Mobile Interface Development for Tongue-Controlled Devices Aimed at Training and Real-Life Interaction

BY

SILVIA MADDALENA ROSSI
B.S., Politecnico di Milano, Milan, Italy, 2018

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Bioengineering
in the Graduate College of the
University of Illinois at Chicago, 2020

Chicago, Illinois

Defense Committee:

Hananeh Esmailbeigi, Chair and Advisor

Joseph Michaelis

Enrico Caiani, Politecnico di Milano

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my advisor, Dr. Esmailbeigi. Not only for proposing me an amazing and stimulating project to develop throughout the months of research in the Wearable Technology and Sensory Enhancement Laboratory (WTSE Lab, UIC), but also for always being there to push me, support me and motivate me to do my best.

Moreover, I want to thank both other members of the committee, Professor Michaelis and Professor Caiani, for always being available and taking the time to give me precious insight for the project.

Many thanks to Nicholas Marjanovic, Giulia Soresini, Davide Lacca and Davide Bondavalli, together with the rest of WTSE Lab, for their constant indications, suggestions and support.

Thank you to Mallika, Maryam, Mahal and Anya of the Early Researcher Scholars Program for taking an interest in this work and deciding to make part of this challenge their project.

A special thanks goes to Francesco Sgherzi, whose feedback and insight in React Native, together with his patience, were of great support.

I would also like to thank all my friends, both the ones in Chicago and the ones back at home, and my family for always being there, through the ups and downs of this incredible academic experience.

TABLE OF CONTENTS

CHAPTER **PAGE**

1. INTRODUCTION [56] 1

1.1 Upper Limb Impairment 1

1.2 Available Assistive Devices 2

1.3 Intraoral Devices 4

1.4 Enabling Technology and Quality of Life 7

1.5 State of the Art: interfaces for Accessible Interaction 8

1.6 Tongue Training 9

2. MATERIALS & METHODS [56] 11

2.1 Hardware 11

2.1.1 O-UIC Device 11

2.1.2 Tongue Trackpad 13

2.2 Preliminary data collection 14

2.3 Interface Development 15

2.3.1 React Native 16

2.3.2 Custom-made components 17

2.4 Usability and inclusivity of the interface 21

2.5 Bluetooth communication 22

2.5.1 O-UIC device BLE control 23

2.5.2 BLE control for mock-wheelchair 25

2.6 Navigation between screens 27

2.6.1 SOS call 27

2.6.2 Generic phone-call 27

2.6.3 Email and Messaging 29

2.7 Functioning Evaluation 30
2.7.1 O-UIC Interface 32
2.7.2 Tongue Trackpad 32
2.8 Keystroke Level Model 33
2.9 Tongue Training Interface 34
2.10 User Feedback Survey 36
3. RESULTS AND DISCUSSION [56] 38
3.1 Survey Data 38
3.2 Developed Functionalities 40
3.3 Usability and inclusivity 44
3.4 Fitts Law 46
3.5 Keystroke-Level Model 48
3.6 Functioning Evaluation 50
3.7 Observations and Discussion 52
3.8 Tongue Training 56
3.9 Feedback Survey 60
4. CONCLUSIONS AND FUTURE STEPS [56] 62
5. CITED LITERATURE 64
6. VITA 70

LIST OF TABLES

TABLE **PAGE**

TABLE 1: TASKS FOR O-UIC DEVICE INTERFACE TESTING 32

TABLE 2:TASKS FOR TONGUE TRACKPAD INTERFACE TESTING 32

TABLE 3: KLM TIMES FOR EACH FINITE ACTION NECESSARY TO EXECUTE A TASK 33

TABLE 4: SEQUENCES OF ACTIONS NECESSARY TO EXECUTE EACH TASK IN EACH INTERFACE (TASK FOUR IS ONLY EXECUTABLE WITH THE TONGUE TRACKPAD) 34

TABLE 5: DEVELOPED FUNCTIONALITIES; THE X IDENTIFIES WHICH VERSION OF THE INTERFACES SUPPORTS THE DEVELOPED FUNCTIONALITY 43

TABLE 6: DIFFICULTY INDEX OF THE DEFAULT KEYBOARD COMPARED TO THE CUSTOM-MADE ONE 46

TABLE 7: THEORETICAL TIMES NEEDED TO COMPLETE THE THREE TASKS USING THE TWO DEVICES 48

TABLE 8: SECONDS NEEDED BY EXPERT USER TO COMPLETE THE TASKS PRESENTED DURING THE EVALUATION OF THE INTERFACES. 49

TABLE 9:SECONDS NEEDED TO COMPLETE TASKS AND ERRORS MADE BY EXPERT USER INTERACTING WITH THE O-UIC DEVICE AND THE APPLICATION 50

TABLE 10: SECONDS NEEDED TO COMPLETE THE TASKS AND ERRORS MADE AN EXPERT USER INTERACTING WITH THE APPLICATION WITH THE TONGUE TRACKPAD 51

TABLE 11: SECONDS NEEDED TO COMPLETE THE TASKS AND ERRORS MADE AN EXPERT USER INTERACTING WITH THE PHONE'S DEFAULT SYSTEM WITH THE TONGUE TRACKPAD 52

LIST OF TABLES (continued)

TABLE 12: USER RESPONSES TO THE ONLINE SURVEY COLLECTING FEEDBACK ON THE DEVELOPED APPLICATIONS 60

LIST OF FIGURES

FIGURE **PAGE**

Figure 1:Effects of spinal cord injuries as found in “Spinal Cord Injury Facts and Figures at a Glance Re-Hospitalization” by Level et al. 2

Figure 2: a) O-UIC device with 8 capacitive pads; b) Tongue Trackpad device with capacitive rows and columns 6

Figure 3: Code snippet of the communication between the O-UIC device and the mobile phone’s application. The resulting string of characters is stored in a characteristic of the BLE communication 12

Figure 4: Communication between (a) O-UIC device and (b) Tongue Trackpad and application. The first entails a direct handling of the information on the application’s end, the latter has the OS of the phone bridging between device and application. The device’s output is seen as a usual control for HID mouse cursor..... 14

Figure 5:Specific color palette used during the application's development..... 18

Figure 6: O-UIC device with eight capacitive pads numbered to illustrate the location of the input given by the user 23

Figure 7: Information flow between the user and the screens. The reducer acts as a control center, receiving the instructions from the screen, executing the requested case and forwarding the resulting command 25

Figure 8: Communication flow from the application to the Elegoo’s Arduino Uno board. The BLE communication is between the App itself and an Espruino Puck.js, which is connected via serial communication to the Arduino board. 25

Figure 9:Example code of the firmware updated on the Puck.js to set the characteristic and service needed and control the serial communication to the Arduino 26

LIST OF FIGURES (continued)

Figure 10: Code snippet from the O-UIC PhoneScreen showing how the commands received from the reducer are decoded in the screen itself and transformed into numbers to call 28

Figure 11: Example of the code use to print the console logs 31

Figure 12:: Example of a console.log of the execution of 6 tasks by the expert user, interacting with the two versions of the app through the O-UIC device (a) and Tongue Trackpad (b). The timestamps are used to calculate the time needed by the user to perform the tasks 31

Figure 13: Screenshots of the presented functionalities for the section regarding the O-UIC Device 37

Figure 14: Screenshots of the presented functionalities for the section regarding the Tongue Trackpad . 37

Figure 15: Survey data; a) respondents' living situation; b)desired functionalities of a tongue-controlled wearable device; c)type of ailment of the individuals who answered the survey; d)gender distribution of the individuals who answered..... 38

Figure 16: Developed screens for the implemented functionalities; a) Home screen; b) Wheelchair Control screen; c) Phone screen for the O-UIC; d) Phone screen for the Tongue Trackpad; e) Email screen for the O-UIC; f) Email/SMS screen for the Tongue Trackpad. 41

Figure 17: a) Custom-made keyboard with wider buttons; b) default Android keyboard 47

Figure 18: a) Custom made phone keyboard with distanced and well-defined buttons; b) default phone keyboard 48

Figure 19: Image inserted in the O-UIC device's phone interface to represent the link between the pads and the letters 53

Figure 20: Recipient and body field in the email and SMS screens 54

Figure 21: Screens developed to test hypotheses one and two; a) trains the vertical movement of the tongue from the front of the palate to the back; b) trains the horizontal movement from the left of the palate towards the right 57

LIST OF FIGURES (continued)

Figure 22: Screens developed to test hypotheses three, four, five and six; a) trains elevation and maintenance of the tongue-palate contact, tested in hypothesis four b) trains the speed in elevation of the tongue; c) the screenshot refers to the game developed for the fifth and sixth hypothesis. Since the goal is the same, click on the dot as it moves around, the layout was left identical; what changes is the speed with which the dot moves around..... 58

Figure 23: a) Data representation of the scores for two games, shown as an example. The user can update the graphs to show all the recent repetitions of the games; b) Notes section, where either the user or eventually a physician can add comments regarding the progress. 58

Figure 24: Results of trials of the games with the Tongue Trackpad; (a) results for 10 seconds holding game and tapping on the palate game; (b) results for vertical and horizontal movements; (c) results of games clicking on the dot appearing randomly and accelerating. The last game displays the frequency of tapping the dot, in Hertz, while the other games display the scores. 59

LIST OF ABBREVIATIONS

WHO ..... World Health Organization

CNS ..... Central Nervous System

USA ..... United States of America

BCI ..... Brain Computer Interface

EEG ..... Electroencephalography

EMG ..... Electromyography

SNR ..... Signal-to-Noise Ratio

WTSE ..... Wearable Technology and Sensory Enhancement Laboratory

BLE ..... Bluetooth Low Energy

ACAT ..... Assistive Context Aware Toolkit

ALS ..... Amyotrophic Lateral Sclerosis

IRB ..... Institutional Review Board

O-UIC ..... Oral User Interface Controller

IDE ..... Integrated Development Environment

MAC ..... Media Access Control

OS ..... Operative System

UUID ..... Universal Unique Identifier

T9 ..... Text on 9 [keys]

dpi ..... Device-independent-pixels

TTS ..... TongueToSpeech

LIST OF ABBREVIATIONS (continued)

UIC-C ..... User Interface Cursor-Controller

SOS ..... Save Our Ship (morse code alarm)

IoT ..... Internet of Things

ERSP ..... Early Research Scholars Program

SUMMARY

Allowing individuals with disabilities to the upper limbs to have access to smartphones is an ongoing challenge. One solution, developed in the Wearable Technology and Sensory Enhancement laboratory, comprises of Bluetooth-enabled assistive devices to be placed discreetly inside the oral cavity. For this work, two devices have been adopted: one button-based and one trackpad-based. The devices are positioned on the palate, encased in a discreet dental retainer, and controlled by the tongue. This work presents a custom-made application to be paired with the two tongue-controlled assistive devices.

The application aims at allowing simple interaction by the user, who is enabled to perform a plethora of different functionalities, identified through a survey distributed among disabled individuals. The most requested functionalities are access to 911 calls, interaction with keyboards and communication means, and control of a wheelchair, which have been implemented.

To develop the needed interfaces, the framework React Native was adopted, due to its cross-platform compatibility and the high flexibility it allows. The application was then tested on an Android phone, to assess its performance. Keystroke Level Model analysis of the main functionalities was executed, demonstrating the theoretical usability of the interfaces. Interaction by an expert user further tested the functioning of the application, comparing it to the default environment of the phone. This confirmed the hypothesized simplifications estimated by applying Fitts Law's principle to the design of custom-made components, such as a keyboard.

A tongue training environment was included in the application's development, aiming to increase the strength and precision of the tongue and ease interaction with the assistive technology placed on the palate. The environment is made up of six games, each focusing on a specific movement required by the user.

The interfaces, jointly with the assistive devices, could represent a solution to close the technological gap that involves people with injuries that paralyze the upper limbs.

CONTRIBUTION OF AUTHORS

Chapter 1 is an introduction to the context of the research and its specific aims, as well as to the state of the art of the specific field. It contains the same information as the introduction of a paper published in the *17th International Conference on Computers Helping People with Special Needs [56]*, of which I am the first author. Co-authors are my advisor, Dr. Hananeh Esmailbeigi, who assisted me throughout the whole research project, with essential insight and guidance, and Nicholas Marjanovic, who helped with the research process.

Chapter 2 contains references to a paper published in the *17th International Conference on Computers Helping People with Special Needs [56]*. It is a description of the materials and methods adopted throughout the presented research. I am the first author of the paper and am the principal researcher. I have developed the applications and conducted the described testing protocols, as well as analyzed the collected data. The co-authors of the paper are my advisor, Dr. Hananeh Esmailbeigi, who assisted me throughout the entire project with essential insight and guidance, and Nicholas Marjanovic, who helped the research process.

Chapter 3 contains references to a paper published in the *17th International Conference on Computers Helping People with Special Needs [56]*. It includes the results of the research, some of which are mentioned in the referenced paper. I am the first author of the paper and am the principal researcher. I have developed the applications and conducted the described testing protocols, as well as analyzed the collected data. The co-authors of the paper are my advisor, Dr. Hananeh Esmailbeigi, who assisted me throughout the entire project with essential insight and guidance, and Nicholas Marjanovic, who helped the research process.

Chapter 4 contains references to a paper published in the *17th International Conference on Computers Helping People with Special Needs [56]*. It includes the conclusions of the work and the possible future developments, which are also mentioned in the published paper. I am the first author of the paper and am the principal researcher. I have developed the applications and conducted the described testing protocols, as well as analyzed the collected data. The co-authors of the paper are my advisor, Dr. Hananeh Esmailbeigi, who assisted me throughout the entire project with essential insight and guidance, and Nicholas Marjanovic, who helped the research process.

INTRODUCTION [56]

1.1 Upper Limb Impairment

According to the "World report on disability", presented by the World Health Organization (WHO) and World Bank, 13.3% of the global population experiences some form of disability, either mental or physical [1]. For this thesis work, we will concentrate on totally or partially paralyzing disabilities. In the USA, 1.9% of the population lives with paralysis [2][3], due to stroke in 34% of cases, spinal cord injuries in 27%, multiple sclerosis for 19% and to cerebral palsy in 8%. All other causes amount to 12%.

Stroke can be defined as a neurological deficit due to focalized damage to the Central Nervous System (CNS). This is often linked to infarction or hemorrhaging in the interested area [5]. When this happens to impact on the motor cortex, specifically the areas of it which are intended for the control of the upper limbs, either one or both hands and arms are not controlled efficiently anymore. These instances can be approached both on a compensatory level, with assistive devices which intend to substitute the lost function, but also on a rehabilitation level. A complete gain of the lost function is difficult, but different approaches have shown results in the post-stroke rehabilitation, both with and without the aid of devices for that purpose. This may mean either a complete loss of function or a high diminishment in the finetuning of movements. Both these conditions are a strong obstacle in the interaction with essential technologies such as phones, computers and tablets.

