České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačové grafiky a interakce

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Šimon Rudolf**

Studijní program: Otevřená informatika (magisterský)
Obor: Softwarové inženýrství

Název tématu: **Podpora přístupnosti v textovém procesoru**

Pokyny pro vypracování:

Prostudujte problematiku detekce myoelektrických signálů a způsoby, jak je možné využít elektromyografické senzory jako vstupní periferní zařízení počítače. Zaměřte se na asistivní technologie. Realizujte zásuvný modul (plug-in) do vybraného textového procesoru, který umožní běžný vstup a manipulaci s textem pomocí vstupních myoelektrických signálů. Navrhněte několik způsobů, jak mapovat rozsah vstupních hodnot na konkrétní akce uživatelského rozhraní. Realizovaný prototyp podrobte testům použitelnosti, ideálně s cílovou skupinou lidí s postižením horních končetin.

Seznam odborné literatury:

Susumu Harada, T. Scott Saponas, and James A. Landay. 2007. VoicePen: augmenting pen input with simultaneous non-linguisitic vocalization. In Proceedings of the 9th international conference on Multimodal interfaces (ICMI '07). ACM, New York, NY, USA, 178-185. DOI=10.1145/1322192.1322225 http://doi.acm.org/10.1145/1322192.1322225

Torsten Felzer and Bernd Freisleben. 2002. HaWCoS: the "hands-free" wheelchair control system. In Proceedings of the fifth international ACM conference on Assistive technologies (Assets '02). ACM, New York, NY, USA, 127-134. DOI=10.1145/638249.638273 http://doi.acm.org/10.1145/638249.638273

Tonio Wandmacher, Jean-Yves Antoine, Franck Poirier, and Jean-Paul D233;parte. 2008. Sibylle, An Assistive Communication System Adapting to the Context and Its User. ACM Trans. Access. Comput. 1, 1, Article 6 (May 2008), 30 pages. DOI=10.1145/1361203.1361209 http://doi.acm.org/10.1145/1361203.1361209

Vedoucí: Ing. Adam Sporka, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015


L.S.

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 25. 2. 2014

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

**Master's thesis**

# Accessibility Plug-in for a Word Processor

*Šimon Rudolf*

Supervisor: Ing. Adam Sporka, Ph.D.

Study Programme: Open Informatics

Field of Study: Software Engineering

May, 2015

# Acknowledgements

I would like to thank Adam Sporka, the supervisor of my thesis, for his worthwhile advices, patient guidance and devoted time. Big thanks also go to Antonín Pošusta, the developer of the hardware device, and Ondřej Poláček for his comments and recommendations. Finally, I would like to thank my girlfriend and family for their support in my studies.

This research has been supported by projects TextAble (LH12070; Ministry of Education, Youth and Sports of the Czech Rep.) and Design of special user interfaces (grant no. SGS13/213/OHK3/3T/13, FIS 161 – 832130C000).

## Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used. I have no objection to usage of this work in compliance with the act §60 Zákon c. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague on May 7, 2015

. . . . . .

# Abstract

Most computer users use standard peripherals, keyboard and mouse. It is difficult to imagine the work without these devices. However, a number of users exist who are unable to operate these devices due to their physical impairment. This thesis presents a solution for these users. As a part of project TextAble, this thesis deals with a problem of designing a virtual keyboard layout which is controlled by signals from myoelectric device developed in the Academy of Science of the Czech Republic.

The keyboard is developed for usage in Microsoft Word text processor as a virtual on-screen keyboard and provides not only typing of characters, it also allows users to navigate through the text, to format and edit it. Several scanning methods are implemented. Apart from the description of the developed application, and its testing, this thesis also contains a survey of assistive technologies used for text entry and some existing text entry systems.



# Abstrakt

Většina uživatelů počítačů ke své práci používá standardní zařízení jako je klávesnice nebo myš. Bez těchto periferií je velmi těžké si tuto práci představit. Existuje ovšem skupina uživatelů, která tyto zařízení používat nemůže, a to kvůli jejich tělesnému postižení. Tato práce nabízí řešení pro tyto uživatele. Jako součást projektu TextAble se totiž zabývá návrhem rozložení virtuální klávesnice, která bude ovládána myoelektrickými signály, které sníma zařízení vyvíjené na Akademii Věd České republiky.

Klávesnice je vyvinuta jako doplněk programu Microsoft Word, která ve formě virtuální klávesnice nabízí možnost nejen psát text, ale i formátování textu a navigaci v dokumentu. K ovládání rozhraní je implementováno několik metod skenování. Vedle popisu vyvinuté aplikace a jejího otestování obsahuje práce také rešerši metod používaných pro textový vstup se zaměřením na asistivní technologie.

# Contents

**List of Figures** **ix**

**List of Tables** **xi**

**1 Introduction** **1**

1.1 Motivation . . . . . 2

1.2 Organization of the Thesis . . . . . 2

**2 Background** **5**

2.1 Assistive Technology . . . . . 5

2.1.1 Selection Techniques . . . . . 7

2.1.2 Low-level Input Techniques . . . . . 9

2.1.3 Interaction Modalities . . . . . 10

2.2 Myoelectric Signals . . . . . 11

2.2.1 Myoelectric Devices . . . . . 11

**3 Requirements Analysis** **15**

3.1 User Requirements . . . . . 15

3.2 Functional Requirements . . . . . 16

3.3 Non-functional Requirements . . . . . 17

**4 Design** **19**

4.1 Types of Activities Supported . . . . . 21

4.2 Types Of Signals Supported . . . . . 21

4.3 Layout Prototypes . . . . . 22

4.3.1 Linear Scanning . . . . . 22

4.3.2 N-ary Scanning . . . . . 23

4.3.3 Two Dimensional Scanning . . . . . 24

4.3.4 Cursor Scanning . . . 25
4.3.5 Layout with Contexts . . . 26

**5 Implementation** **29**
5.1 Technologies . . . 29
5.2 Architecture . . . 31
5.3 The Algorithm . . . 32
5.4 Implemented application . . . 33
5.4.1 Application functionality . . . 34
5.4.2 Signals mapping . . . 36
5.4.3 Application appearance . . . 36

**6 Testing and Evaluation** **39**
6.1 User Test . . . 39
6.1.1 The First Session . . . 40
6.1.2 The Second Session . . . 40
6.1.3 Output from the Testing . . . 42
6.2 Comparison of the Designed Layouts . . . 44

**7 Conclusion** **49**

**Bibliography** **53**

**A User Manual** **55**
A.1 Configuration . . . 55
A.2 Controlling the Keyboard . . . 56
A.2.1 Mapping of Input Signals . . . 56

**B Content of the Attached CD** **59**

# List of Figures


2.1 Scanning. . . . . 8
2.2 Weight-balanced binary tree of items in an on-screen keyboard. . . . . 9
2.3 MYO armband. . . . . 12
2.4 Emotiv EPOC headset. . . . . 13
2.5 NIA device. . . . . 13

3.1 Use cases for the application. . . . . 16

4.1 Block diagram of the context. . . . . 20
4.2 Layout for Linear Scanning. . . . . 23
4.3 Layout for Binary Scanning. . . . . 24
4.4 Layout for Two Dimensional Scanning. . . . . 25
4.5 Layout for Cursor Scanning. . . . . 26
4.6 Layout for Optimized Cursor Scanning. . . . . 26
4.7 Difference in layouts based on context. . . . . 27

5.1 Ribbon menu in MS Word. . . . . 30
5.2 Ribbon menu in MS Word after pressing Alt key. . . . . 30
5.3 Ribbon menu in MS Word with displayed shortcuts. . . . . 30
5.4 UML diagram of the main parts of the application. . . . . 31
5.5 Inital screen of application. . . . . 37
5.6 Generated layout. . . . . 38
5.7 Application appearance. . . . . 38

6.1 Pseudo-QWERTY layout. . . . . 43
6.2 Pseudo-QWERTY layout with Undo. . . . . 43

A.1 Mapping of Input Signals. . . . . 57

# List of Tables

6.1 Number of interactions needed to write the text using only trigger input 45
6.2 Measured performance in simulation using only trigger input . . . . . 46
6.3 Number of interactions needed to write the text using analog input . . . 47
6.4 Measured performance in simulation using analog input . . . . . 47

# Chapter 1

# Introduction

Text input is one of the most important use cases when talking about interaction between the human and the computer. A typical user of computer commonly employs standard peripherals such as mouse or keyboard and it is hard to imagine working without them. If any of these devices breaks, all of a sudden, the user realizes, how dependent on that device he is. In short-term period, the user can handle that with the help of a virtual keyboard provided by the operating system in case of a broken keyboard or with help of touchpad in case of a broken mouse when working with notebook. Although it can be sufficient and comfortable solution for some users, it will probably be very inappropriate for most of them. Luckily, this is only a temporary state for them. After the device gets fixed or replaced by a new one, users can return to the usual way of controlling the computer.

However, a number of users exist who are unable to operate these standard devices due to their physical impairment at all. For them computer is not as good assistant as it could be. Computers are not designed to provide sufficient support for them. If you ever tried to write with help of built-in mouse-controlled software keyboard, you know that it is not very easy and comfortable to write more than a few sentences. Entering text is then especially challenging for people who struggle with common interaction methods due to their impairments. Handicapped people being faced with this problem every day.

Fortunately there are nowadays many both hardware and software products which simplify the interaction with computers. Joysticks or devices, which are controlled by movement of head or eyes, can be examples of such hardware products, virtual keyboards could be examples of software products. The application developed in this thesis is one of them.

This thesis deals with the problem of text input for motor-impaired people. It is being

carried out in the context of the project TextAble whose aim is to develop a limited input device based on myoelectric signals. The goal is to develop a virtual keyboard that could be employed in Microsoft Word allowing the user to write and format the text as well as navigate in the document.

## 1.1 Motivation

Nowadays, computers are a fixed part of our life and we can hardly imagine it without them. We use it at work and when studying, in our free time for communicating with friends, reading news, shopping etc. It is not very difficult to use the computer for most of the people, but as I have already mentioned, there is a group of users that can't use computer or it is nearly impossible for them to do so. Assistive technology could be a solution for them, as it enables or simplifies the interaction with computers to users with significant issues. Speech commands can be suitable for people with motor impairments, blind people can use braille displays to operate the computer.

It is a task for the healthy people to enable the less lucky ones to use the computers as comfortably and effectively as possible, because these people need to feel that they belong to the society to the same degree that we do. The need of socialization is one of the basic needs of a human being.

The main goal of this thesis is therefore developing a virtual keyboard which will simplify the text input for motor-impaired people, based on myoelectric signals. The developed application should also be tested with users from target group and prove whether or not this approach is suitable.

## 1.2 Organization of the Thesis

Chapter 2 contains basic information about myoelectric signals and a survey of existing text input methods. Assistive technologies which deal with the problem of text entry is also presented.

Chapter 3 describes the application's requirements. User requirements as well as functional and non-functional requirements are presented.

The model of the application is described in Chapter 4. Layout prototypes are pre-

sented with their description, advantages and disadvantages.

Implementation of the application is discussed in Chapter 5. The application's architecture and the integration with the hardware myoelectric device is described.

Chapter 6 contains the description of testing the implemented application and its evaluation. Also findings from testing are discussed and possible improvements of the application are presented.

Finally Chapter 7 provides a presentation of the conclusions of this work.

Appendix A contains the user manual for the application. Content of the CD attached to this document can be found in appendix B.

