

EMPWRD: Enhanced Modular Platform for People with Rigid Disabilities

by

Moustafa Dafer

Submitted in partial fulfilment of the requirements  
for the degree of Master in Computer Science

at

Dalhousie University  
Halifax, Nova Scotia  
August 2019

© Copyright by Moustafa Dafer, 2019

## TABLE OF CONTENTS

List of Tables .....	iv
List of Figures .....	v
List of Abbreviations Used .....	viii
Glossary.....	ix
Abstract.....	xii
Acknowledgements.....	xiii
Chapter 1 Introduction and Overview.....	1
1.1 Introduction.....	1
1.2 Purpose and Inspiration.....	2
1.3 Objectives.....	3
1.4 Outline .....	4
Chapter 2 Literature Review .....	5
2.1 Input Methods for Mobility Challenged Users .....	5
2.2 Scanning Cluster Ambiguous Keyboards.....	11
2.2.1 Introduction to Cluster Ambiguous Keyboards .....	11
2.2.2 Letter Arrangements in Cluster Ambiguous Keyboards.....	13
2.2.3 Scanning Keyboards .....	15
2.2.4 Keyboards for People with Motor Impairments.....	17
2.3 Research Gaps.....	22
Chapter 3 Proposed EMPWRD Framework and System.....	23
3.1 Framework Characteristics and Requirements .....	23
3.2 EMPWRD System Components and Design .....	25
3.2.1 Frontend and UI .....	26

3.2.2	User-side Backend .....	33
3.2.3	Server-side Backend .....	36
3.3	Sample Input Device: Webcam.....	37
3.4	Sample Module: Self-hosted GSM.....	39
3.4.1	AI Thinker A7 Development Board:.....	40
3.4.2	The Communication Protocol and NodeJS API.....	43
3.4.3	Audio Configuration and Streaming (VOIP).....	44
3.4.4	GSM Module Watchdog .....	46
3.4.5	Overall System and Integration .....	48
3.4.6	Other Considerations and SaaS Solutions .....	49
Chapter 4	Novel Algorithms for the Design.....	51
4.1	Auto-Correct Algorithm for Cluster Ambiguous Keyboards .....	51
4.2	Input-Adaptive Scanning Grid .....	58
4.3	Input Calibration Module .....	63
Chapter 5	Comparisons and Experimental Analysis.....	71
5.1	Cheek Muscle Input Methods Comparison and APDS-9960 .....	71
5.2	Stephen Hawking's System: Assistive Context-Aware Toolkit (ACAT) .....	76
Chapter 6	Conclusions and Future Work .....	85
References	.....	88

## LIST OF TABLES

Table 1 Raspberry Pi NodeJS Server API .....	44
Table 2 Comparison Between Self-Hosted and SaaS GSM .....	50
Table 3 TCRT5000 vs APDS-9960 .....	73
Table 4 Feature Summary of Different Input Methods .....	74
Table 5 Comparative Overview of ACAT vs EMPWRD .....	84

## LIST OF FIGURES

Figure 1 Facial Expression Detection Using a Camera.....	5
Figure 2 Thermal Image Examples.....	6
Figure 3 EMG Facial Patches.....	7
Figure 4 PANDA Type Ring Resonator.....	7
Figure 5 EEG Headset.....	8
Figure 6 Sample Piezoelectric Sensor.....	9
Figure 7 Breathing Pressure Sensors [57].....	9
Figure 8 TCRT5000 Picture and Top View.....	10
Figure 9 Stephen Hawking and Cheek Movement Sensor.....	10
Figure 10 T9 Keyboard Layout.....	12
Figure 11 Full QWERTY Keyboard Next to Three Ambiguous Keyboards [73].....	13
Figure 12 QWERTY Layout in a Cluster Ambiguous Keyboard [5].....	14
Figure 13 Screenshot of OneKey Application with 4 keys [80].....	15
Figure 14 Linear Scanning Ambiguous Keyboard Concept [80].....	16
Figure 15 Row-based Scanning in Ambiguous Keyboard Concept [80].....	16
Figure 16 Dvorak Keyboard Layout [101].....	18
Figure 17 Metropolis Keyboard [104].....	19
Figure 18 Dasher: The Zooming Interface [107].....	19
Figure 19 Clavicom NG: (1) word propositions, (2) AZERTY keyboard [110].....	20
Figure 20 K-Hermes: (1) Word proposals, (2) 3-letters-per-key keyboard [110].....	21
Figure 21 Device Control Page and Dock Controller.....	26
Figure 22 Dock Controller in Start Screen.....	27
Figure 23 Input Sensor TCRT5000 Mounted on an Eye-Glasses Frame.....	28
Figure 24 Phone Call Interface.....	29
Figure 25 Contact Selection Panel.....	30
Figure 26 Call Initiation Confirmation Dialog.....	31
Figure 27 Common Phrases Panel.....	32
Figure 28 Input-Calibration Module Initial Screen.....	32

Figure 29 Device Control Module: Microcontroller, and Controlled Socket .....	34
Figure 30 Microcontroller (Client) Board and TCRT5000 .....	35
Figure 31 Affectiva's AFFDEX SDK Implementation .....	38
Figure 32 Raspberry Pi 3 Model B+ [115] .....	40
Figure 33 AI Thinker A7 GSM Development Board Used [1] .....	40
Figure 34 AT Commands for Testing A7 GSM Module Using SSCOM tool .....	42
Figure 35 The Different Modules of The System .....	43
Figure 36 External USB Audio Card [116] .....	45
Figure 37 Audio Connections and Streaming Over the Web .....	45
Figure 38 Relay Module Used for Resetting GSM Module [118] .....	47
Figure 39 Reset Command from Issuing to Execution .....	47
Figure 40 Overall System Components and Communication Flow .....	48
Figure 41 Self-Hosted GSM Module Used in Demo .....	49
Figure 42 Cluster Ambiguous Keyboards' Hashing Algorithm .....	51
Figure 43 Hashed Dictionary for Cluster Ambiguous Keyboard .....	52
Figure 44 Chanti Software Featuring a Delete Sequence Button .....	53
Figure 45 3-Gram Dictionary with New Sentence Identifiers .....	54
Figure 46 Traditional Flow of NLP in Ambiguous Keyboards .....	55
Figure 47 Text Correction Component .....	56
Figure 48 Overview of Proposed Text Correction Algorithm .....	57
Figure 49 Scanner Blocks (1): Suggestions Block, (2) Letters Block .....	58
Figure 50 Single-Input Behavior in EMPWRD Scanning Grid Algorithm .....	59
Figure 51 Flow of Traditional Scanning Keyboards .....	61
Figure 52 Scanning Cycle in Manual Mode .....	62
Figure 53 Stephen Hawking and Cheek Movement Sensor .....	63
Figure 54 TCRT5000 Picture and Top View .....	63
Figure 55 Calibration Initialization Tutorial in Action .....	64
Figure 56 Ready State Waiting for Input .....	65
Figure 57 Signal Indicator Showing Good Signal Strength .....	66
Figure 58 Calibration Fails After a False Positive Occurs .....	66
Figure 59 Groups of Signal Strengths .....	67

Figure 60 The Process of Sensor Calibration .....	68
Figure 61 Final Signal Strength Ruled Out by Multi-Trial Voting System .....	70
Figure 62 TCRT5000 (left) and APDS-9960 sensor (right) .....	72
Figure 63 Facial Expression Detection in ACAT [22] .....	76
Figure 64 ACAT’s Scanning Grid Keyboard for Pointer Controls [22].....	77
Figure 65 Full Scanning Grid Keyboard With Predicted Words in ACAT [22] .....	78
Figure 66 Scan Speed Configuration Window in ACAT [22] .....	78
Figure 67 Step 1: Block Selection In ACAT’s Scanning Grid Keyboard [22] .....	79
Figure 68 Step 2: Row Selection In ACAT’s Scanning Grid Keyboard [22].....	80
Figure 69 Step 3: Column Selection In ACAT’s Scanning Grid Keyboard [22] .....	80
Figure 70 Alphabetic Layout Scanning Grid Keyboard in ACAT [22] .....	81
Figure 71 ACAT’s Talk Window and Talk-Related Scanning Grid Keyboard [22] ....	81
Figure 72 Cluster Ambiguous Keyboard Featured in EMPWRD .....	83

## LIST OF ABBREVIATIONS USED

ACAT	Assistive Context-Aware Toolkit
API	Application Programming Interface
ECG	Electro-Cardiogram
EEG	Electro-Encephalogram
EOG	Electro-Oculogram
GPIO	General Purpose Input/Output
GPRS	General Packet Radio Services
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HTML	Hypertext Markup Language
IC	Integrated Circuit
IoT	Internet of Things
JS	Javascript
MQTT	Message Queuing Telemetry Transport
NLP	Natural Language Processing
POS	Part of Speech
SaaS	Software as a Service
TTS	Text-To-Speech
UART	Universal Asynchronous Receiver/Transmitter
VOIP	Voice Over Internet Protocol
WebRTC	Web Real-Time Communication

## GLOSSARY

**AI Thinker A7:** “GSM/GPRS/GPS function module. It supports GSM/GPRS Quad-Band (850/900/1800/1900) network. Also, it supports voice calls, SMS messages, GPRS data service and GPS function. The module is controlled by AT command via UART and supports 3.3V and 4.2V logical level.” [1]

**APDS-9960:** Digital Proximity, Ambient Light, RGB and Gesture Sensor. “The APDS-9960 device features advanced Gesture detection, Proximity detection, Digital Ambient Light Sense (ALS) and Color Sense (RGBC)”. [2]

In this report, we are interested in using this sensor as a single-pixel camera only (using it is RGB sensors).

**API:** Application Programming Interface, is an intermediary software that allows two applications to talk to each other. [3]

**AT commands:** “AT commands are instructions used to control a modem. AT is the abbreviation of ATtention. Every command line starts with "AT" or "at". That's why modem commands are called AT commands.” [4]

**Cluster Ambiguous Keyboard:** a cluster ambiguous keyboard is a keyboard that combines multiple letters in one key. [5]

**GPIO:** General Purpose Input/Output, which means a pin can be programmed to act as input (e.g. to read values from sensors) or as an output (e.g. To control devices).[6]

**GPRS:** General Packet Radio Services, is a wireless communications service that is a packet-based, and features data rates between 56 and 114 kbps over a continuous Internet connection for mobile devices. [7]

**GPS:** Global Positioning System, satellite positioning system that is used for determining the geo-location of a receiver with direct line of sight to multiple satellites.[8]

**GSM:** Global System for Mobile Communications, is the European standard for second generation (2G) digital cellular networks used by mobile devices. [9]

**IC:** Integrated Circuit (aka monolithic integrated circuit, chip, and microchip), is a collection of electronic components connected together and placed on a chip. [10]

**IoT:** Internet of Things, “The Internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction” [11].

**Microcontroller:** “A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.” [12]

**MQTT:** Message Queuing Telemetry Transport, is an extremely lightweight machine-to-machine connectivity protocol designed with IoT in mind. It features a very small footprint and is designed as a publish/subscribe messaging transport. [13]

**NLP:** Natural Language Processing, is the field that deals with making computers able to process natural human languages. [14]

**NodeJS:** Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. It allows the use of JavaScript for server-side scripting [15][16]. Node.js has an event-driven architecture that features asynchronous I/O. This helps optimize throughput and scalability in web applications with many input/output operations; it is also developed with real-time Web applications in mind (e.g., real-time communication programs and browser games). [17]

**Raspberry Pi:** is a series of small and cheap computers that are meant for teaching kids programming. [18] The raspberry pi became very popular among technology tinkerers and enthusiasts because it features a 40-pin GPIO header built in. [19]

**RESTful API:** A RESTful API is an API that uses HTTP requests to GET, PUT, POST, and DELETE data. It is preferred over other types of APIs since it requires less bandwidth. [20]

**SaaS:** Software as a service, is a form of software that is centrally hosted and accessed via a subscription rather than being installed on every computer. [21]

**Scanning Grid:** A scanning grid is a grid of predefined buttons and layout, where the focus is rotated along buttons automatically, thus allowing the choice of any button using a single binary input. The grid can be in the form of a keyboard for example. [22]

**TCRT-5000:** is a reflectivity sensor that features an infrared emitter and a phototransistor, and is used for the detection of reflective material such as paper, IBM cards, magnetic tapes etc. [23] In this report, we are interested in the use of this sensor for cheek muscle contraction detection.

**TTS:** “Text to Speech is used to artificially produce human speech through computerized means. Text to speech converts written language in to speech” [24]

**UART:** Universal Asynchronous Receiver/Transmitter, is a physical circuit in a microcontroller or a stand-alone IC that is responsible for transmitting and receiving serial data. [25]

**VOIP:** Voice Over Internet Protocol, “is a technology that makes phone calls possible from any Internet-connected device with a microphone and speakers” [26].

**Watchdog:** a piece of hardware or software that constantly checks whether a device or a function is ready. If the function is not responsive for a specified period of time, then the watchdog resets it. [27]

**WebRTC:** Web Real-Time Communication, “is a free, open project that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs” [28]. It is very famous for its capability of audio and video streaming.

## **ABSTRACT**

Neurological disorders such as amyotrophic lateral sclerosis (ALS) can lead to severe mobility limitations. ALS has even been described as “locked-in” syndrome, since the brain is functional, but the personality is locked in a body with almost no voluntary control. The objective is to design, implement, and test a sensor based platform called EMPWRD which empowers such users with the ability to speak and express themselves using state-of-the-art tools such as an input-adaptive grid scanner and a Natural Language Processing (NLP) powered cluster ambiguous keyboard; it also helps them control devices in their surroundings. The platform is modular and is compatible with any device that supports HTML and Javascript; thus, it can be easily customized. We present here the platform in addition to a Text-To-Speech (TTS) and a Global System for Mobile Communications (GSM) module that enable users to make live phone calls, along with some comparisons and experimental analysis.

## **ACKNOWLEDGEMENTS**

First and foremost, praises and thanks to God, the Almighty, for making everything possible in the successful completion of this work.

I would like to express the deepest appreciation to my supervisor, Prof. Srinivas Sampalli for his encouragement, guidance, and patience that he demonstrated during my study and for providing all the opportunities necessary in making this work possible.

I take immense pleasure in expressing my sincere gratitude towards my guides and mentors, Prof. Hiroyuki Ohno, Prof. Aaron Newman, Prof. Rached Zantout, Prof. Mohamad El Abed, and the team of Halifax Learning for their exemplary motivation and creative suggestions.

I wish to extend my gratitude to my colleagues and friends Mir Masood Ali, Anirudh Koul, Robbie MacGregor, Achla Parashar, Alaa Maarouf, Ahmad Shebl, Abdullah Khayyal, Sogra Memon, Dinesh Shenoy, Nazmul Islam Riyadh, and Bader Aldughayfiq, not only for their valuable ideas and help, but also for their constant support, encouragement, and their consistent presence whenever I needed them.

I'm also thankful to MYTech Lab members, who have always been supportive and enthusiastic, never once showing any signs of boredom while listening to me talk about this project tens of times during our weekly meetings.

I am deeply indebted to my family, relatives, and friends who have enlightened my way with enchanting encouragement and support, for their patience and infinite kindness, and for always being there for me.

Last but not least, I would like to thank all those who helped proofreading this study. They too have contributed towards this project's success.

# **CHAPTER 1      INTRODUCTION AND OVERVIEW**

## **1.1 INTRODUCTION**

Since the beginning of time, humans have known the importance of communication. Language has allowed us to express our thoughts and live as part of a community. Modern communication inventions such as the phone and the Internet have even allowed communications beyond the boundaries of land and the limitations of sound travel. However, of what importance are these systems for someone who has lost the ability to communicate?

After all, humans are thought-sharing and thought-provoking creatures. There are a lot of reasons that may force individuals to change their methods of communication. People who lose their ability to speak, whether at birth or due to other causes that emerge during one's lifetime, learn to communicate through other methods such as sign language and writing.

Switching from one communication method to another depends on the available methods and the different abilities one possesses. A person who cannot speak using their mouth, needs their hands and their fingers to use sign language; similarly, someone who's lost their fingers, won't be able to write using the traditionally common method; they could, for example, use speech to text technologies, which requires them to use their mouth and voice, or they may train to use another part of their body to do that instead of their fingers.

When thinking about such circumstances, a very important question comes to mind. How can we communicate with people who have lost control over their body? We are referring to people who are completely capable of thinking and understanding their surroundings but have lost all means of voluntary control over their muscles, a case that is becoming more prevalent, unfortunately, due to the increasing cases of neurological disorders and other causes such as spinal injuries caused by accidents. Such patients may lose control over some body parts or even become completely

paralyzed. It is worth mentioning that it has been recently discovered by advances in brain scanning and analysis techniques that some coma-state patients are aware of their surroundings and should not be deemed almost-dead as previously thought.

## **1.2 PURPOSE AND INSPIRATION**

As time goes by, miraculous technologies emerge, breaking the boundaries that we once thought were unbreakable. One can easily imagine how the story of a scientist such as that of Stephen Hawking could have ended, hadn't it been for the technology that empowered him with the ability to share his thoughts. Stephen Hawking was not only able to help us understand physics better, he even wrote books and published his theories by controlling a single muscle, or a group of muscles that acted as a single input method.

It is both sad and alarming how such cases are becoming more common due to car accidents and different neurological disorders; even having a single case like this is way too many. This project has been inspired by the likes of Stephen Hawking and other human beings who have lost their basic right of expressing themselves and is planned to be tested with patients so that the benefit is maximized; we are currently in talks to bring this solution to a patient in Nova Scotia for initial testing. And what greater purpose can we achieve other than giving those who can't speak a voice, and empower people who have no means of expression or communication with the means to "live" and share their thoughts and emotions again?

While it is true that technology has improved drastically and that now we are able to communicate with such cases where only one input method is available, even when no muscles can be moved at all (i.e. using brain waves EEG [29]), we are concerned with the rate of typing that limits the user's means of communication. Stephen Hawking was able to type at an average rate of one to two word/s per minute until 2014 where SwiftKey prediction doubled the rate [30], that is a whole five minutes for a single short sentence. It is also worth mentioning that Hawking did not actually type the whole word; natural language processing algorithms are estimated to have

guessed the correct word after the second or third letter. The latest system that he used, which was developed by Intel, has been made open source [22]. Imagine how much more books and theories Hawking might have been able to potentially produce if he had the means to communicate faster.

### **1.3 OBJECTIVES**

The primary goal of this thesis is to design, implement, and test a sensor network platform called EMPWRD. Towards the realization of this goal, we have the following objectives.

First, it presents a framework along with an applicable solution that help users with severe motor disabilities express themselves and control devices within their surroundings.

Second, we aim to find an accurate and easy to use sensor that is also comfortable to be used for prolonged periods of time and provide a comparison of it alongside the existing sensor solutions.

Third, we attempt to provide a new method for auto-complete suggestions used in cluster ambiguous keyboards so that they cater for word corrections as well.

Fourth, we intend to address an input-adaptive grid scanner that functions differently as more inputs are identified.

Fifth, we present a binary-input calibration method that makes sure that the sensor is functioning properly and can be generalized to any binary-input sensor.

Sixth, we showcase a GSM-enabled module for the system that enables users to make phone calls.

Seventh, we conduct a comparative overview of the proposed framework and of “ACAT”, the system that was developed by Intel, and later integrated technologies from SwiftKey, specifically to cater for Stephen Hawking’s needs.

Overall, the presented solution integrates multiple innovative concepts and algorithms that enhance scanning cluster ambiguous keyboards, and binary-input-device position calibration methods.

## **1.4 OUTLINE**

Chapter 2 presents a literature review on “Input Methods for Mobility Challenged Users” and “Scanning Cluster Ambiguous Keyboards” and highlights the research gaps that we intend to address. Chapter 3 introduces our proposed framework “EMPWRD” along with a sample system that showcases it. Chapter 4 describes novel approaches regarding algorithms used in scanning cluster ambiguous keyboards, input-adaptive scanning grids in cluster ambiguous keyboard systems, and a binary-input calibration technique. A comparison between different cheek-muscle input solutions and a new proposed solution are given in Chapter 5 alongside a comparative overview of EMPWRD and ACAT systems. Chapter 6 gives the conclusions and suggests future work.

