile feedback. A learning algorithm could remember these words and in turn expand the dictionary.

Integration of error correction methods

Instead of multiple presses of the 'change' button, only one push of the button would suffice to launch a list of words with the same key combination and the user could choose the intended word directly by means of a numbered key. A dictionary with more than 6000 words is advisable.

List of words with same key combination

5.2.3 Data Mode

Temporal limitations

A new blocking algorithm has to be developed for the date entry. This algorithm should control the user's typing in consideration of specific temporal limitations and grammatical correctness. For instance, when the user types in "31" in the day field, only matching months should be allowed to be entered subsequently, such as January or March, since February or April, for example, do not have 31 days.

5.2.4 Alternative Prototypes

Reduce noise

There may be alternative techniques for realizing the keypad than those described in chapter 3. There may be a solution that would produce less noise. In case we kept the approach of using magnets, research could be done on how to reduce the sound of the magnets, and a slim keyboard could be set under the magnets instead of the switch matrix we chose.

5.2.5 Full Keyboard

Possible application areas

After the evaluation of the prototype it turned out that the implementation of a full keyboard could be desirable. Programming editors that have a strict grammar depending on the programming language, and text editors with XT9 system (section 1.4) and a huge dictionary could be an interesting application for a full size DEKtf keyboard. One thing we would have to consider is that the user is able to use shortcuts in every system state.

5.2.6 Evaluation

Users copy data from paper

In future user tests the user should copy data from a paper to get his focus away from the screen. The results could be different than the results we achieved with our user tests. The log file should be extended as to include the data to be

copied as well. This would make it easier for the analyzing tool to find the errors made by the user. Additionally, the user should be asked before the user test if tactile feedback is a useful approach, to see if the system changes the opinion of the participant.

Extended log file

Appendix A

Questionnaire

User Test – Haptic Keyboard Prototype for Data Entry – A. Hoffmann

media computing group

Questionnaire after the User Test of DEKtf
(Data Entry on a Keyboard with Tactile Feedback):

Age:

Studies:

Gender:

Male	Female

SMS per month

<5	5-20	20-50	50-100	>100

How often do you use T9?

Never	Sometimes	Always

Previous T9 experience

None	Little	Normal	Advanced	Professional

Touch typing: How many fingers do you use while typing on a normal keyboard?

1	2	3-6	8-10	10

I am typing a lot of text on my computer, palmtop/mobile device or mobile phone.

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

Describe your daily typing tasks (select multiple answers if applicable)?

1) Secretary (Office Applications)	2) Programmer	3) Worker (e-mail, Internet)	4)Gamer (games)	5) Chatter (chat rooms and/or instant messengers)
6) Mobile working (Palm, iPhone, ...)	7) Caller (SMS, Calling)	8) Other:		

About the hardware:

I felt comfortable using the keypad.

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

If you disagree or strongly disagree, please give a short explanation.

The labels on the key on the keypad prototype hardware were very important.

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

Figure A.1: Questionnaire User Test Page 1

User Test – Haptic Keyboard Prototype for Data Entry – A. Hoffmann

When did you look for the labels on the keypad prototype?

Never	Only at the beginning	From time to time	All the time

Which fingers did you use to type on the keypad prototype: (if you don't remember, no problem, just leave this question blank.)

Thumb	Index finger	Middle finger	Ring finger	Pinky

The keys were easy to press (pressure resistance).

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

The loudness of the keyboard was disturbing.

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

How would you improve the keypad prototype (Hardware)?

Did you experience hardware problems during the user test?

Did you miss any functionality with the Keypad Prototype (Hardware)?

About the software:

The GUI (Graphical User Interface, what you see on the monitor) was clearly structured and easy to use.

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

The labels on the monitor keypad were important to use the prototype.

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

How often did you look at the labels on the monitor keypad?

Never	Only at the beginning	From time to time	All the time

Do you have any suggestions how we could improve the Software or Graphical User Interface?

Did you experience software problems during the user test?

Figure A.2: Questionnaire User Test Page 2

User Test – Haptic Keyboard Prototype for Data Entry – A. Hoffmann

Did you miss any functionality with the Software Prototype?

Generally questions about the whole system:

What did you like about the system TEKtf?

Using tactile feedback is generally a useful approach.