When the paralysis is due to an injury in the spinal cord, rehabilitation is often not possible. A spinal cord injury causes the interruption of communication in the nervous pathways. The symptoms are usually comprised of loss of function, control and sensation in the body parts located inferiorly to the injury level. The injury can be either considered complete, with total loss of sensation and muscle function, or incomplete, with some residual nervous signals. As can be seen in Figure 1, in 47,6% of cases the cases of spinal cord injury in the USA the outcome is incomplete

tetraplegia, namely the incomplete loss of function to all four limbs. 19,9% of cases result in incomplete paraplegia, effecting only the lower limbs. Complete loss of function is experienced by 31,9% of patients, respectively 19,6% for paraplegia, and 12,3% for tetraplegia. Only less than 1% of individuals who have been, at some point, affected by spinal cord injury are able to completely recover and revert to their previous condition. [6][7]

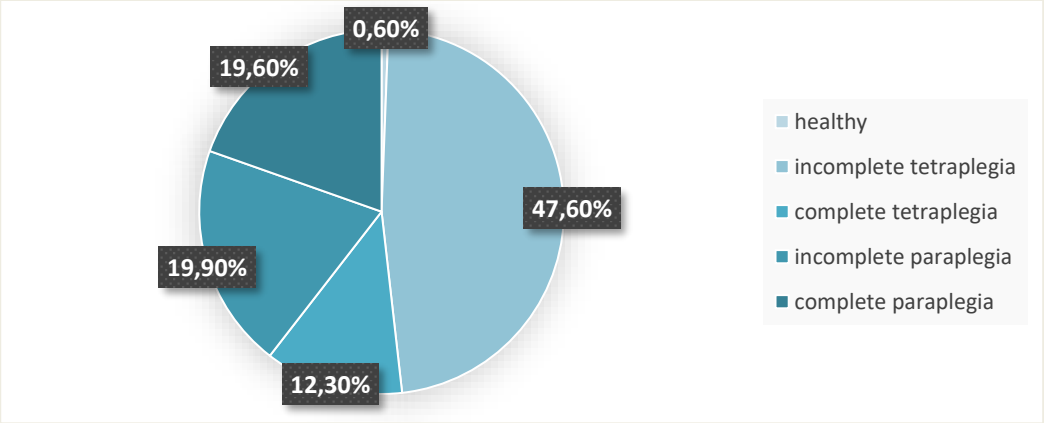

Figure 1:Effects of spinal cord injuries as found in “Spinal Cord Injury Facts and Figures at a Glance Re-Hospitalization” by Level et al.

These conditions, in the cases which affect the upper limbs, are key in the exclusion of individuals from having full access to the new and always evolving technological advancements. Not having complete control of ones arms, hands and fingers, is an increasing cause of isolation for disabled individuals, who are forced to have assistance in interacting with modern day technology, such as tablets, computers and smartphones, either in the form of personal assistance (thus incurring in a complete lack of privacy) or assistive devices.

1.2 Available Assistive Devices

The range of available assistive devices, used for both mobility and interaction with technological interfaces, is wide and strongly differentiated.

Devices based on bio-signal acquisition include all mechanisms of interface or mobility control which are related to the reading and analysis of bio signals. They include brain-computer interfaces (BCI), centered on electroencephalography (EEG), and electromyography (EMG) procedures. Both incorporate surface electrodes, positioned either on the scalp or on the limbs where subjects still retain varying degrees of control. The signal from the electrodes is recorded, filtered and classified by algorithms which vary in speed and precision. The application of electrodes, both wet electrodes, used in combination with gel, and dry ones is often complex and, on top of providing varying stages of discomfort for the users, entails high levels of noise due to reciprocal movement between skin and electrode or general interferences which render the signal-to-noise ratio (SNR) not always optimal for a correct classification of controls. EMG and EEG are often also used in combination.

Regarding the mechanical-based control systems, the most common are sip-and-puff and switches. The former are comprised of a tube, position in the users' mouth. By regulation of the strength and direction of the airflow (so, by "sipping" and "puffing" with different intensities), the user can interact with various systems and interfaces. Another common device family is made up of switch-based interface-controllers. These are devices made up of buttons and the closure of the switches linked to the buttons is used as input. These are often used either as learning aides or as controls for mobility devices, such as electric scooters or motorized wheelchairs. The detected input acts as either activation or deactivation of certain actions. The use of this technology requires some residual control of motion in the upper limbs, at least in the hands and fingers. [8]

In the most severe cases of quadriplegia or similarly disabling diseases, the preferred technology consists of eye-tracking based computer systems. This technology uses a recognition system for the position of the eye and its movements. These systems usually involve cameras and support structures, which are mounted in specific environments, such as beds and chairs, often in hospitals or hospices.

An alternative can be found in BLUI, an interface built based on a microphone, therefore adaptable to all kinds of computers, which recognizes the area against which the user blows. The environment is event-based; therefore, the interface reacts differently according to the different directions of the airflow [9].

A common denominator in the presented devices, though, is the lack of discreetness. Although many of these propose a viable solution for the individuals making use of them, the devices are either cumbersome or eye-catching. An alternative can be found in speech recognition software. This technology can be found in everyday systems, such as the latest types of smartphones and computers. This alternative only requires a dedicated software, which is usually compatible with most environments, and a functioning microphone. This can also easily integrate in smart-homes and Internet of Things (IoT) environments. The drawbacks can be found in the absence of privacy for the user, who must distinctly voice all interactions they wish to achieve with the device, and in the difficulty the software manifests with the recognition of specific accents and inflections. An effective alternative can be found in silent-speech recognition devices, which rely on the EMG recordings from electrodes positioned on the lower half of an individual's face. These may compensate for the absence of privacy, but are extremely cumbersome and invasive, being placed directly on the subject's face.[10]

All assistive devices which attract further attention to individuals, who already have characteristics that may differentiate them from their peers, are contributing to emphasize the stigma which is linked to disability. This is often linked to a high abandonment rate for assistive devices, with users preferring compensating mechanisms over the unwanted attention such devices entail[11]

1.3 Intraoral Devices

In most of the previously described conditions which lead to disability, the subjects retain a nearly perfect control of the tongue muscle. This is due to it bypassing the spinal cord and having a direct

link to the brain stem. It is, in fact, controlled by the brain stem through the lower cranial nerves. Specifically, the Hypoglossal nerve controls all extrinsic and intrinsic muscles of the tongue, responsible for all movements. [12] The only exception is the palatoglossus muscle, responsible for the initiation of swallowing, which is controlled by the Vagus nerve.[13]

These characteristics make the tongue the ideal method of interaction for those individuals who have lost use of their upper and lower limbs. Recent research has, therefore, focused on the development of intraoral tools to be controlled by contacts of the tongue.

The state of the art for these devices sees many prototypes, most of which still result cumbersome for the user, or highly invasive. They either need an activation unit, which might be either stuck [14] or pierced in the tongue, or external receptive elements [15][16][17].

Always relying on the tongue as an interface, but similar to the 'sip and puff' for what regards the physical aspect of the device, which develops outside of the user's mouth, we can find the technology developed at Carnegie Mellon University, comprised of a physical component, much like a flower, with petals the user can interact with to achieve different tasks. [18]

The WTSE Lab has been working on the development of retainer-like devices to be positioned on the upper palate of an individual, containing electronic components and enabling the user to interact with different setups through this device. The devices are independent from any accessory element and don't require neither an activation unit nor external elements. This, combined with the wireless BLE communication, allows the device to be minimally invasive.

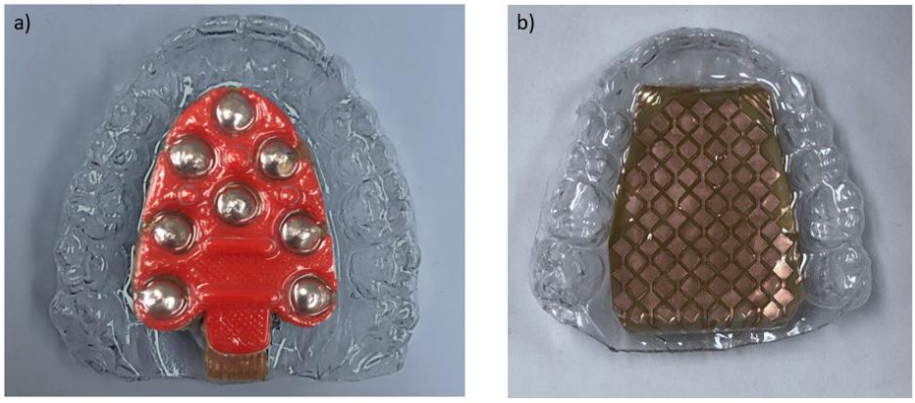

Figure 2: a) O-UIC device with 8 capacitive pads; b) Tongue Trackpad device with capacitive rows and columns

The first device shown in Figure 2 is the third generation of oral assistive technologies developed in the WTSE Lab. It was built upon the UIC-C [19] and the TTS [20] and is called Oral User Interface Controller (O-UIC) [21]. It is based on eight capacitive sensors, positioned in reachable sections of the palate. Each capacitive pad is able to detect the touch of the tongue. The information – which pad is being touched – is then sent via Bluetooth Low Energy (BLE) and is read by the developed interface.

Further advancements have been achieved in the WTSE Laboratory, producing the fourth generation of the device, a Tongue Trackpad positioned on the palate of the user, which acts as a HID mouse.

This work presents the two iterations of a mobile application intended as accessible user interfaces, each iteration adapted to one of the two devices presented in Figure 2.

1.4 Enabling Technology and Quality of Life

One may easily think that, with the progress technology makes on an everyday basis, disabled individuals are bound to be more and more integrated in the society. This has been proven to not always be the case. Disabled people face participation barriers at every step, as devices and software can be badly tailored to their specific needs and difficult or inaccessible. This results on a great gap between the online presence of abled and disabled people, with the latter being 20% less likely to be on the internet. [22]

There has always been a great gap between the plethora of available mainstream technology and the specific and dedicated assistive devices. These have, usually, been extremely expensive. The smaller the intended user group, the more specialized the device, the higher the cost. Furthermore, the offered technology has shown to be limited in versatility and in the number of functionalities each device offered.

A tailoring of the software, hardware and integration to the specific needs of a user class, with a strong consideration of the adaptability of such solution, is the necessary step towards the integration of disabled individuals in society.[23]

The ideal situation would be the one where mainstream technology, ideally already owned by the intended user, is made compatible with the proposed software and hardware. This aims at bridging the gap between the user and the device. [22] This is the principle that was followed in the development of the presented work. The proposed interface is an application, deployed on the Android Operational System but applicable to iOS as well, which enables the user to interact, through the assistive devices presented, with the basic but essential functionalities of a smartphone.

1.5 State of the Art: interfaces for Accessible Interaction

The variety of interfaces, either for computer-based systems or for smartphones, controlled by assistive devices, is limited. The devices presented in paragraphs 1.2 and 1.3 typically do not provide a custom user interface and are, therefore, used to interact with systems that have not been thought for those interactions. For example, smartphone interfaces are thought for touch-based interactions. This results in a difficulty for the users, which contributes to the high rate of abandonment of assistive devices.

An example of a technology which combines the hardware with an interface is the custom built ACAT [Assistive Context Aware Toolkit] [24]. This was designed specifically with a single, notorious, user in mind: Professor Stephen Hawking, who has been affected by ALS (Amyotrophic Lateral Sclerosis) for most of his life. The interaction is based on the recognition of movements of facial muscles. The different movements are connected to specific commands with which the interface is controlled. Professor Hawking is therefore able to interact with a computer in order to select letters, words and phrases based on his personal patterns, learned by the software, which has been trained on his lectures and speeches. These are then read aloud and this permits him to communicate, an ability lost due to his disease. [25]

An application which, though, allows any user to reproduce normal interactions with a computer or phone, which is controlled by a non-invasive and discreet oral device is not yet available.

There are cases in which the assistive devices presented allow the user to control the entire Operating System. The Tongue Trackpad, for example, offers this possibility. This, while being without doubt useful, especially when thinking of an expert user who has developed great dexterity with the technology, is often uncomfortable for the user. The mobile interfaces in particular have been optimized for touch-based interaction, not for a cursor controlled from an external device. While still functional, the combination of cursor and mobile interface may result strenuous for the

user, especially due to the presence of small buttons and icons. A custom-made application which allows for the most important functions while strongly focusing on ease of use is one possible solution to enable more individuals to access the new technologies. The interface is intended to be modular, allowing for easy future development and personalization, based on each subject's needs. The implemented functionalities have been chosen based on literature and data collected from a survey. They intend to cover the most essential and immediate needs and can be accompanied by leisure-related functionalities.

1.6 Tongue Training

It has been demonstrated that the tongue's muscle can undergo training, like all other muscles of the human body. [26] By continuously executing the same movements, tasks or exercises, the tongue increases both in its strength and its precision. Practicing the common movements necessary to interact effectively with an assistive device, such as the Tongue Trackpad or the O-UIC device, can result in an increase in precision, dexterity and ease of use.

Literature review has revealed common practice exercises, based especially on therapies aimed at thwarting the effects of dysphagia, though a general lack of interactive environments for such exercises, which combine visual feedback and entertaining or motivational setups, has emerged.

Dysphagia is a common symptom, related to either advancing tongue sarcopenia or neuromuscular consequences of brain injuries. It entails a loss of function in the swallowing motion. The movements that require exercise in order to counter the effects of dysphagia are principally four: *elevation, lateralization, protrusion* and *swallowing* [27]; these exercises increase lingual strength and therefore counter the symptom. The elevation of the tongue refers to the contact of either the anterior or posterior area of the tongue against the palate. Lateralization is the movement to the sides, either from the center towards one of the two cheeks or from one to the other. Protrusion is the act of pushing the tongue outside the oral cavity, surpassing the teeth and extending as far as

possible. Lastly, the act of swallowing, either forced -meaning with nothing to swallow- or natural -therefore including a bolus to be swallowed-, is the act itself, which can be practiced.

The rehabilitation through these four movements can be divided in two different approaches, direct or indirect. The direct approach regards the swallowing movement and includes either a liquid or solid edible prop, that the patient is expected to swallow and ingest. The indirect approach, which is of greater relevance for this work, is made up of different combinations of the first three actions described: lateralization, elevation and protrusion.

To achieve the goal, enhancing the performance of potential users of the Tongue Trackpad, and the related application, the specific movements which constitute tongue-palate interaction must be practiced. The main actions include clicking, therefore an exercise compatible with the aforementioned elevation action, and pointing, a movement from one point to another either laterally or front-to-back and vice versa. The diagonal movement is also one of the milestones to accomplish with the intention of reaching better ease in the control of the application.

The second version of the interface, related specifically to the Tongue Trackpad, has been equipped with a training area, where specific interactive games allow the user to address their aim of enhancement of specific tongue abilities to improve the experience of interaction with the application and an operative system, being it computer or mobile based.

MATERIALS & METHODS [56]

2.1 Hardware

Two versions of the interface were developed, each one optimized to interact with one of the two devices presented. The functioning of both devices is, in a way, similar, but they completely diverge when it comes to the communication with the mobile phone and the application. The touch of the tongue is, in both cases, detected by capacitive sensors which trigger a voltage change when touched. This is due to a parallel capacitance that develops when the tongue contacts the pad or the line-column intersection. The thresholds have been optimized for interaction with the tongue in the impervious environment which is the mouth, characterized by high humidity and the presence of saliva.

2.1.1 O-UIC Device

The O-UIC device is based on an Espruino MDBT42Q board with an NRF52832BLE chip, which incorporates both the capacitive sensing and the BLE communication necessary for this device to function as needed.[28] [29]

The communication between the O-UIC device and the developed application is based on a Bluetooth Low Energy (BLE) connection, directly from the device to the application. This implies that the distinction between commands happens at a software level. Only the detection of the touch is handled by the firmware uploaded on the device. The code used to detect the touches was developed during the implementation of the O-UIC. [21] This code was originally divided in two modes, one where the output of the device is transmitted to the receiving computer as Human Interface Device (HID) keyboard commands, and a debugging mode which writes the information on the console, by serial communication. The debugging mode of the code was used as a starting point which lead, after little change, to the functional communication with the application.