## CHAPTER 2 LITERATURE REVIEW

### 2.1 INPUT METHODS FOR MOBILITY CHALLENGED USERS

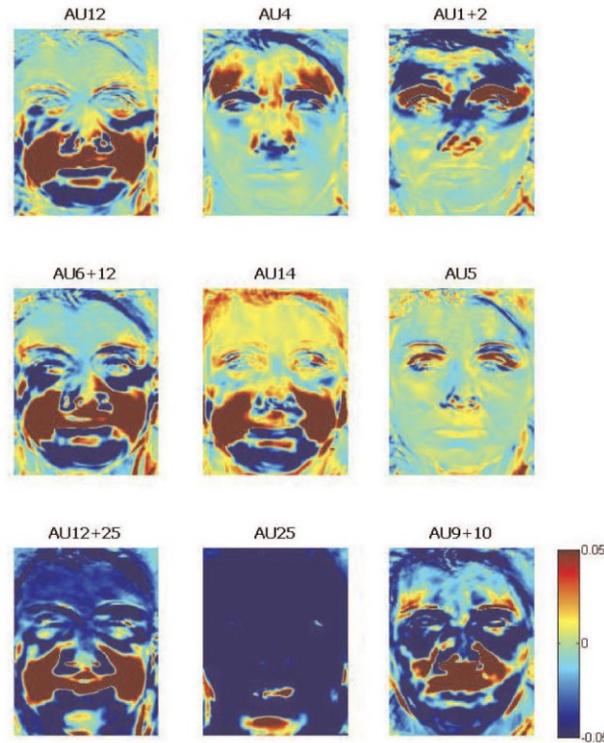
HCI (Human Computer Interaction) is the study of methods of interaction between humans and computers such that the application is as pleasant and easy to use as possible [31]; one very interesting application of HCI, is the recognition of facial expressions and hand gestures as a means for data input [32]. Facial expression recognition is done by using a camera and implementing computer vision algorithms [33]. Such systems usually yield very good and accurate results, exceeding 94% mean accuracy when machine learning algorithms are used to detect multiple expressions automatically [34]. Other similar used approaches are eye-ball movement tracking [35], blink detection [36], and eye-gaze detection [37].

As one may imagine however, a camera may pose a huge obstacle in such applications as it jeopardizes the user's privacy, and there could be legal issues preventing the use of a body worn camera [38].



*Figure 1 Facial Expression Detection Using a Camera*

Thermal imaging has also been used to detect facial muscle contraction [39]. Although it is more secure than facial expression detection, it has still been known to raise some privacy concerns [40].



*Figure 2 Thermal Image Examples*

Electromyography (EMG) is the technique of measuring the electrical activity of a skeletal muscle with respect to a reference [41]. EMG is a hot research topic as well when it comes to controlling devices based on HCI such as controlling a wheelchair using multichannel forehead bio-signals electromyography [42]. EMG can be used for controlling a prosthetic hand [43], for example, and can be used as an input switch by detecting facial muscle contractions as well [44]. However, EMG patches can be irritating for prolonged use; they also usually require gel and skin preparation before being applied [45].



Figure 3 EMG Facial Patches

It is worth mentioning that one of the most relied on technologies for facial muscle contraction and facial expression detection is optical detection and image processing [46] [47]; however, optical sensors do not necessarily mean a camera that captures a full image in order to be processed; other types of optical sensors include optical PANDA type ring resonators [48] and other infrared sensors [49].

A PANDA type ring resonator consists of an Add/Drop filter connected to two ring resonators, one sensing unit and one reference ring [50].

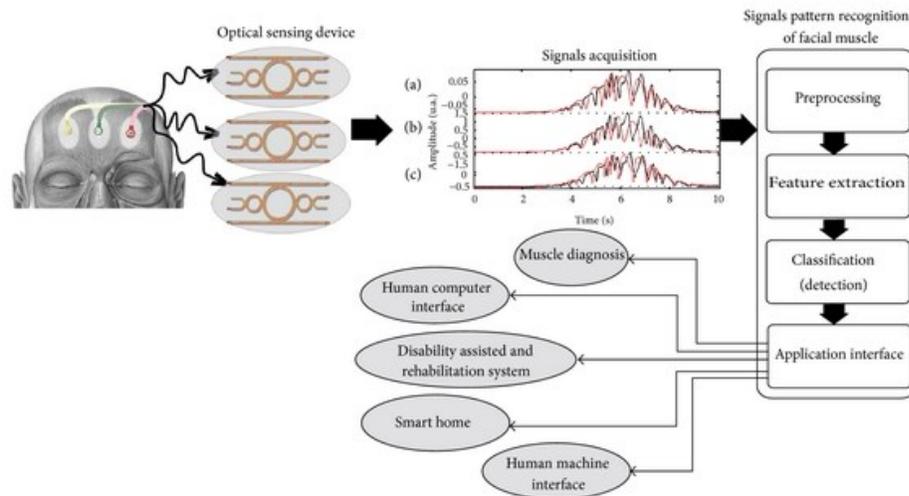
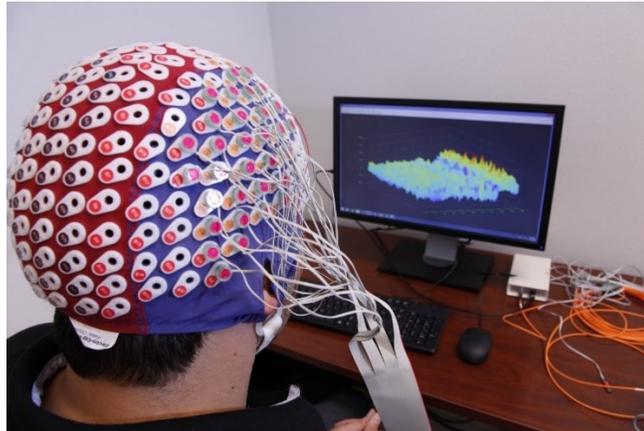


Figure 4 PANDA Type Ring Resonator

EEG (Electro-Encephalogram) and EOG (Electro-Oculogram) are also among the leading research topics when it comes to HCI, especially in application for patients with complete paralysis. EEG is the use of brainwaves in order to predict a specific

thought, a letter, or an action; there also exists solutions that utilize EEG for typing on a virtual keyboard [51]. Similar solutions that utilize EOG, which is the measurement in the change of electrical potential between the cornea and ocular fundus of the eye [52], instead also exist [53].



*Figure 5 EEG Headset*

Although new EEG headsets that do not require gel and that can detect some facial expressions as well with a minimal number of nodes are being developed, the accuracy and comfort are not the best yet. This is because wireless EEG headsets are still a science fiction; EEG also requires the user to train the system on the thoughts before it can be used. In addition, the typing rate using EEG only is less than a single character per minute which is too slow for users who can move at least one muscle [37].

The use of piezoelectric sensors has also been suggested in the literature [54]. A piezoelectric sensor converts a force to an electric signal [55]; the force may be the result of strain or pressure.

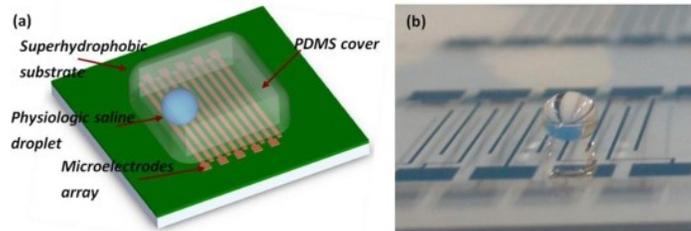


Figure 6 Sample Piezoelectric Sensor

Even utilizing breath pressure has recently been utilized, where a text-entry system called VIWA was developed to detect patterns in breathing pressure. The study of the system reported 99.8% accuracy and a typing rate of up to 7.9 words/minute [56].

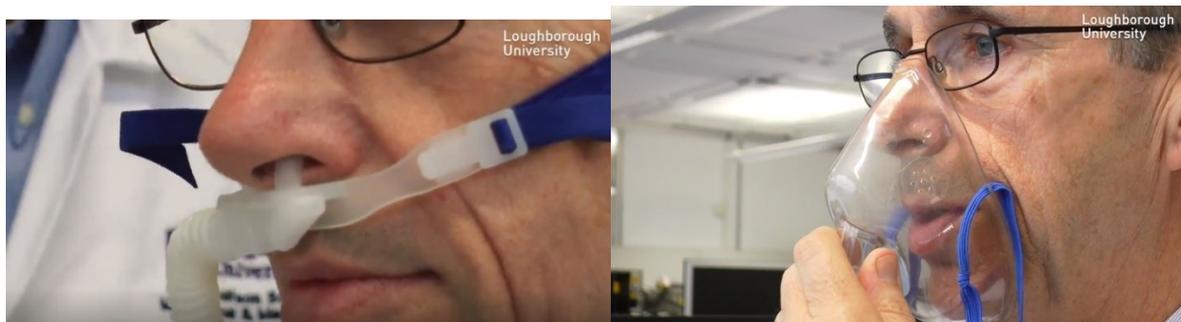


Figure 7 Breathing Pressure Sensors [57]

It is also important to note that multi-modality systems exist where an integration of multiple techniques is used. This typically means that each technique will output a quantifiable result, which is processed by a voting system that decides whether the final result is a valid voluntary input or not. Such systems include solutions that combine EEG and EOG [58], or EEG and Eye Blink Detection [37].

## Focus

In this thesis, we focus on the detection of specific facial muscle contractions to provide HCI solutions for users who have lost control over their body muscles and can only move some specific facial ones instead. Examples include ALS (amyotrophic lateral sclerosis), which afflicted Stephen Hawking, botulism, polio, stroke, cerebral

palsy, and neuropathy which is increasing in numbers due to diabetes and automobile injuries [59].

### Common solution

Throughout our research to find suitable sensors that satisfy the aforementioned objectives, we found that few papers address this research area. The widely used method is to use a reflectivity sensor which is most commonly packaged in what is known as TCRT5000 sensor (Figure 8 [23]). A similar sensor is used in the latest system that was developed for Stephen Hawking. To the best of our knowledge, the exact sensor components used in his solution are not published, but from the pictures and videos, such as the one shown in Figure 9 [60], we can speculate that it is a smaller version of TCRT5000.

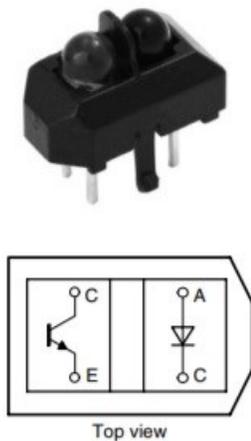


Figure 8 TCRT5000

Picture and Top View

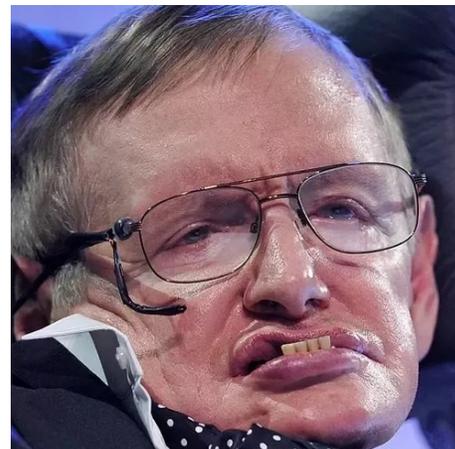


Figure 9 Stephen Hawking and

Cheek Movement Sensor

In short, the TCRT 5000 sensor is made up of two simple components, one infrared transmitter and one phototransistor in a leaded package that blocks visible light [23]. Depending on the surface that the sensor faces, the receiver would read different values, thus measuring the reflectivity of the surface.

## **2.2 SCANNING CLUSTER AMBIGUOUS KEYBOARDS**

### 2.2.1 Introduction to Cluster Ambiguous Keyboards

Text entry remains an important part of HCI with small devices. As devices used for text entry became smaller - such as the case with mobile phones - the need for smaller keyboards arose, and thus, methods for combining multiple letters into a single key were created.

Cluster Ambiguous Keyboards are keyboards that feature clusters of letters. Each cluster is associated with one key only. Thus, algorithms to determine which letter is intended by pressing a certain key have been well studied.

A dictionary based predictive disambiguation (DBPD) algorithm is often used for this purpose, such as the case of T9™ from Tegic Inc. [61] which groups multiple letters into a single key, where a key is pressed once for the desired letter, therefore significantly reducing the number of keys needed for typing any desired word. The keystroke sequence is matched to words in the dictionary. If there is more than one matching word, users will have to cycle through these words by pressing a “next” button.

Higher disambiguation means less “next” button key presses, which translates to less keystrokes required and, thus, less fatigue and time spent. Traditional methods only used word frequency to solve the problem of ambiguity. Boggess [62] presented two simple prediction algorithms, and Masui [63] utilized further developed techniques. Inverso et al [64] proposed a context-sensitive algorithm for word suggestions rather than mainly depending on the frequency of the word in a dictionary.



Figure 10 T9 Keyboard Layout

An algorithm that makes use of semantic analysis has also been proposed by Li et al. [65]. The algorithm suggests future words based on the contextual information of the text and is thus desirable for users with motor difficulties since it may save more keystrokes.

Syntactic analysis is also a well-known challenge in the field of natural language processing where POS tagging is desired. The Viterbi algorithm [66] is the most common n-gram approach for that purpose. For a word  $w$ , and a set of possible POS tags  $\{t_1 \dots t_n\}$  the Viterbi algorithm calculates the probability  $P_i$  of the word  $w$  having each POS tag  $t_i$ . Then the syntactic algorithm can be defined as follows:

$$POS(w) = \max(P_i) \quad (1)$$

Gong [67] utilized both the semantic and syntactical contexts in the preceding texts together for better disambiguation performance. Semantic analysis relies on the co-occurrence of words where a relatedness model is proposed. The model can be derived from a set of training corpora. The algorithm proposed for applying semantic analysis is as follows:

$$SEM(w) = \sum_i \frac{C(Stem(w), Stem(w_i))}{C(Stem(w_i))} \quad (2)$$

Where  $w$  is a candidate word and  $w_i$  are those words that precede  $w$ .  $C(Stem(w))$  is the frequency of the stem of word  $w$  in the training corpus.  $C(Stem(w), Stem(w_i))$  is

the frequency of both stems of words  $w$  and  $w_i$  exist in the same contexts in the training corpus. A sentence is an example of a context; hence,  $SEM(w)$  may refer to the co-occurrence relatedness of the word  $w$  based on all the preceding words. It is worth mentioning that the stem of a word can be derived by a computer stemmer [68].

Gong also proposes an improved disambiguation algorithm that orders the list of words proposed based on the overall score that is the result of all semantic, syntactic, and frequency analyses as follows [67]:

$$Overall(w) = \alpha * Freq(w) + \beta * SEM(w) + \gamma * POS(w) \quad (3)$$

where  $Freq(w)$  is the frequency score achieved from common DBPD methods and where  $0 \leq \alpha, \beta, \gamma \leq 1$  are the combination variables, whose ideal values can be found by incrementally modifying their combinations.

### 2.2.2 Letter Arrangements in Cluster Ambiguous Keyboards

A crucial decision in designing a keyboard is the number of keys (NKEY) featured in this keyboard, not to be confused with the arrangements of the letters bound to a single key [69]. Based on the theoretical model proposed by Soukoreff and MacKenzie [70], Fitts' Law [71] can be a good indicator of movement time whereas the Hickey-Hyman Law [72] can be utilized for predicting the visual scanning time.

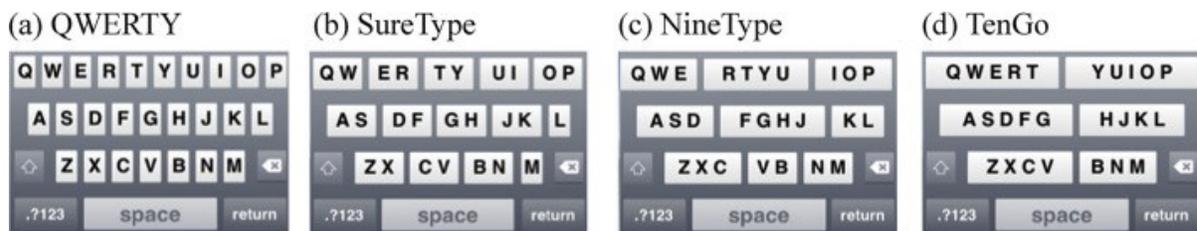


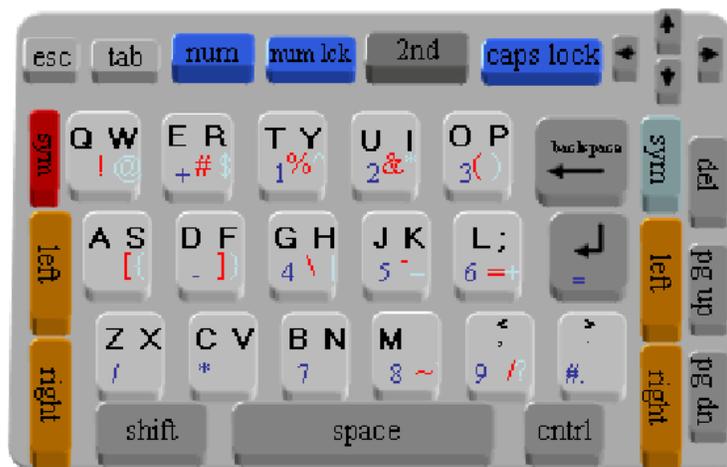
Figure 11 Full QWERTY Keyboard Next to Three Ambiguous Keyboards [73]

As we start grouping multiple letters and binding them to a single key, a very important question arises, how do we arrange the letters optimally?

Keyboards come in different sizes and may feature different layouts. MacKenzie and Tanaka-Ishii [69] categorized keyboard arrangements into three main groups: mobile phone keypad variants; QWERTY keypad variants; and fewer key keypad variants.

Huang and Wu [73] reviewed these categories and optimized them into the following categories: optimized; alphabetic-like; and QWERTY-like.

As for the optimized category, Levine [69] considered the letter arrangements and the optimization of disambiguation algorithms using dictionaries and statistics based on the size of the keypad. Other researchers proposed arrangements to maximize the expected accuracy of the disambiguation algorithm [74]–[76].



*Figure 12 QWERTY Layout in a Cluster Ambiguous Keyboard [5]*

Other cluster ambiguous keyboards have been designed based on the alphabet, by which the common keypad featured 12 buttons. Foulds et al. [77] proposed the TOC layout which switches the placements of three characters (t, o, and c). Less-Tap, the keyboard proposed by Pavlovych and Stuerzlinger [78], arranges the letters based on each letter's frequency, which means that the most frequent letter can be accessed by one key press, the second frequent letter bound to a key can be accessed by two key presses and so on. It is undeniable that the QWERTY layout has become very popular, as a result, many researchers have designed various ambiguous keyboards

based on the QWERTY arrangement such as EQ3 by Eatoni ([www.eatoni.com](http://www.eatoni.com)) and the QWERTY phone [69]. QWERTY-like cluster ambiguous keyboards have also been researched such as TenGo [79] which features the QWERTY layout over 6 keys only.



*Figure 13 Screenshot of OneKey Application with 4 keys [80]*

### 2.2.3 Scanning Keyboards

Depending on the severity of motor impairment, the user may not be able to control more than a single switch, which translates to binary input; this is where scanning keyboards come into the picture. Scanning keyboards cycle through a group of keys automatically based on timed interval, which can be as quick as 0.3 milliseconds per key [81], and only require the switch to be used once per desired selection. The interval can also be adaptive and adjust in runtime according to the user's performance [82]. Once a key is selected, the default action for the scanner is to go back to the home position and restart scanning; hence, it makes sense to arrange the letters that are most frequently used near the beginning position of the scanner in order to optimize the speed and hence the rate of input [83]–[86]. Another optimization

suggestion is the use of block scanning, where a bulk of keys is scanned and, once selected, individual key scan starts [87]–[89].

Since scanning grids are usually meant for people with severe motor impairments, they are usually bundled with cluster ambiguous keyboards, thus it is not intuitive how non-dictionary words can be formed. Several researchers have addressed this issue with potential solutions [81], [85], [88], [90], [91].

Figure 14 and Figure 15 show how scanners work as a function of time.