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree
Comments:				

A system like TEKtf would help to avoid errors during writing long letters?

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

A system like the TEKtf prototype could be useful to avoid errors when writing Short Messages (SMS)?

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

A cell phone with haptic keys could be very useful (integrated in the mobile phone keypad)?

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

Imagine applying these principles to a full keyboard. All keys without sense in the context would be blocked.

A system like this would help to avoid errors.

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

A system like this would help to speed-up writing

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

Suggestions, ideas, improvements, comments:

Thank you for participating in this User Test.

Page 3 of 3

Figure A.3: Questionnaire User Test Page 3

Appendix B

Main Class

```
import java.io.*;
import java.util.*;
import javax.comm.*;
import java.awt.*;
import javax.swing.*;
import ft.*;


public class dektf {
  public void FTDemo(){
    // This function loads the dictionary
       and initializes the variables for
       the dictionary mode
  }

  private int control(int c){
    // This funcions receives a key and
       controls if it makes sense in the
       context for the dictionary-mode. An
       integer gives the result back.
  }

  private int hBE(String command){
    // This is the main procedure of the
       program. Here every pressed key gets
        its meaning. This function provides
        the demonstrated entry in the input
        field. Furthermore, in the Multi-
       Tap mode the timertask() gets
       started and in the Data mode the
       format of the input field is defined
       .
  }

  private void nextkey(){
    // This method unlocks all keys and
       sends every key as an integer to the
        control function. All keys that
       make no sense in the context are
       saved in a string. This string is
       send to the sent function.
  }

  public void button(char fi){
```

```
  // This function receives a char with
     the name of the key that was pressed
      on the hardware keypad. The
     belonging visual button receives
     then a ''doClick'' event. In case we
      are in the dictionary mode the
     nextkey function is called to see
     which keys have to be blocked next.
     In addition this function marks the
     active word.
}

public void timeover(int i){
  // Function for the Multi-Tap Mode.
     This function is started when 800ms
     after the last button press have
     been passed. It reinitializes the
     keycounter, that counts how many
     times a button was pressed, the
     lastchar, that remebers the last
     pressed key, and it sets the timeron
      variable to false.
}

public int lastspace(String space){
  // This function finds the last space
     in the text input field. It is
     called to delete words in the
     dictionary mode.
}

public class commListener implements
   SerialPortEventListener{

  public void serialEvent(SerialPortEvent
      event) {
    // This function checks if data from
       the Arduino is available. In case
       data is available it is saved in a
        character. Additionally the key
       is sent to the function textfile()
  }
}
```

```
public void send(String nachricht){  //
   Sends string to Keypad
  // This function sends a message to the
      Arduino.
}

public void textfile(String p){
  // This function saves the actual time
     when receiving the string and saves
     this information together with the
     string to a log file that is
     initialized when starting the
     program.
}

public class label{
  // Labels the Keybuttons of the visual
     Keypad

    public void init(){
      // Inizalize the labels to mode 0 (
         dictionary mode)
    }

    public void set(int modeint){
      // Changes the labels of the visual
          keypad to the depending mode.
         This information is send to the
         log file (textfile()) to
         remember the mode change.
    }

}

public void next(int number){
  // The task counter is actualized,
     depending if the next or back button
      was pressed. The new task
     description gets displayed in the
     text pane. All variables that belong
      to the dictionary mode get
     reinitialized and the text input
     field emptied.
}
```

```
  public static void main(String[] args) {
    // Main function that creates the log
       file with the filename composed of
       date and time. It starts the FTDemo
       function and the GUI.

    SwingUtilities.invokeLater(new Runnable
       () {
      // Opens the connection to the
         Arduino and starts an EvenListener
          : commlistener(). At the end the
         Keyboard is reset.
    });
  }

class Task extends TimerTask{
  public void run(){
    // Creates a new dektf object and
       starts the timeover(0) function.
  }
}
```

