**Abstract**

Text entry is present in various fields of expertise, be it on smartphones, portable end devices or PC’s, via physical keyboard or virtual software-based keyboards. The most common used text entry method being the QWERTY keyboard layout, whether on mobile devices or PC’s. With modern computer keyboard-like text entry methods and layouts on typewriters, dating back to 1829 , various attempts at improving keyboards have been made. This thesis focuses on researching and developing a text entry system and physical device for PC. As part of this thesis a device – altype – was created and evaluated. Altype works with chording, an alternative way of text entrance, via pressing a combination of keys, known as chords, in order to achieve text output. The research on keyboard performance improvement with chording, as well as the development process of altype is depicted in this thesis.

**Devising different Systems**

In terms of the standard keyboard layout, disregarding the different, country specific keyboard layouts, there are on average 105 key options arranged on a flat surface in a rectangular manner. With adjusting a keyboard for text input with one hand, certain changes have to be implemented. For text input only alphanumerical or character keys are of use, thus other keys can temporarily be omitted. To reach a basic level of text input, the character keys can be further reduced. Regarding the development of the different keyboard-prototypes built as a part of this thesis, text input in terms of characters used was limited to the standard Latin alphabet. As the main focus for conceptualising and building said prototypes is towards the efficiency and feasibility of concept, upper- and lowercase distinction will be left out for the time being and therefore only a unicase system will be analysed and implemented. In later versions a modifier key comparable to ‘Shift’ could be implemented additionally.

{TODO transition}

To fulfil the criteria mentioned in section{reference} following characters will be defined as relevant characters and will be referred to as such.

Standard Latin Alphabet , Space, Period(also Full stop or dot) and Comma

As part of measuring the efficiency, the number of errors is recorded, and therefore Backspace is deemed nonessential concerning the relevant characters. However, regarding normal text input use, the existence of a Backspace is warranted, therefore justifying its implementation.

The relevant characters amount to a sum of 30 whose assignment to a smaller number of keys, owed to the practicality and smaller physical form of the handheld text input device, shall be determined in the following.

**Systems of Entry**

The ergonomic layout and physical implementation not taken into consideration for the sake of this assessment, a range for the number of keys needed should be discerned. The amount differentiates by the means of input, of which two will be further looked at. The matter of key input can be separated into two, for reduced keyboard feasible, categories: repeated, single button push sequences and combinatory sequences of multiple keystrokes.

**Repeated, single button push sequence**

In devising a system that operates on a repeated, single button push sequence, the amount of consecutive key presses determine the character that is entered. Each key has an array of alphanumeric characters assigned to them and by depressing said key repeatedly in a set amount of time the character corresponding to the number of key presses is set. This system has its roots in the telephone keypad engineered by Bell Labs in the 1950s{reference}. Although it is optimized and was thoroughly researched on, this system of entry is not restricted to 12 keys but could be further reduced in number. Also, the order in which the characters of the alphabet are set to the keys could be altered to improve efficiency, which will be further evaluated in {reference}.

**Combinatory sequences of multiple keystrokes**

Contrasting to repeated single button push sequences, more than one button can be entered either simultaneously or in a sequential order. Depressing multiple keys at once results in a combinatory output whereas pressing the keys in order results in permutations also called sequenced combinations or combinations with order important. If a repetition of keys is desired, the preferred method of multiple keystrokes would be of a permutational matter as combinations, where all keys are pressed simultaneously, do not account for repetition. Additionally, it has to be said that combination wise, sequenced or not, repetition is a peripheral need as the possibilities of five keys cover enough assignable combinations and permutations to include the relevant keys.

**Combination, sequence order not important**

In the same manner as chorded keyboards{reference}, if the keyboard is able to handle more than one key pressed simultaneously, this opens up the possibility to enter text via combination of keypresses. Commencing with five keys, one for each finger of an able person’s body, and each key is a simple binary button with on/off states, the number of possible combinations equals to 32.

As the absence of any keypresses counts as a combination of five off-state keys, 31 assignable combinations or chords are left which would accommodate for the relevant characters{reference}. Given, that another combination is used as a modifying key, the remaining number of combinations would double. A similar approach could be taken with four binary buttons and one additional modifier key, which leaves 15 combinations, 16 minus the one for no keypresses, as of the four keys and double the amount with the modifying key implemented. This results in 30 assignable combinations which would also accommodate the relevant characters{reference}. For either of the two versions, five buttons are needed and the assignment of characters to the combinations has to be assessed in terms of memorability and efficiency. This will be covered in {reference}.

**Permutations, sequence order is important**

In contrast to the simple binary combinations, a different approach can be taken if the order in which the keys are depressed are given consideration. This concept also is encountered on a standard keyboard, the sequenced key combination Shift+S will display an uppercase S whereas S+Shift has no modifying impact, and nothing assigned so a lowercase ‘s’ will be displayed.

Again, five buttons for each finger are assumed as a starting point of calculation. To calculate the amount of all possible sequenced combinations the sum of all permutations with the number of buttons pressed, ranging from one to five out of five, are determined.

Using five keys with sequence order important amounts in 325 assignable sequenced combinations which can cover the entire extended 8-bit ASCII code including non-printable characters. When using one to four keys out of five possible or even one to three, the resulting sequenced combinations, , would be enough.

As the relevant characters amount to 30, a device with four-keys could be considered when implementing basic text input. When using up to three keys out of four, assignable sequenced combinations are attained, which would still be sufficient.