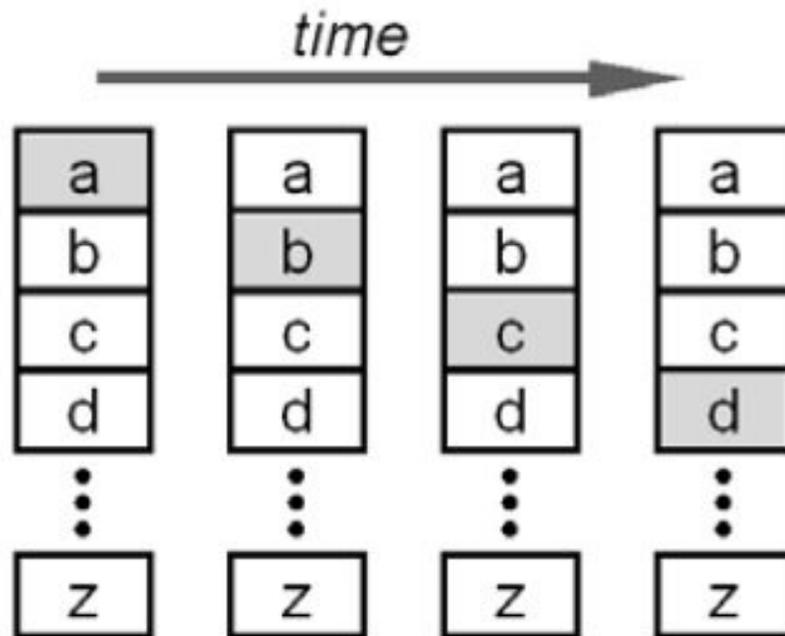


Figure 14 Linear Scanning Ambiguous Keyboard Concept [80]

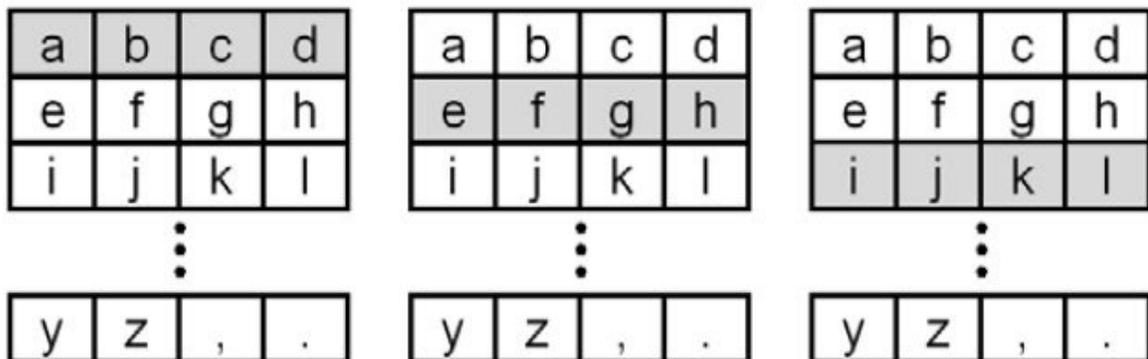


Figure 15 Row-based Scanning in Ambiguous Keyboard Concept [80]

Even though a lot of researchers have found that QWERTY-like keyboards yield better results than both optimized and alphabetic-like keyboards [92]–[95], it is worth noting that these studies do not refer to scanning keyboards; thus their results should not affect the letter arrangement decision of a scanning keyboard, since in such keyboards the user is limited to the speed of the scanner and is not affected by the size of the space occupied by a button.

Mackenzie and Felzer [80] presented characteristics and performance measures, as well as experimental results among different scanning keyboards.

It is worth mentioning that the language served by a certain scanning keyboard may affect its complexity, such as in the case of a language with too many phonetics (e.g. Chinese) [96].

#### 2.2.4 Keyboards for People with Motor Impairments

The problem of online communication is even more important for physically challenged users, such as motor or visually impaired users, because of their special needs. A lot of research have considered methods of text entry for people with motor disabilities [97], [98]. Such keyboards are meant to reduce movements required to form a word, and to increase the typing rate. It is also worth noting that most implementations for such purpose feature word prediction to save time and number of keystrokes [81], [90], [91], [99].

Dvorak [100], for example, places the vowels and most frequently used consonants in the middle row of the keyboard. This unique placement results in simpler finger movements and allows more comfortable input while minimizing physical stress.

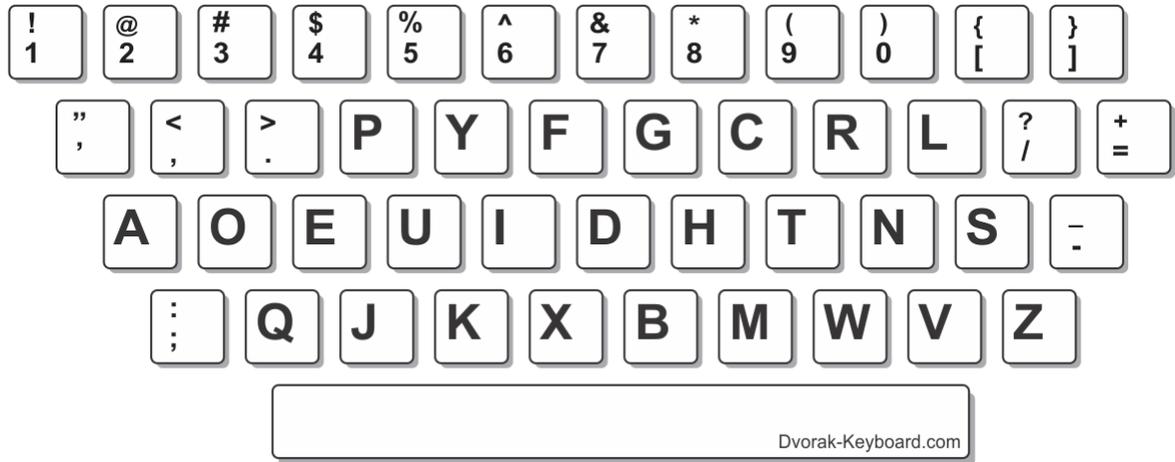


Figure 16 Dvorak Keyboard Layout [101]

XPeRT [102] is a keyboard that is very similar to the QWERTY layout. The idea is to group the most common letters in groups of two so that the distance traveled is smaller when typing text. Similar to the Dvorak principle, OPTI and FITALI [103] place the most frequent letters used in the middle of the keyboard, with two space bars on the sides and a repositioned larger shift bar in the bottom to maximize input rates; this is confirmed by Fitt's law [71] which states that the closer the letters, the faster and less tiring the input. Métropolis [104], as the name suggests, is built on the metropolis algorithm. This keyboard connects letters that are frequently used together such as "t", "h", and "e" - which form the word "the" - so that users can swipe through them instead of tapping each of them and thus saving time and energy.

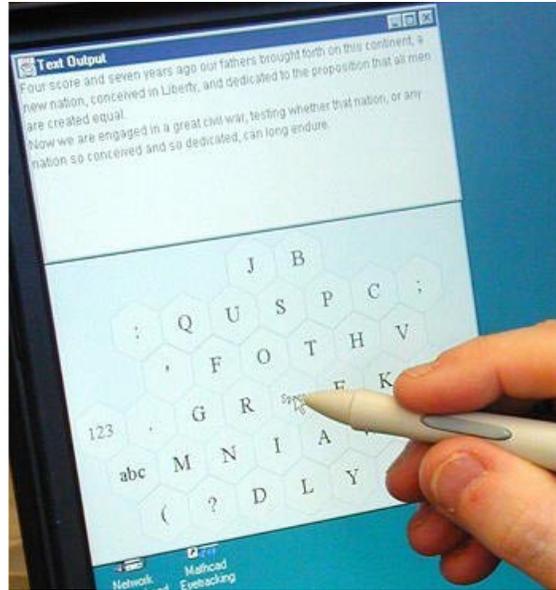


Figure 17 Metropolis Keyboard [104]

Sibylle [105] is a scanning keyboard designed specifically for single binary input where users cannot control more than a single switch.

Dasher [106] is a zooming interface where the user controls where they want to zoom in and when to click. Visual areas are bounded and divided into rectangles that are associated with specific letters so that the user does not have to be very precise about when to click.

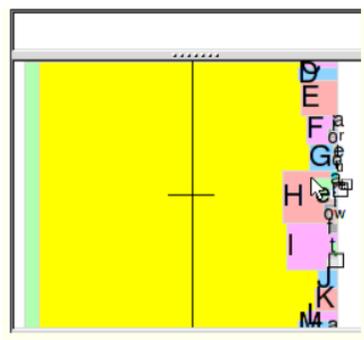
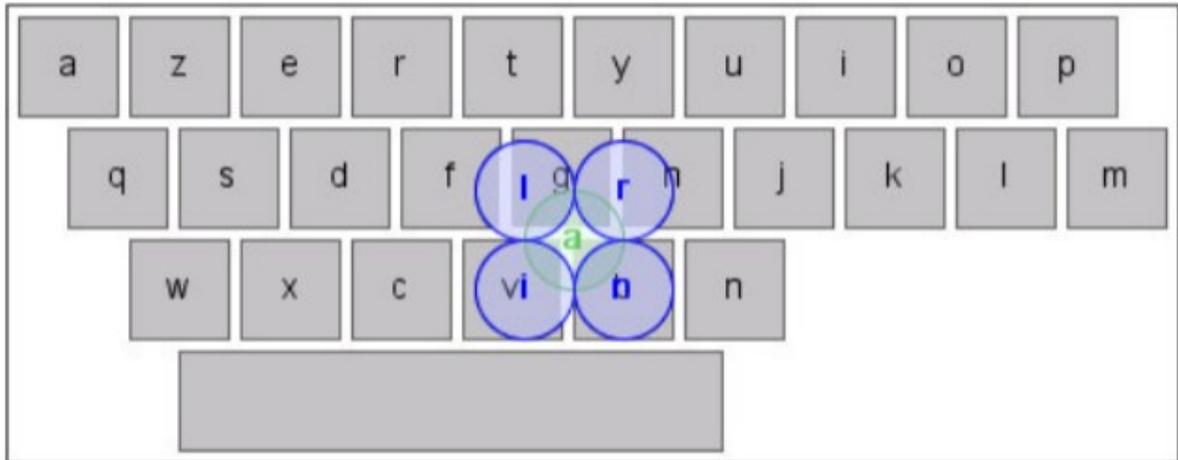


Figure 18 Dasher: The Zooming Interface [107]

KeyGlasses [108] creates four letter glass-like bubbles, based on the most likely letters to be desired next, around the selected key to save movement time and shorten distance travelled.



Chewing Word [109] is a row-based keyboard where the next letters are rearranged automatically based on user's input according to the maximum probability of occurrence.

Clavicom NG [110] utilizes the azerty layout where suggestions are created based on the user input and spaces are automatically inserted once a suggestion has been selected.

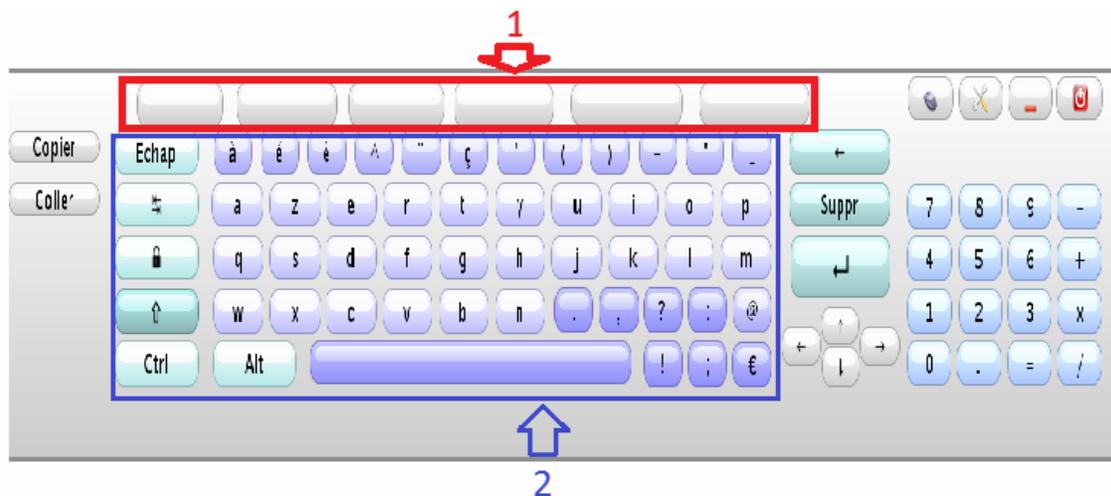


Figure 19 Clavicom NG: (1) word propositions, (2) AZERTY keyboard [110]

K-Hermes [110] is named K for "keyboard" and Hermes after the Greek god for sending messages. K-Hermes is T9-like keyboard. It has the advantage of enabling entry with only 9 keys, but requires pressing the same key multiple times to cycle through the letters associated with that specific key; for example, if the user desires to choose letter "b", they have to click the button labelled "a b c" twice. Propositions that are shown to the left are a set of dictionary words that correspond to the user input.

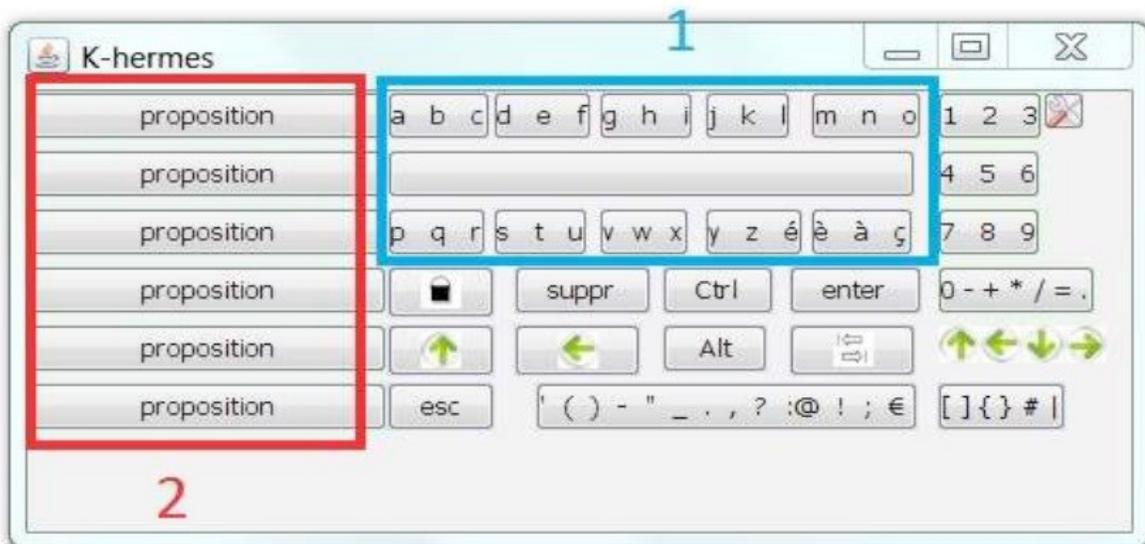


Figure 20 K-Hermes: (1) Word proposals, (2) 3-letters-per-key keyboard [110]

Hence choosing the right keyboard is a crucial element in such systems, especially that such movements required for typing can cause muscle stress if repeated several hundred times a day.

Multiple other solutions for people with motor impairment exist [111], [112]. It is also worth mentioning that toolkits for designing and evaluating different keyboards also exist [113].

For purposes of comparison we use ACAT, the system used by Stephen Hawking, since it was developed and improved by Intel and their partners over a very long period of time, and since Hawking could only control one input switch. Section 5.2 showcases ACAT along with a comparative analysis of ACAT and EMPWRD.

### **2.3 RESEARCH GAPS**

Although accessibility-focused solutions and scanning cluster ambiguous keyboards have been a hot topic over the past few decades, there is always space for improvements and novel approaches.

First, new sensors are being developed and released at a much faster rate nowadays than we used to witness before. Thus, new research regarding potential sensors for accessibility applications is required. We address this issue by proposing the use of APDS-9960 and providing a comparative analysis between existing sensors that are used for cheek muscle movement detection.

Second, the literature in the field of scanning cluster ambiguous keyboards focuses more on the algorithms rather than the usability of the existing solutions. Moreover, the literature in this field mainly tackles disambiguation and auto-complete algorithms, missing the possibility of integrating auto-correct algorithms as well; thus, we propose a method for suggesting corrected words as well as auto-completed ones.

Third, a lot of research has been conducted on scanning keyboards; however, the opportunities of multi-input adaptive scanning have not been tackled sufficiently in the literature; hence, we present a customized scanning solution that tackles multi-input customization.

Fourth, most of the literature focuses on sensors functionality after calibration, however, there is a need for a framework to calibrate and/or test that a binary-input sensor is working properly. We address that by providing a calibration framework that caters specifically for binary input methods, keeping in mind non-binary information availability such as signal strength.

## **CHAPTER 3      PROPOSED EMPWRD FRAMEWORK AND SYSTEM**

In this chapter, we propose a general framework for systems that address limited-input translation into text using a cluster ambiguous keyboard and potentially feature different accessibility options such as TTS and GSM phone calling.

### **3.1    FRAMEWORK CHARACTERISTICS AND REQUIREMENTS**

One would think that with all the technological advancements that we witness today, it would be relatively straightforward to provide a solution that uses some sensor data combined with NLP algorithms to cater for mobility challenged patients; however, after going through the literature review, and testing the different solutions on the market, it became evident that each solution either caters for a very specific type of patients, or optimizes a specific feature at the cost of missing other essential features. Thus, we have designed a framework that combines the essential features with the optimizations we have found either through our experimentation, or by going over the existing solutions. The following characteristics and requirements are the foundation of our framework:

a. Online and offline availability of accessibility systems

The main purpose of accessibility systems is to translate user input to text. The system should work offline as well as online (if necessary). The text may then be used in different ways such as getting converted to speech by utilizing text to speech technologies (TTS) and thus empowering users with a voice.

b. Powered by N-Grams or AI and an automatic scanning grid

Such systems should utilize an innovative combination of natural language processing algorithms and/or artificial intelligence, an automatic scanning grid for limited input(s), a dictionary, and a corpus to potentially speed up the average typing rate.

c. Continuous learning and template updating

Such systems should support template updating so that they become “smarter” with use and for new sentences to be added to the corpus. The user should also be able to add new words to the dictionary so that the user’s vocabulary is not limited to the words in the dictionary alone.

d. Modularity and compatibility

Such systems should be designed with modularity in mind. This is important so that new features can be easily integrated later on. Support of multiple input(s) - potentially from multiple types of sensors - is of utmost importance so that the same system can support the maximum potential number of users, even when their cases are not identical; for the same reason, maximum compatibility should be taken into consideration.

e. On-the-fly real-time configuration

Input-method-modules should be developed in a way that allows changing their configuration - such as thresholds – during runtime without the need to restart the module or, even worse, the whole system, just to recalibrate or change the settings.

f. Dynamic module availability and integration

Device control modules should be developed in a dynamic way where a variable number of devices can be used, and where a malfunction in a module should not affect the performance of other integrated modules.

g. Hardware and protocol standards compatibility

Compatibility may not always seem important, especially when it comes to biomedical applications; however, compatibility means easier integration and faster improvement of the system. Thus, the appliance-control-hub module for example, should be compatible with most if not all appliances and should not be limited to a single smart-communication protocol. Such compatibility may even be achieved through developing an additional module to translate

communications over different protocols. Hence, interconnecting multiple appliance control hubs that support different protocols may be possible.

h. Multi-user collaboration support

Corpora and dictionaries should be customizable and may be merged from different users for better results; this is decided by the remote server (API) and can only be done when the system is online.

i. Sensor criteria

Since the users are going to be using these sensors most of the time they are awake, and even possibly while sleeping to prevent the need for having someone help them to wear and recalibrate the sensors, our goal is threefold. First, the least intrusive sensors should be selected so that they would not jeopardize the user's privacy, nor cause any known security issues in case they get hacked. Second, sensors that do not need additional requirements -such as hydration using a saline solution - are preferred in order to prevent having to recalibrate them at specific intervals. Sensors that do not need to be in contact with the skin are also preferred so as to avoid technical issues especially in the cases of sweat, change in temperature, and difference in skin color. Third, accuracy should be among the most important aspects in sensor selection, and the ability to be calibrated by a caregiver after being worn by the user usually has a major impact on the accuracy as well. A sensor that does not require any calibration would be optimal.

### **3.2 EMPWRD SYSTEM COMPONENTS AND DESIGN**

In this section, we present the EMPWRD system that is based on the presented framework. As suggested by the framework, the system is designed in a completely modular manner where any module is independent of the others. Sample modules such as cheek-muscle-movement input and GSM phone calls are presented in this thesis.

### 3.2.1 Frontend and UI

The frontend is the interface that the user deals with and is composed of the following components:

#### 3.2.1.1 Start Screen

The Start Screen is considered a non-trivial component in the framework since it can take different forms. In the case of a paralyzed user who also cannot see clearly, the Start Screen may represent information for the caregiver, and audio instructions for the user for example. The common example is where the Start Screen serves visual purposes for the user directly where the initial page is loaded.

Figure 21 presents the sample Start Screen intended for users who can see clearly. Each device state is represented with an icon and a switch.

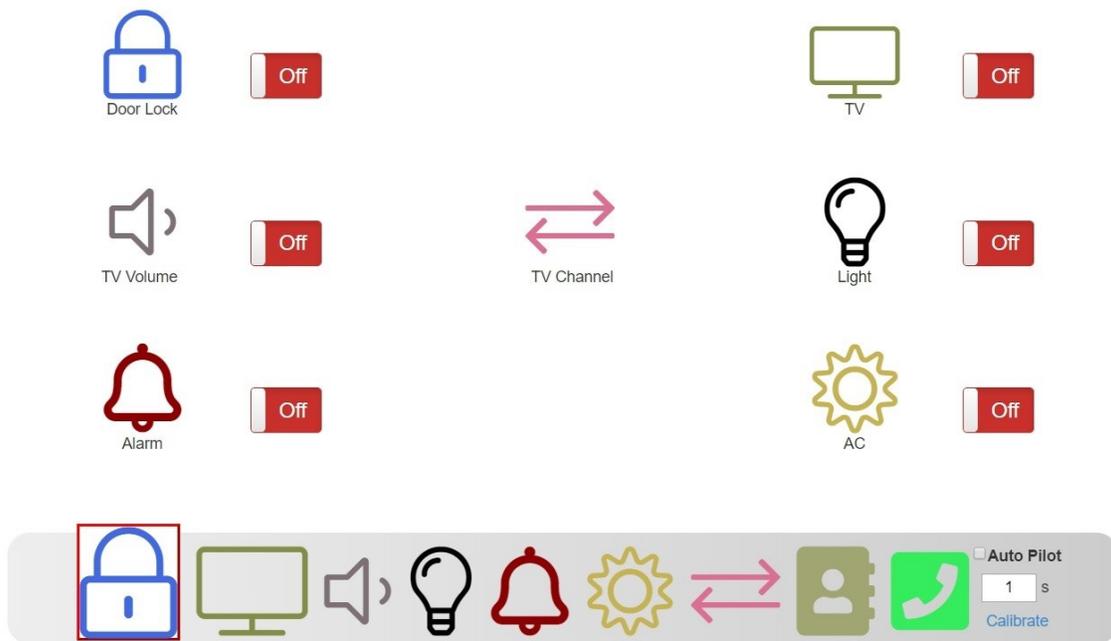


Figure 21 Device Control Page and Dock Controller

### 3.2.1.2 Dock Controller



*Figure 22 Dock Controller in Start Screen*

The dock controller ensures navigation through all sections of the system and acts as the user's interface for the device control module which empowers the user with the ability to control devices in their surroundings. In addition, it provides the caregiver with the option to calibrate the sensors.