Appendix C

Schematic Diagram of Wiring the Prototype

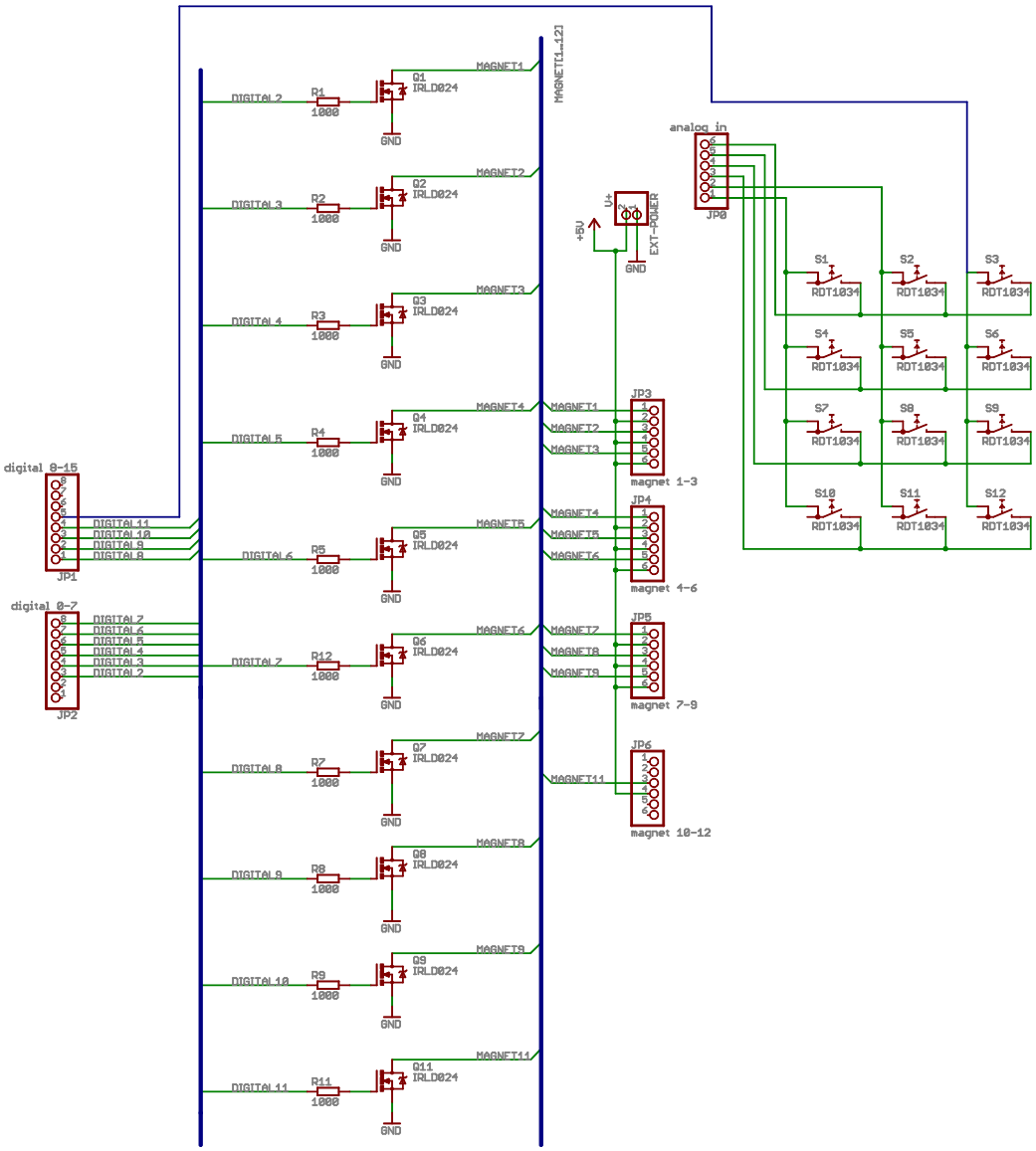

Figure C.1: Schematic Diagram of Wiring the Prototype

Appendix D

Flowchart of Data Entry Blocking Algorithm

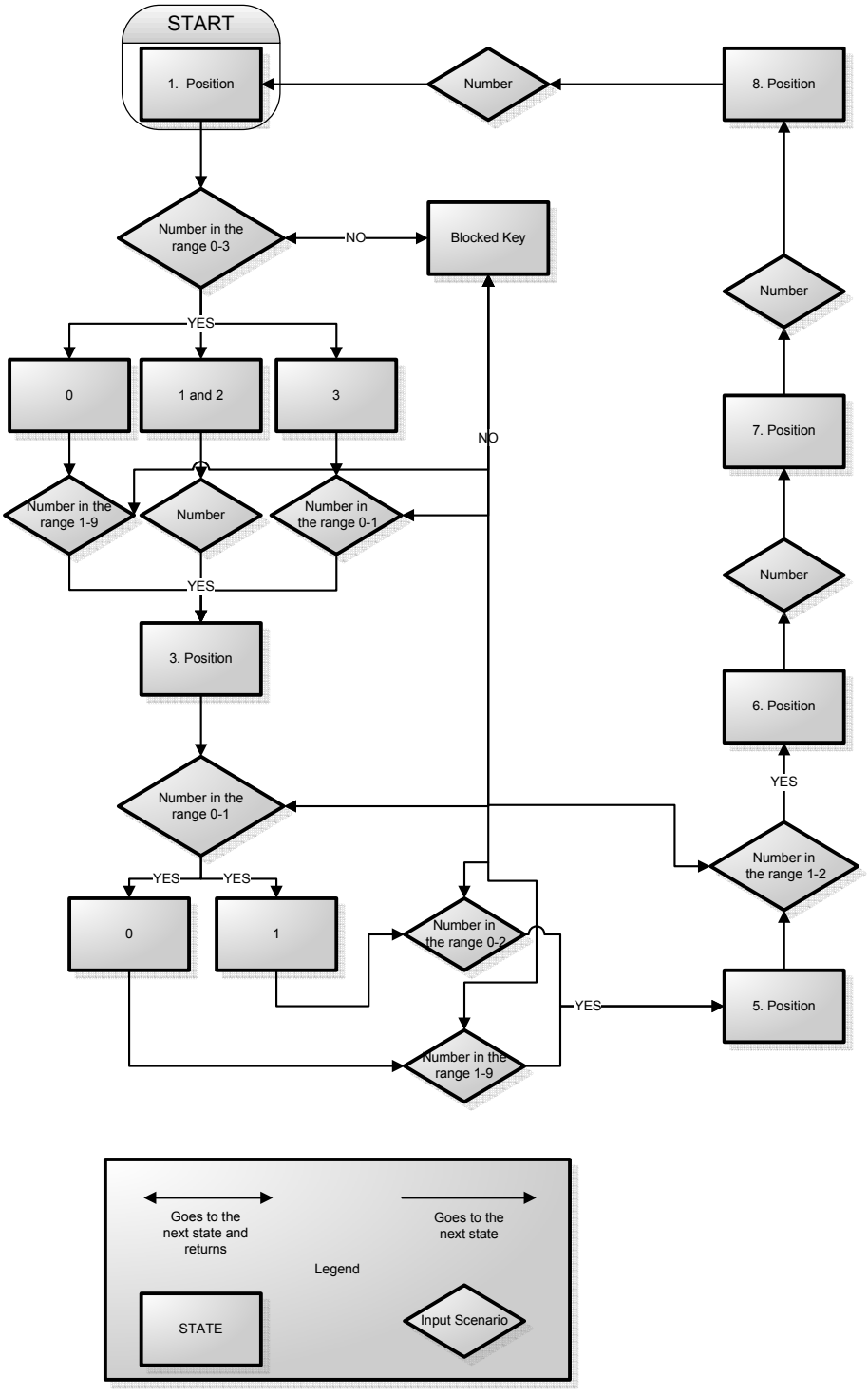

Figure D.1: Flowchart of Data Entry Blocking Algorithm

Bibliography

M. Banzi, D. Cuartielles, D. Mellis, and N. Zambetti. Arduino, 2005. URL `http://www.arduino.cc`.

Richard A. Bolt. "put-that-there": Voice and gesture at the graphics interface. In *International Conference on Computer Graphics and Interactive Techniques*, pages 262–270, July 1980.

P. C. Buzing. Comparing different keyboard layouts: Aspects of qwerty, dvorak, and alphabetical keyboards, 2003.

Vasilios G. Chouvardas, Amalia N. Miliou, and et al. Tactile displays: a short overview and recent developments, 2005.

Scotland Computing Science Department of the University of Glasgow. iphone-haptics, 2008. URL `http://code.google.com/p/iphone-haptics/`.

CompuTouchAS. The hsd (haptic solution for disabled) report, 2002.

Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-computer interaction*. Prentice Hall, New York, NY, USA, 1993. ISBN 0-13-458266-7.