**Extended sequenced combinations**

Although not necessary in a simple implementation, the use of permutations could be further expanded. When considering that once the keys are pressed, the order in which they are released is also important, a bigger sum of combinations can be reached. For five keys this would come to 17685 combinations and with four keys to 748. By using three keys, 51 combinations could be reached.

In extending the sequenced combinations even further, by allowing sequences in which the keys are pressed, released, pressed, released, and so forth with the only condition being that one key must be pressed at any given time an exponentially high number of assignable combinations could be reached.

**Selected Version**

Given all the calculated and evaluated possible combinations and permutations, it was decided that for a simple implementation four buttons with combinations of up to out of three would be suited best. This way, the device could be scalable upwards afterwards as well as remain backward compatible, as the possible combinations, sequential or not, also remain within a bigger set. In the following, this chosen variant of text input will also be called ‘altype’. Footnote: Which’s origin stems from the two words alternative and type being joined together.

**System of Letter sorting and placement**

With the variant of four buttons and a permutational approach of sequential order with up to three feasible key presses out of the four possible selected, the matter of character placement and sorting has to be evaluated. A potential device version with 12 keys and similarities to the mobile phone keypad will also be briefly considered. This section will explore the sorting and overall placement of letters, an in-depth description of the means of entry and exact combination assignment of altype can be found in its respective chapter{reference}.

**Standard keyboard layout**

The origin behind the layout of the standard keyboard can be found in {reference}. As the user might be familiar with QWERTY or a variant thereof, it could be reasoned that sorting the characters in a way that resembles the grouping of QWERTY, it would be easier to familiarise oneself with the system of entry faster as there is no additional need to memorise the letter placement. But as the standard layout is optimized for text entry with ten fingers on a flat surface it was refrained from implementing this system for the device{reference}.

**Alphabetical order**

Knowledge of the Latin alphabet is implied and therefore thought should be given to and placing the letters in alphabetical order to increase the memorability of its assigned combinations. Depending on the matter of implementation, the letters of the alphabet could be grouped into smaller sets and the user should be able to identify all the characters in a group by just being told the first and last letter of said group. For instance, when given the theoretical key with the range inscription ‘a-d’, the user should be able to deduce the characters in between those two without effort. In that way, the letters could be grouped, and each key press divides it into smaller groups until the desired character is reached. This approach will be further examined in {reference}. Alphabetical order could also be used on a 12-key device, as each key has three or four consecutive letters assigned to them and thus making it uncomplicated to either estimate where the desired character is placed, or to memorise the character placement easier.

**Order based on empirical evaluation of frequency of use**

A contending order of letters could be based on what text is entered. Specifically, based on the most common letters within any given input, so that the most frequently used characters will have the easiest combinations assigned in order to type text efficiently. Assuming this, an algorithm was implemented to assess the frequency of use in any given text.

{algorithm}

Basic text input is limited to the standard Latin alphabet, so any non-alphabet character was dismissed. Additionally, no distinction was made between upper- and lowercase letters. Sample text was fed in via a .txt file and the analysis was carried out with python. The first analysis was done on a sample of 10 books. The books were provided in Plain Text by Project Gutenberg{reference}. Books used were the top 10 downloaded Books on the day of access and included colloquial defined literary classics{footnote}. Overall, the sample consisted of more than 2.7 Million characters. Below the letters are sorted after frequency.

{table}

In an attempt to personalise the text input to its user, the algorithm was modified to analyse the personal writing habits of said user. To achieve this, chat logs of various conversations{footnote}(: predominately in English) were downloaded from the communication service provider{reference?} and slightly altered before analysis. Downloaded chatlogs usually contain the date and time as well as the names of conversational participants and also messages from others. Those were removed and then the text was processed. Although the sample of 680 Thousand characters was considerably smaller, it still holds some empirical information.

{table}

The most used letter in both cases is ‘E’, and the first six positions show similarities. In general, the frequency of letter use collected from personal text usage show more resemblance to the most used letters in the German language as listed by the Duden{reference}. This discrepancy could stem from the small but still present German words in the sample which could explain the similarities to both lists.

While over time the use of either frequency-sorted list will increase efficiency, the presumed learning curve for this method is steeper than with sorting after the alphabet. Frequency sorting prevents grouping the characters in consistent and easily completable sets and thus each combination has to be learned individually. In turn, a 12-key device could benefit from such a sorting. As fingers have to hover above the keys, characters could be printed on the keys to guide the user. Additionally, if the most common letters are placed on different keys and not in immediate proximity to each other based on their successive use, the error rate could be decreased. This is due to its difference from the current implementation. Keys have to be pressed repeatedly with a small gap in time in between if two successive characters have to be typed after another, which increases the error rate.

**Altype**

**General Description**

As the name might suggest, altype is an alternative way to enter text using a handheld device. The design, interaction and technicality of the device will be further explained in the upcoming sections. Based on the findings of intensive research on already developed one handed keyboard implementations, presented in {reference}, and the findings of {reference}, the implementation of altype was done as an accompanying project to this thesis and realised during the fifth semester project.

**Project scope**

Main goal of the project was to conceptualise, create and test altype to provide further research bases and help assess, evaluated, and improve the findings of this thesis. In their nature, this thesis and the project were intertwined, and both took benefits from this reciprocal approach.