### 3.2.1.3 Scanning Grid

As this system caters for more than just text entry, a scanning keyboard was not enough. The scanning grid is based on a time interval that may be predefined or adaptive to user capabilities in runtime (as shown in section 2.2.3), and it scans all buttons in the system, except for the sensor calibration button - since that requires assistance from another person and cannot be performed by the users themselves. The main components of the scanner are the dock and the different pages of the system. The scanning grid may also be input-adaptive, as discussed in section 4.2.

### 3.2.1.4 Input method(s)

An input method acts as a binary switch (based on any kind of sensor) and thus it can be modular and can be applied to any muscle or even to brain waves (EEG). A sample input method using TCRT5000 is shown in Figure 23.



*Figure 23 Input Sensor TCRT5000 Mounted on an Eye-Glasses Frame*

#### 3.2.1.5 Cluster ambiguous keyboard and word suggestion module

This is probably the most important part of the system. The text input module is the one responsible for utilizing all the other modules to translate user input into text. This module utilizes a scanning cluster ambiguous keyboard, and uses the dictionary and the corpus to predict desired words with the help of NLP algorithms as well as other custom algorithms; it also provides the functionality of text to speech (TTS) which actually empowers the user with a voice and is discussed in the following section. The letter arrangements of this keyboard as shown in the figures are optimized for bi-gram NLP algorithms according to Lesher et al. results [75].

#### 3.2.1.6 Text-to-speech (TTS) module

The TTS module is a component that not only allows text to be read loudly, but also represents the voice of the user when in a phone or an online call. This component shall be easily customizable, such as changing the volume and switching the voice, tone and pitch.

### 3.2.1.7 Phone Calls User Interface

The integration of the GSM module should be seamless to the user and, since the system features a scanning dock that switches between different elements, it is only natural that we integrate the GSM capability as an element that can be chosen right from the dock.

As shown in Figure 24, the user has to select the green phone icon from the dock in order to access the GSM functionality interface.

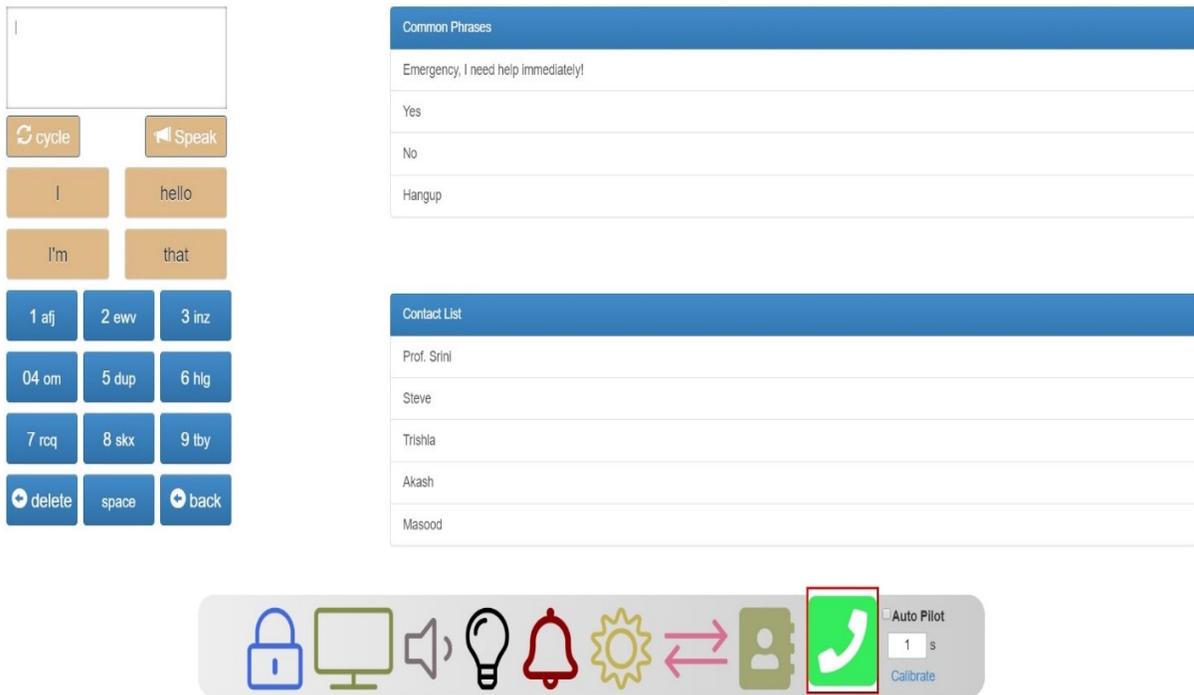


Figure 24 Phone Call Interface

The GSM call interface features three main elements:

a- Scanning ambiguous keyboard and word suggestion modules with TTS

These are the same modules presented in sections 3.2.1.5 and 3.2.1.6.

b- The common phrases component:

This component features dynamic frequently used phrases that help the user express themselves while on a live phone call without the need to type each word from scratch using the scanning keyboard.

c- The contact list:

This is the visual component that allows phone calls over GSM to be made. The user selects from a predefined list the contact that they wish to call. This selection panel is highlighted by the scanning grid as shown in Figure 25.

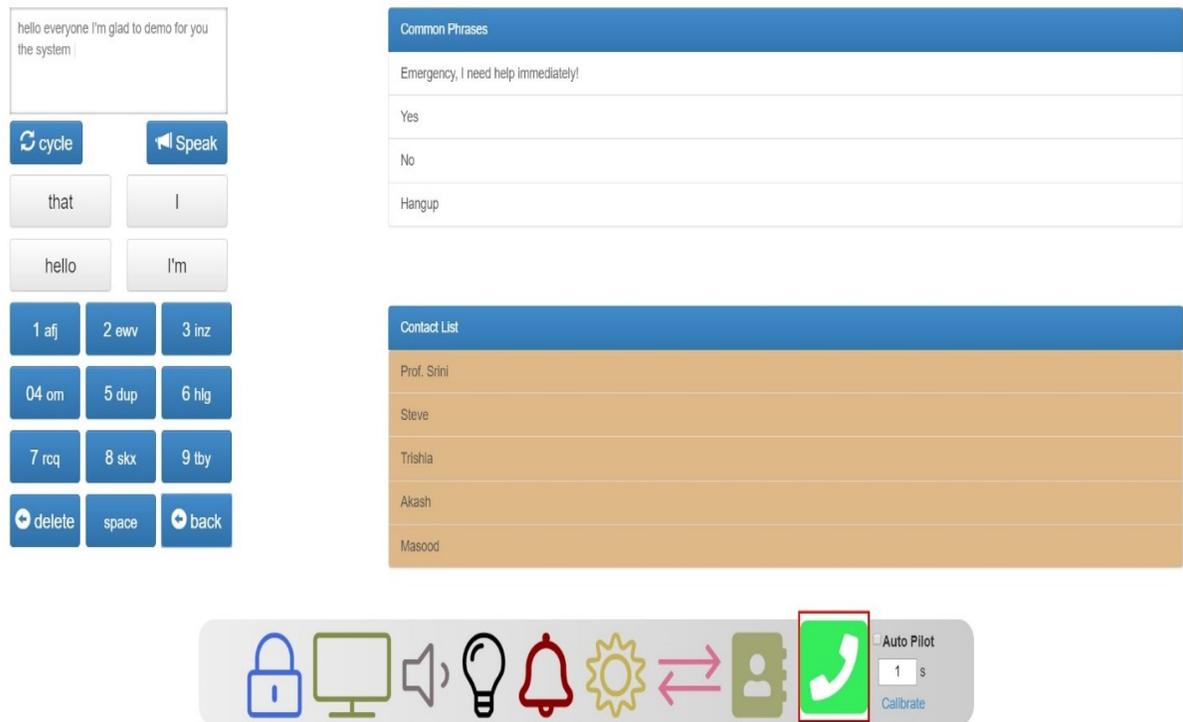
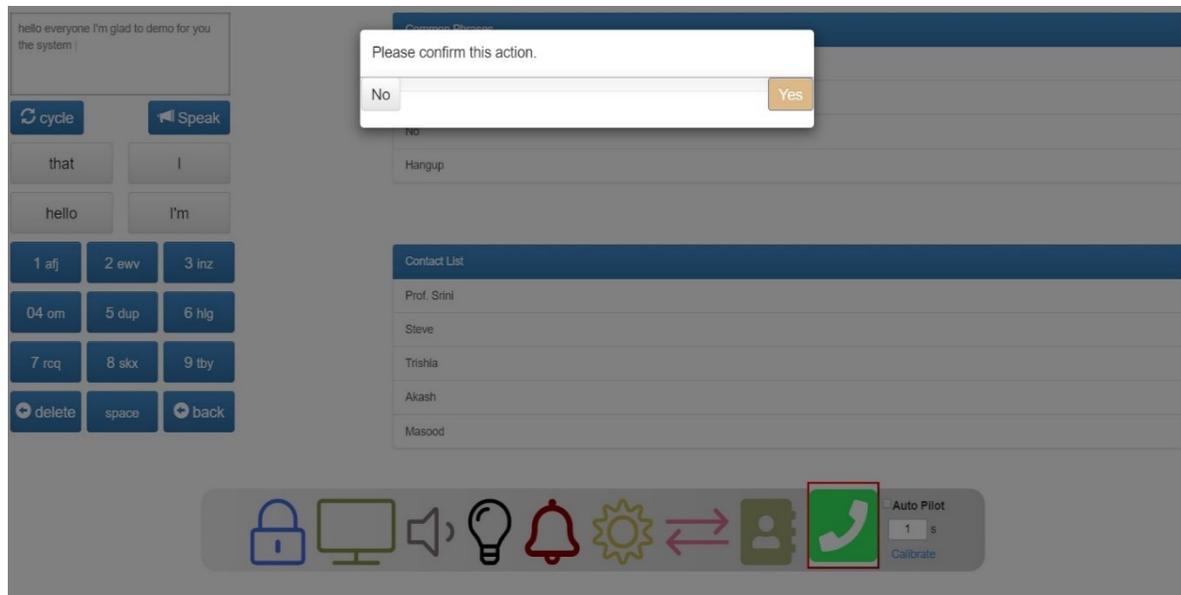


Figure 25 Contact Selection Panel

Once a contact is selected, the confirmation dialog appears so as to prevent accidental call making as shown in Figure 26.



*Figure 26 Call Initiation Confirmation Dialog*

Once the call is confirmed, a command is sent out to initiate the GSM call. The system then waits for the other end of the line to pick up; once the other end joins the call, the system starts by greeting the person with:” Hello [Contact Name], this is a pre-recorded message from [User Name]”. A beep is then heard and the text that is previously entered using the cluster ambiguous keyboard is transmitted after TTS is applied to it. Once the message is over, another beep is heard, and the system reads the following:” [Username] is now listening to you, you can stay on the line and communicate with him/her, or you can hang up”. Both ends are then in a live phone call where the user can either use the common phrases to express themselves or utilize the cluster ambiguous keyboard to speak different sentences using the power of TTS. The user can also hang-up on their end using a button that appears under the same collection of common phrases but requires confirmation before the call ends. It is also worth mentioning that it is possible to cater for voicemails in case the person on the other end does not join the call. In this case, a delay is added to the initial dialog until the beep is heard.



Figure 27 Common Phrases Panel

The modularity of the system also provides the possibility of choosing the mode of operation of the GSM module, or even set up multiple modes simultaneously. We are interested in two main modes for GSM operation: self-hosted and SaaS; a comparative overview is presented in section 3.3.

### 3.2.1.8 Input Calibration Module

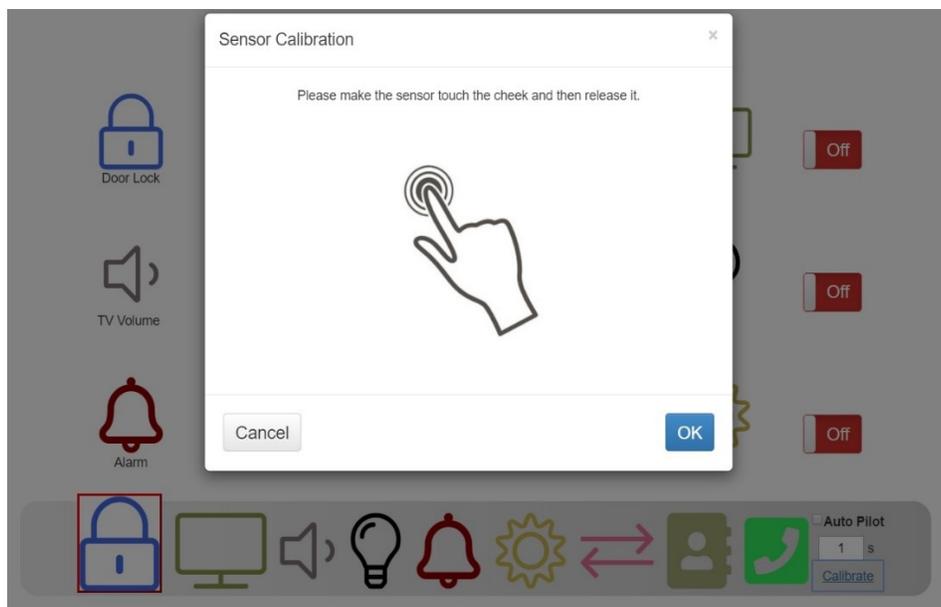


Figure 28 Input-Calibration Module Initial Screen

Some sensors may require calibration before each use (such as EMG), others may require hydration (such as EEG); calibration may also be in the form of optimal placement as in the case of TCRT5000 and APDS-9960. This module is meant to make sure that the sensor is calibrated and well-placed in such a way that ensures its good working condition. The specifics of this component are presented in section 4.3.

### 3.2.2 User-side Backend

#### 3.2.2.1 Dictionary

This module is used to predict the word that is being typed using the cluster ambiguous keyboard. It is hashed when the system initially starts. Typed numbers are then compared with the hashes to detect the words that match these numbers. More details about this module are presented in section 4.1.

#### 3.2.2.2 Corpus Controller

This module is used to predict the next word that the user is going to type before they start typing it. The algorithm is developed where it starts with the highest available n-gram (currently set to 3) and falls back until it reaches 0-gram if necessary (0-gram being the 1-gram of the dictionary words). This module also utilizes another algorithm that merges results from the predictions of the corpus with those of the dictionary to predict the word that is currently being typed.

The corpus is processed when the system initially starts in order to improve processing times in runtime. This module also supports template updating so that it can learn from the user as they utilize it more and more. Corpora from different users can be merged and used as deemed appropriate by the remote server (API).

A proposed algorithm is presented in section 4.1.

#### 3.2.2.3 Device control module

This module is used to provide control over devices surrounding the user. For example, the user can lock/unlock the door, turn on/off the TV or the AC, switch the TV channel, sound the emergency alarm, etc. This module may be compatible with any number of standard protocols and devices on the market.

Figure 29 presents the device control module prototype.

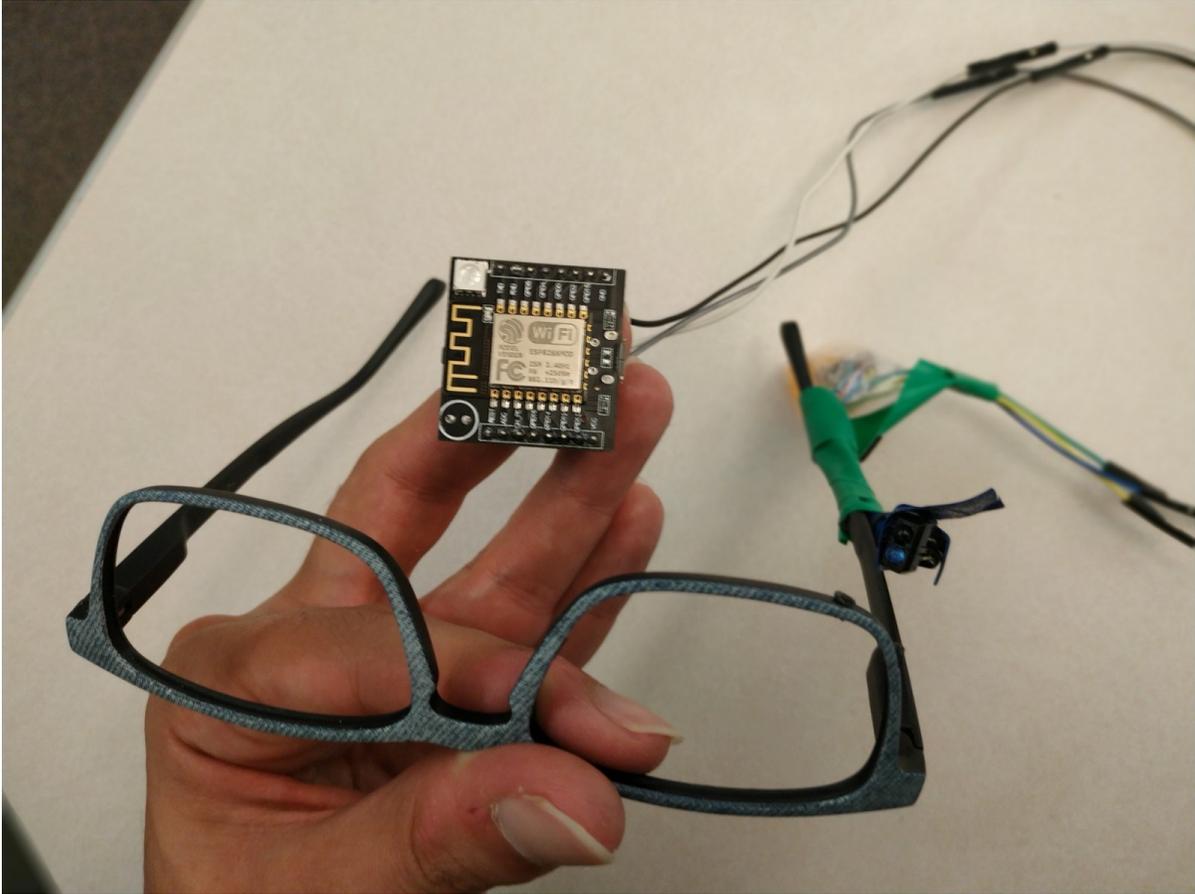


*Figure 29 Device Control Module: Microcontroller, and Controlled Socket*

#### 3.2.2.4 Microcontroller (Client)

The microcontroller on the client side is meant to configure, communicate with, and calibrate the methods of input.

In Figure 30, the client microcontroller is shown connected to the input sensor TCRT5000 and mounted on a frame:



*Figure 30 Microcontroller (Client) Board and TCRT5000*

#### 3.2.2.5 Microcontroller (local server)

The purpose of this microcontroller is not only to process the data received from the input methods and translate them to be used by the frontend, but also to access the software as a service (SaaS) in case that is desired, and to connect to an external display. This microcontroller can be replaced by a computer, but it is listed as a microcontroller since a PC can be an overkill for example.

#### 3.2.2.6 Transmitter

The transmitter can be connected to one or more methods of input and can transfer their data to a receiver over a communication medium.

### 3.2.2.7 Communication Medium

The communication medium must necessarily utilize a real-time communication protocol. Depending on whether the communication is going to be online or offline, the communication medium on the user's end may differ. For example, if Bluetooth is to be used, then the user's device must support Bluetooth and a Bluetooth-based communication module should be used. On the other hand, as in our case, if MQTT for example is used, then the user's device should be connected to the same network as the MQTT broker that is hosted in the server-side Backend as shown in section 3.2.3.1.

### 3.2.2.8 Receiver

The main purpose of the receiver is, as the name suggests, to receive the messages from the transmitter and categorize them based on the input method, especially that multiple methods of input may be connected to a single transmitter.