The communication frequency was changed from the original 10Hz to 2.5 Hz, one string of information sent every 400 milliseconds. The reason this number was chosen was based on the medium reaction time of the human brain, which is 200ms. Due to the complexity of mapping the interface's layout to the location of the pads in the mouth, the time period was doubled. This was also confirmed to be the best frequency by trial and error. The string sent is composed by eight characters, each identifying a different pad, and usually set to "_". Once a specific pad gets touched, not only does the corresponding character change, but it also assumes the number related to the specific pad, making decoding on the app side easier. The passed string, for example after the touching of pad 4, would be "_ _ _ _ 4 _ _ _". The following code represents the segment of the firmware uploaded on the Espruino board to control the serial print of the touches.

```
// Prints Touch Status for each pad

    for (i = 0; i < 8; i++) {
      if (touch[i] > -1) {
        base += touch[i] + " ";
      }
      else {
        base += "_ ";
      }
    }
```

Figure 3: Code snippet of the communication between the O-UIC device and the mobile phone's application. The resulting string of characters is stored in a characteristic of the BLE communication

The decoding of this information is then handled in the code of the interface, which is therefore crucial for the device to interact with smartphones. The decoding process will be explained in detail in paragraph 2.5.1.

2.1.2 Tongue Trackpad

The Tongue Trackpad is the latest version of intra-oral assistive devices developed by WTSE Lab. It is built upon capacitive sensing, the same principle as the O-UIC device, but is based on a matrix of rows and columns of capacitive elements that substitute the previous 8 pads and are able to detect the tongue's touch. The peculiarity of this latest version is the adaptation of the device to function as a Human Interface Device (HID) mouse. This is possible by detecting the incremental movement of the tongue in the x and y direction, so the changes in rows and columns touched. This information is sent via BLE characteristics in two packets, one per direction to whatever device the Tongue Trackpad is connected to. It is then translated into a command to move the cursor in said direction. A third packet of information signals the intention of the user to click in the current position of the cursor.

The main difference between the interaction's methods of the two devices, therefore, relies mainly on where the information is processed and transformed into commands for the interface. The O-UIC device communicates via BLE *directly* with the application, which is therefore responsible for the computational load of the communication. For the tongue trackpad, on the other hand, the translation of the packets' information in commands is handled by the phone (or computer's) Operative System (OS), therefore allowing greater freedom of interaction to the user and creativity to the developer.

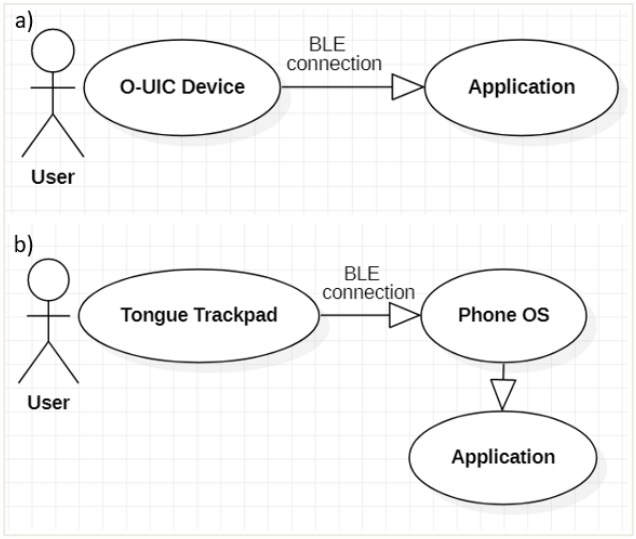

Figure 4: Communication between (a) O-UIC device and (b) Tongue Trackpad and application. The first entails a direct handling of the information on the application's end, the latter has the OS of the phone bridging between device and application. The device's output is seen as a usual control for HID mouse cursor.

2.2 Preliminary data collection

In order to develop an interface which responds to the needs of the target population, made up of individuals suffering from disabilities which impact the upper limbs, a survey was developed through Qualtrics and distributed via email and flyers, upon IRB approval [(Institutional Review Board) protocol number 2017-0550]. [30]

The criteria followed in the development of the interfaces have been the same. In order to produce an application which resulted user friendly and usable, the gold standard process of *iterative design* was followed, adapted to the circumstance of the specificity of the intended user population. The information was first collected by analyzing the target population theoretically, through literature research and analysis. A second iteration and optimization were done following the retrieval of the survey data.

The questions asked in the survey aim to gather a well-rounded knowledge on the necessities of the considered population, looking at all-encompassing details. The survey is divided in sections,

each approaching a different aspect of the inquiry. Overall, it is comprised of 72 questions, nested into one another and therefore not always appearing in toto to each user.

The first section focuses on general data, such as age, type of ailment and living situation, as well as an open-ended question regarding the specifics of everyday challenges faced by the user. This portion was aimed at obtaining a general overview of the intended population.

The central section was developed around inquiries concerning assistive devices previously or currently owned by the users. The aim was to have an idea about the habits and knowledge of the population.

Questions regarding the usual approaches to new assistive devices were asked in the fourth section, in order to analyze the difficulties encountered when approaching new technologies.

To conclude, eight questions were asked, focusing specifically on technology and technological needs of the users. The aim was to obtain information as to what was present in the users' life in terms of technology and what specific functionalities were lacking in the available interfaces for assistive devices. This ensures the correct prior knowledge necessary to approach the development of the interface. The questions asked were both multiple choices and open ended.

2.3 Interface Development

With the purpose of achieving an interface which encompasses all necessary aspects of the intended users' requests, the presented applications were developed. Both are structured in a semi-modular manner, which allows for easy modifications and tailoring to the user's needs, on a code basis. This permits the developer to change the displayed screens, and the sequence of the navigation, so that it presents functions which are effective for the specific user. For the purpose of this work, all developed functionalities have been left accessible. The two iterations of the

application, as mentioned in the presentation of the used hardware, have been intended to be optimized for interaction with the two devices.

The functionalities developed, which will be presented in detail in paragraph 3.2, have been inspired by the answers to the survey described previously. The focus was put on allowing, primarily, communication with others. For this reason, the developed functionalities were focused on an automated SOS call to 911, interaction with the mobile's embedded phone-call system to allow all numbers to be dialed and the compilation of an email, which results in the triggering of the related email app with pre-filled fields.

The development of new functionalities was then limited to interaction with the trackpad. The difference in the nature of the interaction, explained in the previous section, resulted in an ease of development for the functionalities of this second iteration, and a lessening of the computational burden on the application. This allowed for more functions to be developed, starting with a simulation of wheelchair control, mimicked through the communication with an Elegoo car (details can be found in paragraph 2.5.2). The remaining functionalities listed together with the used libraries in the consumptive Table 5, have been added to ensure the best user experience possible.

2.3.1 React Native

The interface application was developed in React Native, a framework developed by Facebook in 2015 specifically for cross platform optimization. It allows to develop applications in JavaScript, the de-facto standard for front-end development, declaring the behavior of the application and then passing it on to the native environment to ensure the best user experience. [31]

The code can be written once and then optimized, regarding the authorizations necessary for the app to function (for example Bluetooth and position necessary for BLE communication) and

library installation, for both Android and iOS. For the purpose of this work, the application was finalized only for Android, but can be easily optimized for iOS in future developments.

Together with the simplicity of the coding language and the adaptability to both most common operative systems, React Native was chosen due to the extensive number of libraries which are continuously built and optimized by the developer's community. This, though, entails a drawback: most of the libraries and functionalities are still in beta version, therefore not yet optimized for use and application development, and this has proven to be a hindrance in some parts of the coding process.

2.3.2 Custom-made components

React Native allows the developer to create custom components, such as buttons and other elements of the screen's functioning. Using this possibility, it is possible to include special features to the screens. This was used to implement components which are especially versatile and adaptable, to answer to those requirements that developing an interface for assistive devices has. Having self-made components also allows for a more personalized (and modifiable) look and feel of the whole interface. The buttons and icons are both clickable (so controllable with the trackpad or the finger) and act as placeholders to indicate to the user which pads relate to which actions.

With the aid of an open source online resource [32], a personalized palette was created. The colors are in tune with each other and the choices made aim at creating a welcoming and peaceful feeling in the user. The choice of the primary color, starting point for the creation of the palette, is a light blue (Figure 5). This color was demonstrated being between the most chosen internationally for the development of country-specific websites, according to a 2006 study which analyzed 15 countries and the predominance of color in their websites. The most present tones were black and white, but blue and all its shades was the first color present in all the countries analyzed, thus confirming the tendency to choose this color in interface design [33].

Figure 5:Specific color palette used during the application's development

The custom-made components called `My_Button` and `Icon`, used in all screens as buttons and touch-sensitive icons, follow the same idea. Taking advantage of a wrapper, `TouchableOpacity` [34], which is a parent view, that wraps around whatever one positions inside it, it is possible to interact with contents. It is able to recognize touch events, in particular the `onPress()` handler, which calls a predetermined function, or set of functions, to be executed when the user interacts with the wrapped elements. This was selected to be used to create clickable icons and custom made buttons because it gives greater freedom and allows the customization of the area one can interact with, making it possible to increment the surface and therefore facilitate the user's interaction with the application.

Using the `My_Button` component, the keyboard used in the messaging and email functionalities and the number buttons in the phone screen were developed. They were built aiming at optimizing the distribution of the key-buttons to facilitate cursor interaction. The principle followed was the optimization of the ratio between distance and the area of the button itself, based on Fitts Law [35]. According to Fitts' studies, the time required for a person to point to a certain target is appraisable with a formula that takes into account the distance from the beginning and the width of the objective. The original law presented by Fitts derives from a parallel between information transfer and human movement, which is said to be modellable with Shannon's 17th theorem (2.1).

$$C = B \log_2 \frac{S+N}{N} \tag{2.1}$$

The channel capacity (C) is compared to the index of performance, obtained by dividing the Index of Difficulty (ID) of a proposed task (2.2) by the time needed to execute it. B is the bandwidth of the considered target and is therefore constant. The ID was identified by Fitts as (2.2), where A is the distance from the target and W is the width of the target itself.

$$ID = \log_2 \left(\frac{2A}{W}\right) \tag{2.2}$$

The original formulation of Fitts Law is a regression of the Movement Time (MT) on the ID obtaining (2.3), with *a* and *b* being empirically determined constants of the regression.

$$MT = a + b \log_2 \left(\frac{2A}{W}\right) \tag{2.3}$$

This has been demonstrated to apply to Human Computer Interaction and can therefore be used to estimate the improvements in usability introduced by custom-made components such as the keyboard [36]. The introduction of a constant *c* is a modification to Fitts Law proposed by Weford and confirmed by Fitts himself. It is a constant equal to either 0,5 or 1, added in order for the difficulty index to never be negative, even for overlapping targets

$$MT = a + b \log_2 \left(\frac{A}{W} + c\right) \text{ [37]} \tag{2.4}$$

The final version of the Difficulty Index is given by the second part of (2.4):

$$\log_2 \left(\frac{A}{W} + c\right) \tag{2.5}$$

Considering that the width and height of the screen are constant, therefore the maximum distance between targets (A) can also be considered constant, we can compare exclusively the effect of the width of the button in the calculation of the ID, which can be computed, for this work, as 1/W.

The android development guide was used as a source as to how big and distanced each pad is in the default screens. The distance between small elements, such as icons or small keys, is set to 4

dpi (device-independent-pixels), each equivalent to one physical pixel on a 160dpi screen. Applying this logic to the *Xiaomi mi mix 2* phone [dpi: 403 [38]], used to execute the tasks presented in paragraph 2.7, the dimensions in pixels of the inter-button distance is equivalent to 10.1 pixels, obtained as follows:

$$1:160 = x:403$$

$$x = 2.51\,\frac{px}{dpi}$$

$$4dpi * 2.51\,\frac{px}{dpi} = 10.1px \tag{2.6}$$

Multiplying this value by twice the number of buttons present in the keyboard, considering a margin present on both sides, the number of pixels taken up by margin is equivalent to 201.5 pixels. To identify the maximum width of the native key-pads, the screen definition – subtracted of the margin pixels – was divided by the number of keypads present in the most dense row (the first, with 10 pads), thus obtaining 87.85 pixels, equivalent to 35.14 dpi. The dimensions of the buttons of the custom-made keyboard have been set to 60dpi. The horizontal margin of each button is 10dpi, so equivalent to 25.2 pixels.

The buttons present in the default phone screen are wider than the keypads, thus an intervention to increase the width was not strictly necessary. What was necessary was a better definition of the edges of the button, to make the target more easily identifiable. The distance between the buttons was also optimized, in the custom application, to reduce unvoluntary clicks.

2.4 Usability and inclusivity of the interface

To ensure the best user experience, the gold standard of usability principles was applied, to the extent that was possible considering the peculiar use case of the presented application. These principles, presented by Jakob Nielsen and Rolf Molich in 1990 [39], encompass all good practices that should be adopted in the development of user interfaces. All aspects of this presented work were ideated and brought forth keeping in mind the *Heuristics* presented in Nielsen's studies. The nine principles presented in 1990 were enhanced and dogmatized in 10 postulates that define interface development [40].

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

Considering the nature of the developed interfaces, both aiming at a user population comprised of individuals with more or less severe ailments, another important aspect that was considered was the inclusivity of the final product. To ensure the highest level possible, the principles presented by Hammad et al. were analyzed and adapted to the particular case [41]. These rely on six cardinal elements (derived from Nielsen's heuristics) that must be respected to achieve an inclusive interface, which were followed in this work:

1. Readability
2. Affordance
3. Error tolerance
4. Organization and color coordination
5. Natural flow of information
6. Vigilance to users' abilities

2.5 Bluetooth communication

Both iterations of the interface require Bluetooth Low Energy connection. This relies on the communication through services and characteristics that are at the base of BLE communication protocol. A service is a specific packet of information, which comprises multiple characteristics. All services and characteristics are each identified by a UUID (Universal Unique Identifier). Each characteristic is described by descriptors, that identify if a characteristic is readable, writeable and notifiable. A writeable characteristic allows the connected phones and computers to send information to the device. A readable characteristic allows to read what the device is communicating.

In the specific case of the O-UIC, the characteristic where the string of touches was stored and transferred was not readable, but notifiable, which identifies a characteristic that continuously sends data to the receiving device.

The O-UIC device, as illustrated in Figure 4, relies completely on Bluetooth communication to control the interface. This connection is handled on the application side, and the code for it was written during the development. For what regards the second iteration and the connection to the Tongue Trackpad, this is handled by the Operative System of the phone. The BLE connection, in this case, is needed to control a wheelchair-like Arduino Elegoo car.

The React Native library which was used to handle all BLE communications is `react-native-ble-plx`, which was found to be the most effective for the required use. It allows to both read and write on the Bluetooth characteristics of the devices connected to the application. [42]

For both uses the initial process of BLE connection and identification of the correct characteristic to interact with was handled the same way. The whole connection process and handling of the communication was done in the `componentDidMount()` function, which identifies the instant in which the screen first loads. For the O-UIC device's interface this screen is the first one the user

loads, called `SigninScreen`; for the trackpad interface this happens in the `commandScreen`, where the handling of the wheelchair controls happens.

The first step required to create a so called `BleManager`, a component which would be the bridge between the outside device and the application itself. The library provides a function, nested in the manager itself, which allows to scan and identify all available devices. From these, only the device with the correct name was extracted. For both applications the device's name and the specific services and characteristics used were saved in constants initialized at the beginning of the code. The id (equivalent to the MAC address of the chosen device) is extracted and used to identify all services and characteristics. At this point the two processes divide and differ.

2.5.1 O-UIC device BLE control

The value passed from the device's characteristic to the application first needs to be decoded by the first screen of the interface and saved in a variable. The string of underscores and numbers is read, and the first touched pad is isolated and saved. The distribution of the numbers on the pads can be seen in Figure 6.