**Physical implementation**

**Design**

The question arose in which manner the device should be implemented physically. Two options were considered before starting, with it either being a handheld or table or surface bound device. During the first few stages of conceptualising and experimenting a table-bound device, consisting of an old keyboard with its unused keys removed{picture}, was used. But ultimately it was decided on a handheld device.

Unlike the table bound prototype with the keys facing the user, the finalised version of altype is held in one hand with the keys facing away from the user. On the backside, four keys are located in such a way, that the user’s fingers sit directly atop them in a relaxed hand state with the device enclosed between the four fingers and the thumb, as indicated in {picture}. In an early concept of the device{picture}, a fifth button for the thumb was contemplated but given the calculations and the practicability of a four-button implementation mentioned in{reference} omitted.

Albeit not part of the main focus, the design of the physical casing was treated as well. Still not the final version, but the produced casing provides a solid shell for the PCB, cables as well as breadboard plus buttons{picture}. It was constructed with the thought of a sleek, minimalistic remote that can fit into the user’s hand without disturbing its natural resting position.

**Keyboard analysis**

As a starting point, discarded keyboards{footnote}: provided by the IT Centre Hagenberg(?) were dismantled and their system of operation analysed. Once opened up, the keyboard consist of four main parts, a PCB (printed circuit board) with the contact pins{picture}, two membranes with conductive tracings{picture} also called the key matrix and physical buttons as well as a silicone sheet with outdents which uphold the buttons{picture}. Once a key is depressed, two tracing lines on the separate but overlaying membranes connect and the circuit, connecting with the pins on the circuit board, is completed. Thus, the circuit board which is connected to the computer sends a letter, number, or command.

The keyboard used was {reference}. Not all keyboards have the same number of pins or have the same key mapping within the key matrices but most keyboards which operate with membranes function in a similar way. The following numbers and statements apply to the keyboard used.

**Principle of operation**

The pins on the PCB are grouped into two sections, consisting of 8 and 18 pins, with each group connected to one membrane. Hereafter, the two groups are called ‘Side A’ with eight pins and ‘Side B’ with the remaining 18. In combining two pins with one from each side, the circuit is closed, and a key-event registered. By joining every possible two pins, a total amount of 144 combinations can be achieved, but not all combinations are assigned to keys. In the table{reference} the keymapping can be seen. For instance, by connecting pin number five of Side A with pin number eight of Side B, the letter S is produced.

{table}

**Keyboard hacking**

For the physical implementation of altype, only four buttons are needed. Consequently, by joining one pin of Side A to a row of buttons, and the remaining connector pins of the buttons to separate pins on Side B, four different letters are produced which can be intercepted before being processed by the computer. The planned keymapping can be seen in {picture} where the buttons are plugged into a breadboard.

Cables were soldered to the contact pins of the PBS and connected according to the schemata in{reference}. The keycodes and corresponding letters produced when pressing the buttons are as indicated below:

Button 1: Index finger – Q – 81

Button 2: Middle finger – P - 80

Button 3: Ring finger – I - 73

Button 4: Little finger – R - 82

Depending on different prototypes and thus different keyboard types with different keymaps and pin assignments, the keycodes vary and have to be adapted accordingly.

**Interaction Technicality**

Provided by the technicality of the input via the buttons discussed in {reference} and the considerations in {reference} regarding the arrangement of letters, the following was established:

*By a combination of three out of four buttons with sequence order important, a letter, sorted by their occurrence in the alphabet, is entered and further processed by the computer.*

With the foundation set, a system of assigning combinations to letters was devised. As mentioned in {reference}, arranging letters by frequency and assigning the most used to the easiest combinations could work with the requirement of training hours on that system. Also, a guide would have to be printed out in full, either on the device itself or on a screen with which the device is used together. This reasons why a different method was explored.

**Intuitive character/combination assignment**

As proposed in {reference}, when only the boundaries of a group of letters within the alphabet are given, the letters not stated explicitly can be deduced. On the basis of this assumption, by grouping consecutive letters into three groups, the user can decide on a group of letters by pressing the button that selects the desired group. If, by pressing the button once, the letter that initiates the grouping is selected, only the remaining have to be decided on. As with the initial choice the other group choices become void, the three remaining buttons can be repurposed and assigned new values. The letters, with the first letter of the group exempt, are split into three groups, again. In pressing the second button, this group is chosen. As with pressing the first button, if now both buttons are to be released, the first letter of this group is chosen.

By dividing the first selection of groups into three and not four, as four buttons are available, the alphabet can be split into two groups of nine and one group of eight letters. With the second decision, the letters are either split into groups with the size of three, three and two or three, three and one. Based on this calculation and the continuation of the beforementioned method of leaving out the first letter of the group, with the third decision two buttons and two, or one, letters are left to choose from. The two remaining letters are each assigned to a button, which’s depression then determines the final letter. An example of the entire selection process is shown and described in the following.

**Selection Process**

On the start of each new letter selection process the letters are grouped and shown as in {picture}. A singular button press of the fourth button will produce a whitespace, by holding the button down, the choices of the characters period, comma and question mark are presented. Once button one is pressed and the group ‘a-i’ is chosen, the next selectable groups are shown with their corresponding buttons as in{picture}. Now the second button is pressed and thus selecting the group ‘b-d’. If the two depressed keys were to be let go now, the letter ‘b’ would be selected. By holding down the key, once again the next choices are presented, now with only two letters left, each remaining button stands for a standalone character. In this specific selection process of the buttons one and two already pressed, the remaining two options would be ‘c’ and ‘d’ as can be seen in {picture}.