## 3.2.3 Server-side Backend

### 3.2.3.1 Communication medium

Whether communication is setup over a local network, or even over the Internet, the speed shall not affect the user's concentration or cause any distraction while using the system. As EMPWRD was developed with maximum compatibility in mind, we opted to use MQTT as our communication protocol. Not having Bluetooth as a requirement enabled us to create a portable system that can run on any browser-enabled device regardless of the operating system (e.g. Smart TV, Computer, TV Box, Phone, Tablet, or even gaming consoles, etc....). It is worth mentioning that using Bluetooth would probably yield better results than MQTT in terms of responsiveness to user input in cases of network congestion for example; in that case, a communication protocol on the server side is not going to be needed for the transfer of messages between the sensor and the system and the sensor must feature Bluetooth as well; however, if compatibility is required amidst an unstable Internet connection, then an MQTT broker can be setup locally without the need for an Internet connection; this is preferred even when an Internet connection is available because users may be relying on the system

heavily for everyday tasks, and interrupting the service may pose a huge risk on the users' health since the service is the means for them to ask for help and control their surrounding devices.

#### 3.2.3.2 GSM Module

It is hard to imagine a life with no telephone or mobile communications in our age. However, what value are telecommunications for people who have lost the ability to speak or – more generally- to communicate?

Part of this thesis is aimed at enabling such people with a voice; since the system is modular, it is possible to add GSM integration as a separate module. Through this module, it is possible to extend the mission to provide such telecommunication ability to people who are going to use the system. Section 3.3 presents a self-hosted GSM module as well as a quick comparison with SaaS GSM solutions.

#### 3.2.3.3 Remote Server (API)

This module is intended to serve as an API that interconnects and enhances dictionaries and NLP results by categorizing users and merging their information. It acts as a method that enhances AI results by providing additional information based on the choices of similar users.

This module may also collect data anonymously from users to enhance the system performance based on their usage.

### **3.3 SAMPLE INPUT DEVICE: WEBCAM**

As discussed in Chapter 2, there are many sensors that can be used as input devices to cater for accessibility needs. One of the most comfortable methods is the use of a camera-based solution such as facial expressions recognition [33] or eye-gaze detection [37].

In order to cater for input devices of different types, the system expects binary input states that can be bound to different events. An example of a binary input event is a

cheek muscle contraction, where the system receives an event that triggers the action bound to that event (e.g. Letter selection); a long-press can be triggered by implementing a timer on the input sensor end, where the system only receives the event trigger for a long press; this makes integration of any input device seamless since even non-binary devices can be used by specifying binary states based on input; Event-driven approach also allows the use of multiple sensors simultaneously, whether each sensor is responsible for specific triggers, or with multiple sensors triggering the same action.

The goal is for the system to be dynamic and modular enough to cater for each user's specific needs.

For demo purposes, we have chosen Affectiva's AFFDEX SDK [114] to integrate with the system. Figure 31 presents a sample webpage running the SDK.

The screenshot displays a web application interface for emotion tracking. On the left, a video feed shows a man with glasses and a beard, with numerous small white dots (facial markers) overlaid on his face. Below the video are three buttons: 'Start', 'Stop', and 'Reset'. To the right of the video, the 'EMOTION TRACKING RESULTS' section shows the following data:

```

EMOTION TRACKING RESULTS
Timestamp: 23.32
Number of faces found: 1
Appearance: {"gender":"Male","glasses":"Yes","age":"Under 18","ethnicity":"Caucasian"}
Emotions:
{"joy":0,"sadness":0,"disgust":0,"contempt":0,"anger":0,"fear":0,"surprise":0,"valence":0,"engagement":.3}
Expressions:
{"smile":.16,"innerBrowRaise":0,"browRaise":0,"browFurrow":0,"noseWrinkle":0,"upperLipRaise":0,"lipCornerDepressor":0,"chinRaise":.3,"lipPucker":0,"lipPress":.50,"lipSuck":.5,"mouthOpen":0,"smirk":.14,"eyeClosure":0,"attention":.96,"lidTighten":0,"jawDrop":0,"dimpler":.68,"eyeWiden":.3,"cheekRaise":0,"lipStretch":.28}
Emoji: 😊
    
```

Below this, the 'DETECTOR LOG MSGS' section shows:

```

DETECTOR LOG MSGS
Clicked the start button
Webcam access allowed
The detector reports initialized
    
```

Affectiva JS SDK CameraDetector to track different emotions.

**Instructions**

Press the start button to start the detector.  
 When a face is detected, the probabilities of the different emotions are written to the DOM.  
 Press the stop button to end the detector.

*Figure 31 Affectiva's AFFDEX SDK Implementation*

Once camera access is granted and processing starts, the SDK finds faces and draws markers over the video in real-time. Although Affectiva is famous for emotion detection using facial expressions, we are more interested in the facial expressions themselves such as “browRaise”, “mouthOpen”, “dimpler”, etc...

Since the SDK provides the “strength” of the detected feature, we need to implement a converter that triggers system events as if the sensor was binary, hence we apply a threshold for each feature depending on the user’s needs. Once the threshold requirements are met, we register the even in the system.

It is important to note that the user does not see what’s shown in Figure 31; instead, the user sees the regular frontend as shown in 3.2.1, and all the processing is done in the background.

We provide an overall comparison of different input methods in Chapter 5.

### **3.4 SAMPLE MODULE: SELF-HOSTED GSM**

Self-hosted GSM can be very expensive depending on the range provided, signal strength, hardware use, security measures taken, and maintenance, however, since we are only interested in using GSM as a client -initiating and receiving calls using a third party GSM provider as opposed to setting our own towers and access points- it is relatively cheap.

We have opted to use Raspberry Pie with AI Thinker A7 module in order to create a GSM client. The Raspberry Pi is not only used as a GSM client, but also as a web server that features an API which utilizes a real-time communication protocol for integration with other modules. The Raspberry Pi server is used so that we can control the AI Thinker A7 right from our user interface. However, although controlling the AI Thinker A7 board can be achieved without using the Raspberry Pi as a server that features an API, we are interested in that capability as it maximizes compatibility and modularity; it is also worth mentioning that, since the API is accessible over the web, the Raspberry Pi along with the AI Thinker A7 can be placed somewhere separate, and potentially far, from all other modules, as long as Internet connectivity is ensured.

In this thesis, we are using the latest Raspberry Pi as of the writing of this report, Raspberry Pi 3 Model B+, not only because we require its fast performance, but also because it is equipped with a Wi-Fi module out-of-the-box. This version features a

1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet support (with separate PoE HAT).



Figure 32 Raspberry Pi 3 Model B+ [115]

### 3.4.1 AI Thinker A7 Development Board:

Although the AI Thinker A7 chip features GPS, GPRS, and GSM, we are only interested in it for its GSM capabilities.

Below is the development board used for the AI Thinker A7 GSM module:

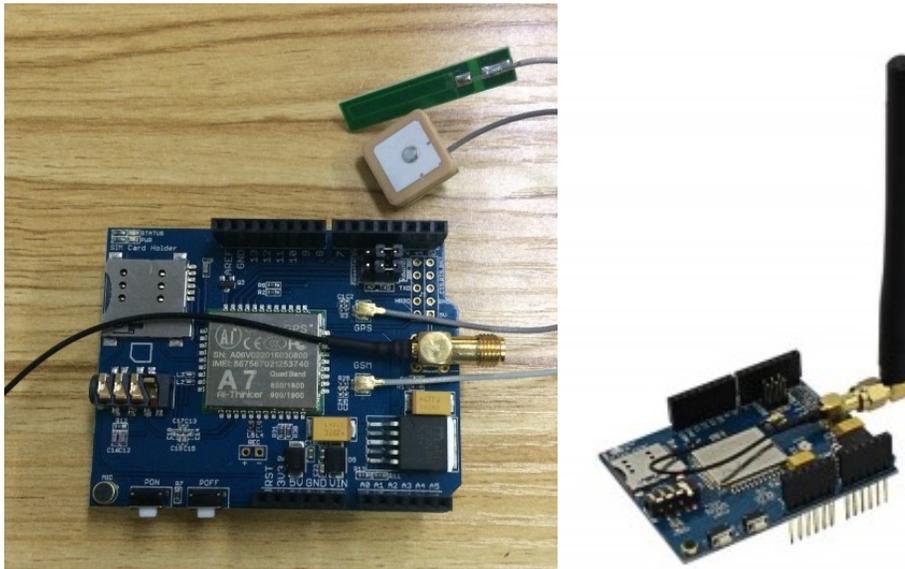


Figure 33 AI Thinker A7 GSM Development Board Used [1]

The AI Thinker A7 features a serial API in the form of AT commands.

Since we are only interested in this chip's GSM capabilities, we have interfaced the AT commands that are respective to GSM, ignoring other commands relative to features such as GPS, GPRS, and GSM.

The following is a list of the AT commands we are most interested in:

**AT:** this is the main attention command, returns OK if the device is ready. We also use this command to check whether the device has hanged and requires a reset.

**ATD[PhoneNumber]:** this is the command that initiates phone calls. It accepts the phone number in its international format (including the area code).

**ATH:** This is the command to hang up an ongoing call.

**ATA:** This is the answer command, where an incoming call in the "ringing" state can be answered.

Figure 34 shows some of the commonly used AT commands as well as the result of the status command "AT" as sent over the SSCOM Serial port tool:

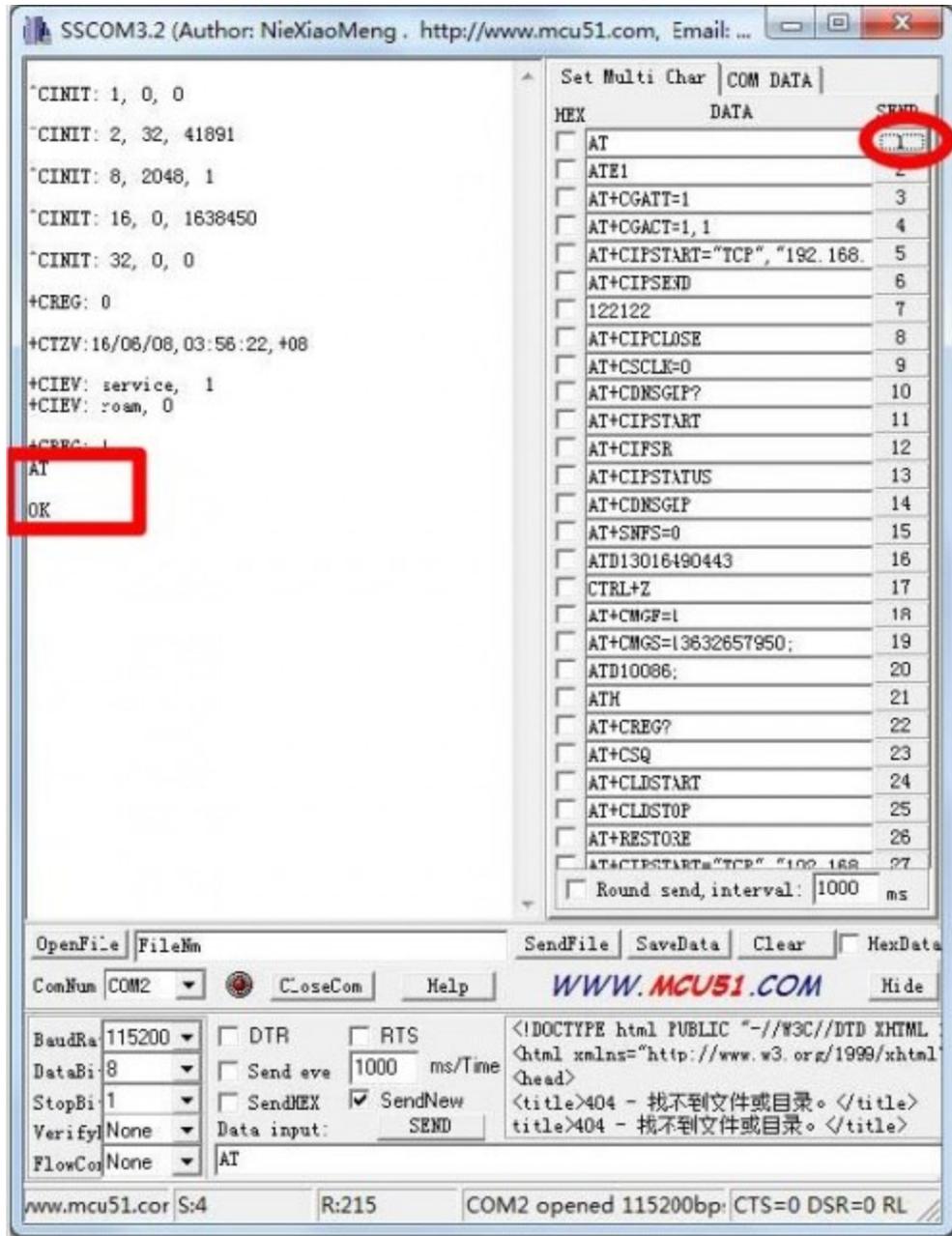


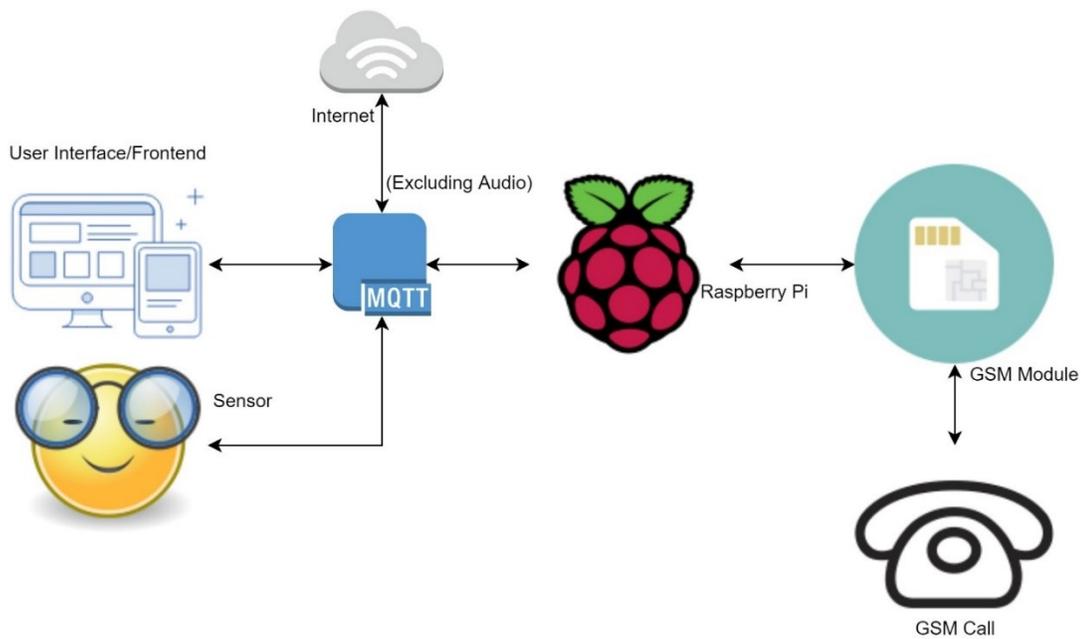
Figure 34 AT Commands for Testing A7 GSM Module Using SSCOM tool

As we can see in Figure 34, the default baud rate for communicating with the AT Thinker A7 is 115200. Although any serial port tool can be used, the SSCOM tool features quick commands, where the allowed commands can be loaded for easy access.

### 3.4.2 The Communication Protocol and NodeJS API

In order to achieve the desired module, it is essential to develop an asynchronous multi-channel API that can communicate with other modules and, at the same time, control the GSM module using GPIO and adjust accordingly to its feedback. Thus, we have chosen NodeJS to develop our server which can be accessed by other modules over the API that we have developed. Since we need multiple channels (e.g. 2-way communication with other modules and 2-way communication with GSM module), and since we are already using MQTT to connect different modules, we chose MQTT as the communication protocol to integrate this module as well.

Figure 35 represents the components of the system and shows how modularity is achieved:



*Figure 35 The Different Modules of The System*

Since we are using NodeJS as our server on the Raspberry Pi, it was easy to mirror the AI Thinker's A7 AT commands API with slight renaming as follows:

Table 1 Raspberry Pi NodeJS Server API

AI Thinker A7 Command	NodeJS API	Description
AT	at()	At, to check if the readiness of the device
ATD[PhoneNumber]	Call(PhoneNumber)	To call a phone number
ATH	Hangup()	To hang up an active call
ATA	Answer()	To Answer an incoming call
[CommandWithParameters]	Direct(Command)	To forward a direct command over the API to the GSM module
Reset Pin does not work as expected when the module hangs, so we introduce a separate watchdog.	Reset()	To reset the GSM module manually or by the watchdog (more details in part D)

### 3.4.3 Audio Configuration and Streaming (VOIP)

As shown in Figure 35, all modules are connected through MQTT, except for the AI Thinker A7 which is considered part of the same module that includes the Raspberry Pi and is connected serially to it;

Since the Raspberry Pi does not feature a microphone or audio-in out-of-the-box, we added an external USB sound card with microphone input to cater for the incoming voice from the GSM module (i.e. other end of the phone call).

The USB sound card that we used in our demo is shown below:



Figure 36 External USB Audio Card [116]

Other USB sound cards of higher quality exist, but we chose this one as it is one of the cheapest and does the job with sufficient results.

The audio configuration is shown in the following diagram:

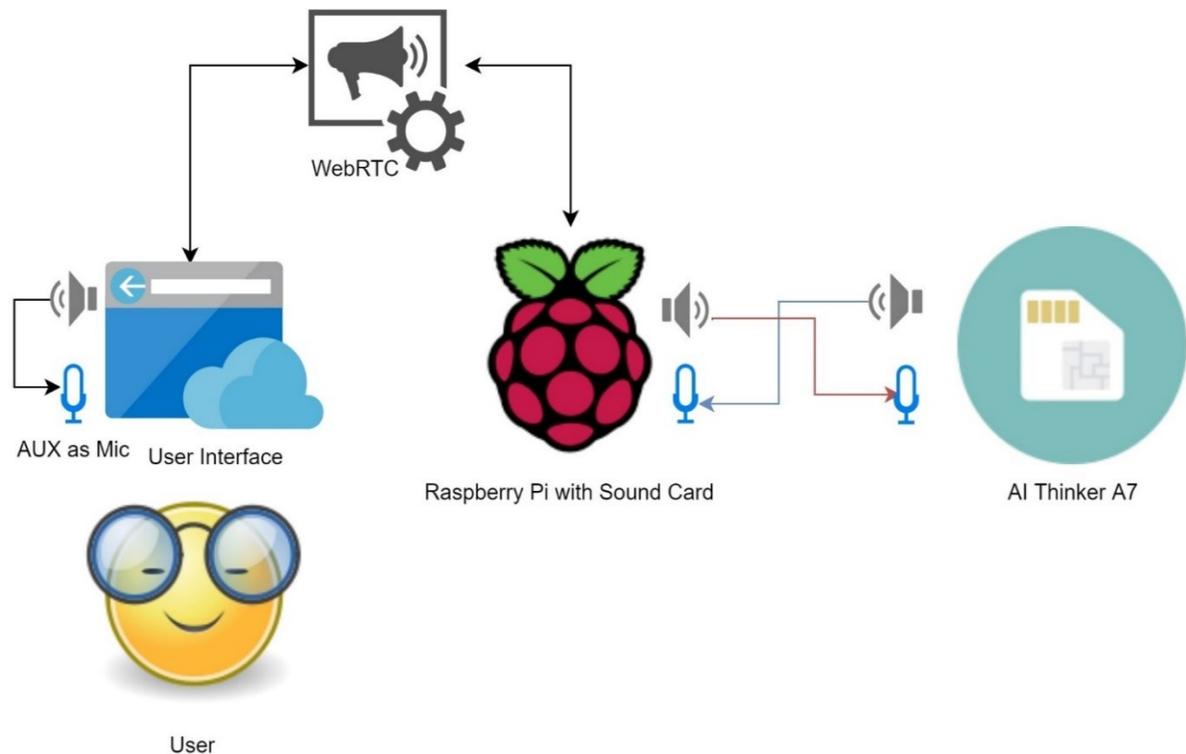


Figure 37 Audio Connections and Streaming Over the Web

With all the configuration done, the hardware setup, and the API ready, we still needed a way to transfer audio from the GSM module, over the Internet, to the user's machine. Although we're heavily relying on MQTT, which we are using as a real time communication protocol for communication between the different modules, MQTT is not a good fit for this purpose since it is optimized for small messages for IoT, not for real time audio streams. Our options were to either use a paid service such as Wowza Audio Streaming [117] or to use an implementation of the open-source WebRTC. We have opted to use WebRTC for this thesis, since it is free and fulfills our requirements perfectly.

We set the Raspberry Pi microphone input as the GSM audio output to be forwarded to the user UI; And we set the Raspberry Pi's AUX as the microphone input of the GSM module since AUX is the forwarded audio from the user's machine. Audio is transferred between the user's device and the Raspberry Pi using an implementation of WebRTC. In order for TTS to be forwarded from the user's device to the Raspberry Pi, we set the AUX as the microphone; this, however, introduces a new problem, where the person on the other end of the call will always hear the echo of their voice, since the voice is transferred from their headset to the user's AUX, which is the forwarded back to the other end. Although this is not a deal breaker, we propose two main solutions. First, we can programmatically switch the microphone on and off or (un)mute it only when TTS is in progress. Second, we can apply audio processing to eliminate the echo. We opted for the first approach where we control the sound level of the microphone on the user's device.