# Chapter 2

# Background

History of computer text entry methods dates back to the 18th century. The first idea of a typewriter was mentiond in 1714 by Henry Mill. However, the first typewriter was not constructed until 1808, when an Italian called Pellegrino Turri developed a device to enable his blind relative to write. The first typewriter which was commercially successful was constructed in 1870 by Rasmus Malling-Hansen of Denmark. It was very popular in Europe and it was used in offices in London as late as 1909 (Mares, 1909).

In 1874 a typewriter was developed, which has had the largest influence on the distinctive layout of today's keyboards. It was a typewriter with a QWERTY layout made by Christopher Latham Sholes and Carlos Glidden (Siioles, 1868). With several modifications it is still being used today. Many attempts to improve the layout and make it quicker have been made but all alternatives fail to provide very significant advantages (Liebowitz and Margolis, 1990). Probably the most known attempt of creating a different layout is called Dvorak layout and is named after its inventor Dr. August Dvorak. It was patented in 1936 and its proponents claim that this layout uses less finger motion and increases the type rate compared to the standard QWERTY layout (Liebowitz and Margolis, 1990). However most of today's keyboards have a QWERTY layout.

## 2.1 Assistive Technology

Regardless of the type of the keyboard, a common feature of physical keyboards is the fact that they are controlled by a physical key press. But it is, of course, not necessarily the only way of text input. Today there are many other techniques of text input and they

can even be more appropriate in some special cases, e.g. mobile environment or assistive technology.

Assistive technology is an umbrella term used for various devices and techniques which enable or simplify the interaction with computers for users with important issues. There exists many various interaction techniques that are based on limited movement of hands (e.g. single-switch interfaces), on eye tracking, on detection of myoelectric signals or voice recognition.

When using these limited hardware interfaces, only a small amount of control signals (both in count and diversity) are available. User also needs a specialized software interface (middleware) that maps the signals from the hardware to a regular user's interface actions. The form of such middleware is usually an on-screen keyboard.

There are different types of limited input devices in the assistive technology, ranging from simple triggers and on/off switches to joysticks and eye trackers. These devices differ in modes of operation. Single-switch interfaces (Felzer et al., 2010) enable the selection performing the scan step when talking about scanning. Direct selection could be performed by eye trackers or joystick.

To increase the type rate, an optimal layout of characters for a given keyboard can be found by a computer algorithm. The algorithm evaluates the layout according to some optimization criterion. The optimization criterion can reflects either the number of estimated keystrokes (for direct-selection keyboard), or the number of scan steps and scan selections (for scanning keyboards), or the predicted cursor movement (for keyboards operated by a pointing device).

The layout can be constructed with respect to a statistical language model. Letters with higher frequencies should be placed so they could be reached easier then letters with lower frequencies. Also letters which form common *n-grams* or words should be placed close to each other, when the selection method can utilize this. Finding the layout with the maximal type rate is often an NP-complete optimization problem (Lesher et al., 1998).

A search heuristic based on genetic algorithm can be used for finding the layout with better performance. Such algorithm was used in the work of Harbusch and Kuhn (Harbusch and Kühn, 2003). The whole alphabet in ambiguous keyboard was assigned to only three keys. For finding this distribution of letters to these three keys was used the genetic algorithm.

Another approach is to restrict the search space when using the ambiguous keyboard. An important work has been done for layouts which follow the alphabetical order. For example, Gong and Tarasewich (Gong and Tarasewich, 2011) presented optimal layouts

for 4, 8, 9 and 12-key ambiguous keyboards. Similar work has been done by Kim et al. (Kim and Kim, 2009), who compared several methods for layout optimization of multi-tap text input. MacKenzie and Felzer (MacKenzie and Felzer, 2010) showed that a three-key layout is optimal for scanning an ambiguous keyboard.

### 2.1.1 Selection Techniques

There exists several methods which are used for input selection in augmentative assistive devices. I will mention direct selection, encoding and scanning.

**Direct Selection** This method is much the same as the ordinary way of pressing a key. Beside software changes, someone with a minor disability or poor motor control can use a keyguard. It is a template that fits over a standard keyboard with openings above the key tops. To press the key, the user has to put his hand on this template and push a finger through one of the holes. The hole then avoids minor errors caused by tremors. Also above mentioned software changes can result in ignoring errors, such as brief key presses or repeated activations within a short time frame. Next example could be the situation when the shift key is first pressed and then software interprets the next key press as if the shift key was still held down (Jones, 1998).

**Encoding** Next method I would like to mention is encoding. In this case a sequence of actions produced by user results in the coded input. Morse code is a well-known example of encoding method. This code consists of a sequence of short transmissions (dots), longer transmissions (dashes) and a pause to indicate inter-symbol spacing. Similar scheme or even this scheme could be used by users with disabilities. It can offer a number of advantages to someone with motor impairment. For example when user has learnt the code, he no longer needs to concentrate on the input device (Jones, 1998). Some training is of course needed before a user becomes capable of this. For first time users, it is slow and unreliable, as it totally depends on the user internalising the meaning of the sequences. Visual representation of the codes can help users during the training.

**Scanning** A scanning method uses an array of characters just like a standard keyboard. It differs mainly in a select time of a key. In standard keyboard any key may be selected practically at the same time, i.e. it has random access. The scanning keyboard differs in both layout and method of selecting which key to activate (Jones, 1998).

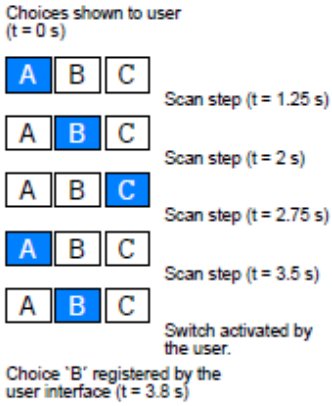


Figure 2.1: Scanning.

Scanning works as a sequential access system. Each key in the layout could be selected. Keys are focused one after another until the required one is reached. This could be done automatically or by manually. The user then activates their switch and the character at that position is selected. See Figure 2.1. The easiest type of scanning is linear scanning, where at each moment only one key is focused. Keys at the end of the sequence are hard to reach as user needs to perform many scan steps. This is obviously too slow to be acceptable. There exist ways how to improve this method. One way of speeding up the selection process is when using grouping. Each group is focussed sequentially and when user selects one of that group, the scan continues among the keys in that group. Recognition rather than recall is one of the most important advantages when using scanning keyboards. This approach has less cognitive demands when compared to the systems using some form of coding (Jones, 1998).

For example selecting with use of a single-trigger interface requires scanning (MacKenzie and Felzer, 2010). An interface consisting of two switches allows automatic scanning, selection performed by one switch and undo by triggering the other, or manual scanning performed by the first switch and the other then confirms. Analogue value input could be performed by a pressure-sensitive puff by altering the intensity of the blowing.

There exists many ways of how the scanning can be implemented. Here is the list of some:

- Selection by scanning: Waiting until the right choice is shown, then triggering.
- Cursor selection: Moving the cursor by arrow keys and pressing Enter.

- 2D scannnig: The user chooses the row by and then the column by scanning.
- Binary scannnig: Items are recursively split into two halves until a single item is highlighted.
- Pointer selection: Moving a screen cursor over the target and pressing a trigger or hovering.

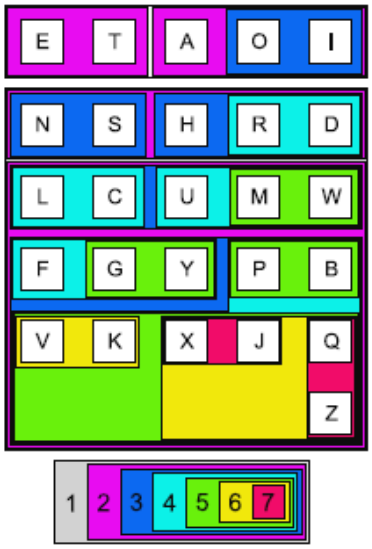


Figure 2.2: Weight-balanced binary tree of items in an on-screen keyboard.

The selection can be 1-of-$N$, i.e. all options are shown at once, or hierarchical, where the user is given a choice to select a group, then a subgroup etc., until the user reaches the actual option to choose from. Figure 2.2 demonstrates how item are organized in a binary hierarchical scanning, optimized according to the letter frequency.

### 2.1.2 Low-level Input Techniques

There are many types of physical interaction devices. Here is an overview of some:

- Trigger. A switch whose function is to enable the user to send a trigger signal. It can be realized as a simple button and the signal is sent when the button is pressed by a palm of the hand. Pressure sensor that is mounted on user's forehead can be also used, when the user can't use the simple button. Triggers have a constant selection time.

- On/off switch. It is similar to a trigger. It differs in possibility of reporting the time when being pressed and also released.

- Group of triggers. This interaction device consists of several independent triggers (or on/off switch). The functions assigned to these triggers typically complement each other. An example could be the 4-way arrow keys and a confirmation key.

- Analog input. These devices can produce a value, which can be then mapped by middleware to some visual feature in the user interface. The device quantify some of the action performed by the user, e.g. the amount of pressure applied to a pad..

### 2.1.3 Interaction Modalities

There are many interaction modalities which could be used by motor-impaired people to control the computer. I will describe the most common modalities for the text input in the following part.

#### Acoustic Interaction

Acoustic input modality is one of the possible solutions for text input. Nowadays there are many automatic speech recognition (ASR) systems, which can be used for entering text by dictation in word-by-word manner or letter-by-letter manner. Motor-impaired people reach similar performance as able-bodied people with an ASR system (Sears et al., 2001). However, users with speech impairments (e.g. dysarthria) report poor accuracy of ASR systems.

#### Gaze Interaction

Gaze interaction works on the principle of computing the user's focal point by measuring the position of eyes. This can be used, for example, for controlling the cursor and has been used in many virtual keyboards. The issue with the absence of some kind of selection technique could be solved by using dwell-time approach. In this method, the selection is made after predefined timeout, while the pointer stays in a small radius.

### Interaction via Bio Signals

The last technique which I would like to mention is interaction via bio signals. There are two main approaches that could be used for text input: *intentional muscle contractions* and *brain-computer interaction*. For the first possibility, EMG signals are measured. They are produced by skeletal muscles. In the second approach, EEG signal are recorded along the scalp.

## 2.2 Myoelectric Signals

Myoelectric impulses are an electrical activity produced by skeletal muscles. They can be detected as electrical potential generated by muscle cells when these cells are electrically or neurologically activated. To generate these signals, it is not necessary to physically move with the target part of the body, one only has to clench the muscle. Myoelectric signals have frequencies ranging from a few hertz to about 300 Hz, and voltages ranging from approximately 10 microvolts to 1 millivolt.

Myoelectric signals are of interest to the developers of prosthetic devices, such as artificial limbs. The signals can also be used to facilitate the operation of a computer using small voluntary muscle movements, such as blinking the eyelids.

Myoelectric signals are detected by placing electrodes on the skin. Two electrodes are positioned in a way there is a voltage between them when a myoelectric signal occurs. One of them is conventionally called active (the more active part of the nerve or muscle), the second is called reference (the less electrically active areas). Voltage change between the electrodes is then measured and evaluated. If the region under the active electrode is charged negatively, the recorded deflection is negative, while if it is charged positively, then the deflection is positive. If the voltage between the two electrodes stays the same, then this state is identical with the state of the rest. However such a state practically never occurs, since between the sensing electrodes and the muscles is the tissue which changes the characteristics of the measured potentials.