Figure 6: O-UIC device with eight capacitive pads numbered to illustrate the location of the input given by the user

This variable will then be memorized in a `reducer`, which is a native component of React Native that stores the states of the variable, the actions that may be triggered by specific dispatches and the resulting changes in the variable's state.[43]

For the aims of this application a global reducer was created, which receives the data from the screen that handles the BLE communication and then elaborates this depending on the screen that is currently loaded. This is achieved by having each screen pass information to the reducer (called *"*`deviceValue`*"*). This information controls which kind of decoding of the command must be implemented and therefore what kind of information must be passed back to the screen to control it.

In particular, the different decoding methods regard navigation, typing and numbers.

Whenever a pad is touched, it dispatches a '`TOUCHED`' action that has the number of the pad as payload. This then is handled differently if we find ourselves in one of three states: the keyboard state – where the touches are seen as T9 commands- the number state -with a T9 based compilation of phone numbers -, and the navigation state -where each touched electrode causes the navigation to a different screen. For the first two, the typed number or sentence is passed over to the screen we find ourselves in and displayed or used there, in the navigation state the number of the pad that has been touched is passed over once again to the screen and used to direct the navigation. The selection of which state is to be executed is handled by the information passed from the screens to the reducer. Every time a screen is loaded it communicates to the reducer the enabling or disabling of keyboard, numbers and navigation mode. Once that is saved in the reducer, touching a pad triggers a different action depending on the state the screen is in.

The handling of typing for both numbers and letters is similar. It is based on the re-invention of a T9 style keyboard, which had been proven usable in the development of the device [21]. An array

of arrays containing four letters/numbers (or commands like `'xx'` to delete or `'BACK'` to trigger a change of screen) was created. Every time a pad is touched, the first index of the array assumes that number, the second index goes to zero: this way the first element of the sub array is selected. If the same pad is touched in less than one second, time compatible with the sampling frequency chosen for this use, the second index is incremented, selecting the next element of the sub array. The state element `word` or `number` is then updated, containing the typed sentence or number. This is then passed on to the screen, and for each screen the details of the decoding process are explained in the respective paragraph.

The schematic of the interconnection between the screens and the state is illustrated in Figure 7.

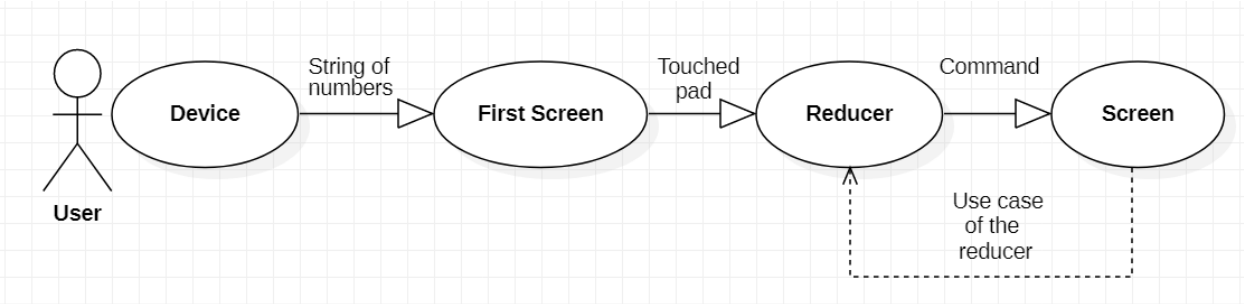

Figure 7: Information flow between the user and the screens. The reducer acts as a control center, receiving the instructions from the screen, executing the requested case and forwarding the resulting command

2.5.2 BLE control for mock-wheelchair

In order to simulate the control of a BLE wheelchair, a connection with an Elegoo Arduino car was established. This connection consists of three different steps, as can be seen in Figure 8.

Figure 8: Communication flow from the application to the Elegoo's Arduino Uno board. The BLE communication is between the App itself and an Espruino Puck.js, which is connected via serial communication to the Arduino board.

The first part is the code uploaded on the Arduino placed on the car, which is the example code provided with the Elegoo kit. The only change that was made regards the speed of the motors, which was set to 80 to allow for a more realistic control. The second part is the code uploaded on the Puck.js board, constituted of the same module as the O-UIC device and the same chip [29][28]. This has two sections, one dedicated to creating the characteristics needed for the Bluetooth communication. This characteristic has to be writeable in order for the application to communicate with it. It has a specific identification number, the UUID, which is searched and recognized by the application's BleManager. The second part of the Espruino code is dedicated to the passing of the commands to the Arduino it is connected to. It writes on the serial port and sends a character ("b", "f", "l", "r" or "s") to communicate the direction – backwards, forwards, left and right- and weather the car should stop. The Arduino decodes this character and controls the motors accordingly. On the application's side, four buttons trigger the writing of a corresponding character (with the same encoding as the Arduino-Puck.js communication) on the identified characteristic. When each button is pressed, the application writes the corresponding letter in the BLE characteristic of the Puck.js, which then decodes it and writes it on the serial communication to the Arduino. The firmware uploaded on the Espruino board can be seen in Figure 9 [44].

```
NRF.setServices({
  "6e400001-b5a3-f393-e0a9-e50e24dcca9e":
{"6e400002-b5a3-f393-e0a9-e50e24dcca9e": {
      readable: false, writable: true,
      onWrite: function(evt) {
        toggle(String.fromCharCode(evt.data[0]));
        console.log("Got ", evt.data);}}}});

function toggle(command) {

  if(command === 'f') {
    digitalPulse(LED2, 1, 500);
    Serial1.write('f');
  }
…}

Serial1.setup(9600, {rx:29, tx:28};
```

Figure 9:Example code of the firmware updated on the Puck.js to set the characteristic and service needed and control the serial communication to the Arduino

2.6 Navigation between screens

To handle the navigation, the gold standard of React Native navigation is adopted, `react-native-stack-navigation`, which is based on a stack-like method. The screens are rendered one at a time, each called from the previous one. In order to avoid an overload of memory usage, instead of piling the stack with screens, each re-rendered a new time at every call, the "`replace`" function was used; this pushes the current screen from the stack and replaces it with the called one. This allows more memory space for in-screen computation, necessary to handle both the Bluetooth communication and the keyboards created in the second iteration. Each screen was developed aiming at allowing one of the necessary functionalities

2.6.1 SOS call

In order to allow users to call 911 with ease, with no need to type out the number and risk errors and delays, one of the developed functionalities is an immediate call to 911 triggered by the press of an icon on the screen by the cursor or the press of pad 3 (positioned at the top of the device, as shown in Figure 6). This is obtained with a simple line of code that calls the `react-native-immediate-phone-call` library's function, that triggers a call to whatever valid number has been passed on to the function, as shown below. The variable `click` is the value passed on by `deviceValue`.

```
if (click === 3) {RNImmediatePhoneCall.immediatePhoneCall('911')}
```

2.6.2 Generic phone-call

Using the same library that was installed for the emergency call, a functionality to allow users to call whatever number they'd want was developed. This differs in the interface optimized for the O-UIC device and the one for the Tongue Trackpad.

In the first one, it has been deemed important to explicit the commands and numbers linked with each pad, therefore an image of the schematic was added to the screen. The rest of the screen is only composed by a textbox, where the typed number appears, and an icon exemplifying the possibility to call. This icon is wrapped in a `TouchableOpacity` component which, as explained in the general overview of the app, renders the enclosed elements reactive to clicks. This was added to allow anyone to trigger the call once the number has been typed, either by use of the O-UIC device or by direct interaction with the screen. The code reported in Figure 10 shows two key elements in the functioning of the phone-call. First, the calling of three functions: these are essential to dispatch to `deviceValue` the information regarding what part of the reducer's function this screen needs. The navigation is disabled, the keyboard as well, whilst the number-typing function is enabled.

Secondly, it is visible how the screen receives two different variables, saved in number and in control. In the first, the typed phone number is saved, and displayed in the textbox, as well as passed to the state that updates what gets called by the function. The latter, control, is used to pass

```
    this.props.navigationDisabled();
    this.props.keyboardDisabled();
    this.props.numberEnabled();

    const number = this.props.numero;
    const control = this.props.pippo;

    if (control === 'BACK') { console.log(new Date() + ' PHONE.BACK'); navigate('Ho
me'); }
    if (control === 'CALL') {
      console.log(new Date() + ' PHONE.CALL'); RNImmediatePhoneCall.immediatePhoneC
all(this.state.phoneNumber);
    }
    if (control != 'BACK' && control != 'CALL') {
      this.state = { phoneNumber: number }
      console.log(new Date() + ' PHONE NUMBER ' + number);
    }
```

Figure 10: Code snippet from the O-UIC PhoneScreen showing how the commands received from the reducer are decoded in the screen itself and transformed into numbers to call

the current selected item in the array. It is used to pass on commands such as call or back, to allow the user functionalities other than simply typing.

For the Tongue Trackpad interface, the screen comprises of big buttons with which the user can type the phone number, stored in an internal state which gets updated when each button is pressed. The same storage system accounts for the pressed pads in the O-UIC version.

The phone number is than, in both cases, passed on to `RNImmediatePhoneCall.immediatePhoneCall('{phone number}')` which triggers the call.

2.6.3 Email and Messaging

Like the phone call, also the compiling of an email has been developed in two versions, one specific to the O-UIC device and one for the Tongue Trackpad.

For the O-UIC device, two screens were created for this functionality, one to allow users to practice typing (called `KeyboardScreen`) and one specifically for the compiling and sending of the email. This latter functionality, though, has a limitation: due to the required password and username necessary to enter an email account, the developed interface only extends to filling in different elements of the email, but still requires the send button to be pressed in the mailing application. This is a strong limitation for the interaction with the O-UIC device, which is only able to interact inside the React Native interface. This hindrance is not present in the second iteration, as the trackpad's cursor can press send in the original mailing app. To allow a user to, anyway, fill in the body of the email using the O-UIC device a similar code to that presented in the phone call functionality was developed, allowing `deviceValue` to pass both the wholly typed sentence and the commands (to send the mail, go back to homepage or delete if a mistake occurs).

The possibility to send emails (and messages) was also implemented for the Trackpad device, with some modifications. The presence of a custom keyboard allows the user to select the letters to

compose the desired address and body of the email to send. The choice to keep the order of the keys based on the QWERTY model was adopted, though the number of buttons on each line does not allow the complete reproduction of the standard keyboard.

2.7 Functioning Evaluation

To test the functioning of the developed interfaces, an experienced user was asked to execute a set of assignments with each device, as well as complete the same tasks using the default interface of the phone with the Tongue Trackpad and using a single finger. For all tests, the recorded times refer to the first time the user was asked to perform such tasks, though the user is defined as an "expert", due to ample knowledge of the technology and practice in tongue-palate interaction.

Using the `console.log()` function, strategically placed messages were added to the code. This way, once the test was over it was possible to extract the console's log and review a trace of the whole process, along with timestamps of each action. This provides a well-structured overview of every step taken by the user and allows a further confirmation of the time needed for the different tasks.

The messages were positioned at the loading of each screen, and then linked to each action which can be executed inside of the screens. The function `componentDidMount()` is used to identify the moment in which the screen first loads; it is placed at the beginning of each screens file and contains the functions necessary for the correct functioning of the screen together with the `console.log('NAME SCREEN LOADED')` message. An example can be seen in the code fragment added below.

```
componentDidMount() {

 console.log(new Date() + 'KEYBOARD SCREEN LOADED'); //timestamp and scre
en

  }
```

Figure 11: Example of the code use to print the console logs

Along with the timestamps and information regarding the loaded screens, the `console.log()` function was used to log information regarding the actions taken in each screen. For example, for the home screen the logged information regards the pressed buttons linked to navigation. Similar approaches were applied to all screens. In the following figure, an example of the logging information for the *expert user* is available. The logging of the commands sent by the device to the application, this limited to the O-UIC device, is also present, and is useful to recognize which pad is identified by the device as "touched".

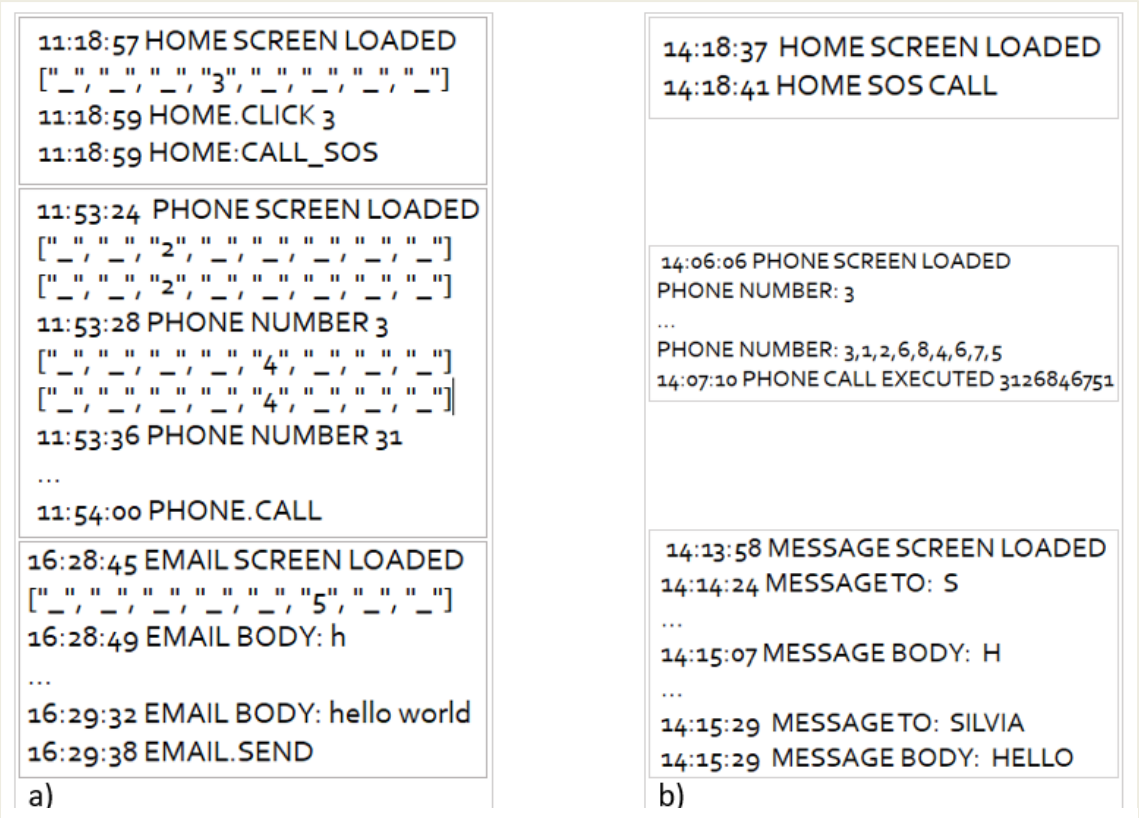

Figure 12:: Example of a console.log of the execution of 6 tasks by the expert user, interacting with the two versions of the app through the O-UIC device (a) and Tongue Trackpad (b). The timestamps are used to calculate the time needed by the user to perform the tasks

2.7.1 O-UIC Interface

The expert user was asked to perform three tasks interacting through the O-UIC device with the application installed on a Xiaomi mi mix 2 (Android 9.0.1). The tested tasks are summarized in the table below.

TABLE 1: TASKS FOR O-UIC DEVICE INTERFACE TESTING

Task 1	Open the application and trigger the SOS call
Task 2	Open the phone screen, type in a phone number and call
Task 3	Open the email screen and send an email typing '*hello world'*

2.7.2 Tongue Trackpad

The same process was followed for the Tongue Trackpad. The same tasks were executed by the same user, with the Tongue Trackpad interacting with the default environment of a Xiaomi mi mix 2 (Android 9.0.1) and interacting with the custom application. The tasks (Table 2) are the same as the ones used for the O-UIC device, with the addition of a fourth task, verifying the sending of a Short Message Service (SMS).