#### 3.4.4 GSM Module Watchdog

Based on our testing, we have found that the AI Thinker A7 Development board may become unresponsive with time, and that resetting it using the built-in reset pin does not always yield the desired result. Hence, we have integrated an additional relay in the form of an external circuit so that we can control the power flow to the GSM module. In addition, we have implemented, as part of the software hosted on the Raspberry Pi, a watchdog that controls this relay so that we can ensure continuous operation. Figure 38 shows the relay module that we have chosen for our demo, as

we needed a 3.3v compatible relay because that's the power limit of the Raspberry Pi's GPIO pins. Otherwise, we would have needed to integrate more external circuitry to cater for that.

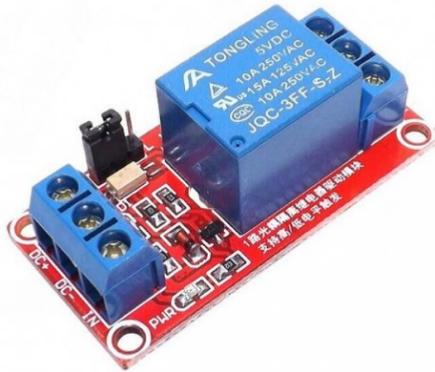


Figure 38 Relay Module Used for Resetting GSM Module [118]

Although we have developed an automatic watchdog, we also extended the API that we have developed for the GSM module to cater for manually but remotely resetting the AI Thinker A7 for maximum compatibility. Figure 39 explains how the manual reset command works.

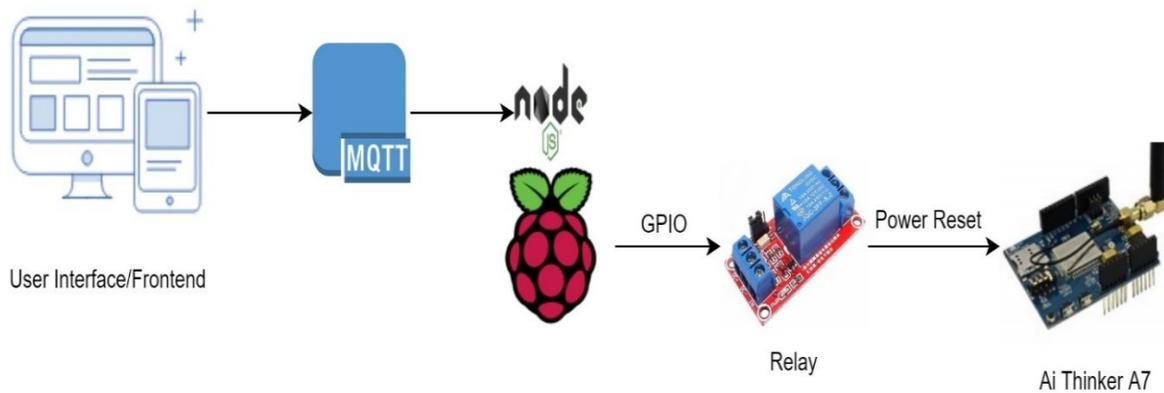


Figure 39 Reset Command from Issuing to Execution

### 3.4.5 Overall System and Integration

The overall system with the self-hosted GSM module utilizes two main communication protocols -namely WebRTC for audio streaming and MQTT for IoT and small messages- a computer (the Raspberry Pi ), a relay as an external circuit that acts as a watchdog, and a GSM chip – the AI Thinker A7 Development Board in our case. Figure 40 shows the overall system components including the communication protocols.

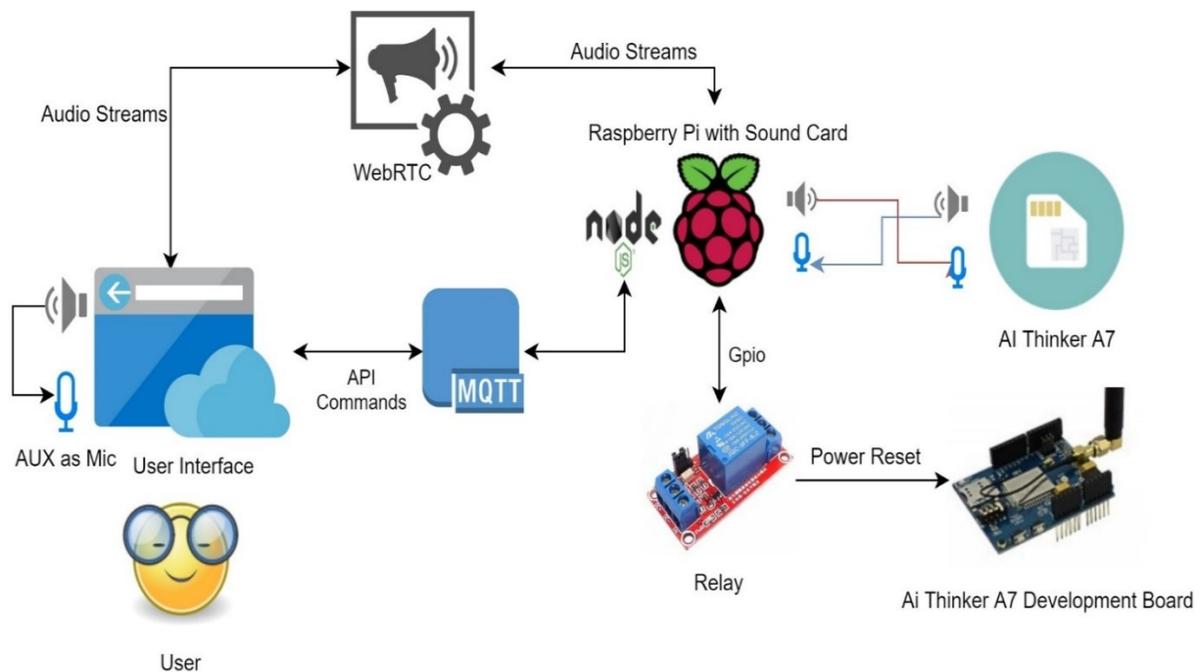


Figure 40 Overall System Components and Communication Flow

Although power failure is very rare, we have configured the Raspberry Pi to run the relevant services and scripts on startup, just in case any emergency or power outage occurs; even though a an uninterruptible power supply (UPS) should be put in place so as to cater for that without causing any damage to any component.

Figure 41 shows the final GSM module -without the power supply- used in our demo. We removed the cover of the USB sound card and soldered the audio connections to

the PCB directly. We also placed the Raspberry Pi in a transparent Acrylic cover and fixed the relay and the AI Thinker A7 Development board on top of it.



*Figure 41 Self-Hosted GSM Module Used in Demo*

#### 3.4.6 Other Considerations and SaaS Solutions

Since the proposed framework is modular, such GSM module can be replaced with a SaaS solution instead. While SaaS implementation is going to cost more (usually pay per minute as opposed to the local GSM service provider), hardware maintenance is almost negligible (based on the service's reputation and guaranteed uptime) as opposed to the self-hosted one. Moreover, signal strength and power considerations are not to be taken into consideration in the case of SaaS implementations. However, a self-hosted module may improve privacy, as in the case of managed communication encryption between the user's microcontroller and the GSM module. In addition, a self-hosted GSM module will work even when an Internet connection is not available and will not require changes in terms of software on the long run, whereas the SaaS solution provider may require changes due to API updates.

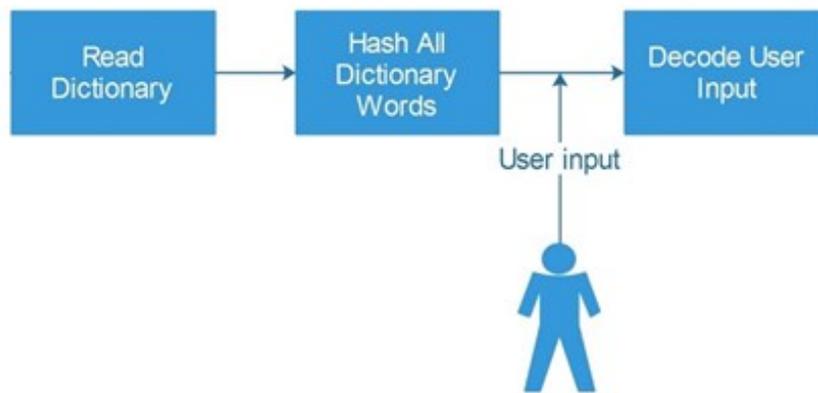
*Table 2 Comparison Between Self-Hosted and SaaS GSM*

	Self-Hosted	SaaS
Cost (Long-term)	Lower	Higher
Setup	Hardware and software	Only Software
Maintenance	Hardware only	Software only
Signal Strength	Must be taken into consideration	Dependent on Internet connectivity
Internet Connection	Not required	Required
Power Consumption	Low	N/A
Privacy and Encryption	In-house encryption	Dependent on service provider

## CHAPTER 4 NOVEL ALGORITHMS FOR THE DESIGN

### 4.1 AUTO-CORRECT ALGORITHM FOR CLUSTER AMBIGUOUS KEYBOARDS

In this section, we present our implementation of a scanning cluster ambiguous keyboard, along with the methodology used in creating the dictionary, implementing the disambiguation algorithm, and creating the list of suggested words. Figure 42 shows the process of creating a list of suggested words for a simple cluster ambiguous keyboard.



*Figure 42 Cluster Ambiguous Keyboards' Hashing Algorithm*

The process starts by reading a dictionary of all words that can be suggested. It is worth noting that the dictionary may contain a word in different forms, as opposed to the corpus used by the disambiguation algorithm where the stems of the words are used. All words in this dictionary are then hashed according to the number of the key that each letter is bound to. For example, consider the cluster ambiguous keyboard where the first two keys are bound to: {a,b,c} and {d,e,f} respectively. The word “add” would be hashed into “122” where 1 is the number of the key that ‘a’ is bound to, and 2 is the number of key that “d” is bound to.

Figure 43 presents an example of a hashed dictionary.

```
▶ 764537: (2) ["smiler", "smiles"]
▶ 764687: ["points"]
▶ 764689: ["pointy"]
▶ 764733: (2) ["poised", "soiree"]
▶ 764737: (3) ["poises", "sniper", "snipes"]
▶ 764757: ["smirks"]
▶ 764766: (2) ["pogrom", "poison"]
▶ 764835: ["snivel"]
▶ 764837: ["rogues"]
▶ 764847: ["smiths"]
▶ 764849: ["smithy"]
▶ 765223: ["solace"]
▶ 765263: ["poland"]
▶ 765337: (2) ["polder", "solder"]
```

*Figure 43 Hashed Dictionary for Cluster Ambiguous Keyboard*

As we can see in Figure 43, a hash can represent one or more words; this is where the disambiguation algorithm comes into the picture. In this section, we are not interested in a specific disambiguation algorithm, nor in finding the best NLP solution; we are interested in proposing a novel component in scanning cluster ambiguous keyboards, which is text correction.

While the disambiguation algorithm and the models for suggesting the desired words accurately are of utmost importance, it is also very important to take into consideration the mistakes that a user might make while selecting letters, especially when scanning is based on a time interval.

Since the user selects a button through a binary-input method, it is possible for the response to be delayed and, thus, the wrong button to be selected. This kind of mistakes does not occur in keyboards where physical contact with a button is necessary (e.g. physical and touch-screen keyboards). Such mistakes in text entry costs the user a lot of effort and time. And since the user cannot see the portion of the word that they have entered, the hash that is not disambiguated yet is of no use to them.

It is also worth noting that the suggestions present at this step are inaccurate because they are the result of the disambiguation of a mistyped hash. A very important question comes to mind, how do we fix such mistakes?

It is not very surprising to find that a lot of scanning ambiguous keyboards in the literature do not offer a “backspace” or “delete” button, since most of them are meant to test and analyze the results of different optimizations such as enhanced algorithms or letter arrangements. Chanti [119] for example, features a button to delete the last sequence as shown in Figure 44. We are inclined to follow their example because we think that it is not very helpful to delete the last key since the whole word only lives in the user’s mind, and it is not possible to show them the portion that they have typed so far. We have, however, added comparing these two options to the future work suggestions in Chapter 6.

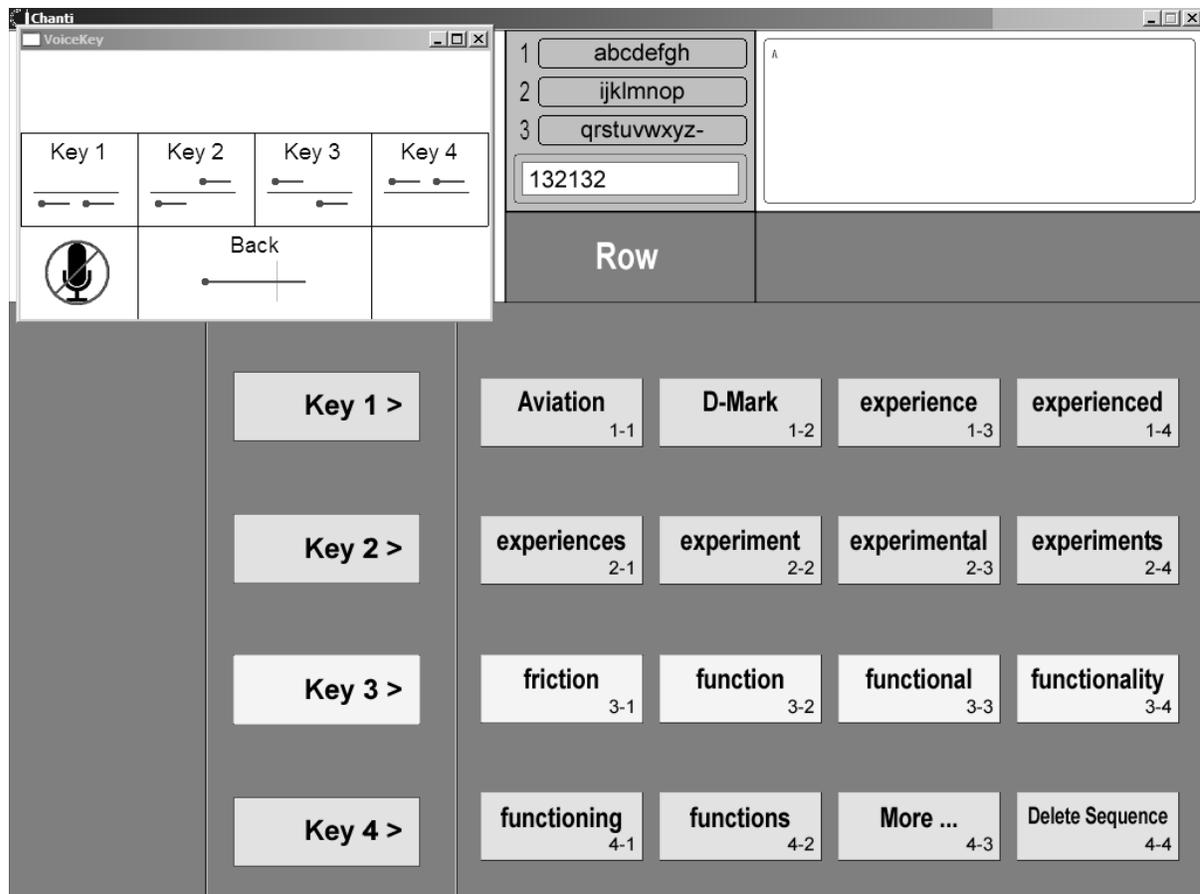


Figure 44 Chanti Software Featuring a Delete Sequence Button

Before we present our suggested method for implementing both, a disambiguation algorithm and a text correction algorithm in the same solution, we need to present a sample algorithm as an example. This sample algorithm will then be used to showcase the full algorithm that combines both disambiguation and text-correction functionalities.

```

▼ 2: Array(12)
  ▶ 0: {word: "I would like", count: 2, size: 3}
  ▶ 1: {word: "would like to", count: 2, size: 3}
  ▶ 2: {word: "to talk to", count: 1, size: 3}
  ▶ 3: {word: "$new$ Hello there!", count: 1, size: 3}
  ▶ 4: {word: "there! I would", count: 1, size: 3}
  ▶ 5: {word: "like to talk", count: 1, size: 3}
  ▶ 6: {word: "Hello there! I", count: 1, size: 3}
  ▶ 7: {word: "talk to you", count: 1, size: 3}
  ▶ 8: {word: "to you soon!", count: 1, size: 3}
  ▶ 9: {word: "you soon! $new$", count: 1, size: 3}
  ▶ 10: {word: "soon! $new$ I", count: 1, size: 3}
  ▶ 11: {word: "$new$ I would", count: 1, size: 3}
  length: 12

```

Figure 45 3-Gram Dictionary with New Sentence Identifiers

Figure 45 shows the result of tri-gram NLP algorithm. The simple algorithm without smoothing or optimizations [120], [121] can be represented by the following equation [122]:

$$P(w_i | w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})} \quad (4)$$

Where  $w_{i-2}$  is the word preceding the current word by two positions,  $w_{i-1}$  being the word preceding the current one, and  $w_i$  being the current word. The word with the highest probability is suggested first.

Example: The fox jumped.

$$P(w_{jumped} | w_{the}, w_{fox}) = \frac{\text{Count}(w_{the}, w_{fox}, w_{jumped})}{\text{Count}(w_{the}, w_{fox})} \quad (5)$$

In our system, we are using tri-grams with fall back to bi-grams and uni-grams; all words that exist in the cluster ambiguous keyboard's dictionary are added to the uni-gram corpus to avoid the case where words may exist in the former but not the latter. We are also using Laplace smoothing which can be represented as follows for trigram smoothing [123]:

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i) + 1}{\text{Count}(w_{i-2}, w_{i-1}) + v} \quad (6)$$

Where  $v$  stands for vocabulary size and represents all the words considered.

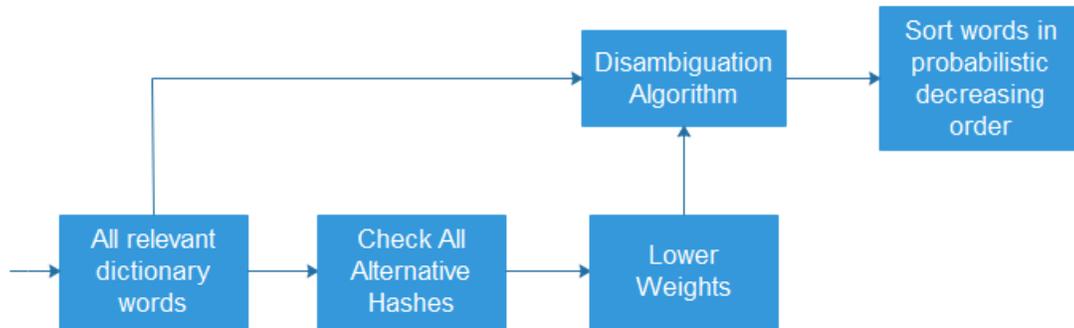
The flow of such algorithm in traditional systems is shown in Figure 46.



*Figure 46 Traditional Flow of NLP in Ambiguous Keyboards*

The user inputs some text (using numbered keys), all candidate words are processed according to the NLP algorithms implemented (N-Grams with smoothing and fallback in our case), the user then is presented with the most probable words as suggestions.

We propose a simple text correction algorithm as shown in Figure 47:



*Figure 47 Text Correction Component*

After getting all the words that are relevant to the hash entered by the user, a list of all alternative hashes is created. Alternative hashes are then down-weighted in order to keep the actual suggestions on top. All hashes are then processed by N-Grams keeping in mind the down-weighted values. Alternative hashes may simply be the combination of single decrements of each digit of the hash by one. This is based on the assumption that most mistakes are due to the delay in activating the binary input.

Example: Hash: 234

Alternative hashes: 134, 224, 233, 124, 223, 133, and 123

Since this method can add a lot of processing, especially in the case of long words, an optimized approach can be used. Suggested computation optimizations are as follows:

- 1- Setting a mistake-threshold: in this case, the researcher assumes that the number of mistakes cannot exceed a specific number so as to limit the computation power required according to the microcontroller being used on the user's end. So, for the above example, and for a threshold of 1 mistake, the possible alternative hashes become: 134, 224, and 233 only.
- 2- Create another dictionary of words and/or keys that the user has deleted before. Example: user enters 123, then deletes the word. The word 123 is added to the dictionary of mistakes and the last key "3" is added to the corpus of mistakes. The most probable keys to be hit by mistake are then considered

for alternatives. This is based on the assumption that the source of mistakes is actually visual, and probably not reflexive. Even though our intuition says that this method is very unlikely to yield good results, only experimental analysis (suggested in future work in chapter 6) can prove what works well and what does not.