### 2.2.1 Myoelectric Devices

Myoelectric devices are analyzed in many studies and researches. Most of them, however, focus on the problem of controlling the prostheses of missing limbs or their parts. These

products like prostheses could be used as a basis to control the mouse cursor or keyboard itself, but their price (tens of thousands of US dollars) exceeds the financial possibilities of the target users. Therefore, instead of these, I will focus on devices that do not offer as many options as the whole prosthesis but on the other hand are more affordable and practical.

Probably the most known device in this area is called MYO. MYO is a wireless armband which can be fastened to the forearm of the left or the right hand. Then, it records myoelectric signals, processes them and send them to the evaluation program via Bluetooth technology. It is compatible with Mac, Windows, iOS and Android operating systems[1]. The MYO armband is shown in Figure2.3.

Figure 2.3: MYO armband.

Next product that I would like to mention is the Emotiv EPOC. It is a headset, i.e. it is attached to the head, and it processes signals using electroencephalography (EEG), electromyography (EMG) and 2 gyroscopes. All three components are combined to make the final output, which is then used for triggering mapped events on the target device. It is clear that it is not a purely EMG device, nor can it be (easily) separated. The product is suitable for cursor controlling, mainly thanks to the gyroscope, which allows a simple interpretation of movement of the head in all directions [2]. The Emotiv EPOC headsed is shown in Figure2.4.

[1]https://www.thalmic.com/en/myo/techspecs
[2]https://emotiv.com/epoc.php

Figure 2.4: Emotiv EPOC headset.

The last device I would like to mention is NIA. The name is an abbreviation for Neural Impulse Actuator. It is a brain-computer interface (BCI) device developed by OCZ Technology. BCI devices try to avoid using the classic input devices like keyboard and mouse by reading electrical activity from the user's head, preferably via EEG. NIA has been developed as a gaming device which communicates via Universal Serial Bus (USB). As of May 27, 2011, the NIA is no longer being manufactured and has been end-of-lifed[3]. The Emotiv EPOC headsed is shown in Figure2.5.

Figure 2.5: NIA device.

[3]http://www.ocztechnology.com/nia-game-controller.html

# Chapter 3

# Requirements Analysis

This section describes the attributes which are required from the software. Very important part of this are the requirements on the user as the application is being developed for motor-impaired people. These user are quite a specific group and a detailed analysis has to be made. Also functional and non-functional requirements are presented.

## 3.1 User Requirements

When developing a software application or, in general, doing any work connected with Human-Computer Interaction (HCI), it is very important to know who are the target users, what are their experiences and abilities. Then, the developer can be aware of their needs and be less influenced by his own assumptions.

Project TextAble is aimed on people with impairment of upper limbs who cannot comfortably, or at all use standard computer peripherals like mouse and keyboard and therefore are not able to achieve the same type rate and accuracy as healthy people. Their abilities can vary widely, of course. Some persons are able to move their hands, but not to type. Others cannot move their hands at all, but are able to press something. Last but not least, there are also persons who are not able to move anything else than their eyes.

The time a disabled person needs to perform an action may even vary for him, depending for example on his current state of health or mood (Sporka et al., 2011). The application should therefore try to reduce the number of steps and the time to achieve a goal (e.g. write a character). Also, the application should offer several ways for achieving

the goal to allow the user to choose the most appropriate one for him.

I have to mention that the target users of the text entry application developed in this thesis are people with physical disabilities, not mental disorders. A completely different approach would certainly be required if the application was targeted to mentally handicapped users.

## 3.2 Functional Requirements

There are several basic operations which have to be supported by the application. Some of them are generally expected to be found in any text processor. In Figure 3.1 these requirements are shown in form of a Use Case diagram.

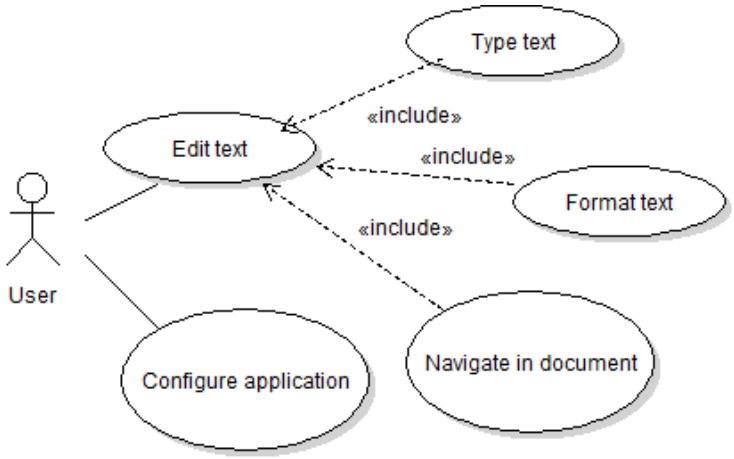


Figure 3.1: Use cases for the application.

1. **Type text**
   Definition: Program allows to input text in the document.
   Description: User is able to put text into document. He is also able to delete the text and perform block operations (e.g. Copy/Paste)

2. **Format text**
   Definition: Program allows formatting text.
   Description: User is able to format the text. Both the one he is writing or the one that is already written.

(a) **Decorate text**
Definition: Program allows decorating the text.
Description: User is able to set text decorators on text, that means using *Bold, Italic* and *Underline*. He can also set the font size.

(b) **Text align**
Definition: Program allows setting the align of the text.
Description: User is able to set the align of the text to right, left or center.

3. **Navigate in document**
Definition: Program allows moving of the caret in document.
Description: User is able to move through the document without a mouse.

(a) **Move the caret without marking the text**
Definition: Program allows moving of the caret in document without marking the text.
Description: User is able to move through the document without marking the text.

(b) **Move the caret with marking the text**
Definition: Program allows moving of the caret in document and mark the text.
Description: User is able to move through the document with marking the text.

4. **Configure the application**
Definition: Program allows to configure the application according to user's preferences.
Description: User is able to choose, what the layout should look like, choose the behavior of the user interface (e.g. type of scanning) or number the of input signals.

## 3.3 Non-functional Requirements

There are also several non-functional requirements which have to be met.

- **Limited number of input signals**
The application should be controllable by using only a few signals according to the

possibilities of the hardware device.

- **Target environment**
  The application should be developed for to be used in Microsoft Word text processor, that means it will run on Microsoft Window operating system only.

- **Respect UI standards**
  The application should respect standards typical for user interfaces. For example, the designed user interface should use standard icons and symbols.

# Chapter 4

# Design

In this chapter, I present a description of the design of the application which will operate as a middleware between the hardware device and and target text processor. It is a generic on-screen virtual keyboard whose layout is built based on a number of parameters, including the available user's physical interaction unit or the type of text-processing work the user is performing.

Text entry and it's editing operated by these limited input devices is a very time-consuming effort. Thus it is very important to optimize as much as possible the corresponding middleware in order to maximize the possible speed of typing. By optimizing the layout it could takes less amount of time to access different options presented.

As it was already mentioned in the previous chapter, it is known that the conditions of the motor-impaired people and therefore also their capacity to operate the input devices may vary in time. Also, it is obvious that for different tasks (writing up, proofreading, typesetting, etc.), the frequency of corresponding functions of the work-processing environment (typing, nagivation, setting the format, etc.) differ. Therefore it is a great advantage that the middleware is capable of a response to these circumstances. This adaptation requires that the layout of the on-screen keyboard and toolbar is constructed automatically, because the combinatorics of this task prevents from manual authoring. The necessary functionality of the middleware is shown in Figure 4.1.

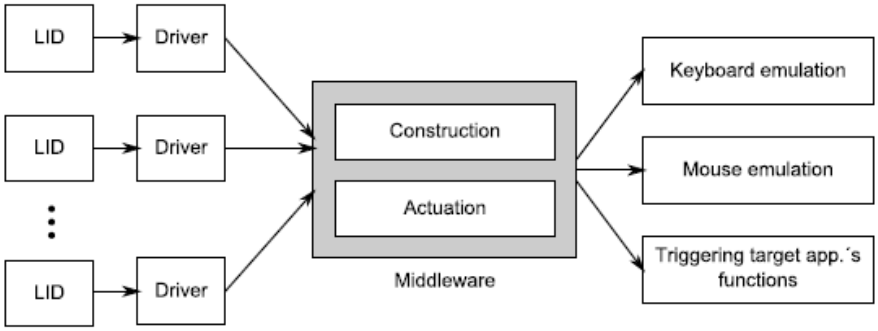


Figure 4.1: Block diagram of the context.

The layout of the keyboard should be constructed according to the type of the available limited-input devices and a simple language model. The layout is also affected by the profile of the task the user needs to accomplish within the environment of the word processor. The algorithm for building the layout is intended for fixed layouts only, i.e. the layout is not being changed depending on the previously entered text or operations that were performed.

The frequency of characters is a well researched area and we can reuse some of its observations. For example in English, the seven most common letters are E, T, A, O, I, N and S[1]. The frequency of unrepresented characters, e.g. Space, Enter, Tab, etc. has also been researched thoroughly (Soukoreff and MacKenzie, 2003), but not as much, as the represented characters.

The frequency of actions which can format the text (bold, italics, align) is, unfortunately, unexplored. It is mainly due to the reason that formatting the text is very a disproportional operation. There are texts which have no formatting, so there are only represented and unrepresented characters. Then we can have a paper, in which there is only formatted text. One line is aligned to the left, the other to the right, one word is underlined and the other is written in italics. To give the user possibility Anyway, it is very important to take the frequency of characters and actions into account, because it can increase the type rate significantly.

[1] `http://en.wikipedia.org/wiki/Letter_frequency`

## 4.1 Types of Activities Supported

Composing and editing text is an activity which combines a number of tasks. It is not only the typing, it could also be cursor movement, block operations or text formatting. Thus it is important to give the user the possibility to set a profile of the writing session. In our model, we split the text editing operations into three categories:

- Typing, i.e. entering new text,
- Cursor movement, i.e. navigation,
- Formatting, i.e. layout specification.

The user can choose which use case he prefers. The preference will be mediated by the possibility of giving weight to the use cases. The more the user prefers one of the use cases, the more weight he will give to it. The reason of this parameter is clear. It is unnecessary to have controls on the keyboard which do formatting, when the user wants only to write. Equally, it is useless to have controls for writing characters when the user wants to format a text that is already written. He will surely use the controls for navigating through the document and also formatting controls.

When the user sets a preference of one of these groups to 0, items from the group will no appear in the layout. On the other hide, when the user puts greater weight on other group, the items from this group will be placed in the layout in such a position that it will be easier and quicker to reach them.

There will also be the possibility to divide the layout into parts, where every part, let us call it a sub-layout, will contain only items from one category. This can help the user with orientation in the whole interface but this will also mean that the costs for reaching some items could be higher.

## 4.2 Types Of Signals Supported

As it was mentioned in Chapter 2, there are many types of interaction devices. The hardware device, which works with myoelectric signals, can recognize two of the types, and thus our application is designed to cooperate with these signals:

- Trigger. This signal is sent every time the user clenches the muscle. It has only two values - 0 and 1. When the muscle is in tension, the device sends 1, otherwise 0. When the muscle is in tension for longer period, the device sends these signals in intervals defined by the controller of the device, and so it can operate also as an On/Off switch. It can be used for advancing the state of a system by one step, for scan selecting or triggering a concrete action.

- Analog signal. This signal carries an analog value ranging from 0 to 100. 0 is sent when the muscle is relaxed. 100 is sent when the muscle is in maximal tension. The quantity can be then reported to the application which maps it to some visual feature. It can, for example, select the immediate row, when talking about two-dimensional scanning.