TABLE 2:TASKS FOR TONGUE TRACKPAD INTERFACE TESTING

Task 1	Call 911
Task 2	Open the phone system, type in a phone number and call
Task 3	Open the email screen and send an email typing '*HELLO WORLD'*
Task 4	Open the messaging system and send 'SILVIA' the text 'HELLO'

2.8 Keystroke Level Model

To assess the theoretical performance of the screens, the Keystroke Level Model (KLM) approach was adopted. This model is used to assess the usability, in particular the efficiency, of a specific user interface. It is a theoretical computation of the estimated time needed to complete a task. The task is broken down in finite elements, each relative to a specific interaction the user has with the machine.

TABLE 3: KLM TIMES FOR EACH FINITE ACTION NECESSARY TO EXECUTE A TASK

Action	Operator	Duration [s]
Key or button press	K	0.20
Pointing	P	1.10
Drawing	D	varies
Mental preparation	M	1.35
Homing from mouse to keyboard and vice versa	H	0.4
Representation of the response	R	depends on system

The theoretical calculations of the times were performed for both interfaces, for the same tasks that were proposed in the functioning evaluation. The only operands that were used were K, P and M. [45]

The sequences of actions necessary to perform each action were identified by studying the interaction expected from an expert user when completing the tasks. An ideally completed task was considered, where no errors were made. This was done to assess the usability of the interface after a transitioning period in which the user gets familiar with the intended movements of the tongue and eliminates the unnecessary errors due to inexperience. The identified sequences can be found in Table 4. Mental preparation was included at every step, to account for the need of a user to identify the correct location of the pad or the button.

TABLE 4: SEQUENCES OF ACTIONS NECESSARY TO EXECUTE EACH TASK IN EACH INTERFACE (TASK FOUR IS ONLY EXECUTABLE WITH THE TONGUE TRACKPAD)

	911 Call	Phone call	Email	SMS
O-UIC	M+P+K	[M+P+K+K]*3 + [M+K+K]*4 + [M+K]*3 + M+P+K	M+K+[M+K*2]*2+ [M+K*3]*2+M+K*4+ [M+P+K*3]*2+[M+P+K*4]*4	---
Tongue Trackpad	M+P+K	[M+P+K] *11	[M+P+K]*13	[M+P+K]*14

2.9 Tongue Training Interface

To allow the user to train their tongue in those movements presented in paragraph 1.6, six interactive games were developed, each aiming at the enhancement of one specific ability of the tongue. Six corresponding hypotheses were identified, which will be tested and validated, or disregarded, in future developments of this work.

- Hypothesis I: horizontal and vertical exercises improve tongue strength and precision [46]
- Hypothesis II: horizontal exercises improve vertical strength and precision (and vice versa) [47]
- Hypothesis III: elevation exercises increase the speed of the tongue's elevation to the palate [26]
- Hypothesis IV: elevation and maintenance of an elevated position of the tongue increase tongue strength. (based on dysphagia therapy presented by Namiki et al. [48])
- Hypothesis V: training increases precision of tongue movement towards random points on the screen [49]
- Hypothesis VI: training increases speed of tongue motion against the palate. [49]

To develop the games, which will be presented in paragraph 3.7, the same principles that were described for the development of the interfaces were followed. The layout and colors are maintained in accordance to the rest of the work, to allow for coherence and consistency of look and feel. The interfaces are kept simple, with easy instructions and straight forward layouts. The incentive needed to keep users focused and motivated during the training is achieved by the exploitation of the `setTimeout()` function, which allows to call an action that is executed after a set number of milliseconds. In particular, all interfaces were set to have a 60000ms countdown start when the screen is loaded, giving the user a minute to interact with the game and achieve as many points as possible, except for the screen developed for hypothesis four which, taking into account the 10 seconds holding period, allows the user to exploit a longer time period, up to 90s. Once the timer ends, the user is either brought back to the selection page, where they can choose the next game (if a minimum number of points is achieved, or the screen reloads to give the user the chance to perform the game again.

The data is stored locally, exploiting the React Native storage Async Storage [50], which allows to save strings of information and retrieve them in different screens. Restarting the application does not affect the storage, and the data are secured unless the user desires to delete them. The user can see the data in graphs, created with the react native library `react-native-chart-kit` [51].

The games were first tested, during the development, with one finger interacting with the application on a physical android phone, Xiaomi mi MIX2, and also with an on-screen emulator to verify the functioning with a cursor interaction. An expert user was also asked to play the games, three times each, using the Tongue Trackpad, to assess the functioning of the code and its compatibility with the device.

2.10 User Feedback Survey

Once all applications were developed and implemented, a follow-up survey was distributed to the target population via an online Qualtrics questionnaire. The survey included a description of the applications and aimed to assess the user's feedback in regards to the implemented functionalities for both the application customed to the Tongue Trackpad and to the O-UIC Device. The survey was structured in two parts, corresponding to each device which included a description of the three core functionalities and a photograph of all screens and devices presented. The survey presented the potential users with five multiple choice questions based on an agreement scale and two open ended questions. The following figures depict the introductory descriptions for both devices and applications.

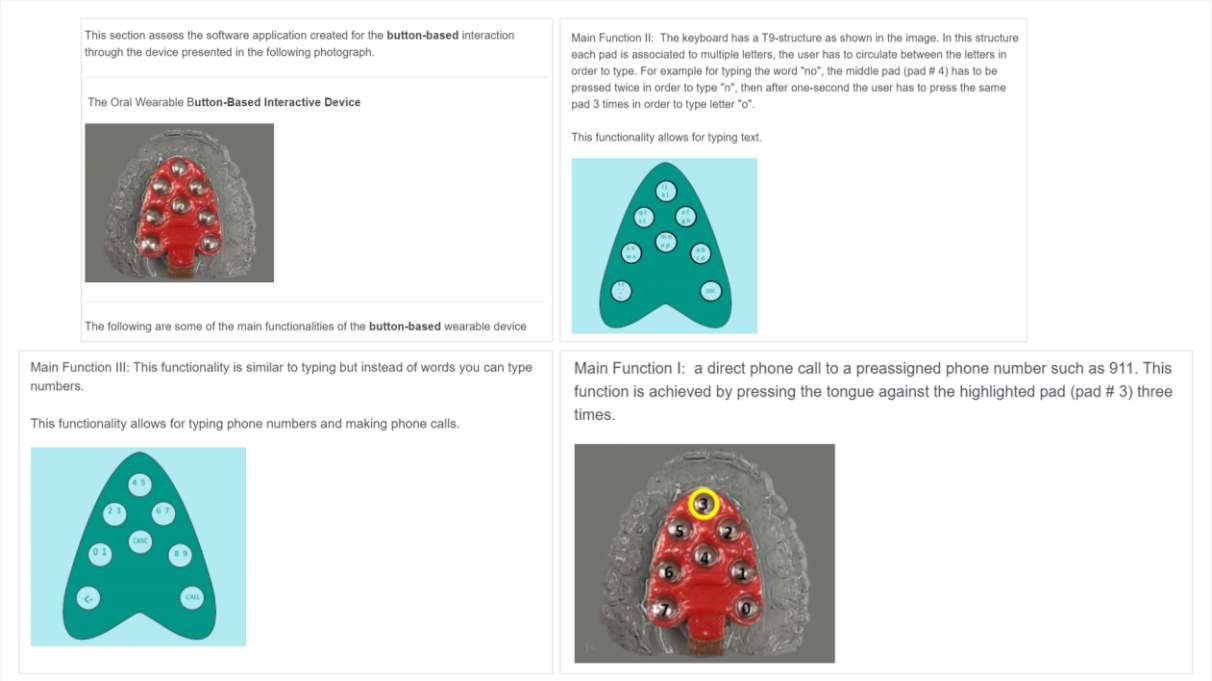

Figure 13: Screenshots of the presented functionalities for the section regarding the O-UIC Device

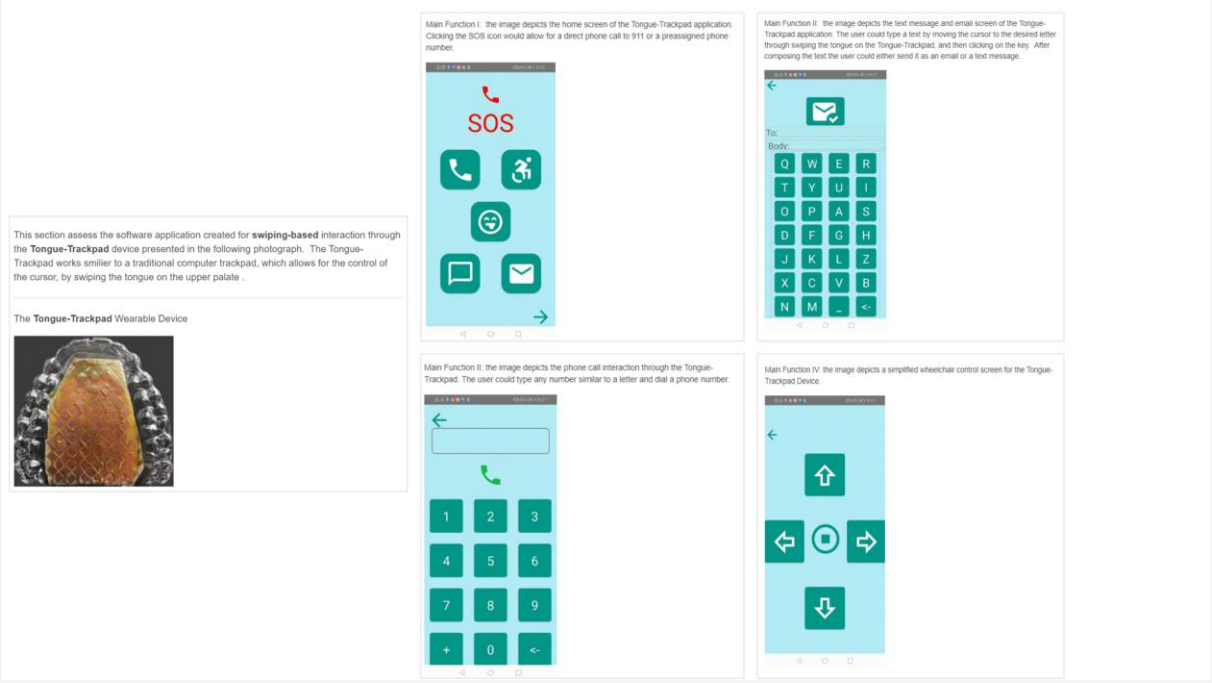

Figure 14: Screenshots of the presented functionalities for the section regarding the Tongue Trackpad

RESULTS AND DISCUSSION [56]

3.1 Survey Data

Eighteen potential users filled the survey in its entirety, answering directly. One more complete answer was registered, by a family member of a potential user on their behalf. All answers were by users of age comprised between 18 and 74. Of these, only 31.58% declared to be living alone, while the rest relies on family assistance due to a lack of independence. Of the eighteen surveyed, 50% expressed interest in adopting this technology to interact with either phones or tablets. The most requested functionalities for the application, selected each by 14% of the users, were control of a wheelchair, easy access to 911 calls and interactions with keyboards. Open ended questions confirmed, in 23% of the cases, the need to have easy access to keyboards and communication. These functionalities were therefore implemented in the application. The possibility to interact with an on-screen keyboard and have easy access to a 911 call is available in both iterations, the

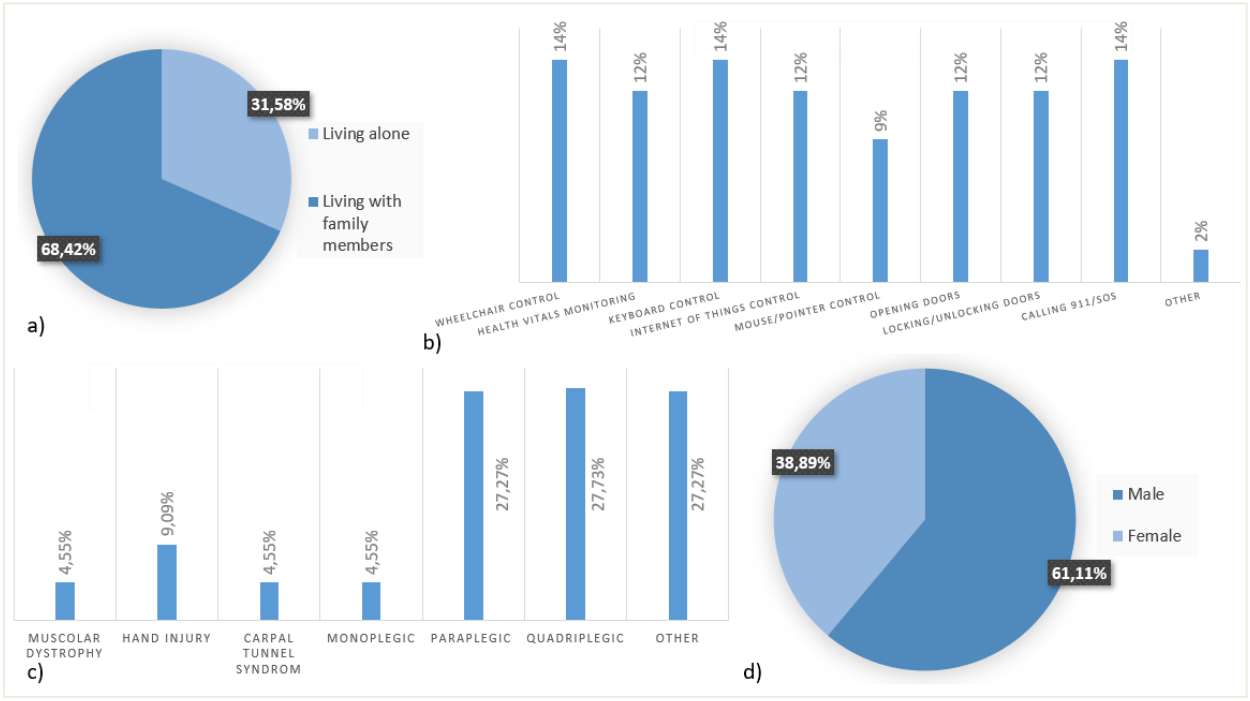

Figure 15: Survey data; a) respondents' living situation; b)desired functionalities of a tongue-controlled wearable device; c)type of ailment of the individuals who answered the survey; d)gender distribution of the individuals who answered

control of the wheelchair only in the Tongue Trackpad version. A summary of this data can be found in Figure 15.

Complaints about the current available technology that facilitates the use of one's hands, but does not bypass it, have been registered. In particular, two subjects mentioned discomfort in the use of their hands, though they retain partial control of them. This indicates the possibility to expand the target population to those users who, though still having partial motor control in the upper limbs, may experience hardship in leveraging them in interacting with technology. Using the proposed assistive devices combined with the interfaces the user could mix up the input to the phone, using both hands and tongue. Therefore, all the screens that were developed were both controllable through the input devices and by touch of fingers.

Most answers that were recorded were completed by the potential users through the adoption of voice recognition software, that were though identified as a less than ideal option by those individuals who filled in the open ended questions regarding any problems linked to current interaction with technological devices. The main issues noted were in the efficiency of recognition of the user's voice, as well as a complaint about the lack of privacy this solution entails.

The analysis of the survey data also highlighted the need to have an interface accessible for poor-sighted users or blind individuals. Using the built-in functionality of React Native

`accessibilityLabel` and `accessibilityHint`, each component was rendered compatible with a voice-over reader of the device the application is ran on.