The overall proposed probabilistic equation may be presented as follows:

$$W_t = W \left( \sum_j \alpha * P(w_j | w_{j-2}, w_{j-1}) \right) \cup W \left( \sum_i \beta * P(w_i | w_{i-2}, w_{i-1}) \right) \quad (7)$$

where  $W_t$  represents all the resulting suggestions;  $W_j$  represents words whose source is alternative hashes, whereas  $W_i$  represents words whose source is the original input hash, and  $\alpha$  and  $\beta$  are weights applied to words according to their source.

Figure 48 presents the overall flow diagram.

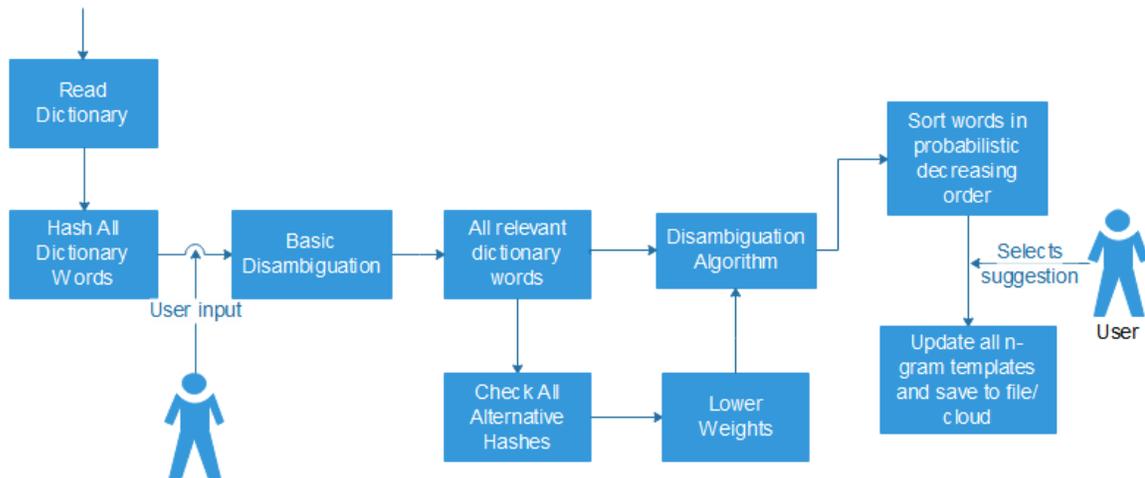


Figure 48 Overview of Proposed Text Correction Algorithm

## 4.2 INPUT-ADAPTIVE SCANNING GRID

ACAT features mapping different input methods to different shortcuts [22]. However, the number of inputs cannot be configured to change the flow of the scanner in ACAT.

The idea is to have a scanner that changes its behavior according to the number of inputs in order to maximize the typing rate.

### Auto-pilot mode:

This mode is intended to be used where only one or two methods of input are available. The auto-pilot mode allows a user to select the needed button while the system navigates through the different buttons automatically. This mode can also be used when more than two input devices are present, where the additional inputs are mapped to different shortcuts.

Figure 49 shows the blocks of the scanner.

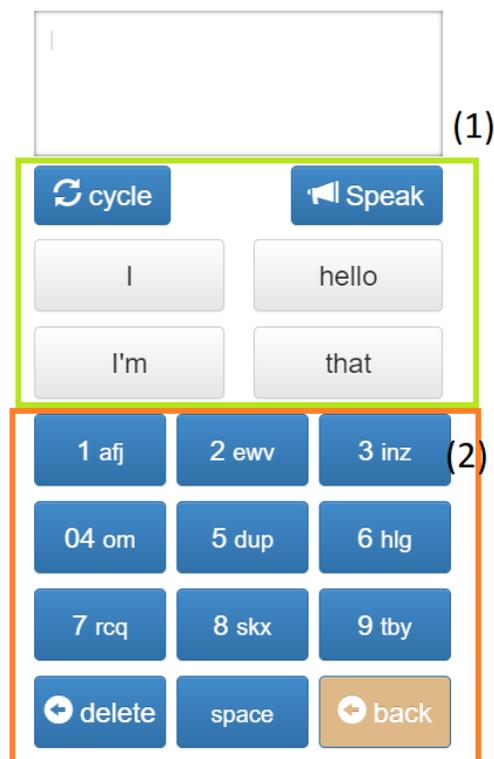


Figure 49 Scanner Blocks (1): Suggestions Block, (2) Letters Block

The back button is always scanned before the scanner restarts a cycle; however, the back button can have different functions depending on the block being scanned. The cycle or “next” button replaces the suggestions with the next ones (next words being less desirable/probable based on the disambiguation algorithm). The speak button activates the TTS module where the contents of the text box are read out loud.

Single Input Method:

The user utilizes the input method to select the desired button as the scanner cycles through the available buttons. The flow in single-input mode is shown in Figure 50.

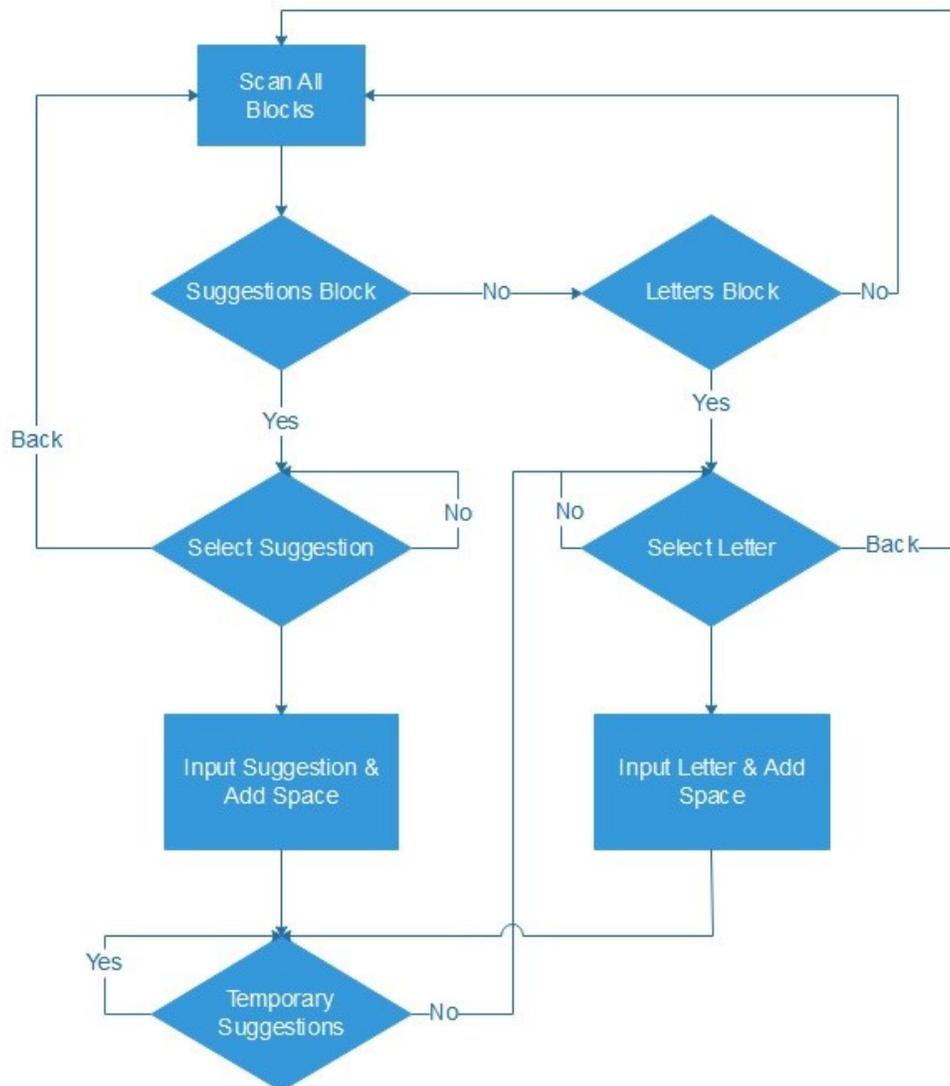
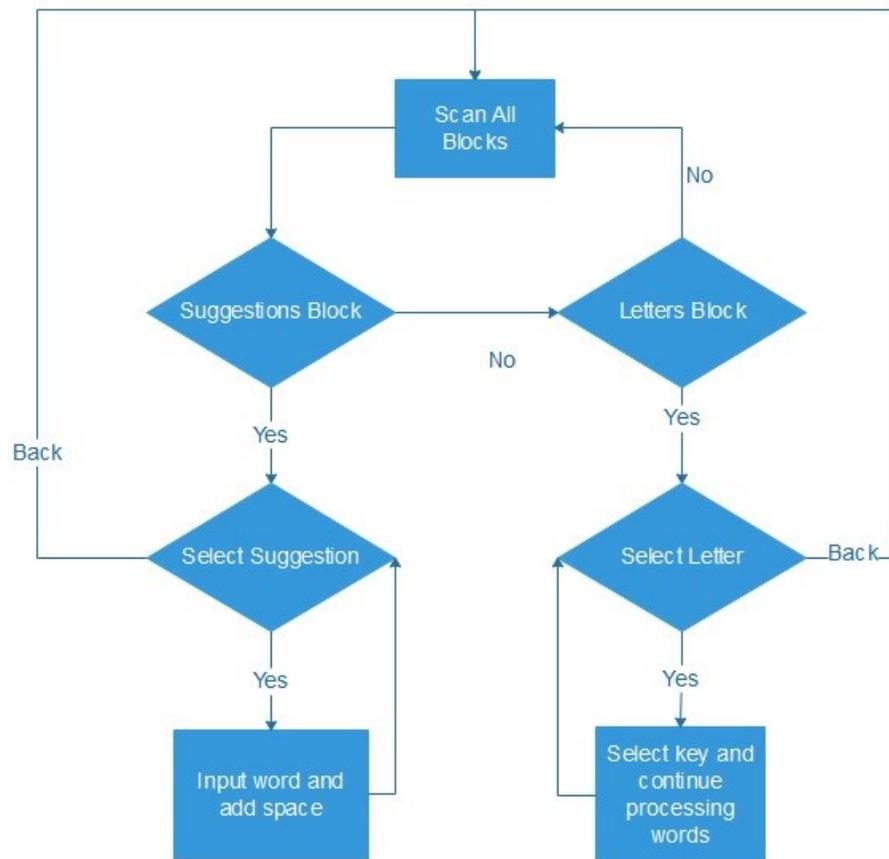


Figure 50 Single-Input Behavior in EMPWRD Scanning Grid Algorithm

Scanners in the literature reset to the initial state after a letter or a word is entered. However, we are proposing a phase of temporary suggestions where the users can go directly to that block or wait and be automatically redirected back to the letters block, thus saving time and additional keystrokes.

In our case, the scanner starts by cycling through all blocks. When the letters block is selected, the scanner starts cycling through individual letters. If the back button is selected, then the scanner goes back to cycling through all blocks; otherwise, once a key is selected, then that key is processed by the disambiguation algorithm and the scanner goes back to the suggestions block and halts there temporarily. When the suggestions block is selected, the scanner starts cycling through the buttons of the suggestions block. If the back button is selected, then the scanner goes back to cycling through all blocks; otherwise, the selected word is added to the text box, a space is added at the end automatically, the disambiguation algorithm is prepared to start processing the next word, and the scanner goes back to the suggestions block and halts there temporarily. When the scanner halts on the suggestions temporarily, it is up to the user to select the suggestions block and continue through the flow normally or wait until the scanner exits the temporary state and starts scanning the letters again. The flow of a traditional scanning keyboard is shown in Figure 51.



*Figure 51 Flow of Traditional Scanning Keyboards*

### Dual Input Methods:

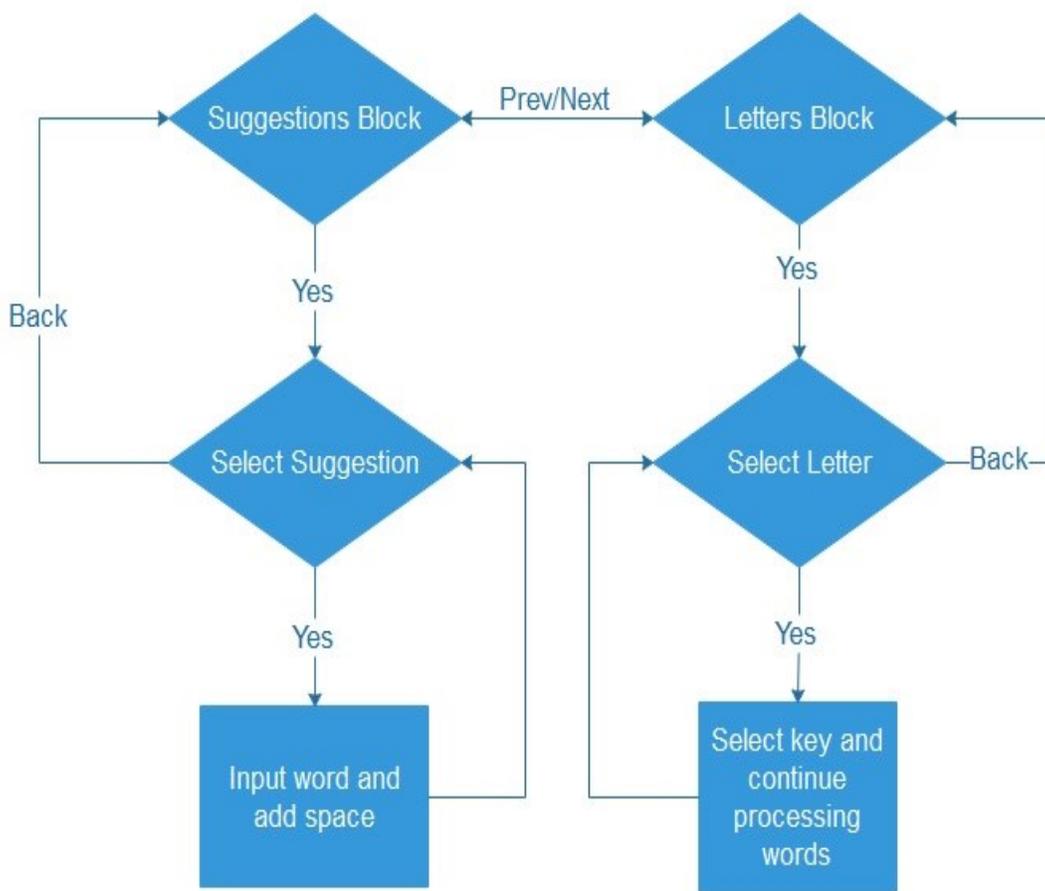
The auto-pilot mode in dual input methods acts in a similar way to that described in the Single Input Method, but the user can undo the last auto-pilot move by using the second input method.

### Manual-pilot mode

The rules of the manual-pilot mode and the way the cursor moves between buttons is different from that of the auto-pilot mode to maximize the user comfort and minimize the time needed to access the different components of the system.

In this mode, the user controls the movement of the selector; one input method is used to go forward while the other is used for selecting the desired button; in the case of a third input, that input can be used to go one step backwards (reverse scanning step).

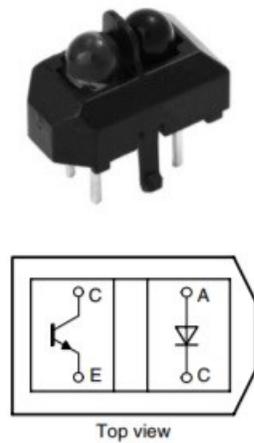
In this mode, there are no temporary suggestions phase, since the user is controlling the scanning step as opposed to it being automatic. Figure 52 describes the flow of the scanning grid cycle in manual mode. The scanner stays in place after any key is selected so as not to surprise the user, especially that the user is controlling the scanning speed (manual cycling through buttons and blocks).



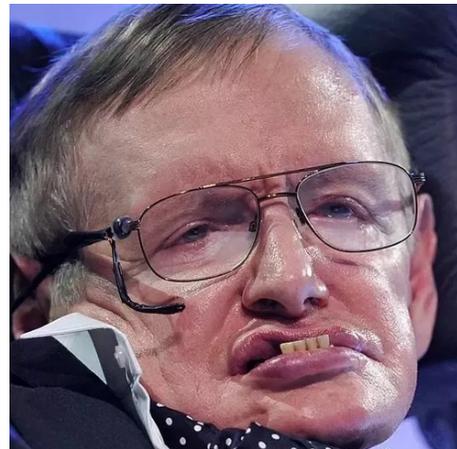
*Figure 52 Scanning Cycle in Manual Mode*

### 4.3 INPUT CALIBRATION MODULE

Regardless of the sensor used for cheek muscle contraction detection, it still needs to be placed in close proximity to the muscle which movement is to be detected. It is worth mentioning that since different people have different face shapes and proportions, we have opted to model the sensor frame in a flexible way that allows it to be adjusted to fit the user's face. This flexibility, however, raises the question of the best location of the sensor in terms of its direction, proximity to the muscle (contactless), and stability. Figure 53 shows Stephen Hawking's sensor mounted on his glasses frame [60].



*Figure 54 TCRT5000 Picture  
and Top View*



*Figure 53 Stephen Hawking and  
Cheek Movement Sensor*

In order to answer this question, we need a framework to evaluate the different factors and provide a customizable solution that caters for every user's needs, the proposed calibration framework serves the purpose of verifying whether the position of the sensor is suitable for the user using it in terms of comfortability, stability, and signal strength.

The proposed calibration framework is composed of 3 main modules:

### 4.3.1- The initialization tutorial

In this module the caregiver is familiarized with the different steps of calibration and how each step works. It also includes a general guideline on how a sensor frame should be initially adjusted before moving to the position examination module. Once this module is activated, all other system components are posed to prevent accidental clicks.

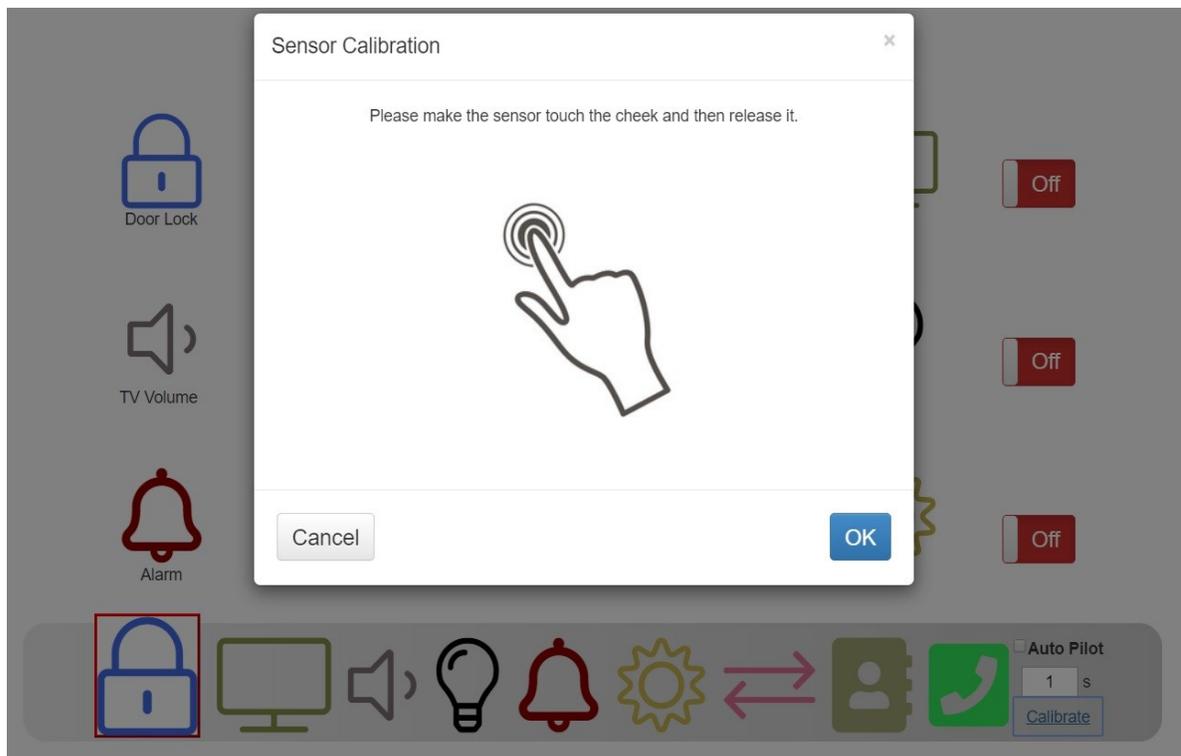


Figure 55 Calibration Initialization Tutorial in Action

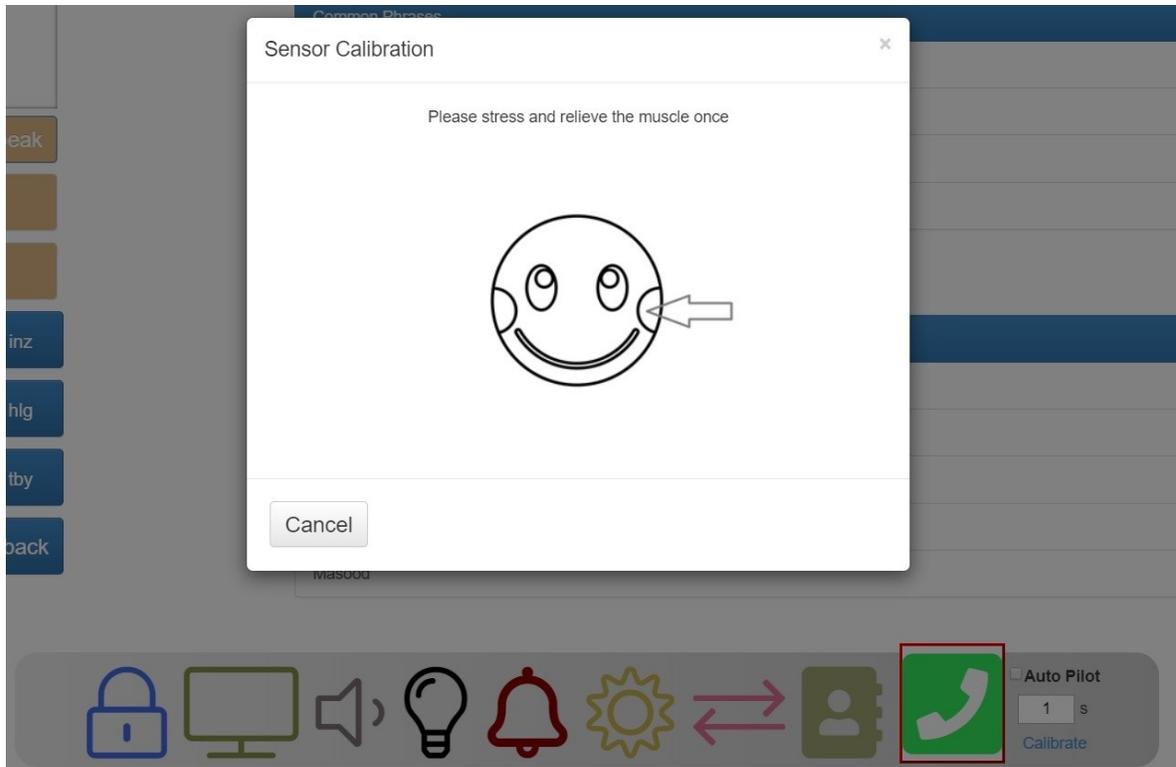
### 4.3.2- The position examination module

In this module the user is instructed to tense the muscle and relax it based on 2 main states:

#### The ready state

In this state, the user is instructed to tense the muscle whenever ready. If a muscle tension is detected, the signal strength is shown to the caregiver for reference and the state changes to the pause state. If muscle tension occurs but is not detected (false

negative), then the sensor positioning should be altered until detection happens, the state will not change until then.