{TODO pictures selection process}

An Overview of the whole grouping of letters as well as the pressed buttons needed to select the letter can be seen in{reference}. The combination can be found out by taking steps of one indentation and down with each group chosen and joining the numbers next to the groups together.

On starting to use and explore altype a user is given a short description of the process, which reads as follows:

*The letters are entered by a combination of keys. On the screen you see four options corresponding to each button. The letter that is underlined is chosen if only that key is pressed. If a key is pressed, the options change. Again, if the letter is underlined, if that key is pressed additionally and then let go, this letter chosen and entered. Depending on the size of the group, one more decision can be made by choosing a button. Once all buttons of the entered combination are let go, the character is entered.*

**Preliminary Conclusion**

By proceeding as introduced in sections {reference} and {reference}, a novice user is guided through the selection process. While, for a novice or inexperienced user, the guide of possible selections and the changing letter groups assigned to each button have to be shown on the screen of the device that altype is used alongside of, a more practised or proficient user would no longer have the need for a guide. By arranging the letters this way and thus making the entered combinations a selection process rather than just combinations assigned to the letters, typing on altype, with this system implemented, should presumably be easier as the entire process is more intuitive and provides an easier way of remembering combinations. Once the need for the guide is surpassed, the initially introduced selection process makes way to muscle memory as well as the assumed, natural pattern recognition of the human brain. At the point of this substitution taking place, typing with altype should become efficient. This assumption will be further examined in {reference}.

**Altype Emulator**

This thesis’ nature is mostly theoretical thus mostly highlighting the thought process and representing scientific exploration based on research{reference}. The goal of the accompanying project altype was to yield physical results and prototypes to open up the possibility of user testing and consequently to corroborate, support or refute the preceding, established theories and concepts. Before going deeper into the conducted usability tests{reference}, the means of empirically collecting and primarily the additional, technical implementation need to be highlighted. In order to systematically approach the remotely carried out design testing and validation, altype had to be otherwise devised and arranged without compromising on functionality and the determined implementation. Hence, an emulator was created that would work with the physical altype version as well as with a standard keyboard where the user could either pick their own keys in place of the four implemented on altype or use the set keys ‘S’, ‘R’, ‘T’, and ‘H’. Those keys were chosen owed to their arrangement on a physical QWERTY keyboard. If the index finger of the left hand is placed on the key ‘H’, the fingers would approximately land on the pre-set keys. This key configuration also works for the right hand, with the index finger on the ‘S’. In this case, the button order within the emulator has to be changed.

The Altype Emulator consists of a main window from which the altype or a prototype version of a 12-key keyboard with letter arrangement based on frequency can be selected and started. The 12-key variant was implemented for a potential control group while user testing. The main window is composed of a menu bar with different menus and an output window, where either the emulator is shown or general information. With each emulator, the window is split in two, the top window being the text output and the bottom window the guide for the chosen system. With altype, the guide is changing and adapting with each press of a button.

The menu bar contains settings, in which the configuration of the buttons of altype can be changed. This is meant for users who want to experience altype from a standard QWERTY keyboard. In addition to settings, the menu-bar has the options of ‘help’ and ‘about’. When clicking on ‘help’, a small window is displayed which features the explanation of use, as described in{reference}. ‘About’ simply displays a small description of the project overall and the link to this thesis. The implementation of the mentioned, additional options will not be further illustrated.

**Code examination**

In order to register key-events the *java.awt.event.KeyListener* interface is used{reference}. The interface provides the class which extends it with three main methods, of which keyPressed and keyReleased are used. While keyPressed registers the buttons and adds them to the combination, only when the keys are released, the combination is logged and checked for a valid combination. The variable KeyEvent e holds a numeric int value which corresponds with the characters pressed key in ASCII code. In a switch case, the pressed key’s ASCII value is substituted with the number of the button pressed in order of appearance and added to the combination.

As some keyboards have trouble registering multiple key-events at once, the emulator might not work with every user as of now. Because some keyboards are also prone to registering a keystroke multiple times, a method was implemented to remove doubles from the combination, as combinations with repetitions allowed create a different set of rules and combinations. Only when the first key entered, saved in the temporary integer variable *firstKey*, is released, the combination entered is logged and checked for validity and then their corresponding letter assignments. This is done in keyReleased, which alongside keyPressed runs and checks key events continuously at runtime of the program. The two mentioned methods are shown in {reference}.

{code} {TODO select code, add label}

In {reference}, two additional methods are called: removeDoubles() and getLetter(). While removeDoubles() just removes duplicate values in the input combination, getLetter() checks the combination for validity and returns the assigned letter, if valid. This is done by positioning the letters of the alphabet in a three-dimensional array and the combinations of three out of four check the location in said array for entered letters. If no letter is assigned, a zero is returned, which is then declared as an invalid Character and not added as a letter in the combined *text* variable which contains all letters typed until this point.

{code}

In the case of a zero being returned no character is entered, as well as when all four buttons are pressed simultaneously. This case is checked before calling getLetter() and thus not included in the character array containing the alphabet. By pressing all four buttons, the user is able to backspace and delete a typing error or an incorrectly selected character. In {reference} the three-dimensional array can be seen. Thought was given to placing the alphabet in a simple array or list and accessing the letters by the indices of a joint combination but discarded as this would produce more empty indices as a 3d array would.