3.2 Developed Functionalities

For each of the presented functionalities, one or more specific screens were implemented. These screens can be seen in Figure 16. All screens have been designed following the principles presented in the Materials and Methods section, and function according to the code that was presented there.

For what regards the Home screen, Figure 16 (a) , it is important to note that the icon that is linked to the emergency call has been positioned in the upper part of the screen. This is both an easy location for the cursor to reach from every point of the screen and an immediate link to the position of the pad on the top of the palate. This pad has been chosen to minimize accidental triggering of the emergency call, but also facilitate its location. The triggers for the most frequent actions were linked to the six pads located in the front of the palate, which are easier to reach and more comfortable to interact with, as the backwards motion of the tongue is the most uncomfortable. This is countered, though, by the closeness of the pads in the front of the device, which have resulted more difficult to discern [21].

As far a as the wheelchair control screen is concerned, the main emphasis was put on simplifying and facilitating the interaction. The buttons have been designed to be big and easily reached, as well as clearly marked. They have been kept close together in order to reduce the time needed to

move from one to the other. This prudence aims at reducing the chances of making a mistake on the user's part, which could be dangerous when controlling a wheelchair.

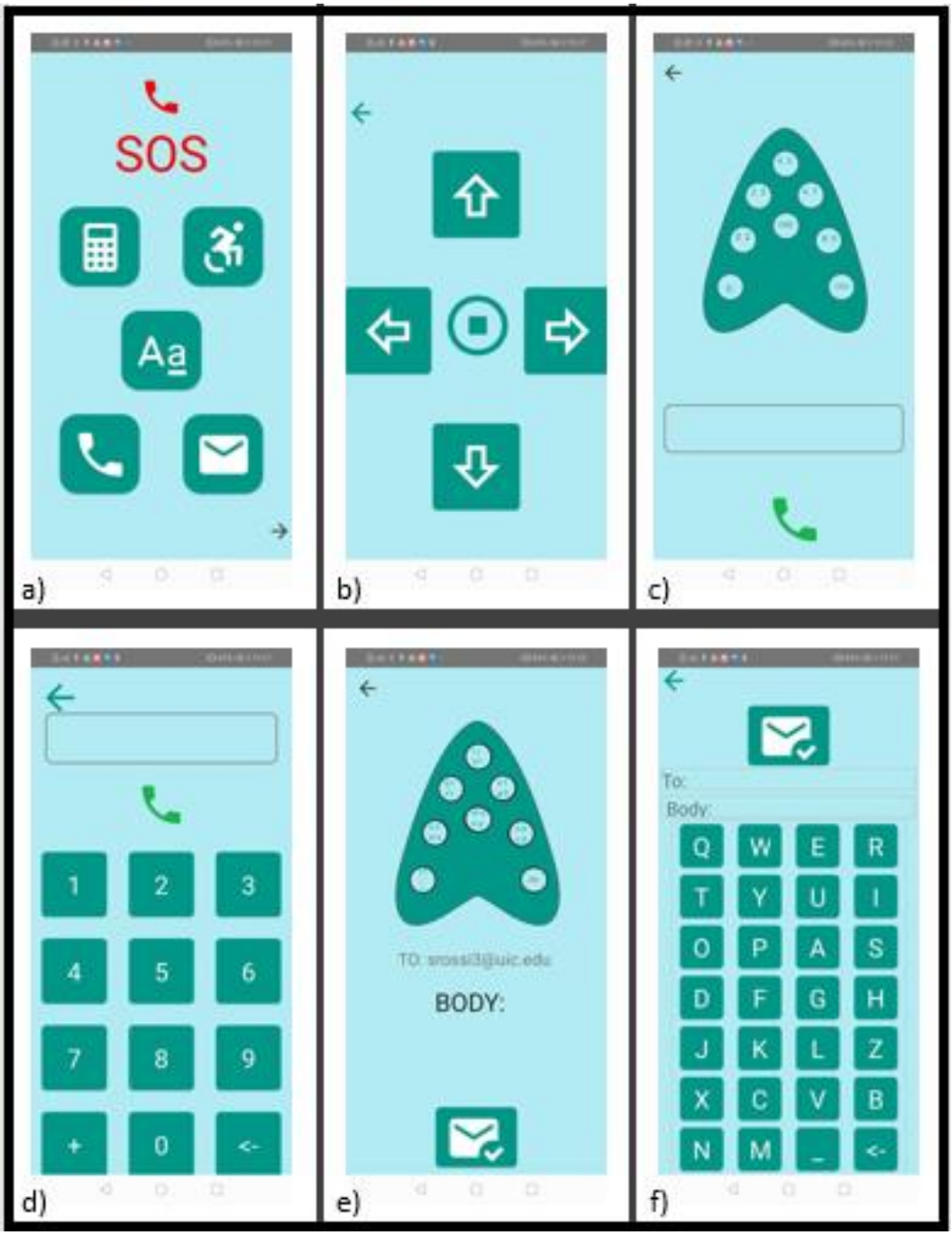

Figure 16: Developed screens for the implemented functionalities; a) Home screen; b) Wheelchair Control screen; c) Phone screen for the O-UIC; d) Phone screen for the Tongue Trackpad; e) Email screen for the O-UIC; f) Email/SMS screen for the Tongue Trackpad.

The screens related to the phone call and typing tasks are very different for the two versions of the application. The O-UIC screens present the user with a replica of the device, useful to identify the location of the letters and numbers of the T9 interaction. As can be seen the design is minimalistic and intuitive, but still encompassing of all necessary functionalities for the phone call to work.

Like the icon in the PhoneScreen, the button with the send-email icon acts as both placeholder and TouchableOpacity for necessities. The typed sentence would appear next to "*BODY:".* The "*to:"* field is precompiled to simplify testing. Both screens, in the Tongue Trackpad version, have big, distanced, buttons which are easily reachable with the cursor.

Table 5 summarizes the developed functionalities, the libraries used for each and the device they are optimized for.

Further functionalities were developed in collaboration with the Early Research Scholars Program (ERSP). The group was supervised in the implementation of three screens, allowing the user to have a custom-made calculator, settings screen and interaction with maps. These functionalities were adapted to the Tongue Trackpad version of the application. The principle followed in the implementation was the simplification of useful functionalities. The calculator screen, much like the phone screen presented before, has more distanced buttons, bigger and with greater contrast to the background, when compared to the default Android calculator. The settings screen addresses the issues of the hardware buttons present on most mobile phones for volume control. The controls have been developed as sliders, though a future implementation could aim at substituting them with buttons, which ensure an even easier interaction. The map interaction was optimized in the simplified identification of one's location, which could be further implemented in an emergency SMS sent to selected contacts. Also, the zoom function which requires two-finger interaction in

the default versions of maps was substituted with two on-screen buttons selectable with the Trackpad-controlled cursor.

TABLE 5: DEVELOPED FUNCTIONALITIES; THE X IDENTIFIES WHICH VERSION OF THE INTERFACES SUPPORTS THE DEVELOPED FUNCTIONALITY

Functionality	Device		Library
	O-UIC	TT	
SOS call	✕	✕	react-native-immediate-phone-call [52]
Generic phone call	✕	✕	react-native-immediate-phone-call [52]
Mock-wheelchair control		✕	react-native-ble-plx [42]
Opening of other Apps		✕	React-native-send-intent [53]
SMS		✕	React-native-send-intent[53]
Email	✕	✕	react-native-email[54]
Tongue Training		✕	Async Storage [50] React-native-chart-kit [51]
Voice-Over	✕	✕	React Native functionality [55]

3.3 Usability and inclusivity

In all screens developed one can see how the principles presented in paragraph 2.4 have been followed and implemented. When focusing on the ten heuristics, one can underline some aspects related to each principle.

The visibility of system status is achieved by the use of `TouchableOpacity` to convey to the user the change in status of the button or icon, with an increase in opacity when pressed. This is, though, lacking in the first iteration, where the commands are executed and only after the execution the result is available for the user.

To ensure both a minimalistic design and a real-world feeling, icons are used to identify the actions linked to specific buttons. Though these icons can appear, to a novice user, somewhat confusing, after a short period of time the icons allow for recognition of what a certain action entails. The size, shape and color of buttons and icons is consistent throughout the application, and across both versions, according to the color palette presented in Figure 5. This is also important for the inclusivity of the interface, as it ensures greater readability. To ensure affordance, regarding the accessibility of the application for blind users, the voice-over function can give the user audio feedback as to what would be triggered by pressing certain buttons, while the visual feedback is ensured by the aforementioned icons. The color of the buttons ensures a good contrast with the icons positioned inside of them, allowing for a greater ease of use. The use of only one color scheme takes into account the issue color-blind users would have in using hue to recognize an element.

To allow the user to easily recover from a wrong navigation, the possibility to go back to the home screen is always clearly marked and coherently positioned. For the O-UIC version, it is always the touch on the same pad (pad 0) that triggers the navigation to the home screen.

The possibilities offered by the second version of the application are limitless, because of the possibility to trigger the opening of any application installed on the phone by adding it to the icons presented. This allows for a greater flexibility, though it requires a back-end intervention to link the installed applications to the buttons on the interface. Future developments of the application could look at an automatization of the process, which would allow the user to automatically add the link to a pre-existing app installed on the phone. Momentarily, the present triggers regard the camera, the car sharing services and a custom-made application, developed in the WTSE Lab which presents to the user a 3D model of the tongue.

The most important precaution taken in preventing important errors regards the SOS call, for the first version of the application. To thwart false calls, the device's pad chosen to trigger the call is pad number 3, positioned on the top of the device and easily recognizable and avoidable during other functions.

Both versions of the application have been thought as an interface to allow users with disabilities to achieve an interaction with the smartphone and those essential functionalities that it offers, and therefore are built upon the concept of vigilance to the user's abilities.

3.4 Fitts Law

TABLE 6: DIFFICULTY INDEX OF THE DEFAULT KEYBOARD COMPARED TO THE CUSTOM-MADE ONE

	Default Keyboard	Custom-made Keyboard
Width (dpi)	35.14	60
Difficulty coefficient (1/dpi)	0.028	0.016

The results of the optimization of the button's width for the custom-made keyboard can be found in Table 6, which also includes the calculated Difficulty Index based on the width of the button.

As can be seen, the increase in width warrants a 43% decrease of the difficulty of interaction with the keyboard, thus allowing for a greater ease of use of the application. In the figure below, the two keyboards are displayed, to highlight the increased usability.

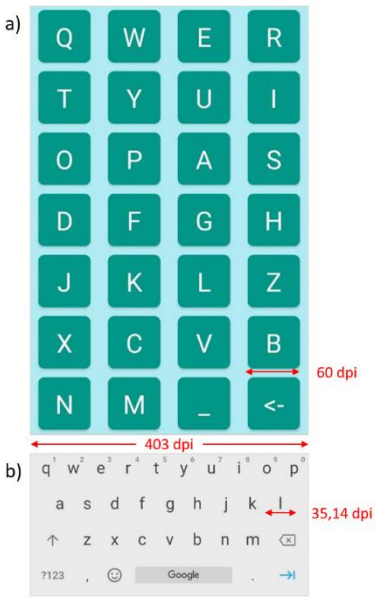

Figure 17: a) Custom-made keyboard with wider buttons; b) default Android keyboard

A sure limitation of the custom-made keyboard is the boundaries it imposes, such as only writing in capitalized letters and not allowing for numerical entries. Further developments could aim to address these issues and allow greater freedom.

Figure 18 shows the custom-made phone screen keyboard and the default one. The greater ease of use is evident in the custom-made screen, where the buttons are not only well defined, but also spaced wider and across a greater area. Also, the easily accessible delete button eases the process of error correction.

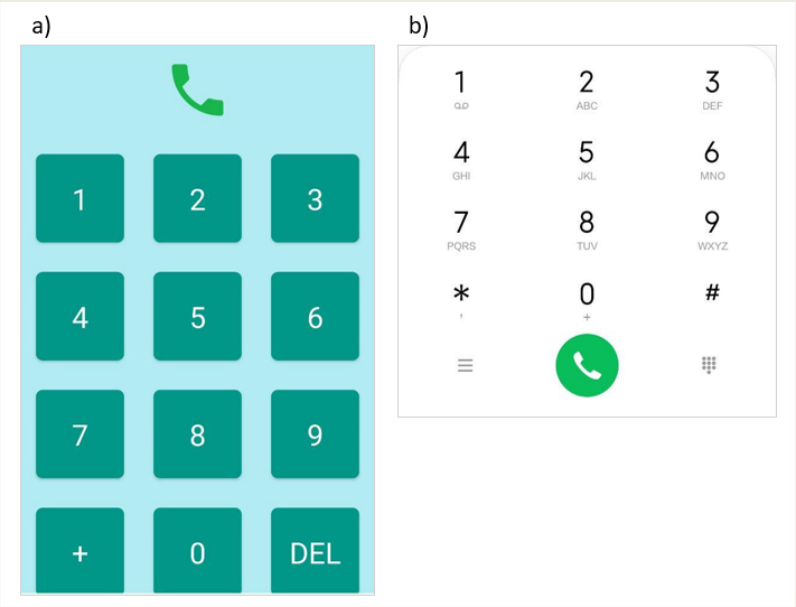

Figure 18: a) Custom made phone keyboard with distanced and well-defined buttons; b) default phone keyboard

3.5 Keystroke-Level Model

The following table summarize the theoretical times, estimated for the three tasks when executed without error by an expert user. Table 8 shows the time needed by the expert user to perform the same tasks with the default interfaces using a single finger.

TABLE 7: THEORETICAL TIMES NEEDED TO COMPLETE THE THREE TASKS USING THE TWO DEVICES

	911 Call	Phone Call	Email
O-UIC	2.65 s	22.85s	30.2s
TT	2.65 s	29.15s	34.45s

TABLE 8: SECONDS NEEDED BY EXPERT USER TO COMPLETE THE TASKS PRESENTED DURING THE EVALUATION OF THE INTERFACES.

911 Call	2s
Phone Call	7s
Email	6s

The time needed to call the emergency number (in the United States, 911) is compatible with the estimated time. It is presumable that the time needed to process the information as to how the emergency call is made would be less with experience, therefore allowing to decrease the time of mental preparation and making the call even faster than with the natural interaction. Regarding the second task, both devices estimate an interaction of more than 21 seconds, positioning themselves at more than triple the time needed during the normal interaction of a user controlling the default operating system with their finger. While this may seem a great difference, needing less than a minute to complete a phone-call when interacting with a tongue-controlled device is an acceptable result. The task of writing ‘hello world’ and sending the email is faster than the writing of both recipient and body of a message, so the third task is simpler for the two devices than the fourth task for the Trackpad. All the results, though, show a time which is more than four times the natural one for a typing task. This is due, for the Tongue Trackpad device, presumably, to the elevated number of pointing motions necessary to reach each letter, when compared to the touch-based selection of a keypad with the finger on the touch screen. For the O-UIC device, the T9

nature of the interaction together with the limited number of pads, that warrant for a high number of clicks to reach each desired letter.

3.6 Functioning Evaluation

The following tables summarize the results of the expert user's interaction with the two versions of the application, when performing the tasks analyzed in the evaluation.

TABLE 9:SECONDS NEEDED TO COMPLETE TASKS AND ERRORS MADE BY EXPERT USER INTERACTING WITH THE O-UIC DEVICE AND THE APPLICATION

	Time needed	Errors made
911 Call	2s	0
Phone Call	36s	4
Email	53s	5

As can be seen, the time needed, and the errors performed are compatible with the needs of a user. Both the phone-call and the simple email are executable in under a minute. Though the errors made by the user when interacting through the device are more than with the natural interaction -where no errors are usually made-, the number is still low enough not to hinder the usability of the device in combination with the custom-made application. It is expected that the time needed lowers, at least to the value estimated with the KLM theoretical calculations. This would be due to a reduction in the errors made by the user with the increase of experience.