C. Doerrer and R. Werthschuetzky. Simulating push-buttons using a haptic display: Requirements on force resolution and force-displacement curve. In *EuroHaptics, Edinburgh, UK*, 2002.

Douglas C. Engelbart. X-y position indicator for a display system, November 1970.

E.B. Goldstein. *Sensation and perception*. Brooks/Cole, 4 edition, 1996.

Joel Gerard Goodwin, Scott Harlan Isensee, Ricky Lee Poston, and I-hsing Tsao. Tactile feedback keyboard, April 2001.

Fabian Hemmert, Gesche Joost, André Knörig, and Reto Wettach. Dynamic knobs: shape change as a means of interaction on a mobile phone. In *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, pages 2309–2314, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-012-X.

Eve Hoggan, Stephen A. Brewster, and Jody Johnston. Investigating the effectiveness of tactile feedback for mobile touchscreens. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1573–1582, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: http://doi.acm.org/10.1145/1357054.1357300.

K.O. Johnson and J.R. Phillips. Tactile spatial resolution. *I. Two-Point discrimination, Gap Detection, Grating Resolution, and Letter Recognition. Journal of Neurophysiology.*, 46 (6):1177–1191, 1981.

Joseph "Jofish" Kaye. Making scents: aromatic output for hci. *interactions*, 11(1):48–61, 2004. ISSN 1072-5520.

Sang-Youn Kim, Jinah Park, and Dong-Soo Kwon. Palpation simulator for laparoscopic surgery with haptic feedback. In *Proc. of International conference Biomedical Engineering*, pages 478–482, 2004.

G. Michelitsch, J. Williams, M. Osen, B. Jimenez, and S. Rapp. Haptic chameleon: a new concept of shape-changing user interface controls with force feedback. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1305–1308, New York, NY, USA, 2004. ACM. ISBN 1-58113-703-6.

Stefan Münch and Rüdiger Dillmann. Haptic output in multimodal user interfaces. In *IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces*, pages 105–112, New York, NY, USA, 1997. ACM. ISBN 0-89791-839-8.

M.L. Nagurka, R. Marklin, and C. Liu. Design of a test rig for measurement of stiffness and damping characteristics of computer keyboard keys. In *Proceedings of the 1999 American Control Conference*, pages 1749–1753, 1999.

Communications Inc. Nuance. *Nuance XT9 Mobile Interface - Enabling text by any means*. 1000 dexter ave. n., ste. 300, seattle wa 98109, 2007.

Ian Oakley, Marilyn Rose McGee, Stephen Brewster, and Philip Gray. Putting the feel in 'look and feel'. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 415–422, New York, NY, USA, 2000. ACM. ISBN 1-58113-216-6.

Franz Penz and Manfred Tscheligi. The feelmouse: an interaction device with force feedback. In Stacey Ashlund, Kevin Mullet, Austin Henderson, Erik Hollnagel, and Ted N. White, editors, *INTERCHI Adjunct Proceedings*, pages 121–122. ACM, 1993. ISBN 0-89791-574-7.

Ely Rabin and Andrew Gordon. Tactile feedback contributes to consistency of finger movements during typing. *Experimental Brain Research*, 155:362–369(8), April 2004.

R. Rashedin and T. Meydan. Solenoid actuator for loudspeaker application, 2005.

Jef Raskin. *The humane interface : new directions for designing interactive systems*, chapter 2.3. Addison-Wesley, Reading, Mass. [u.a.], 2. print. edition, 2000. ISBN 0-201-37937-6.

Johanna Barbara Sattler. *The Converted Left-Hander or the "Knot" in the Brain*. Auer, Donauwörth, 1995.

R. Scheibe, M. Moehring, and B. Froehlich. Tactile feedback at the finger tips for improved direct interaction in immersive environments. *3dui*, 0, 2007.

John Sculley and John A. Byrne. *Odyssey*. Harper & Row Publishers, Inc., New York, NY, USA, 1987. ISBN 0-06-091527-7.

Leonard J. West. The standard and dvorak keyboards revisited: Direct measures of speed. Research in economics, Santa Fe Institute, May 1998.

Andrew Zipern. A keyboard that lets the supremely confident show disdain for qwerty. *The New York Times*, May 26 2005.

Index

Typeset May 28, 2008