One type of signal can be more suitable for some tasks than the other, and we also want to give the user the possibility to choose what will be the most suitable option for him. Every user has different possibilities of how to control his body with the disability he has.

## 4.3 Layout Prototypes

In this section drafts of layout of the virtual keyboard are presented which depend on the scanning method. It is the scanning method which influences the layout most when trying to minimize the number of scan steps and scan selections. The letter and action frequencies are given, so the layout has to be changed accordingly. The items with higher frequencies of occurrence are placed in a way, that they are reachable with less scan steps.

For simplicity, the figures of the layout drafts contain only printable characters. Also, for each layout is shown how letter "$s$" can be selected.

### 4.3.1 Linear Scanning

Linear scanning is probably the simplest scanning method. The layout is shown in Figure 4.2. This method works as follows: items are sequentially highlighted until the desired character is selected. This could be very slow, especially for the characters at the end of the sequence.

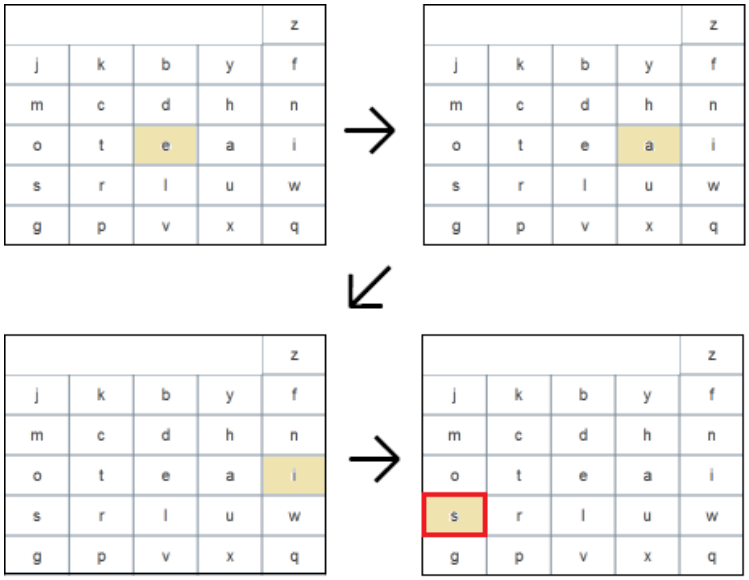


Figure 4.2: Layout for Linear Scanning.

The advantage of this method is that only one trigger is sufficient to control. The trigger can be used for scan selection while scan steps are done automatically or vice versa. In the second case, scan selection can be triggered after some period of time. When two triggers are available, the automatic part is usually replaced by the second trigger. Another option is that the second trigger make the scan step in the opposite direction than the first one. The analog signal has no utilization in this method, as the group of the items is usually too large to choose precisely the one desired. When talking about myoelectric signal, a human is able to differentiate only up to about 8 levels, when clenching muscle.

### 4.3.2 N-ary Scanning

In N-ary scanning is the whole group of items recursively split into $N$ parts until only the desired character is highlighted. It is very similar to an N-ary search algorithm. However, the layout is built based on weight-balanced tree, where all subtrees are of similar identical sums of frequencies of all items located within them. The layout and process of selecting letter $s$ in binary scanning is shown in Figure 4.3. The layouts for both binary and ternary scanning are identical. The difference is only in scanning technique, which is mentioned above.

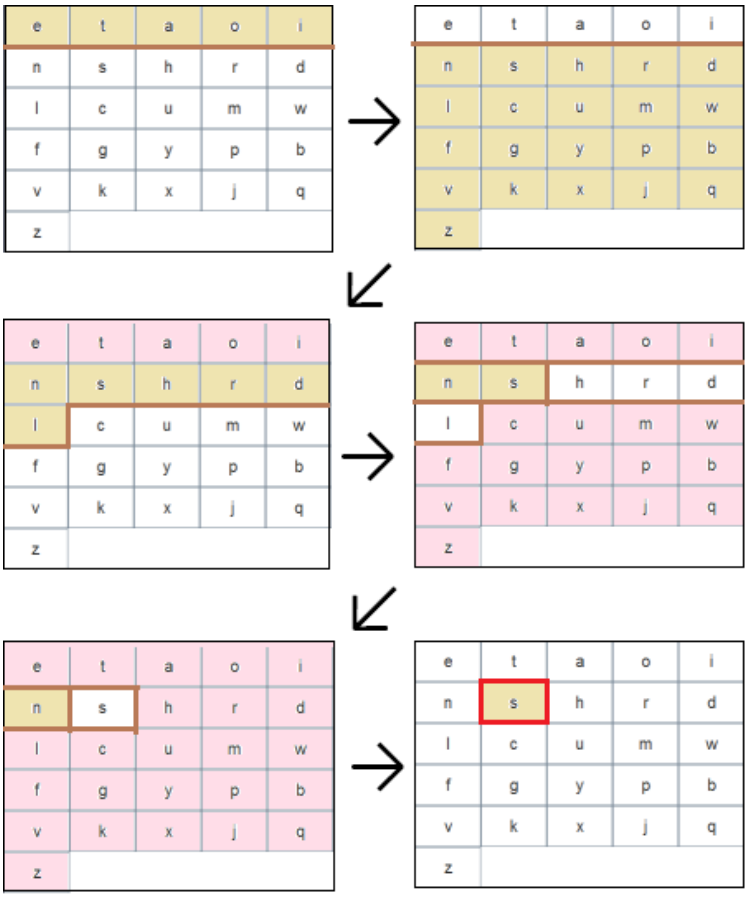


Figure 4.3: Layout for Binary Scanning.

In this technique, the same conditions for signals as in linear scanning could be applied. The only difference is in the usage of an analog signal. When talking about ternary version of this layout, the user is able to instantly select the target group by the analog value, as there will always be maximum of only three groups to choose from. And it should not be difficult to choose the desired one.

### 4.3.3 Two Dimensional Scanning

Two dimensional scanning, also called row-column scanning, works at two levels. At the first level, the rows are sequentially highlighted until the row is selected, in which the target item is placed. Then, the items in the selected row are linearly scanned, until the target item is highlighted. This layout is shown in Figure 4.4

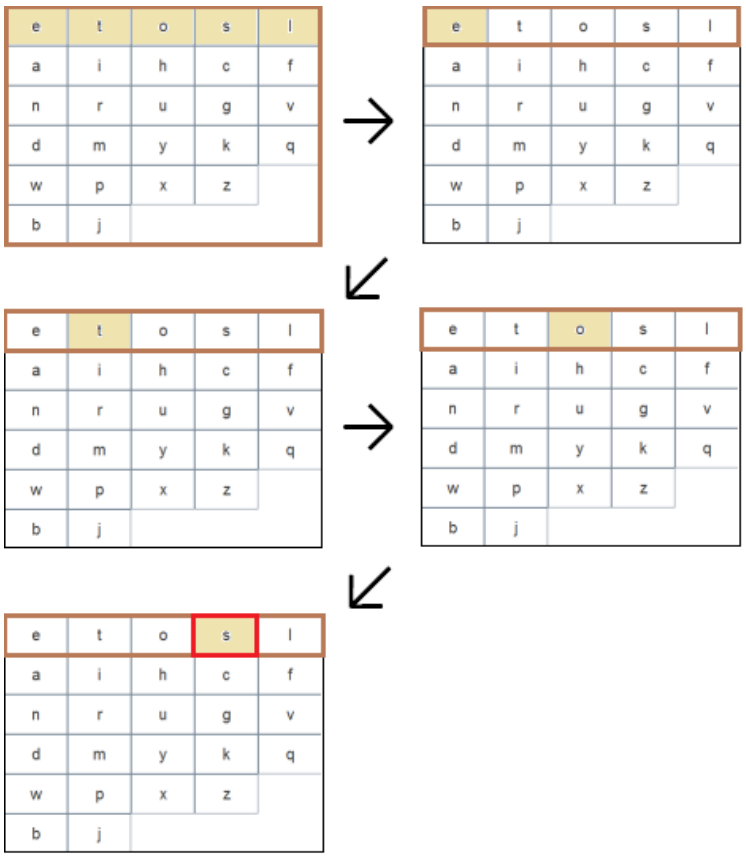


Figure 4.4: Layout for Two Dimensional Scanning.

Again, the conditions for a different number of trigger signals mentioned in methods above applies to this. The analog signal could be used in a similar way as in the case of an *N-ary* scanning. The user is able to choose instantly the row and then the column, which could be very fast.

### 4.3.4 Cursor Scanning

Cursor scanning works in a same way as if the user had a cursor arrows. Only one item can be highlighted at a time. The user can then make scan steps with the help of the arrows and can move on the layout in four directions - left, right, up and down. This layout is shown in Figure 4.5. This layout is also suitable for optimizing the number of scan steps, when taking into account the frequency of bigrams. For example *t* and *h* should be next to each other, as the *th* is the most common bigram in english alphabet. The layout which counts with this is shown in Figure 4.6.

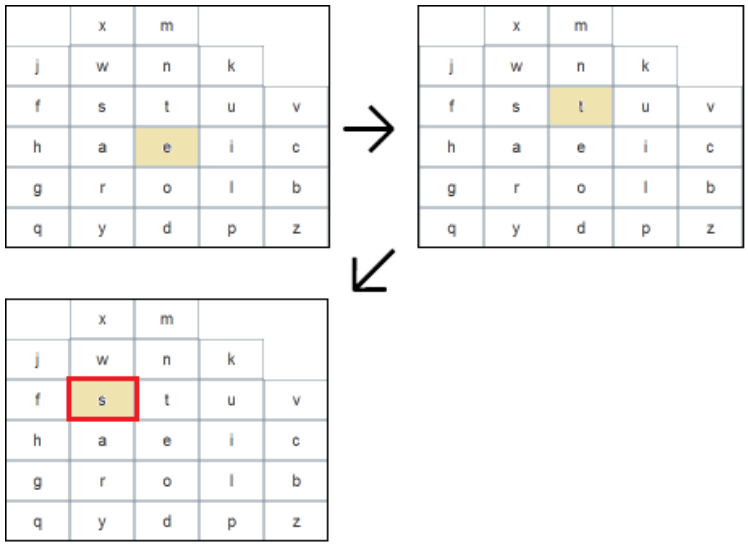


Figure 4.5: Layout for Cursor Scanning.

In this method, at least four triggers are needed. Each one for one "direction". The scan selection can be done after some period of time or by a fifth trigger, if available. The analog signal has no utilization in this method, as in the linear scanning method.

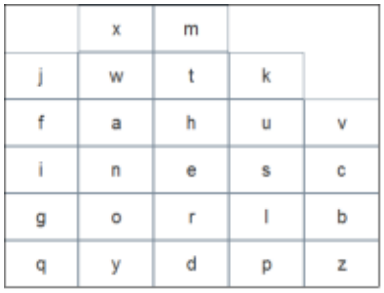


Figure 4.6: Layout for Optimized Cursor Scanning.

### 4.3.5 Layout with Contexts

The last draft presents what the layout will look like, when the division to contexts is disabled or enabled. It is shown in Figure 4.7. The layout on the left consists of only one keyboard including all items (in this example, characters and numbers). On the right there is a layout which consists of two separated parts. One part includes only characters, the other only numbers. This example is built according to the binary scanning method.