{array}

After the selected letter is entered and added to the text, it is output to the main window where the text and the guide can be seen. In a similar manner as the letter selection process, via converting the combination to indices in an 3d array, the guide is determined. The main difference being that the first button determines the second menu option and the second button the third. This offset is due to the selection options given by the guide, previous to pressing any button.

To create conditions and meet requirements set in place for testing the device with novice users, an additional menu was added where on-screen feedback regarding typing speed and accuracy could be enabled and analysed accordingly. This will be treated in {reference}.

{pictures}

**Evaluation of efficiency and accuracy on the base of usability testing**

The study was designed to determine the learning rates, efficiency measured by WPM{footnote}words per minute and accuracy. It was carried out as a longitudinal study, where repeated observations under the same circumstances could be made. In terms of experiment type, it could be classified as an explanatory experiment, as no clear hypothesis was stated, and its raison d’être was mainly to test the design and to aid and validate the preceding predictions and interaction design choices.

**Statement on objectivity based on self-participation**

Important to mention, before further depicting the study as a whole and participants including the recruiting requirements, it has to be mentioned that during the conducted usability testing, I acted as the conductor and observer, but also participated in the study myself. The integrity of the objectivity shall not be harmed, as my understanding and me being the developer of the device does not put me at an advantage over other participants in terms of efficiency and accuracy. My usage of the device and training sessions were strictly recorded. Only comparable data, collected with the same conditions and rules of external participants in effect, was taken into calculation and was analysed.

**Method**

**Participants**

\label{sec:participants}

Five participants were recruited based on their ability to type on a QWERTY keyboard in varying speed ranging from 25- 60 wpm, their technical proficiency, as well as handedness and native tongue or degree of fluency of the English language.

These factors were chosen based on the following reasonings.

Ability to type on a QWERTY keyboard: the user of altype should be familiar with at least one version of the QWERTY keyboard on a daily basis with varying typing speeds. QWERTY versions including different language arrangements as mentioned in{reference} and/or mini-QWERTY on mobile phones.

Technical proficiency: linked with the daily use of a keyboard, the user should be proficient in using an application such as the altype emulator. One added factor that is also linked to technical proficiency in terms of keyboard use is age, as mentioned in{reference}. Therefore, in order to maximise the research scope, a wider age range within the participants was encouraged.

Handedness: because altype can be used with both hands without the need of physical changes, participants with different handedness’ were chosen. All changes made were the assigned order of buttons within the emulator code.

Native tongue: As the used test-set phrases were in the English language, and no inquiry and comparison on performance based on different native tongues was intended, only participants with proficient control over the English language were chosen. Proficiency was measured by daily use of the English language in word and writing and not on the ability to talk and verbally articulate themselves.

Each participant was informed of the significant time commitment required and were motivated by a competition among themselves and the prospect of a small financial reward for the participant achieving the highest relative change on efficiency with using altype. With giving the distributions of factors among participants, I am included in the statistics and therefore the sample number of participants is six.

Half of the participants were female. Not related to gender, also half of the participants dominant hand was the right hand. Two participants were under the age of 25 an one was over the age of 50, with three participants ranging from 26-49 remaining. Four participants were native English speakers and two used written English as part of their daily life due to their education or work with at least two hours of writing per day. Not part of the altype assessment but part of the collected comparison variables, two of the participants typed with one of their index-fingers on the mini-QWERTY keyboard whereas the other four typed with either one or two thumbs. While typing on a standard keyboard, all but one typed with the use of more than 8 fingers on both hands while only one person entered text with up to four fingers, albeit also on two hands.

Not included in the participation requirements but worth mentioning nevertheless, two participants played the piano for more than six years, which could potentially increase their ability to apply the concept of chording on altype.

**Equipment**

Due to the given remote nature of the conducted study, each participant was required to possess an end device capable of running the java emulator as well as a keyboard where up to four simultaneous keystrokes had to be able to be registered. Other than the different locations of subject and conductor, the test was a stationary one with the user sitting at their device running the software. Additionally, a video call was held with the camera of the participant facing their hand during text input and their screen shared while testing and the emulator running. The emulator and usability testing software is self-administered with the conditions within which the experiment is conducted being under researcher supervision.

All participants had their keyboards attached and connected to the end device via a serial cable which continually sent the state of the selected keys to the computer. The serial stream was parsed as text input by the emulator software. To minimise distraction by the printed or engraved letters on their keyboard, the participants were implored to tape over or to cover the keys not used.

**Design**

\label{sec:studydesign}

The study had two independent variables with each having three levels over the span of ten sessions. Expressed differently, the study was structured as a 2x2x10 within-subjects factorial design. The conditions were typing on altype with either hand(two conditions) and typing with the device visible and hidden(two levels) in ten sessions. Totalling at a number of 240 trials( $12 \text{participants} \times 2 \text{conditions} \times 2 \text{levels} \times 10 \text{sessions}$), the dependant variables were WMP and accuracy.

The sessions were scheduled over five weeks with a minimum of one session per week. Each session lasted between 20 and 30 minutes, depending on the participants’ speed. The sessions consisted of two main parts with two levels each, delineated by the set conditions. Each participant started with their dominant hand and switched after completing all tasks and phrases. This was altered each session.

