**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 said{reference} criteria 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.

{TODO label}

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.

{TODO labels, one formula and references}

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.

{TODO label}

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 chose 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 is placed on the key ‘H’, the fingers would approximately land on the pre-set keys.

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**

Objective, motivation,

**Method**

**Participants**

Who, why, specification, motivation

**Equipment**

What, how(why)

**Design**

General layout, specifics on when and how often

**Procedure**

What and how

**Results**

**Data summary**