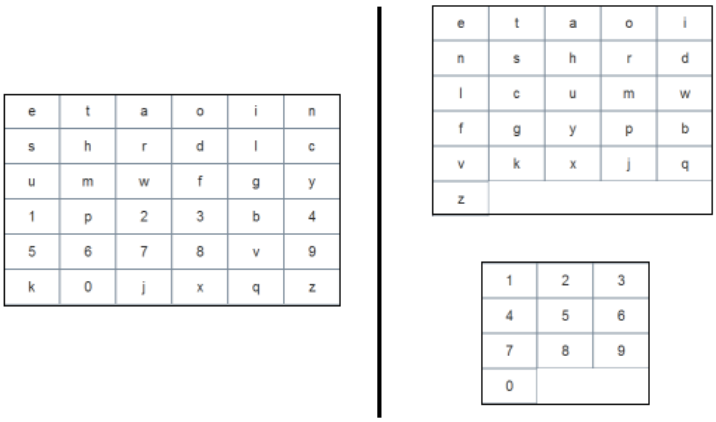


Figure 4.7: Difference in layouts based on context.

# Chapter 5

# Implementation

This chapter presents the implementation of the application. Technologies used for the development and considered approaches are discussed in the first section. The description of the architecture is to be found below. Then, the pseudocode of algorithm which builds the layout is shown. At the end of the chapter screenshots from the developed application are included.

## 5.1 Technologies

There were several possibilities of an environment we could choose. The idea of developing a new text editor from scratch was rejected, as our goal was mainly to prove whether the project TextAble has any perspective. It was therefore decided to extend an existing application. Finally, we selected a form of a plug-in into the Microsoft Word text editor. It has a very friendly API (Application Programming Interface) and the development of a plug-in (in this environment called add-in) is relatively easy. The choice of the development environment was then clear - Microsoft Visual Studio 2010. It is an IDE (Integrated Development Environment) from Microsoft and it is used to develop programs for Microsoft Windows. It has also a built-it support for developing add-ins into Microsoft Office applications. C# was chosen for the application as it is a modern object-oriented language, and there is no reason why any of the other .NET languages should be preferred to it. The application is written for .NET framework version 4.0.

The first idea was to use the built-in Ribbon menu, which was introduced into Microsoft Office applications in version 2007. This menu is in Figure 5.1. The individual

bookmarks and elements of the menu can be accessed by using different keys. After pressing Alt, a letter appears next to each tab, see Figure 5.2. Then, after the key with target letter is pressed, the menu switches to the corresponding tab and individual elements of the tab appears. Again, these elements can be executed by pressing the key with target letter or combination of keys, see Figure 5.3. Unfortunately, I found that this built-in menu cannot be accessed programmatically, so it is impossible to implement the functionality of switching the tabs or access the individual controls[1].

Figure 5.1: Ribbon menu in MS Word.


Figure 5.2: Ribbon menu in MS Word after pressing Alt key.


Figure 5.3: Ribbon menu in MS Word with displayed shortcuts.

The solution was therefore to create our own control. There were two possibilities. Create a new Ribbon menu, which is possible, or create the application in a form of a component called *TaskPane* in Microsoft Word. I chose the second possibility mainly due to the reason of keeping the default Ribbon menu in the Word application. It may

[1] `http://tinyurl.com/blogs-msdn-com`

happen that a user who can use standard keyboard and mouse wants to work with the application and he could be confused with the new Ribbon menu.

## 5.2 Architecture

The UML class diagram of the application is shown in Figure 5.4. The application has four main components. It is the Control, the Model, the Builder and the Scanner.

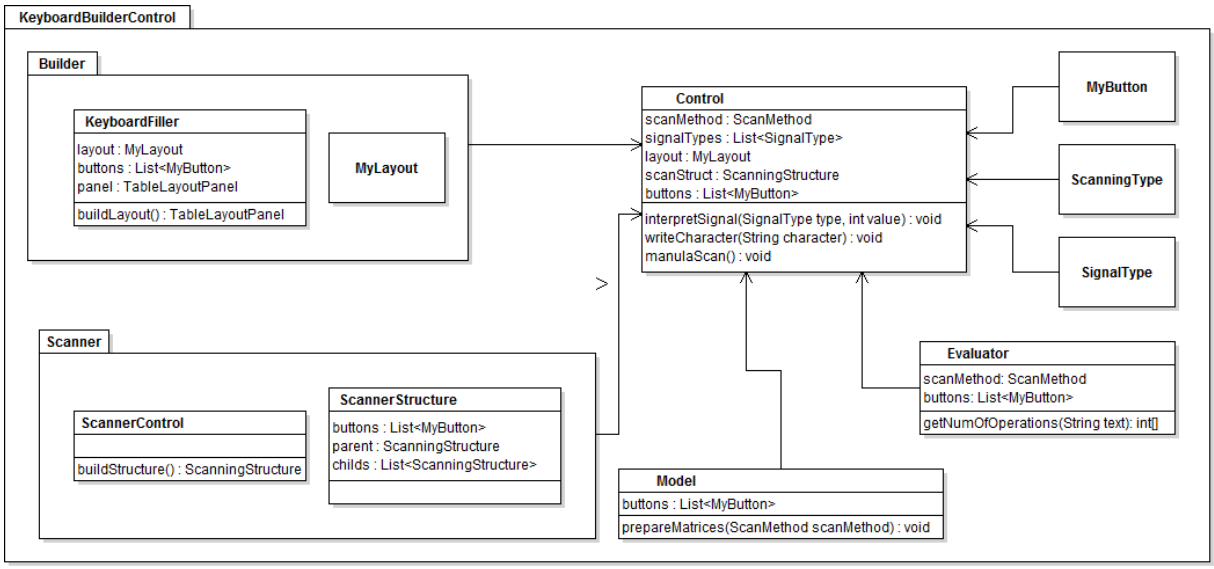


Figure 5.4: UML diagram of the main parts of the application.

**Model**

The Model component prepares the future layout in a form of a two-dimensional array, where the size of the first dimension is equal to the number of items in the layout and the second one is equal to two. In this array $x$ and $y$ coordinates are stored. Each item will be put into the layout according to this array.

**Evaluator**

The Evaluator is used for evaluating the layout prepared by the Model. It returns number of scan steps and scan selections needed when writing input text. It is used when resolving the better layout - with context or without context.

**Builder**

The Builder component fills the target *TableLayoutPane*, which is one of the containers for various controls, with real buttons.

**Scanner**

The Scanner component handles the scanning feature. Depending on the chosen method, it selects the appropriate method and builds the scanning structure from the instances of the *ScanningStructure* class. The whole scanning structure is a tree. Every *ScanningStructure* instance has a parent and children and also contains its items.

**Control**

The Control is the core of the application. It controls the logic. I will mention several important methods of this class.

- *interpretSignal()* - This method interprets the input signal from the hardware device according to the signal type and value and calls the corresponding function.

- *writeCharacter()* - This method calls the API of the Microsoft Word and performs the target action according to the input parameter.

- *manualScan()* - This method simply performs a scan step.

## 5.3 The Algorithm

Bellow is the pseudocode of our algorithm that is capable of constructing the on-screen keyboard layout based on the set of actions (letters to type, or formatting and navigation functions to trigger), available selection method (low-level hardware interface), and the profile of the use cases (i.e. how frequent typing, navigation, or formatting would be).

```
BUILD-LAYOUT
INPUT:
   actions -- a list of keys and operations which are
      accessible by the keyboard. Attributes for each:
      action.frequency -- how frequent is the action
      action.use-case -- which use case it belongs to
   selection-method -- either "Binary", "Ternary",
   use-case-profile -- ratio between Typing,
   Navigation, and Formatting
   spacebar-handled-separately -- true or false
   backspace-handled-separately -- true or false
OUTPUT: keyboard-layout, as the orthogonal matrix of
   items on keyboard
PROCEDURE:
1: CALL INITIALIZE and save result to Array A;
2: Initialize Matrix M,
      in which items will be placed
3: IF selection-method == "Binary" or "Ternary"
      CALL BUILD-N-ARY-LAYOUT (with N = 2 or 3)
      and store the output to M
4: IF selection-method == "2D-scan"
      CALL BUILD-2D-SCAN-LAYOUT
```

```
      and store the output to M
5: IF selection-method == "linear" scan
      CALL LINEAR and store the output to M
6: IF selection-method == "2D-cursor" scan
      CALL CURSOR and store the output to M
7: RETURN M

INITIALIZE
INPUT:
   actions
   use-case-profile
OUTPUT: array
1: SET Array A = array-of-items-on-keyboard
2: FOR EACH element E FROM A,
   E.frequency = E.frequency * use-case-preference
3: IF frequency of E == 0, REMOVE it from A
4: IF spacebar-handled-separately,
   remove spacebar form A
5: IF backspace-handled-separately,
   remove backspace form A
6: RETURN array A

BUILD-N-ARY-LAYOUT
INPUT: array-of-items-on-keyboard, N
OUTPUT: matrix, tree
1: CALL BUILD-N-ARY-TREE and
   store the output to tree T
2: Traverse the T in preorder manner,
   and store nodes N to Matrix M sequentially
   from first row to last row,
   from first column to last column
3: RETURN M

BUILD-N-ARY-TREE
INPUT: array-of-items-on-keyboard, N
OUTPUT: tree
1: Build tree T,
   where sums of frequencies of all subtrees
   of each node in T are equlized
2: RETURN T

BUILD-2D-SCAN-LAYOUT
INPUT: array-of-items-on-keyboard
OUTPUT: matrix
1: Create output matrix M with X columns and
   X or (X+1) rows with minimum X
   so that number of the items would fit.
   Each cell will be empty.
2: FOR EACH element I
   from array-of-items-on-keyboard,
   put I to an empty cell which is nearest
   to the position (1,1), Manhattan metric.
3: RETURN M

BUILD-LINEAR-LAYOUT
INPUT: array-of-items-on-keyboard
OUTPUT: matrix
1: Create output matrix M with X columns
   and X or (X+1) rows with minimum X
   so that number of the items would fit.
   Each cell will be empty.
2: FOR EACH element I from
   array-of-items-on-keyboard, put I
   to an empty cell which is nearest
   to the "center" of M, when moving only
   to the left or right.
3: RETURN M

BUILD-CURSOR-LAYOUT
INPUT: array-of-items-on-keyboard
OUTPUT: matrix
1: Create output matrix M with X columns
   and X or (X+1) rows with minimum X
   so that number of the items would fit.
   Each cell will be empty.
2: FOR EACH element I from
   array-of-items-on-keyboard, put I
   to an empty cell which is nearest
   to the "center" of M, Manhattan metric.
3: RETURN M
```

## 5.4 Implemented application

In this section, overview of application functionality is presented. Then information about mapping signals from the hardware device to actions in the user interface follows. Finally the screenshots of the implemented application are attached.

### 5.4.1 Application functionality

Functionality of the application developed in this thesis is to provide interface for entering and editing the text as well as navigating in the document only with limited number of input signals. I will describe these functions in detail.

**Text enter**

A will not describe, how to enter individual characters as this can be found in section 4.3. I will mention special functions provided by the application. User can use following known keys and functions:

- Backspace - performs deletion of character, could be handled separately by a trigger signal.
- Space - inserts white-space, could be handled separately by a trigger signal.
- Enter - inserts new line into the document, could be handled separately by a trigger signal.
- Shift - enables entering of upper-case letter. After entering the letter, shift is turned off. When dot is entered shift is automatically turned on.
- Caps Lock - enables entering of upper-case letters until it is turned off.

**Text manipulation**

Text can be manipulated using three common functions:

- Copy - selected text is placed into the clipboard.
- Paste - text stored in clipboard is put on the caret position.
- Cut - selected text is placed into the clipboard and removed from document.