Table 10 and Table 11 illustrate the times needed by an expert user to complete the tasks assigned to the functional evaluation of the Tongue Trackpad version of the interface, first in the interface

itself and then in the native environment of the phone. In Table 10 one can see how the number of errors is null, a great advantage when compared to the number of errors performed in similar task with the O-UIC device. The time, though, is greater for the first three tasks, which are equivalent. Though this may seem discouraging, it is arguable that an increase in dexterity, for example through the completion of the proposed tongue-training exercises, could positively affect the speed of a user's control of the cursor, therefore lowering the times. The KLM analysis is presumably the target of the decrease in time needed. This is an interesting aspect to assess in future developments of this work.

TABLE 10: SECONDS NEEDED TO COMPLETE THE TASKS AND ERRORS MADE AN EXPERT USER INTERACTING WITH THE APPLICATION WITH THE TONGUE TRACKPAD

	Time needed	Errors made
911 Call	4s	0
Phone Call	64s	0
Email	67s	0
SMS	91s	0

Table 11 shows the time needed to execute the same tasks as the ones executed within the interface, but in the default environment. Both the time needed, and the errors made, increase when interacting with an interface that has not been ideated and optimized for cursor-based interaction, especially for a cursor controlled by muscles not trained for such movements. Though the time needed to perform a phone call is not significantly higher, the still results more prone to errors.

These results confirm the theoretical simplification that increasing the width of the buttons and distance was expected to provide.

TABLE 11: SECONDS NEEDED TO COMPLETE THE TASKS AND ERRORS MADE AN EXPERT USER INTERACTING WITH THE PHONE'S DEFAULT SYSTEM WITH THE TONGUE TRACKPAD

	Time needed	Errors made
911 Call	21s	0
Phone Call	65s	1
Email	76s	0
SMS	104s	2

3.7 Observations and Discussion

During the performance of the proposed tasks, the expert user was observed and asked to provide feedback regarding the experience of interacting with the interfaces. Though the information gathered by observing only one user's interaction with the interfaces is limited (the study had to be cut short due to Covid-19 pandemic) it is still possible to infer useful cues for modifications and future developments of the applications. The user's feedback was collected after the conclusion of the tasks.

When using the O-UIC device's application, it was noticed that the images representing the link between numbers and letters and the respective pads were useful to the user. It was noticed that the user relied on the images to identify the correct pads to touch and the identify the number of

touches necessary (Figure 19). The modality of interaction, based on a T9-like selection of numbers and letters, was easily identified and applied by the user. Possible modifications to the layout that might aid the user in the interaction could include the engorgement of the images of the device in the phone and email screen, to facilitate the reading of the letters in each pad, and also the experimentation with different distributions of letters and commands between the pads. For example, redistributing the letters on the pads in order to include *x* and *y* and free up the eighth pad to only include the sending option could simplify the action. Multiple Studies have been conducted and have shown that the letters should be distributed according to the frequency of occurrence.

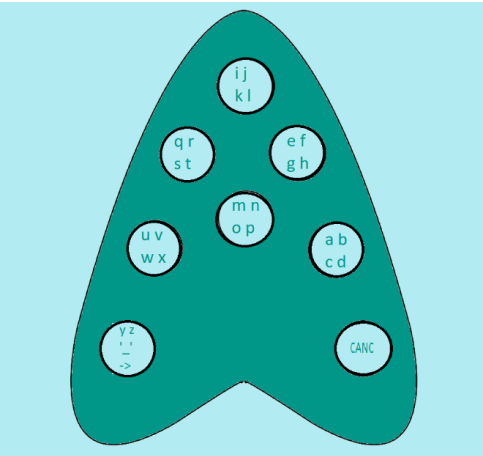

Figure 19: O-UIC device's phone interface displaying the link between the pads and the letters

In regards to the Tongue Trackpad's application, the main focus was centered on the optimization of the interaction with the interface compared to the native phone interface. It was observed that the user performs the tasks with ease, which suggests the user-friendliness of the developed application screens. For the phone-screen, no particular issues were noted . The interface developed for composing emails or SMS were also quickly understood by the user, who managed to interact with them after initial short training. The main issue was identifying how to select the

recipient field or the body field and compile the correct one. This is achieved by clicking on the field itself on the upper part of the screen, as seen in Figure 20. A potential adjustment that could improve the user experience would be a redistribution of the available real-estate. This redistribution should dedicate more space to the input fields, in order to allow for the user to make a simplified selection of the field, also a wider textbox should be incorporated to facilitate readability of the inputted letters. To achieve this, the button which triggers the opening of the default email or SMS application could be made smaller, and possibly moved to a different location on the screen such as the upper right corner. Furthermore, future developments should include changing of the keyboard, and incorporating a punctuation keyboard and a number keyboard, as well as a toggle between the uppercase and lowercase letters. This change could lead to a more encompassing and natural experience for the user.

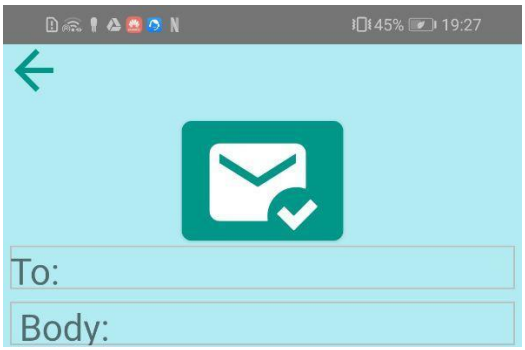

Figure 20: Recipient and body field in the email and SMS screens

The expert user commented on the enhanced ease of use that the increased dimensions of the buttons provided in comparison to the native phone environment, especially in the typing tasks. Furthermore, a comment regarding the concept of the app itself was made: the presence of all the buttons linking to the essential tasks on the homepage facilitates the interaction of the user with the phone, by providing immediate access to the core functionalities required. This could be

leveraged in the future developments to include the possibility for the user to add links to specific applications (which were for the moment limited to camera and weather applications) to be opened from the app. The presence of the big buttons and icons was essential in achieving this simplified access to the functionalities, according to the comments form the user. After the three rounds performed in the tongue training environment, the user demonstrated an interest in the possibility of leveraging the games to effectively improve their performance in the execution of both the games and the everyday life tasks available in the interface.

For both interfaces, what could be improved would be including the possibility to switch the interaction of the device with the application on and off. This would allow the user to only interact with the interfaces when desired, limiting the false triggers and false positive inputs. An issue that would be present when developing this functionality would be due to the impossibility for the user to interact with the smartphone unless through the device itself. This entails that, once the connection is switched off, external input would be necessary to restore the communication and allow the user to control the interface again. A possibility would be the leveraging of voice recognition control, to open and close the application and activate the communication.

Another functionality that would limit false triggers would be adding a confirmation popup to the SOS call. In both interfaces the user achieves the call to 911 either by pressing on the uppermost pad (for the O-UIC) or on the icon (for the Tongue Trackpad) once. To avoid false triggers, an additional pop-up was added for the Tongue Trackpad and three touches are required for the O-UIC.

3.8 Tongue Training

To test hypothesis one and two, both related to the vertical and horizontal movements of the tongue, two games were developed, following the same principle. The user is presented with two squares positioned next to each other in the center of the screen. They are prompted to click on the squares in order, first the one marked as “1” and then the one marked as “2”. Every time the user clicks on a button, they are awarded half a point, making it a whole point for each pair. When the second button is clicked the distance between them increases and the size decreases. This makes it progressively harder for the user. The games are indirectly based on the difficulty index presented in 2.3.2. This time, the distance between the buttons increases making it harder to move from one to the next and requiring more time. Once the minute is up, the user can either go back to the selection screen or restart the game, according to the number of points they gathered, with 5 being the minimum score needed to pass. If the user reaches ten points the buttons disappear and the game ends.

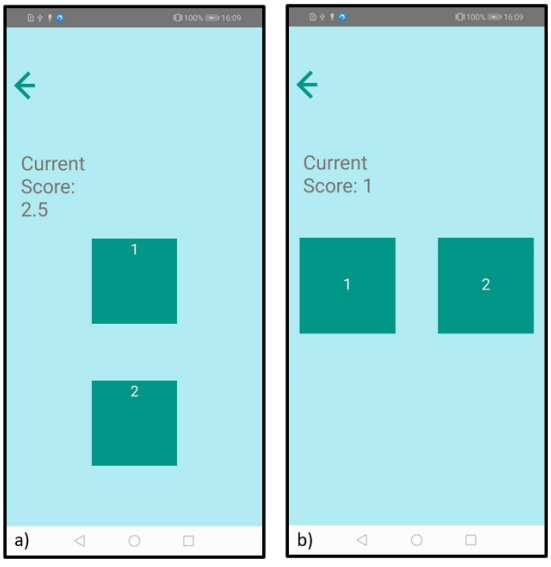

Figure 21: Screens developed to test hypotheses one and two; a) trains the vertical movement of the tongue from the front of the palate to the back; b) trains the horizontal movement from the left of the palate towards the right

To test hypothesis three and four, the user needs to click on a button, as often as possible. For hypothesis three only holding the click for 10 seconds would ensure the point, while any click would provide a point in the other case. To count the 10 seconds, the function `onLongPress ()` was used, that allows the developer to set a timer during which the element must be continuously pressed for the function to then be executed. The layout is different to allow the user to recognize the game at a glance. The developed screens can be seen in Figure 21.

For the last two hypotheses, randomly appearing circles are presented to the user, who is required to click on them as fast as possible. The easier version, testing hypothesis five, only has the dot appear in random locations and staying there until the user clicks on it. To test the speed the user acquires when training, hypothesis six is tested by a dot appearing like in the previous game, but disappearing after a set time, and reappearing in a different location. If the user manages to click on it in time, they gain a point, but the interval in which the dot disappears is reduced. The starting interval is 10s and is reduced by 100ms every time. An example in shown in Figure 22.

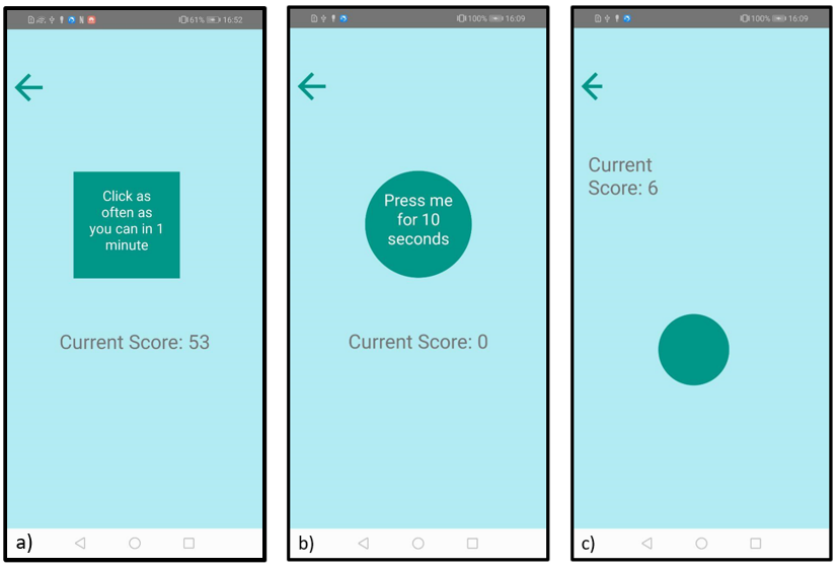

Figure 22: Screens developed to test hypotheses three, four, five and six; a) trains elevation and maintenance of the tongue-palate contact, tested in hypothesis four b) trains the speed in elevation of the tongue; c) the screenshot refers to the game developed for the fifth and sixth hypothesis. Since the goal is the same, click on the dot as it moves around, the layout was left identical; what changes is the speed with which the dot moves around

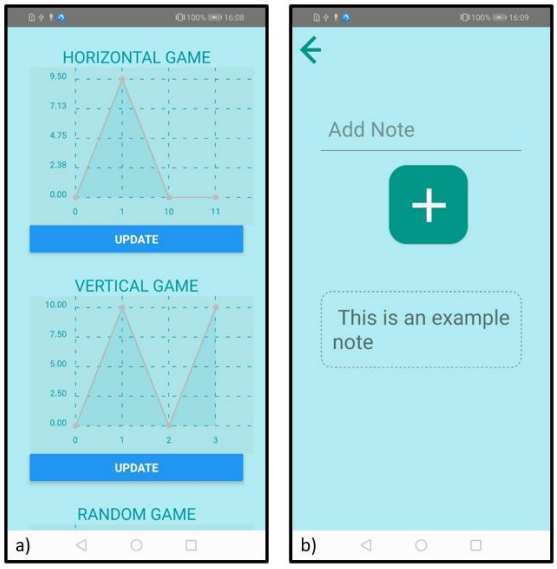

Figure 23: a) Data representation of the scores for two games, shown as an example. The user can update the graphs to show all the recent repetitions of the games; b) Notes section, where either the user or eventually a physician can add comments regarding the progress.

As mentioned in paragraph 2.9, the data is stored locally. There is, therefore, no threat to one's privacy, though developing a cloud-based storage of the information could be an interesting development, especially if the games are used in rehabilitation and clinical settings and require a physician to have access to the data. The scores are then presented to the user in graph form, in a separate screen where a summary of all games is shown. There is also the possibility for the user to add notes, which are also stored asynchronously. These screens can be seen in Figure 23.

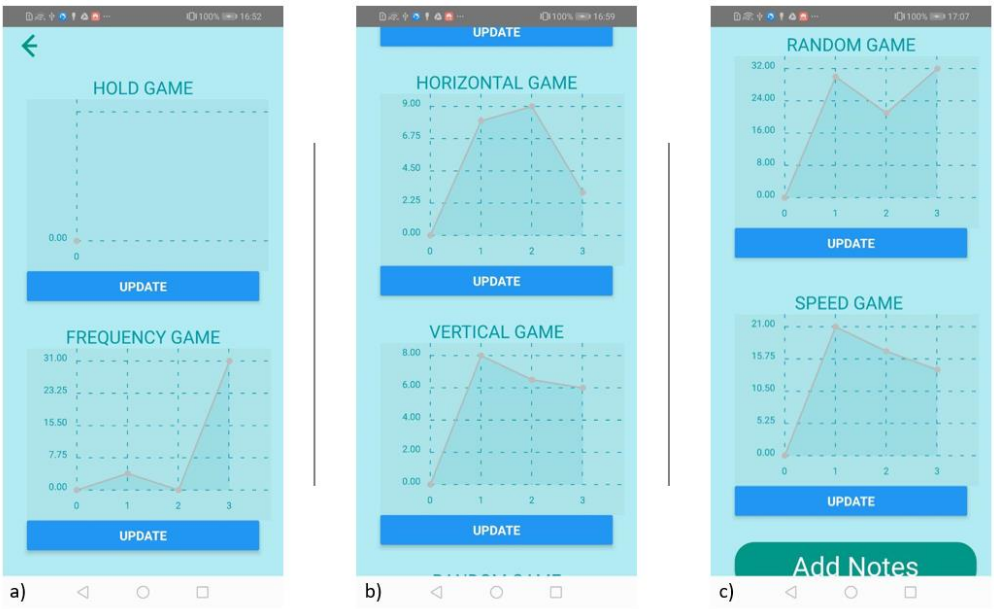

Figure 24: Results of trials of the games with the Tongue Trackpad; (a) results for 10 seconds holding game and tapping on the palate game; (b) results for vertical and horizontal movements; (c) results of games clicking on the dot appearing randomly and accelerating. The last game displays the frequency of tapping the dot, in Hertz, while the other games display the scores.