The participants were shown the keymapping guide alongside the WMP on their screen{picture}. The accuracy was calculated after finishing each phrase. Before starting the measurements, each participant was guided through the alphabet once by highlighting the button that needed to be pressed. A second ‘warm up phrase’, also made up out of the alphabet, was entered without aid but with the guide present still. After finishing the alphabet in full, with 100 percent accuracy, a phrase was shown {picture} and as soon as the participant started typing, the data was recorded, and their performance was recorded. The phrases were representative samples of the English language devised by MacKenzie and Soukoreff and chosen at random out of the given 500 phrases{[reference](https://dl.acm.org/doi/pdf/10.1145/765891.765971?casa_token=77CNLhLESsQAAAAA:S_NqVyKx3XfL82kDFfDC7wszWPx8ffIadOXvGimFU_911uc0pKYagsurn3c7B04u-QtXM_ol8veTwA), [link](http://www.yorku.ca/mack/PhraseSets.zip)}. The phrases are short, were designed to be memorable, and are written with American spelling {reference}.

As the original phrases consisted of upper and lowercase letters, they were converted into lowercase. The phrases lacked punctuation marks and as altype is able to enter punctuation marks in limited form, they were added to the phrases at random with the same probability for each character. In the first four sessions each participant was tasked to complete one phrase per hand and alteration, resulting in four phrases. With each following session, the number of phrases was increased by one every two sessions, amounting to four phrases per hand and alteration in the 10th session.

During the first session, the participants were asked to enter 10 phrases with a standard and mini-QWERTY keyboard as well. This provided a baseline for a comparison with text entry rates of each participant. The need to further evaluate or test the typing rate on QWERTY each session was deemed non-essential, as each participant, prior to the study, typed daily and no increase of WPM or accuracy was expected to happen within the 10 sessions as the use of QWERTY was already well established with participants.

**Means of measurement**

The software measured each button press individually by obtaining the state of each button in a log file, as well as the time each state changed. For this, Java’s *System.currentTimeMillis()* was used. Also noted in the logfiles were the hand used, if the participant was typing with the device covered and the phrases typed. With this data collected, the time needed for each character and the time between characters and buttons can be determined. WPM were calculated from the phrase and the time per character. The accuracy was determined after the phrase was transcribed. The transcribed phrase was declared finished when: 1. The same number of characters as the phrase was reached, 2. the punctuation mark was entered, 3. The number of words determined by whitespaces matched.

Outside the additionally implemented usability test software and testing environment, the simultaneous pressing of all four buttons produces a backspace and deletes the last entered character{reference}.

The accuracy was calculated in percentage relative to the phrase. Errors were counted letter by letter in each word, separated by whitespaces. The accuracy or error rate and WPM were individually written to the log file and shown on screen. The on-screen variables were the session’s calculated average of the reached WMP and accuracy.

**Procedure**

At the beginning of the first session the participants were given a written and a verbal explanation of the task and the overall goal of the study. As the study was held remotely, each participant was supervised individually. The device was shown and explained on how to use. As it was not possible to procure a device for each contestant of the experiment, they were instructed on how to use the emulator with their keyboard instead. Given this circumstance, only the system was tested and evaluated. Also shown in the first session was an example use scenario of the device as well as how to operate and use the software.

The emulator is self-administered but nevertheless each session with each individual was supervised. The participants were then asked to enter the ‘warm up phrase’ described in section \ref{ sec:studydesign}. After completion, the participants were instructed to copy the phrase shown on the screen via the altype emulator and connected keyboard. Their instructed goal was to type as fast as possible and to minimise the error rate.

After finishing the phrase goal on one hand that was set for the respective session, the users were asked to take a short break and then to switch their hands and repeat the process of entering phrases with the other hand. During testing, the participants were also shown their error rates and their overall accuracy as shown in Figure {picture}.

During testing, the camera or webcam of each participant was pointing towards the keyboard and hands to ensure a correct implementation of the conducted study. During blind testing, the participants were asked to either cover their keyboard and the used hand with a piece of cloth or place the keyboard atop their lap and have the view of the keyboard obstructed by the desk their computer or laptop was stationed. During this, the camera was faced at the participant with their hands and their movements in sight.

**Results**

**Data summary**

As a total, 528 phrases were collected in total with 88 phrases for each of the six participants over the time of 10 sessions in five weeks. The entire study amounted to approximately 30 hours with a time expenditure of 5 hours typing with altype on average per participant.

**Text Entry Speeds**

The mean text entry speed for the first session was on average 3,9 WPM, ranging from 2,8 WPM to 4,7 WPM. With continuing sessions, the speed increased, and the participants achieved higher WPM. In the last session an average rate of 17,13 WPM, which is an increase of 439,29 %.

From the collected average WPM of each participant a graph was plotted and based on the power law of practice as explored by Newell and Rosenbloom{reference} an exponential regression curve was derived (see Figure \ref{fig:plotwpm}). The derived equation can be seen in equation \ref{eq:expo}.

\begin{equation}

f(x) = 2.08 \times x^0.4732946

\label{eq:expo}

\end{equation}

{figure}

The $x$-values are the number of phrases typed, whereas $y$ indicates the WPM achieved. Based on Pearson’s correlation coefficient proposed by Karl Pearson in {reference}, the fit of the curves in relation to the collected data was calculated, as can be seen in equation \ref{eq:pearson}.

{equation}

The resulting correlation coefficient $R^2$ is $99,22%$, meaning that then curves are well fitted to the data, accounting for $99,22%$ of the variance within the data.