**Text formatting**

Several decorating function and alignment of text as well as adjustment of font size is implemented in the application:

- Bold - when turned on, text to be written or selected text will be in bold (same symbol on keyboard as in the MS Word environment).

- Italics - when turned on, text to be written or selected text will be in italics (same symbol on keyboard as in the MS Word environment).
- Underline - when turned on, text to be written or selected text will be underlined (same symbol on keyboard as in the MS Word environment).
- Left alignment - align the text to left (similar symbol on keyboard as in the MS Word environment).
- Center alignment - align the text to center of page (similar symbol on keyboard as in the MS Word environment).
- Right alignment - align the text to right (similar symbol on keyboard as in the MS Word environment).
- Adjust font size - increases (symbol '+' on keyboard) or decreases (symbol '-' on keyboard) the font size.

**Navigation in document**

Navigating in the document could be performed by four arrows or by standard keys *Page Up*, *Page Down*, *Home* and *End*. User has also the possibility to adjust the move type and move unit:

- Move type - user can choose from two options - *Move*(caret will be moved without selecting the text) or *Extend* (caret will be moved with selecting the text). On the keyboard can be this selection done by key *Move*.
- Move unit - user can choose from two options - *Small*(caret will be moved by one character to left or right or one line up or down) or *Large* (caret will be moved by one word to left or right or one paragraph up or down). On the keyboard can be this selection done by key *Unit*.
- Move up - moves the caret up (taking into account Move type and Move unit).
- Move down - moves the caret down (taking into account Move type and Move unit).
- Move left - moves the caret left (taking into account Move type and Move unit).
- Move right - moves the caret right (taking into account Move type and Move unit).

- Page up - moves the caret up one page (taking into account Move type). On the keyboard item with label *pgUp*.
- Page down - moves the caret down one page (taking into account Move type). On the keyboard item with label *pgDn*.
- Home - moves the caret at the beginning of line or document (taking into account Move type and Move unit). On the keyboard item with label *Home*.
- End - moves the caret at the end of line or document (taking into account Move type and Move unit). On the keyboard item with label *End*.

**Application controlling**

Several common functions are provided by the application:

- Save - saves the current document.
- Quit - quit the MS Word application.
- Settings - opens the settings dialogue.

### 5.4.2 Signals mapping

There are three ways of mapping signals from hardware device to actions in the user interface. User can choose one of these options on settings dialogue. It could be done statically as it is shown in Figure A.1, when user chooses option *Default*. Next option is to choose option *Automatic*. When this option is selected, the algorithm prepares models for both context and non-context layouts and evaluates them on most suitable text (statically entered in application) according to other parameters entered on settings dialogue. Then the algorithm chooses the model with better result from the evaluator. The signals are then mapped in the same way as in the first option. Last option is to choose *File*. The algorithm will read the configuration and mapping of signals from the file. This is described more properly in user manual, which is located in Appendix A.

### 5.4.3 Application appearance

Figure 5.5 shows the initial screen of the keyboard. It is the screen with settings. The user can set the preferences of use cases, if the layout will be divided into contexts,

number and type of signals, scanning method and name of the document under which it will be saved. The scanning method used on this screen is a two-dimensional scanning by default.

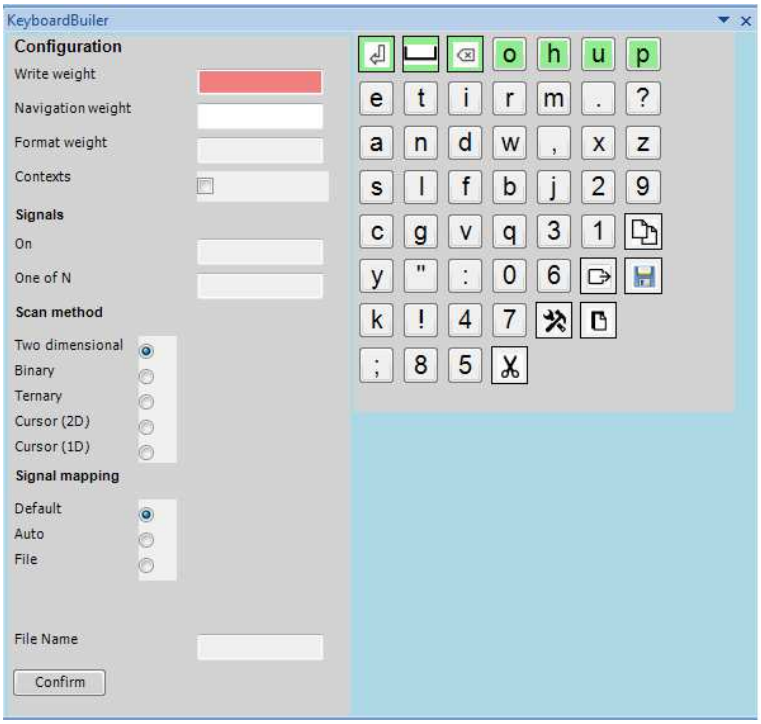


Figure 5.5: Inital screen of application.

After setting the parameters, the user can run the generating of the layout by selecting the *Confirm* button. The algorithm then generates the layout of the virtual keyboard and the user can start his work. This layout is shown in Figure 5.6. This layout is built with the following parameters:

*Write weight*: 5
*Navigation weight*: 0
*Format weight*: 0
*Context*: No
*Number of On signals*: 4
*Number of OneOfN signals*: 0
*Scan method*: Binary

The instructions how to control the application are described in User Manual, which

can be found in Appendix A.

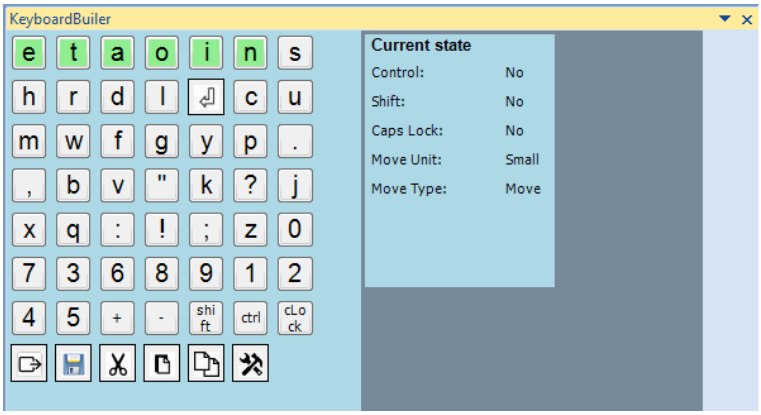


Figure 5.6: Generated layout.

In Figure 5.7 is shown the appearance of the whole MS Word application. Layout in this figure is built with the following parameters:

*Write weight*: 2
*Navigation weight*: 3
*Format weight*: 5
*Context*: No
*Number of On signals*: 3
*Number of OneOfN signals*: 1
*Scan method*: Two dimensional

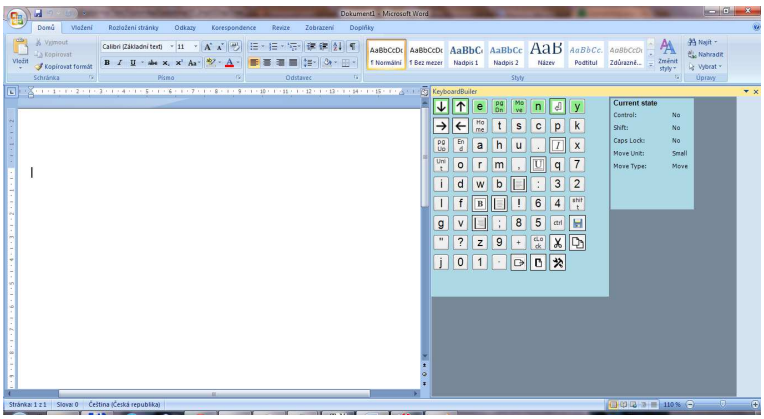

Figure 5.7: Application appearance.

# Chapter 6

# Testing and Evaluation

In this chapter the process of the user test is described. Results from user test with examples of improvements which were found during testing are presented. Also evaluation of the designed layouts is introduced. The designed layouts are compared to each other.

## 6.1 User Test

It was originally planned to test the application with at least three users from the target group, that means users with impairment of the upper limbs. Unfortunately, due to lack of time and also the difficulty to get the target users, only one user tested our application.

This user was, nevertheless, very helpful and participated in two sessions. I will add that the user is between 50 and 60 and has been impaired since the age of 25 due to an accident. His arms are both amputated at the elbow. He uses one prostheses. Since about the time of the accident he has been involved in the design and development of prostheses and other products for similarly impaired people. He uses standard hardware keyboard for interacting with the computer. But he finds this quite uncomfortable and exhausting. He also teaches at the CTU in Prague, on Faculty of Electrical Engineering.

I will describe these two sessions separately, as they were quite different. And I have to mention that both sessions were held more like participatory design sessions that purely testing, as it was rather an improvement of current application. This was the first trial of the whole system with the user from the target group.

### 6.1.1 The First Session

The first session was organized at the end of November and lasted about an hour. The first half of the session was spent with familiarizing the user with the hardware device. This is very important as the comfort of the user stands on the first place. Different sensitivity settings were tried out to best match the user's needs. During this setting, the user mentioned many useful facts and I will try to reproduce them, as they could be very important in the future work. They are listed bellow. This session was prepared to test the application together with only trigger signals from the hardware device. No scenario with analog value was tested. Signals were mapped statically. During this session, two signals were used. Both electrodes were attached to forearm. The first signal was mapped to scan step, the second to scan selection. Type rate in this session is not important to determine as the participant was only familiarizing with the device

Here are listed the findings from the first session:

- **No prediction**. The participant said, that it is nearly impossible for him to think about the prediction and write simultaneously. It is because in the past he participated in many sessions which were related to text input. During some of them, prediction were used and for our participant was this option counterproductive. In my opinion this is caused mainly by the higher age of the participant as he is not as flexible as it would be needed.

- **Simple layout**. The participant would be inclined to the normal QWERTY layout, as the generated layout was too confusing for him. He also suggested using a simple linear scanning as a better solution. He added that it could be only his subjective feeling, but we have to count with it.

- **Forgetting the function of signals**. The user kee forgetting relatively often, which signal has which function and this led to errors, of course.

- **Quite exhausting**. The participant was exhausted after only about 45 minutes. And it has to be said, that the user did not try to write all the time.

### 6.1.2 The Second Session

The second session was held in the middle of December and lasted about an hour and a half. This session was aimed on testing analog input signal. Half an hour was again

spent on configuration and setting of the correct sensitivity of the device. The participant was also familiarizing with the analog signal. In this session, three signal were used. Analog signal, which was used for scan selection and two trigger signals - one for performing scan selection a the other for performing delete action. This second trigger signal was produced by contraction of the muscles on the face. This solution appears quite uncomfortable for the user. The virtual keyboard, which it has been tested with was developed by Antonín Pošusta, a developer of the hardware device. The keyboard had a QWERTY layout, according to the findings from the first session, and used a two-dimensional scanning technique. Three scenarios for the test were prepared, two of them have been partially met. Bellow, only the first scenario is presented. All scenarios can be found on the attached CD.

**Test scenario**
The user was supposed to write short plain text. The text was printed on the paper and placed on the desk in front of the monitor.