The screens in Figure 24 show the results obtained by the expert user when playing the games using the Tongue Trackpad. It is not useful to search for learning curves or improvement patterns in the reported data, as the games were solely performed to assess the functioning of the code. What is important to note, though, is the absence of results in the 10-second elevation game. Due to the long-press press gesture being necessary, but not yet implemented in the device's hardware,

the game is not yet playable with the device. This implementation, together with the effective testing of the hypotheses, is the natural progression of this work.

3.9 Feedback Survey

Eleven potential users participated in the survey, of which five completed the entire survey. The responses are summarized in Table 12. Each user was asked to select (on a scale of 1 to 5, with 1 being "strongly disagree" and 5 being "strongly agree") their agreement level to four statements regarding the application.

TABLE 12: USER RESPONSES TO THE FEEDBACK SURVEY ON THE DEVELOPED APPLICATIONS. THIS TABLE PRESENTS THE MEAN VALUE AND STANDARD DEVIATION (FROM 1 TO 5) OF N=5 RESPONSES

	O-UIC Device	Tongue Trackpad
I think that I would like to use this wearable device and application frequently.	2.4±1.35	2.6 ±1.2
I think that I would be able to use this system independently after it is placed inside the oral cavity.	4.6±0.49	4.4±0.8
I think this system would assist me in my interaction with my smartphone and computer.	3.4±1.02	3.2±1.32
On a scale of 1-5, 5 being the best, what is your overall interest in the wearable device and the associated application.	2.8±0.97	3.2±1.32

As shown by the results, though the sample size is limited, there is a potential interest by the targeted population towards the proposed application and devices. Responses show a strong agreement of the users in regards to the independence the application and devices would allow. The overall opinion asked in question four shows a preference towards the Tongue Trackpad and cursor control system, compared to the O-UIC device and T9 selection (mean 3.2 against mean 2.8).

The users were also presented with open ended questions, in order to provide their general feedback and thoughts. The suggestions that emerged could be a starting point for future development of the application: one user wrote “The typing feature has the letters grouped in alphabetical order. It might be better to group the letters by frequency of use, putting the most frequently used letters in the easiest positions to reach. It would have a longer learning curve, but would result in less fatigue.”, therefore suggesting experimentation with the layout of the letters in the T9-like pads, confirming the idea to test different layouts as a future development that had been hypothesized during the observation of user testing. Another user expressed a preference for the cursor like control, as they stated that double and triple clicks “can cause confusion on repetitive numbers or letters”. There was a suggestion to substitute one pad with a rolling ball that could act as a joystick to control a mouse, reinforcing the preference towards cursor control rather than T9 selection. Which has been developed by the WTSE group in the past. The tongue training environment was not presented to the users in this survey, but one of them suggested it as a possible addition to the application, thus confirming its potential in user engagement.

CONCLUSIONS AND FUTURE STEPS [56]

In this work two smartphone interfaces were developed, that aimed at allowing user-friendly access for individuals with upper limb impairment to core functionalities of smartphones. The interfaces were developed for two tongue-controlled assistive devices. I acknowledge the limited testing of the interfaces in my thesis, partly due to the Covid-19 pandemic that resulted in a shortening of my research period. Conclusions were reached based on only one healthy expert user's interaction with the interface. I acknowledge that the inclusion of more subjects, especially from the target population, could result in different outcomes and that this is a limitation to my thesis, though the presented outcomes and results are the ones that have been currently obtained and represent a starting point for future work.

To further ease the interaction of novice users with the proposed technology, a Tongue Training environment was developed, based on the studies on the biomechanical of the human tongue. The six Tongue Training environment could be used to test the hypotheses that training different movements of the tongue ensures enhancement in the strength and precision of motions: vertical and horizontal movement, elevation of the tongue to the palate and speed and dexterity in selecting a specific point on the screen.

Preliminary functionality tests were conducted. An expert user tested the core functionalities of both devices and the corresponding applications. The observation of these interactions results in understanding the ways the applications should be improved for possible future developments. Some examples are addition of a second verification confirmation in the executing the SOS call, which has been implemented in a final iteration of the design. Furthermore, a feedback survey was distributed online, to assess the target population's response to the proposed application. The responses suggested interest in the devices and applications.

Additional user testing, to further analyze the impact of the design on the user experience, is necessary, together with implementation of supplementary functionalities to cater to diverse user

needs. The testing of the training hypotheses could also be an interesting development for this project. To ensure the most information to be gathered with the future user testing phases, the observations gathered with the preliminary test should be implemented and tested. In particular, the different layouts of the interface, especially regarding the disposition of the letters in the T9 keyboard and the QWERTY keyboard, should be tested, to identify the best solution. Furthermore, the survey that was presented to the users could be adapted to in-person testing, to allow for a comparison between the response given to the applications after the user has experienced them, compared to the responses gathered after an online presentation of the solution.

In conclusion, we expect that through the correct training, a user could reach the desired dexterity to make interacting with their tongue and the palate-mounted assistive device simple. This, combined with the developed custom-made interfaces, could be a solution that allows disabled individuals access to smartphones.

CITED LITERATURE

[1] World Health Organization, "Summary World Report On Disability," *World Health*, pp. 1–24, 2011.

[2] B. S. Armour, E. A. Courtney-Long, M. H. Fox, H. Fredine, and A. Cahill, "Prevalence and causes of paralysis - United States, 2013," *Am. J. Public Health*, vol. 106, no. 10, pp. 1855–1857, 2016.

[3] "Paralysis in the USA." [Online]. Available: https://www.christopherreeve.org/living-with-paralysis/stats-about-paralysis. [Accessed: 23-Apr-2020].

[4] P. Raghavan, "Upper Limb Motor Impairment After Stroke," *Phys. Med. Rehabil. Clin. N. Am.*, vol. 26, no. 4, pp. 599–610, 2015.

[5] R. L. Sacco *et al.*, "An updated definition of stroke for the 21st century: A statement for healthcare professionals from the American heart association/American stroke association," *Stroke*, vol. 44, no. 7, pp. 2064–2089, 2013.

[6] N. Level, "Spinal Cord Injury Facts and Figures at a Glance Re-Hospitalization," 2019.

[7] "Spinal Cord Injuries." [Online]. Available: https://www.themiamiproject.org/resources/statistics/. [Accessed: 23-Apr-2020].

[8] M. D. Saunders, J. P. Smagner, and R. R. Saunders, "Improving methodological and technological analyses of adaptive switch use of individuals with profound multiple impairments," *Behav. Interv.*, vol. 18, no. 4, pp. 227–243, 2003.

[9] S. N. Patel and G. D. Abowd, "Blui: Low-cost localized blowable user interfaces," *UIST Proc. Annu. ACM Symp. User Interface Softaware Technol.*, pp. 217–220, 2007.

[10] P. D. Geoge F. Wittenberg, M.D., "Experience, Cortical Remapping and Recovery in Brain Disease," 2010.

[11] M. Scherer and P. Parette, “Assistive technology use and stigma,” *Educ. Train. Dev. Disabil.*, no. September 2004, 2004.

[12] “Hypoglossal Nerve.” [Online]. Available: https://en.wikipedia.org/wiki/Hypoglossal_nerve. [Accessed: 10-Feb-2020].

[13] “Palatoglossus Muscle.” [Online]. Available: https://en.wikipedia.org/wiki/Palatoglossus_muscle. [Accessed: 10-Feb-2020].

[14] L. N. S. Andreasen Struijk *et al.*, “Development and functional demonstration of a wireless intraoral inductive tongue computer interface for severely disabled persons,” *Disabil. Rehabil. Assist. Technol.*, vol. 12, no. 6, pp. 631–640, 2017.

[15] X. Huo, J. Wang, and M. Ghovanloo, “A magneto-inductive sensor based wireless tongue-computer interface,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 16, no. 5, pp. 497–504, 2008.

[16] Y. Nam, Q. Zhao, A. Cichocki, and S. Choi, “Tongue-rudder: A glossokinetic-potential-based tongue-machine interface,” *IEEE Trans. Biomed. Eng.*, vol. 59, no. 1, pp. 290–299, 2012.

[17] Z. Li, R. Robucci, N. Banerjee, and C. Patel, “Tongue-n-Cheek: Non-contact tongue gesture recognition,” *IPSN 2015 - Proc. 14th Int. Symp. Inf. Process. Sens. Networks (Part CPS Week)*, pp. 95–105, 2015.

[18] R. Slyper, J. Lehman, J. Forlizzi, and J. Hodgins, “A tongue input device for creating conversations,” *UIST’11 - Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, pp. 117–125, 2011.

[19] N. Marjanovic, K. Kerr, R. Aranda, R. Hickey, and H. Esmailbeigi, “Wearable wireless User Interface Cursor-Controller (UIC-C),” *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol.*

Soc. EMBS, pp. 3852–3855, 2017.

[20] N. Marjanovic, G. Piccinini, K. Kerr, and H. Esmailbeigi, "TongueToSpeech (TTS): Wearable wireless assistive device for augmented speech," *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, pp. 3561–3563, 2017.

[21] M. Tomback, "The Oral User Interface Controller (O-UIC): An Assistive Communication Device by," 2019.

[22] L. Reade, "Enabling technology," *Eureka*, vol. 27, no. 6, p. 19, 2007.

[23] F. Scholz, B. Yalcin, and M. Priestley, "Internet access for disabled people: Understanding socio-relational factors in Europe," *Cyberpsychology*, vol. 11, no. 1Special Issue, 2017.

[24] P. Denman, L. Nachman, and S. Prasad, "Designing for 'a' user: Stephen Hawking's UI," *ACM Int. Conf. Proceeding Ser.*, vol. 2, pp. 94–95, 2016.

[25] "ASSISTIVE CONTEXT-AWARE GETTING STARTED GUIDE." pp. 1–31.

[26] H. M. Clark, "Specificity of training in the lingual musculature," *J. Speech, Lang. Hear. Res.*, vol. 55, no. 2, pp. 657–667, 2012.

[27] M. Sasaki, K. Onishi, A. Nakayama, K. Kamata, D. Stefanov, and M. Yamaguchi, "Tongue motor training support system," *2014 36th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBC 2014*, pp. 3582–3585, 2014.

[28] "MDBT42Q," vol. 001. pp. 1–37, 2018.

[29] "nRF52832 - Product Specification v1.0," p. 544, 2016.

[30] WTSE LAB, "Survey," 2019. [Online]. Available: https://uic.ca1.qualtrics.com/jfe/form/SV_2fa1eBm3fcKTlDD. [Accessed: 20-Feb-2020].

[31] "React Native." [Online]. Available: https://engineering.fb.com/android/react-native-bringing-modern-web-techniques-to-mobile/. [Accessed: 23-Feb-2020].

[32] “color-hex.” [Online]. Available: https://www.color-hex.com/. [Accessed: 05-Apr-2020].

[33] I. Kondratova and I. Goldfarb, “Cultural interface design: Global colors study,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4277 LNCS, pp. 926–934, 2006.

[34] “Touchable Opacity.” [Online]. Available: https://reactnative.dev/docs/touchableopacity. [Accessed: 05-Apr-2020].

[35] Paul M. Fitts, “The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement,” *J. Exp. Psychol.*, vol. 47, no. 6, pp. 381–391, 1954.

[36] J. Accot and S. Zhai, “Beyond Fitts’ law: Models for trajectory-based HCI tasks,” *Conf. Hum. Factors Comput. Syst. - Proc.*, pp. 295–302, 1997.

[37] S. (University of T. MacKenzie, “Fitt’s Law,” *Human-Computer Interaction*, vol. 7. pp. 91–139, 1992.

[38] “xiaomi_mi_mix_2.” [Online]. Available: https://www.gsmarena.com/xiaomi_mi_mix_2-8529.php. [Accessed: 06-Apr-2020].

[39] R. Molich and J. Nielsen, “Improving a Human- Computer Dialogue,” vol. 33, no. 3, 1990.

[40] J. Nielsen, “Heuristics.” [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed: 07-Apr-2020].

[41] F. Hammad, P. D. Wastell, and T. Chesney, “CREATING INCLUSIVE USER INTERFACES : TOWARD A COMPREHENSIVE INCLUSIVITY EVALUATION,” pp. 28–38, 2016.

[42] “Ble Plx Library.” [Online]. Available: https://polidea.github.io/react-native-ble-plx/. [Accessed: 01-Mar-2020].

[43] “Reducers.” [Online]. Available: https://redux.js.org/basics/reducers/. [Accessed: 02-Mar-

2020].

[44] “Puck.js BLE.” [Online]. Available: https://www.espruino.com/About+Bluetooth+LE. [Accessed: 30-Apr-2020].

[45] B. E. John and D. E. Kieras, “The GOMS family of user interface analysis techniques: comparison and contrast,” *ACM Trans. Comput. Interact.*, vol. 3, no. 4, pp. 320–351, 1996.

[46] P. Svensson, A. Romaniello, L. Arendt-Nieisen, and B. J. Sessle, “Plasticity in corticomotor control of the human tongue musculature induced by tongue-task training,” *Exp. Brain Res.*, vol. 152, no. 1, pp. 42–51, 2003.

[47] H. M. Clark, K. O’Brien, A. Calleja, and S. N. Corrie, “Effects of directional exercise on lingual strength,” *J. Speech, Lang. Hear. Res.*, vol. 52, no. 4, pp. 1034–1047, 2009.

[48] C. Namiki *et al.*, “Tongue-pressure resistance training improves tongue and suprahyoid muscle functions simultaneously,” *Clin. Interv. Aging*, vol. 14, pp. 601–608, 2019.

[49] H. A. Caltenco, B. Breidegard, and L. N. S. Andreasen Struijk, “On the tip of the tongue: Learning typing and pointing with an intra-oral computer interface,” *Disabil. Rehabil. Assist. Technol.*, vol. 9, no. 4, pp. 307–317, 2014.

[50] “Async Storage.” [Online]. Available: https://github.com/react-native-community/async-storage. [Accessed: 09-Apr-2020].

[51] “react-native-chart-kit.” [Online]. Available: https://github.com/indiespirit/react-native-chart-kit. [Accessed: 09-Apr-2020].

[52] “react-native-immediate-phonecall.” [Online]. Available: https://www.npmjs.com/package/react-native-immediate-phone-call. [Accessed: 02-Mar-2020].

[53] “react-native-send-intent.” [Online]. Available: https://www.npmjs.com/package/react-

native-send-intent. [Accessed: 09-Apr-2020].

[54] “react-native-email.” [Online]. Available: https://www.npmjs.com/package/react-native-email. [Accessed: 02-Mar-2020].

[55] “Accessibility.” [Online]. Available: https://reactnative.dev/docs/accessibility.html. [Accessed: 12-Mar-2020].

[56] S.Rossi, N. Marjanovic, H. Esmailbeigi "*Development of Smart-Phone Interfaces for Tongue Controlled Assistive Devices*" in *17th International Conference on Computers Helping People with Special Needs*, 2020

VITA

NAME:	Silvia Maddalena Rossi
EDUCATION:	Classical High School Diploma (2010-2015), Liceo Ginnasio Statale Giuseppe Parini, Milan, Italy
	Bachelor of Science in Biomedical Engineering (2015-2018), Politecnico di Milano, Milan, Italy
	Master of Science in Biomedical Engineering (2018-present), Politecnico di Milano, Milan, Italy
	Master of Science in Bioengineering (2019-present), University of Illinois at Chicago, Chicago, USA
WORK EXPERIENCE:	Graduate Student Researcher for Richard and Loan Hill Department of Bioengineering, Wearable Technology and Sensory Enhancement Laboratory, University of Illinois at Chicago, Chicago, USA