\subsection[**Error Rate**]%

{Error Rate%

\protect\footnote{In this section, error rate and accuracy are used almost interchangeably, with error rate describing the percentage of errors per phrase, and accuracy describing the inverted error rate: the percentage of correct entered characters.}}%

To evaluate the accuracy of the participants’ transcribed phrases Soukoreff’s and MacKenzie’s total error rate metric{reference} was used. The approach mentioned in their paper allows to incorporate corrected as well as uncorrected mistakes into calculations. The error rate is based on the Minimum String Distance Error Rate\footnote{In their paper two approaches to MSD error rates are mentioned, this thesis focusses on their new and improved version declared as ‘New MSD error rate’} which calculates the error rate depending on two factors (see equation \ref{eq:msderror}). The mean length of the presented and transcribed phrase and the Minimum String Distance, also MSD. $P$ is the presented phrase, $T$ the transcribed text and $\overline{S\_A}$ the mean phrase length.

The already implemented MSD Java class{reference} provided and implemented by Soukoreff and MacKenzie was used and modified accordingly in order to work with altype and their proposed changes to the MSD error rate\footnote{The published software by Soukoreff and MacKenzie was implemented in 2002, their changes to the error metric was proposed in 2003.}.

\begin{equation}

MSD error rate = \frac{MSD(P,T)}{\overline{S\_A}}

\label{eq:msderror}

\end{equation}

\paragraph{MSD} is a method to determine the needed number of changes to the transcribed text in order to achieve the presented phrase.

With MSD the possibility to determine between corrected and uncorrected errors is absent, so additionally the error metric Key Strokes per Character (KSPC), also proposed by Soukoreff and MacKenzie{reference} was utilised.

\paragraph{KSPC} calculates the relation of the number of keystrokes to the transcribed phrase by examining the keystrokes used to enter said phrase (see equation \ref{eq:kspc}). If a participant corrects an error, they press all four buttons and produce a backspace deleting the error. This keystroke is also counted with KSPC. This means that, when a phrase is 100% correct, KSPC amounts to 1,0. If, for instance a letter of the phrase \textit{‘sphinx of black quartz judge my vow’} with 29 characters is misspelled and instead \textit{‘sphinx of bladk quartz judge my uow’} is entered but then corrected, the KSPC would amount to 1,14. This is due to the length of the input stream, where two backspaces and two character-corrections are added to the length.

\begin{equation}

KSPC = \frac{|inputStream|){|transcribedPhrase| }

\label{eq:kspc}

\end{equation}

Also affected by carried out character corrections is, in addition to the KSPC value, the speed of text entry.

Because participants tended to not correct their mistakes and trade efficiency over accuracy, most errors were left uncorrected and thus rendering the KSPC less significant than the MSD error rate.

In Figure \ref{fig:ploterror} the average error rates per phrase for both conditions (see Section \ref{sec: studydesign }) are displayed. All error rates after the third session are less than 6,7%. With the completion of the third session, 12 phrases were entered by each participant.

While average error rates started at 15.84% in the first session with four phrases, by session two (8 phrases completed) the error rates decreased to an average of 9,25%.

\begin{figure}

\label{fig:ploterror}

\subsection{General Observations}

As initially mentioned in Section \ref{sec:studydesign}, the condition of which hand was used to type was also evaluated. While in the first session a slight difference in WPM could be detected whether a participant used their dominant hand or not, with their dominant hand performing better by an average of 0,64 WPM additionally, by session two, no real observation regarding this could be made. Handedness will also be discussed in Chapter \ref{cha:discussions}.

\section{QWERTY WPM as baseline comparator}

As mentioned in Section \ref{sec:studydesign}, the text entry rate on QWERTY and mini-QWERTY was also measured in order to provide a normalised metric comparator for altype WPM. In Table \ref{tab:comparator}, the average typing rate of QWERTY can be seen as WPM with the WPM of mini-QWERTY and altype relative to QWERTY. In this statistic, the average text entry rate after 88 completed phrases was chosen as an indicator as to how altype compares to QWERTY per participant and on average. Regarding the table, it can be observed that altype rates show consistency for the most part. This could further indicate that, with using the WPM of a user, the text entry rate with altype could be predicted. As the number of participants was small and the performance of one a single participant influences the result strongly, this metric has to be applied carefully.

\section{Comparison physical device versus emulator}

Due to the physical device not being part of the usability testing with additional participants, the data collected on the device stems from personal use. After participating in self-monitored sessions, the same methods in place for testing on the emulator were used on the physical implementation. Rates were slower by just 0,06\%. It was noticed, that typing on the physical device required more conscious effort to press the buttons, as the buttons were harder to press, resulting in 93,52\% performance of the emulator.

\ref{sec:physicalalterations}

<https://kilthub.cmu.edu/articles/journal_contribution/Mechanisms_of_skill_acquisition_and_the_law_of_practice/6607196>

<https://books.google.at/books?id=60aL0zlT-90C&source=gbs_navlinks_s>

<https://dl.acm.org/doi/10.1145/642611.642632>

<http://www.yorku.ca/mack/MSD/MSD.html>

**Findings and discussion of limitations**

\section{**Findings**}

Handedness:

Using altype made a small to nearly non existing difference with using it with the dominant or non- dominant hand. While at first participants were slightly slower with the non-dominant hand, after the first session no difference could be observed. Participants that achieved generally lower WPM at the start of the study, did so with both hands. Mentioned briefly in Section \ref{sec:participants}, the participants with experience in playing the piano obtained higher WMP rates in the beginning but other participants caught up or surpassed their WMP score by session 4.