*Introduction*
*Our intention was to develop a system allowing a person to type text using muscles. We developed a plug-in for MS Word.*

The participant managed to write approximately half of the given text in 45 minutes. We can determine the WPM (Words per Minute) value, which is one of the most frequent methods for determining the type rate. Standard word is defined as a string of five characters. In our case was written approx. 60 characters in 45 minutes, that gives us approx. 0.26 WPM. For *error rate* several metrics are used. One of the most used is KSPC (Keystrokes per Character). In our case the KSPC was very high. I can't determine the exact rate, but from observations, the KSPC was at least 8. It was increased firstly by using scanning technique, secondly by lots of type errors and corrections done by participant.
Again the notes mentioned by user during the second session are listed here:

- **Quite exhausting**. The participant repeated, that this method for text input is too exhausting. He said was is feasible but uncomfortable. He also thinks that this is not the right way.

- **Need of assistance**. He claims it is very problematic to be dependent on someone

else to help him into the prosthesis.

- **Simple application**. He considers the prosthesis with moving fingers useless for the practical life and says, that the society goes the opposite way, when trying to improve the devices a adding new functionalities. He is afraid of a failure of these device.

- **Uncomfortable electrodes location**. The participant, after some time, complained about pain muscle spasms in the face.

- **Need of undo function**. There is a big need of an undo function to return into the previous state. In current solution, only delete function is implemented.

- **Placement of the keyboard on screen**. The participant suggested that the placement of the keyboard should dynamically change according to current position of the cursor.

### 6.1.3 Output from the Testing

The testing showed that text input based on the myoelectric signal could be a very exhausting process. In both sessions the user was tired after approximately 45 minutes. This is not a very pleasant finding. The participant found this method feasible but very uncomfortable. There was not significant improvement of type rate in second session if any. In my opinion this was mainly cased by usage of new type of signal for user - the analog signal. The participant was familiarizing with the analog signal for a long time.

However, many findings which could be used in future work were identified. The findings could be used not only in the context of the TextAble project, but in every project that deals with the problem of text input for disabled people. Probably one of the most important findings is that the simpler the application is, the better. This is especially true for older people who are not as adaptable as younger ones.

According to findings during the user test, a pseudo-QWERTY layout was designed and implemented. It is shown in Figure 6.1. It is designed in a way to combine advantages of standard QWERTY layout and ideal layout for two-dimensional scanning, i.e. get close to square layout. This layout could be probably more acceptable for user at the cost of more interactions to achieve a goal.

| | | | | | |
|---|---|---|---|---|---|
| q | w | e | r | t | y |
| u | i | o | p | ⏎ | . |
| a | s | d | f | g | h |
| j | k | l | , | ? | ! |
| z | x | c | v | b | n |
| m | " | : | - | ; | |

Figure 6.1: Pseudo-QWERTY layout.

Also performing *Undo* function easily was a big need for the participant during the test. When no separate signal handles this function, the layout might look like Figure 6.2. The first item on every row is assigned to perform *Undo*, as during the testing it turned out to be very frequent action, especially for inexperienced users.

| | | | | | | |
|---|---|---|---|---|---|---|
| ↩ | q | w | e | r | t | y |
| ↩ | u | i | o | p | ⏎ | . |
| ↩ | a | s | d | f | g | h |
| ↩ | j | k | l | , | ? | ! |
| ↩ | z | x | c | v | b | n |
| ↩ | m | " | : | - | ; | |

Figure 6.2: Pseudo-QWERTY layout with Undo.

This layout together with all, which were developed, are evaluated in next section.

## 6.2 Comparison of the Designed Layouts

All designed layouts presented before 4.3 as well as the layout designed after user test are compared in this section. To compare their efficiency, the program which measures needed scan steps and scan selection was developed in Java and source code for this program is on attached CD. The error rate is not included in the measurement.

For simplicity layouts consisting only from letters of English alphabet and some other printable characters were compared. References to layouts, which are presented in the below tables have only information character. The real layouts that were evaluated are also present on the attached CD. The text was generated [1] with respect to frequency of letters in English language. The text has 393 characters including spaces. It is worded as follows:

*Lorem ipsum dolor sit amet, dolor tollit viderer quo ut, mel cu lobortis postulant? Eam id decore euismod, ei nostro verterem sea? Impetus persecuti in mel, suas enim ei nam? Mei mandamus qualisque ea, te cum dicunt lobortis splendide. Aliquip sapientem eos ad, no sea diam vivendo. Ut nec phaedrum dissentiunt comprehensam, ignota philosophia pro in, an qualisque scripserit reprehendunt mei.*

In Table 6.1 is shown the number of scan steps and selections needed to write the text using only trigger signals. The number of signals are chosen to match the minimal number needed while avoiding an automatic processes, e.g. auto-scan or auto-selection. This is described for each layout in section 4.3. The QWERTY layout is designed for two-dimensional scanning, as it was the most suitable method for the participant.

The best results has the layout for cursor scanning, however to control this layout, 5 signals are needed, which could be a problem in some cases. Layouts for linear and cursor scanning have the lowest number of scan selections because this action is performed only when target character should be written. But the number of scan steps is the highest for linear scanning and for cursor scanning it is lower at the cost of higher number of signals.

Layouts for binary and ternary are the same as already discussed before. They need the highest number of scan selections, as for the characters with the lowest frequencies it may take a long time to get there. And also the participant complained during the testing, that these methods are too confusing for him. These are probably the worst methods designed.

Two dimensional layout looks as the best solution, when only two signals are possible.

[1] `http://http://generator.lorem-ipsum.info/`

The number of selections is relatively low. For each character user has to perform two scan selections - one for choosing the row, one for choosing the character in the row. Number of scan steps differs according to the layout as shown in Table 6.1 when looking on two-dimensional layout and QWERTY layout which both uses two-dimensional scanning technique. The number of scan steps in QWERTY layout is about half greater then in two dimensional layout. And it is up to user to choose his preference. On the one hand better performance (two-dimensional layout), on the other hand more suitable layout (QWERTY layout).

| **Layout Type** | **Figure** | **# of signals** | **# of scan steps** | **# of scan selections** |
|---|---|---|---|---|
| Linear | 4.2 | 3 | 2720 | 399 |
| Two Dimensional | 4.4 | 2 | 1127 | 798 |
| Binary | 4.3 | 2 | 863 | 1928 |
| Ternary | 4.3 | 2 | 1489 | 1489 |
| Cursor | 4.5 | 5 | 1310 | 399 |
| QWERTY | 6.1 | 2 | 1720 | 798 |

Table 6.1: Number of interactions needed to write the text using only trigger input

In Table 6.2 is shown some computed theoretical performance measures. It is *scan steps per character* (SPC), *selections per scan steps* (SPS), *keystrokes per character* (KSPC) and *words per minute* (WPM). To compute these theoretical measures, we need following values:

- C: Number of characters in the input text.
- S: Number of scan steps.
- G: Number of scan selections.
- D1: Scan delay for trigger signal.
- D2: Scan delay for analog signal.
- D3: Time needed by the user to make a selection.

Values $S$ and $G$ are shown in table 6.1. Values *D1*=0.375s , *D2*=2s and *D3*=0.75s was set according to observations and pre-studies.

The SPC measure is in this case defined as follows:

$SPC = \frac{S+C}{G}$

The SPS measure is in this case defined as follows:

$SPC = \frac{G}{S}$

The KSPC is similar to SPC (Poláček, 2014).

The WPM measure is in this case defined as follows:

$SPC = \frac{C}{D1*S+D3*G} * 60 * \frac{1}{5}$

| **Layout Type** | **Figure** | **# of signals** | **SPC** | **SPS** | **KSPC** | **WPM** |
|---|---|---|---|---|---|---|
| Linear | 4.2 | 3 | 7.9 | 0.2 | 7.9 | 2.1 |
| Two Dimensional | 4.4 | 2 | 4.9 | 0.7 | 4.9 | 4.1 |
| Binary | 4.3 | 2 | 7.1 | 2.2 | 7.1 | 3.4 |
| Ternary | 4.3 | 2 | 7.5 | 1 | 7.5 | 2.8 |
| Cursor | 4.5 | 5 | 4.3 | 0.3 | 4.3 | 4.1 |
| Cursor Opti | 4.6 | 5 | 4.1 | 0.3 | 4.1 | 4.4 |
| QWERTY | 6.1 | 2 | 6.4 | 0.5 | 6.4 | 3 |

Table 6.2: Measured performance in simulation using only trigger input

Next Table 6.1 shows the number of scan selections when analog input is used. Analog input is used in a way that the user clench the muscle to highlight the desired character or group of characters and then confirms this choice by another trigger signal. In the table is therefore only the number of scan selections, as the scan steps are made by clenching the muscle. Four designed layouts are suitable for cooperating with analog input.

The numbers of scan selections are the same as in Table 6.1 and it is no coincidence. Layouts for binary and ternary scanning have higher numbers of scan selections, but we should remember, that it could be much easier to differentiate only two or three levels of clenching the muscle. And in these layouts, there will never be a need for higher count. Binary scanning has maximum of two levels, ternary accordingly three levels. This could me much more comfortable for user and can lead to lower error rate.

In case of two-dimensional layout and QWERTY layout is the number equal. The cause is discussed above. The disadvantage when using two dimensional scanning technique together with analog input is the number of levels of clenching the muscle needed.

In our designated layouts, there could be up to eight levels and this could be very difficult for user to differentiate them. When using analog input, the two dimensional layout which is built according to frequencies of letters looses the advantage of lower scan steps needed. This means that it will certainly be better to offer user layout which he prefers. And probably this layout will be QWERTY layout.

| **Layout Type** | **Figure** | **# of signals** | **# of scan selections** |
|---|---|---|---|
| Two Dimensional | 4.4 | 2 | 798 |
| Binary | 4.3 | 2 | 1928 |
| Ternary | 4.3 | 2 | 1489 |
| QWERTY | 6.1 | 2 | 798 |

Table 6.3: Number of interactions needed to write the text using analog input

In Table 6.4 is shown one computed theoretical performance measures and it is *words per minute* (WPM). Other values, which were computed above only for trigger signals and are shown in table 6.2, are in this case pointless, as number of scan steps can't be determined when using analog signal. But we can count with the time needed to select the right item in the layout which is 2 seconds, as discussed above.

The WPM measure is in this case defined as follows:
$WPM = \frac{C}{D2*S+D3} * 60 * \frac{1}{5}$

| **Layout Type** | **Figure** | **# of signals** | **WPM** |
|---|---|---|---|
| Two Dimensional | 4.4 | 2 | 7.8 |
| Binary | 4.3 | 2 | 3.2 |
| Ternary | 4.3 | 2 | 4.2 |
| QWERTY | 6.1 | 2 | 7.8 |

Table 6.4: Measured performance in simulation using analog input

# Chapter 7

# Conclusion

In this thesis a generic on-screen virtual keyboard has been developed whose layout is built based on a number of parameters. It is targeted to people with upper-limbs impairment, who cannot use standard computer peripherals. The application is a part of the TextAble project, which tries to develop a solution of a text entry system controlled by myoelectric signals.

Most of the text entry systems available today, even those targeted to disabled users, only contain a traditional on-screen keyboard that looks like a physical one. Typing with such systems is usually very slow. However, there are some novelty text entry systems that provide an alternative (and a faster) way of entering the text. The application developed in this thesis also describes nontraditional ways of typing. This application is designed to be controlled by myoelectric signals.

The application was designed as a plug-in into the Microsoft Word text editor. It provides functions for typing text as well as formatting the text and navigating in the document.

Before designing the application, a survey of existing text entry systems with focus on assistive technology had been done. The results are also presented in this thesis - some important and also interesting representatives of physical and virtual keyboards as well as alternative text entry systems are listed in Chapter 2.

The advantage of our application is in the optimization of the layout, as the algorithm builds the layout trying to lower the number of user's interactions, which have to be made to achieve his goal (i.e. write a text, format a text). This optimization is based on letter frequency. Also, the layout is built with respect to selected scan method and use case preferences. There are three use cases distinguished - typing, formatting and navigation. The architecture of the application is presented in Chapter 5.

Some usability testing was done with the application, which took place in the Academy of Science of the Czech Republic. It was good to have a chance to try the whole application in a real environment. However, due to time constraints and problem with getting users from the target group, only one of these users tested the application. The testing has shown that text input based on myoelectric signal is not a very suitable solution as it is a very exhausting method. Also, the generated layout of the virtual keyboard was not accepted very well, as it was very confusing for the user to orientate in it. A simple QWERTY layout will be much better, according to the user. A lot of interesting ideas for improvements and new features were also suggested by the user during the usability testing. They have to be taken into account for the future work when working on any other project dealing with problem of text input for motor-impaired people. Some ideas and examples, how to improve the layout are shown in Chapter 6.

Hopefully, more of these projects will be realized and will help the handicapped users to access the computers more easily and be able to communicate better with other people around them. This thesis aims to be one of these projects.

# Bibliography

Adams, R. (1999), 'Bar code 1, chapter bar code specifications'. http://www.adams1.com/.

Barber, C. (1997), *Beyond the Desktop*, Academic Press.

Card, S. K. and Moran, T. P. (1983), *The psychology of human-computer interaction*, Lawrence Erlbaum Associates Inc.

Felzer, T., MacKenzie, I. S., Beckerle, P. and Rinderknecht, S. (2010), Qanti: A Software Tool for Quick Ambiguous Non-standard Text Input, *in* K. Miesenberger, J. Klaus, W. Zagler and A. Karshmer, eds, 'Computers Helping People with Special Needs', Vol. 6180 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 128–135.

Goldberg, D. and Richardson, C. (1993), Touch-typing with a stylus, *in* 'Conference proceedings on Human factors in computing systems', ACM, New York, NY, USA, pp. 80–87.

Gong, J. and Tarasewich, P. (2011), Alphabetically constrained keypad designs for text entry on mobile devices, *in* 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', CHI '05, ACM, New York, NY, USA, pp. 211–220.

Gopher, D. and Raij, D. (1988), 'Typing on a two-handed chord keyboard: Will QWERTY become obsolete?', *IEEE Transactions on Systems, Man, and Cybernetics* **18**(1), 601–609.

Harbusch, K. and Kühn, M. (2003), Towards an adaptive communication aid with text input from ambiguous keyboards, *in* 'Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 2', EACL '03, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 207–210.

Jones, P. E. (1998), Virtual keyboard with scanning and augmented by prediction, *in* 'Proceedings of the 2nd European Conference on Disability, Virtual Reality and Associated Technologies', IOS Press, pp. 45–51.

Karlsdottir, R. (1997), 'Comparison of cursive models for handwriting instruction', *Perceptual and Motor Skills* **85**(3), 1171–1184.

Kim, H. and Kim, Y.-H. (2009), Optimal designs of ambiguous mobile keypad with alphabetical constraints, *in* 'Proceedings of the 11th Annual conference on Genetic and evolutionary computation', GECCO '09, ACM, New York, NY, USA, pp. 1931–1932.

Lesher, G., Moulton, B. and Higginbotham, D. (1998), 'Optimal character arrangements for ambiguous keyboards', *Rehabilitation Engineering, IEEE Transactions on* **6**(4), 415–423.

Liebowitz, S. and Margolis, S. E. (1990), 'The Fable of the Keys', *Journal of Law and Economics* **33**(1), 1–26.

MacKenzie, I. S. and Felzer, T. (2010), 'Sak: Scanning ambiguous keyboard for efficient one-key text entry', *ACM Trans. Comput.-Hum. Interact.* **17**(3), 11:1–11:39.

MacKenzie, I. S., Nonnecke, B., Riddersma, S., McQueen, C. and Meltz, M. (1994), 'Alphanumeric entry on pen-based computers', *International Journal of Human-Computer Studies* **41**(3), 775–792.

MacKenzie, I. S. and Zhang, X. S. (1997), The immediate usability of graffiti, ACM, New York, NY, USA, pp. 129–137.

MacKenzie, I. S., Zhang, X. S. and Soukoreff, R. W. (1999), 'Text entry using soft keyboards', *Behaviour and Information Technology* **18**(4), 235–244.

Mankoff, J. and Abowd, G. D. (1998), Cirrin: a word-level unistroke keyboard for pen input, *in* 'Proceedings of the 11th annual ACM symposium on User interface software and technology', ACM, New York, NY, USA, pp. 213–214.

Mares, G. C. (1909), *The History of the Typewriter*, Post Era Pubns.

Matias, E., MacKenzie, S. I. and Buxton, W. (1993), 'Half-QWERTY: a one-handed keyboard facilitating skill transfer from QWERTY', pp. 88–94.

Perlin, K. (1998), Quikwriting: continuous stylus-based text entry, *in* 'Proceedings of the 11th annual ACM symposium on User interface software and technology', ACM, New York, NY, USA, pp. 215–216.

Poláček, O. (2014), Designing Text Entry Methods for Non-Verbal Vocal Input, PhD thesis, CTU Prague.

Sears, A., Karat, C.-M., Oseitutu, K., Karimullah, A. and Feng, J. (2001), 'Productivity, satisfaction, and interaction strategies of individuals with spinal cord injuries and traditional users interacting with speech recognition software', *Universal Access in the Information Society* **1**(1), 4–15.

Siioles, C. (1868), 'Improvement in type-writing machines'. US Patent 79,265.

Soukoreff, R. W. and MacKenzie, I. S. (2003), Input-based Language Modelling in the Design of High Performance Text Input Techniques, *in* 'Proceedings of Graphifcs Interface 2003', Canadian Information Processing Society, Toronto, Canada, pp. 89–96.

Sporka, A. J., Felzer, T., Kurniawan, S. H., Poláček, O., Haiduk, P. and MacKenzie, I. S. (2011), Chanti: predictive text entry using non-verbal vocal input, *in* 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', CHI '11, ACM, New York, NY, USA, pp. 2463–2472.

Yin, P.-Y. and Su, E.-P. (2011), Optimal character arrangement for ambiguous keyboards using a PSO-based algorithm, *in* 'Natural Computation (ICNC), 2011 Seventh International Conference on', Vol. 4, pp. 2194–2198.

# Appendix A

# User Manual

The keyboard is not a standalone application. It is designed to run in Microsoft Word text editor. The process of starting the application is described on attached CD. The application could be also started when the project is built and run in Microsoft Visual Studio. The project is compatible with version 2010 and higher.

## A.1 Configuration

In the configuration window of the application user can set his preferences. This window is shown in Figure 5.5. Every item is described below:

- Write weight - sets the preference for writing
- Navigation weight - sets the preference for navigation
- Format weight - sets the preference for formatting
- Contexts - enables/disables division to contexts
- Signals On - number of trigger signals
- Signals One of N - number of analog signals
- Scan method - selection of scanning method
- Signal mapping - selection of the way of mapping signals
- File Name - name of the file under which will the document be saved

In the configuration window is used the two-dimensional scanning method. Active control has red background. Advancing to next control can be done by selecting *Enter* item. When user wants to advance to generated layout and wants to start with his work, he has to select *Confirm* button. This could be done by selecting every character (i.e. letter or number).

## A.2 Controlling the Keyboard

User can choose one of several keyboard layouts according to selected scanning method. These layouts have been described in chapter 4 of this thesis. Please refer to these sections for explanation of how each of the layouts works. The description is written in a form that should be acceptable for everyone, so it would be redundant to include the explanation here again.

### A.2.1 Mapping of Input Signals

To be able to control the keyboard properly, it is necessary to be aware of how outputs from the hardware device are mapped to actions in the keyboard. Each device typically provides one or more "signals". For example a single switch provides one output, which is trigged by pressing the switch. A device consisting of three switches provides three distinct outputs, each from one of the switches. In our hardware device, trigger signals and analog signal are recognized. In Figure A.1 is shown the *default* mapping of these signals to the actions it the keyboard.

| | Contexts | # of triggers | # of analog | Map # 1 | Map # 2 | Map # 3 | Map # 4 | Map # 5 | Map analog |
|---|---|---|---|---|---|---|---|---|---|
| 1 | N | 2 | 0 | Select | Step | X | X | X | X |
| 2 | N | 3 | 0 | Select | Step | Back | X | X | X |
| 3 | N | 4 | 0 | Select | Step | Back | Space | X | X |
| 4 | N | 5 | 0 | Select | Step | Back | Space | Enter | X |
| 5 | N | 1 | 1 | Select | X | X | X | X | Step |
| 6 | N | 2 | 1 | Select | Back | X | X | X | Step |
| 7 | N | 3 | 1 | Select | Back | Space | X | X | Step |
| 8 | N | 4 | 1 | Select | Back | Space | Enter | X | Step |
| 9 | Y | 3 | 0 | Select | Step | Context | X | X | X |
| 10 | Y | 4 | 0 | Select | Step | Context | Back | X | X |
| 11 | Y | 5 | 0 | Select | Step | Context | Back | Space | X |
| 12 | Y | 2 | 1 | Select | Context | X | X | X | Step |
| 13 | Y | 3 | 1 | Select | Context | Back | X | X | Step |
| 14 | Y | 4 | 1 | Select | Context | Back | Space | X | Step |

Figure A.1: Mapping of Input Signals.

Description of the mapping follows:

- Contexts - indicates if the context is enabled or disabled
- # of triggers - number of trigger signals
- # of analog - number of analog signals
- Map #1 - action mapped to first trigger
- Map #2 - action mapped to second trigger
- Map #3 - action mapped to third trigger
- Map #4 - action mapped to fourth trigger
- Map #5 - action mapped to fifth trigger
- Map analog - action mapped to analog input

If user chooses the *auto* option on settings screen, the algorithm decides, if the generated layout will be divided into contexts or if the layout will be one matrix. User has also the possibility to map the signals according to his/her preferences. If the user select *file* option, the signals will be mapped according to file "mapping.txt" which should be placed in the same directory as this Add-in (dll file). Each line in the file represents one signal and it consists of two parts - type of signal (*A* for *analog*, *T* for *trigger*) and action in user interface (*select* for scan selection, *step* for scan step, *back* for backspace, *space* for space, *enter* for enter, *context* for changing context). Example of form of such file follows:

```
A step
T select
T context
```

In this configuration, the first signal will be analog and will serve for performing scan steps. The second signal will be simple trigger signal and will perform scan selection. The last signal will be also trigger signal and will be used for switching among the contexts.

# Appendix B

# Content of the Attached CD

Here is the structure of directories on the attached CD:

- `bin`
  - `KeyboardBuilder.dll` ..... compiled application
  - `howto.txt` ..... information about how to run the application
- `src`
  - `KeyboardBuilder` ..... source codes of the keyboard builder
- `test`
  - `layouts` ..... keyboard layouts used for the evaluation
  - `scenarios` ..... scenarios used for testing
- `text`
  - `latex` ..... source files of this text
  - `rudolsim-thesis.pdf` ..... this text
- `thesis-assignment.pdf` ..... scanned version of the official assignment