{figure}

The ability to control individual fingers reliably at first from reading sheet music could have had an influencing factor on the results.

What was also discovered, was, that while the first phrase of the second hand used each session(alternating from dominant to non-dominant hand each session) was taped slower or had a higher error rate, on average the second hand performed better than the first. This could stem from the exercise already completed with the first hand, with the switch from one hand to another temporarily affecting the first phrase while inverting the movements. Still, because the order in which the keys are pressed is the same and typed by the same finger but on the other hand, exercising and memorising the system on one, benefits the other hand as well.

Ideally, with a bigger sample size of participants, the analysing of handedness would have been completely split. This would result in four test groups, two groups for each hand: use of altype on dominant hand and non-dominant hand. Although there should be no difference between left- and right-handed participants, it still would present an interesting metric.

Small sample size:

Because of the small sample size, partly dictated by the considerable amount of time testing occupied, the data collected presents a smaller pool of information to make assumptions of. Despite fewer participants and only 10 sessions per participant, the presented data in Section \ref{} holds valid information, and feedback of participants also helped in understanding the requirements needed to improve altype further.

Testing of the physical device:

The chance of also testing the physical implementation of altype with participants would have been greatly appreciated but testing the system via the implemented emulator provided enough insight to find potential improvements for altype.

Exercise:

As with most things, more time spent using and exercising writing with altype would potentially increase WPM and also the errors produced by accidentally choosing the wrong button sequence. If the equation for the power law of practice (see Equation \ref{}) was evaluated further than phrase 88 after session 10, with a total of 20 sessions and approximately 250 phrases completed, WPM could reach up to 28 WPM.

\section[Limitations and Improvements]{Potential improvements based on found limitations}

During testing several issues emerged, which, albeit not needing immediate correction or improvement, would refine altype overall if implemented.

\section{Code Enhancement}

\paragraph{Timed Key Sequence:}

One limitation that arose, was that a wrong sequence was entered when a finger was lifted prematurely resulting on an error because of unintentional finger movement. If, instead of pressing down all keys until the desired selection is reached, a sequence could also be entered by rapidly pressing the keys in correct order, this could be eliminated. For this, the optimal time span has to be determined, with giving thought to time between key presses and characters. If an experienced user would type two sequences consecutively after another, an invalid sequence would be entered if the pause needed in between key presses was not adhered to. This would have to be implemented additionally to the current method of operation, because the system implemented as-is was optimised to be accommodating for novice users.

\paragraph{Correction of sequence:}

If a wrong key was pressed unintentionally, which happened more frequent in the first few sessions, the user should be able to correct the keypress without the need to enter the wrong character, then deleting it and then entering the correct sequence. This in turn would only work with keeping keys pressed down while selecting the character. If a wrong key is pressed, the user should be able to release said key and select the correct one instead, without adding the wrongly pressed key to the selection sequence.

\paragraph{Backspace:}

Remarks given by participants was on the use of backspace. Currently, pressing all four keys at once will delete the last entered character. Participants noted that it would be helpful to also implement the backspace option in such way, that when the keys are held down longer, characters would be deleted continuously until nothing is left. The longer the key is pressed, the faster the deletion process should happen. This could be implemented by checking the time passed while holding down the keys and continuing to delete one character per time instance, with the time instance decreasing the longer the keys remain pressed.

Another note concerning the backspace sequence was, that it differed from the other key press modalities and thus was standing our or not as intuitive. There were disagreements among participants whether this affects typing in a positive or negative manner. The sequence for backspace was deliberately chosen this way, to accentuate the backspace and set it apart from other sequences as it is rather an action than a character and thus should not be grouped with other characters.

\section{Physical Alterations}

\label{sec:physicalalterations}

The physical flaws described here are based on my personal use of the device. As the device was modelled and built for my hand and then optimised and adapted accordingly, other users might have different or not the same problems I encountered.

\paragraph{Handedness:}

The emulator makes no difference in handedness, the only change being the order of keys which are simply inverted. Thus, the usability testing found that there is no difference in performance regarding left- or right-handed use. Albeit the ability to use the physical altype device on both hands as well by turning the device upside down and switching the order of keys, it has to be stated that the device is optimised for right-handed use. The buttons are indented based on the length of the fingers wrapped around the curve of the device and hereby the arrangement is not entirely symmetrical. The asymmetrical placement of buttons can be seen in Figure \ref{fig:} in Chapter \ref{cha:}. Either the device is implemented for either hand separately or a bit of the fitting placement of buttons relative to the fingers is compromised in order to be able to turn the device upside down and use it with the left hand.

\paragraph{Hand Positioning:}

While the device fits well into the hand, and also the buttons are reached comfortably, it was found hard to press buttons resulting in less WPM. This could be due to the buttons but also the small amount of force available for pressing the buttons in the half-opened hand state. Further prototyping would have to be conducted in order to optimise the use. Three possible solutions could be looked into. Either different buttons with a smaller resistance are chosen, the device is increased in its depth or an entirely different layout is chosen. A different layout version could for instance be a flatter device, comparable to a smartphone with buttons on one of the longer sides\footnote{Somewhat similar to the volume keys on the side of the smartphone} . Changing the physical device could result in slightly higher WPM rates, comparable to the text entry rates achieved on the emulator.