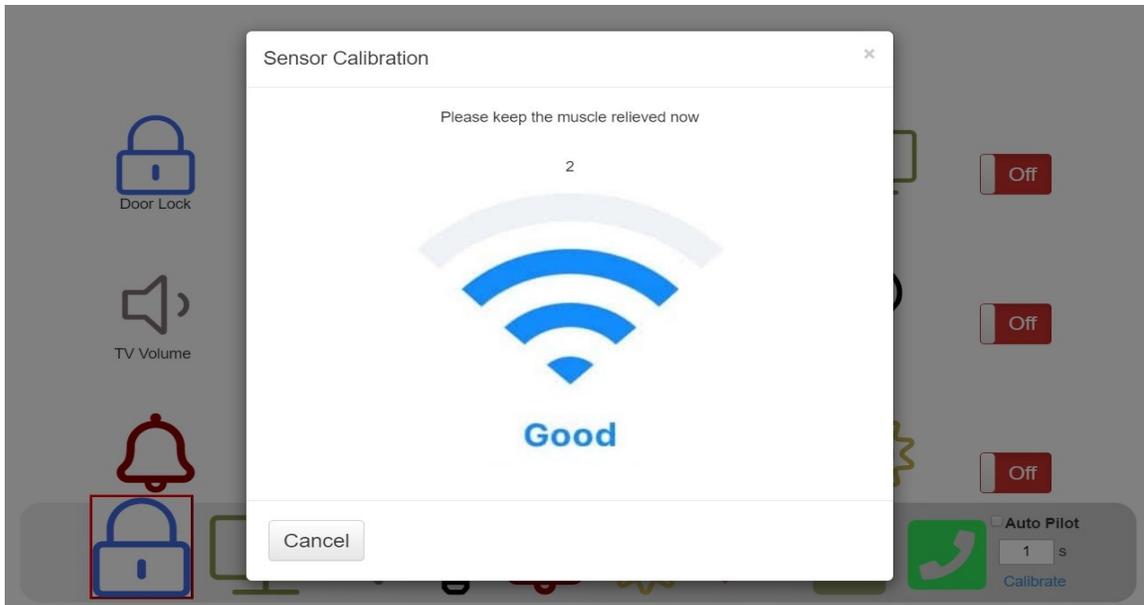
*Figure 56 Ready State Waiting for Input*

In the case of a false positive - where a non-existent tension is detected- the state is changed to the pause state indicating an unstable sensor positioning.

Based on trial and error, the result is almost always a fail in the pause state when a false positive occurs. The result cannot be a direct fail in the ready state.

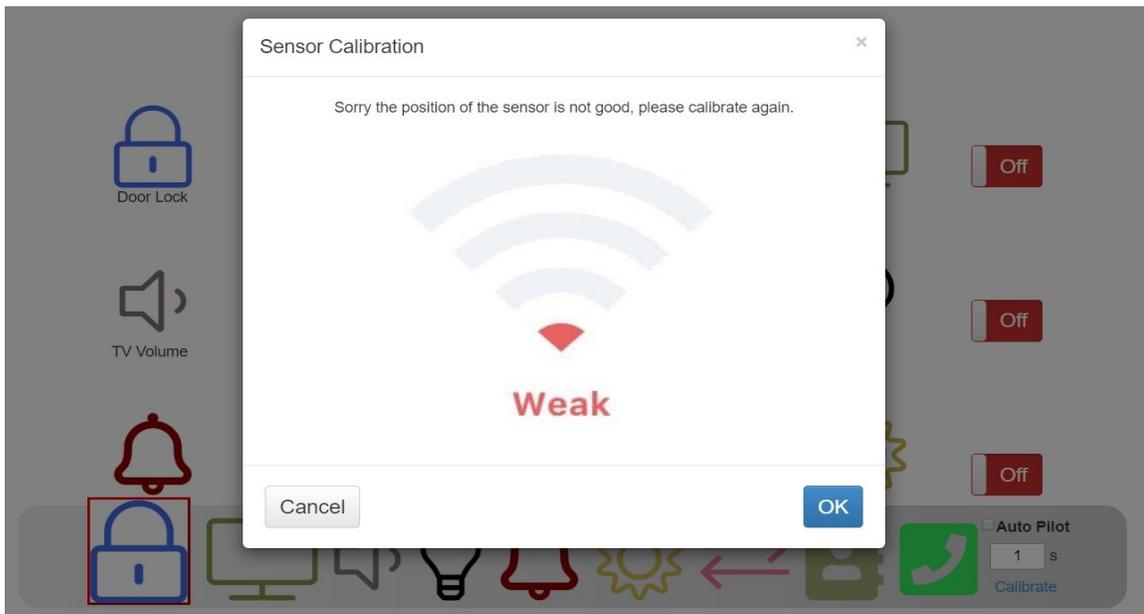
### **The pause state:**

This state shows the signal strength calculated in the ready state. The signal strength indicates the level of reliability of detection; it refers to the signal of the reading itself (which is calculated by the microcontroller connected to the sensor) and should not be confused with the signal of the Internet connection or the communication channel.



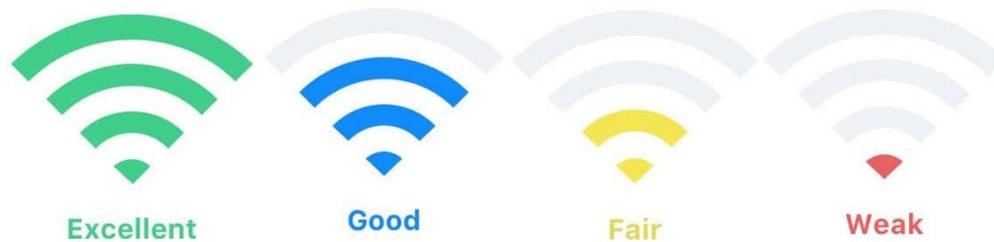
*Figure 57 Signal Indicator Showing Good Signal Strength*

In this state, the user is instructed to relax the muscle for a specified duration of time. This is done in order to rule out false positives which indicate instability. If muscle tension is detected during this state, the result of calibration is a direct fail because that means unstable readings, otherwise it is a success for this trial.



*Figure 58 Calibration Fails After a False Positive Occurs*

Signal strength is defined in one of three groups: “Excellent”, “Good”, and “Fair”, whereas “Weak” denotes a fail.



*Figure 59 Groups of Signal Strengths*

#### **4.3.3- The results module (Multi-Trial Voting System)**

The purpose of the ready state is to make sure that muscle tension is being detected, whereas the purpose of the pause state is to ensure that artifact readings (false positives) are ruled out, thus leading to stability in detection.

These steps - “a” and “b” - are grouped into one trial; multiple trials are required to ensure stability; if any single trial fails, then the overall result is a fail and calibration must be restarted. Hence, we can say that these steps are repeated for increased accuracy. If the failed state is not reached after multiple trials are completed, then calibration is considered successful.

However, a successful calibration does not always mean a stable reading; the overall stability is deduced from the total of the signal strengths calculated at the end of each trial.

Based on trial and error, we have found that three trials are a good balance between user-friendliness and reading stability, so we are using three trials for calibration in the current implementation.

The calibration process can be summarized as shown in the following diagram:

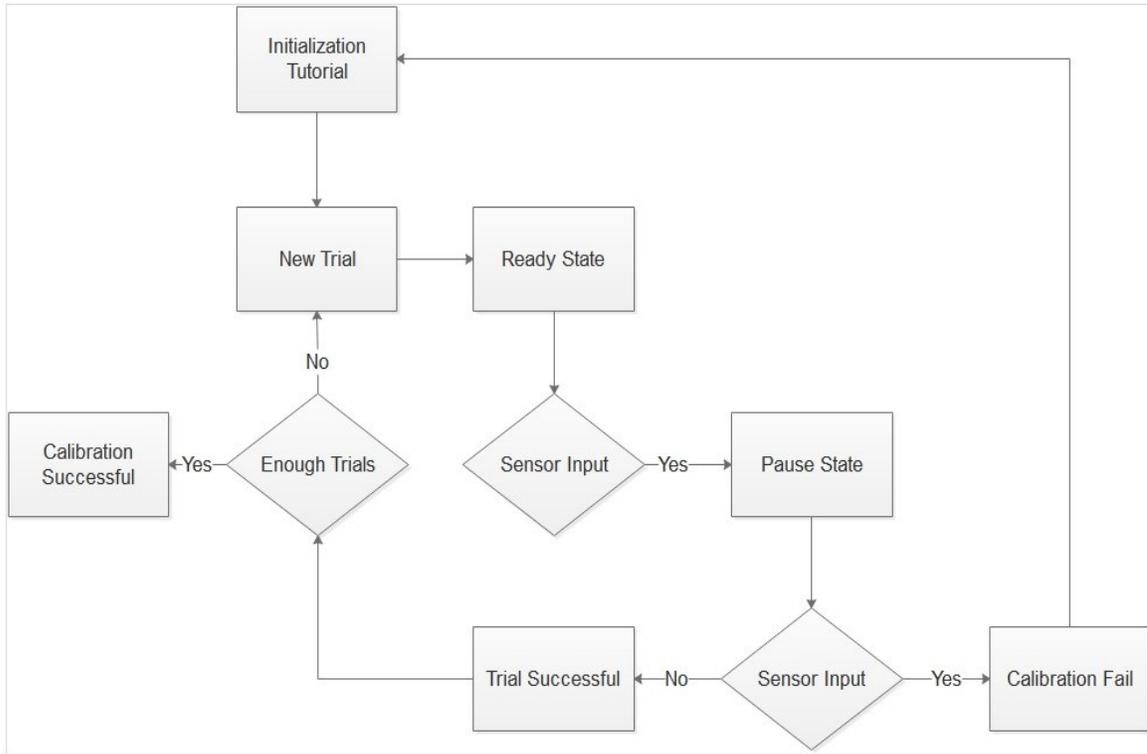


Figure 60 The Process of Sensor Calibration

For simplicity, and since the period of time by which calibration takes place is not very long, we are using the mean of signal strengths as calculated at the end of every “ready” state without assigning any weights; the equation is as follows:

$$\text{Signal} = \text{Enum} (P_x) \quad (7)$$

$$P_x = \frac{1}{n} \sum_{i=1}^n (x_i) \quad (8)$$

where  $x$  is the signal strength enumerable, and where “Fair”, “Good”, and “Excellent” are enumerated by the integers 1, 2, and 3 respectively.

Example:



However, in the case where more trials are required, and the calibration process stretches a longer period of time, or in the case of continuous calibration, a weighted average algorithm may come in handy. A simple example of a weighted average algorithm is as follows:

$$Signal = Enum (P_x) \quad (9)$$

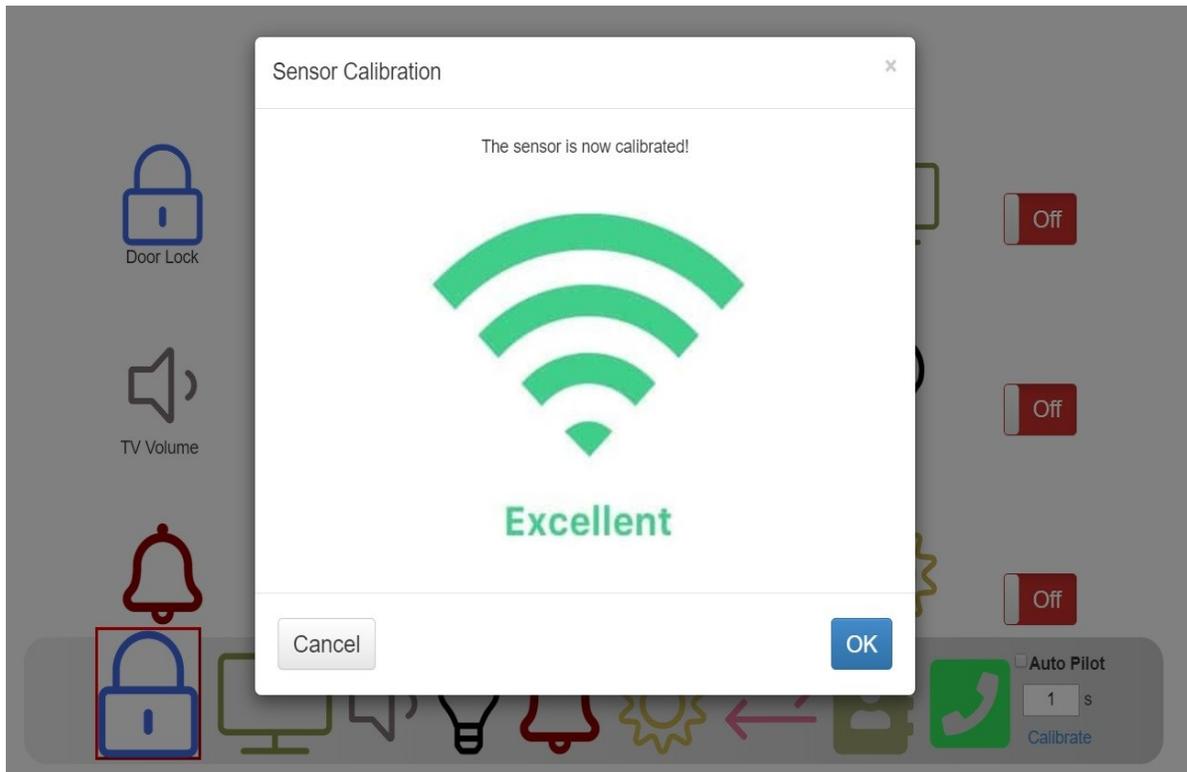
$$P_x = \frac{\sum_{i=1}^n (i^k) x_i}{\sum_{i=1}^n i^k} \quad (11)$$

where  $x$  is the signal strength enumerable, and where “Fair”, “Good”, and “Excellent” are enumerated by the integers 1, 2, and 3 respectively.  $K$  represents a number that determines the factor of the weight assigned to individual iteration values as the series progresses. The series may be limited to a specific number of latest readings instead of taking all readings since the initialization of the system.

*Example:*



Other algorithms that also feature smoothing exist [124]; the algorithm presented above is just a sample as means to show how weights can be added in order to favor the latest readings over the older ones. This may specifically prove helpful in the case of continuous calibration, which may help predict when the next calibration may be required. It is also worth mentioning that continuous calibration may not only yield better results, but also be used as a model of biometrics to provide the means for authenticating the user.



*Figure 61 Final Signal Strength Ruled Out by Multi-Trial Voting System*

## **CHAPTER 5      COMPARISONS AND EXPERIMENTAL ANALYSIS**

### **5.1    CHEEK MUSCLE INPUT METHODS COMPARISON AND APDS-9960**

In chapter 1, we presented a number of methods for detecting voluntary input from users with severe motor disabilities. We assess these methods in terms of comfortability, privacy, training required, cost, accuracy, and calibration requirement.

We also propose a new method as a potential sensory measure for detecting voluntary input through the measurement of ambient light (ALM) in combination with pattern recognition using a one-pixel camera (APDS-9960 sensor).

After developing our own version of the TCRT5000 based sensor and experimenting with different mounting positions and different lighting conditions, we came to the conclusion that although it works, it may not be the optimal sensor for our case where users have to wear it most of the time. It was obvious that our sensor had to rely on waves in order to stay contactless, the sensor also had to be non-intrusive so an image producing camera was out of the question; at the same time, the sensor is meant to be worn by the user most of the time on the head, so we had to be careful about which type of waves we use lest we introduce any side effects or harmful behavior; it was clear that we were stuck with light waves, namely visible light and infrared; that is when it struck us that if reflectivity works, then ambient light index and/or a single RGB pixel may be utilized to detect a cheek muscle contraction as well.

According to our experimentation, APDS-9960 sensor, which is advertised as a gesture sensor, and is less known for its single pixel camera and ambient level sensor, ended up to be a much better replacement of the TCRT5000, mainly due the difference in how reflectivity index and ambient light value are affected by movement of the whole face (that includes involuntary movements of cheek due to other muscle contractions such as blinking or even trembling), which leads to noise, and movement of the cheek which is our desired signal source.

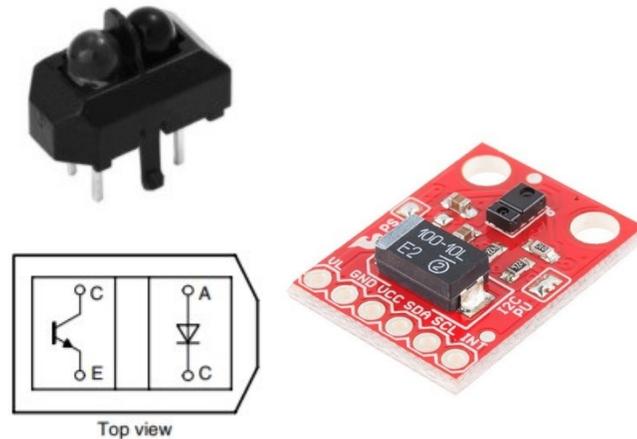


Figure 62 TCRT5000 (left) and APDS-9960 sensor (right)

We also found out that, in addition to the improvement of the results just by replacing the reflectivity sensor with the ambient light sensor, combining the results of the ambient light sensor with that of the single pixel camera may potentially improve the SNR ratio and thus the APDS-9960 would be a major improvement over the TCRT5000. It may also be possible to use the RGB values to aid in calibrating the sensor to find the best spot over the cheek once the device is set on the user's head.

One of the drawbacks that we faced with APDS-9960 though, is that it needs to be reinitialized every time it got disconnected from the microcontroller, where as the TCRT did not require initialization in the first place. This should not be of importance in the final product for two main reasons:

- 1- The headset can be designed in a way that the sensor is never disconnected while the microcontroller is collecting data (except in the case of failure of course).
- 2- A simple hard reset circuit can be added to the APDS-9960 so that it is restarted and reinitialized by the microcontroller in case of disconnection or temporary failure. This circuit can be as simple as a transistor, this of course would require some minor modification in the software to cater for that as well.

The following table represents a quick comparison of both sensors according to our findings:

Table 3 TCRT5000 vs APDS-9960

	TCRT5000/L	APDS-9960	Notes/Conclusion
Picture			APDS-9960 features a gesture sensor which occupies a lot of space. We can create our own smaller version by removing the gesture sensor.
Dimensions (mm)	10.2 x 5.8 x 7	17.8 x 17.8 x 3.2	TCRT5000 needs additional circuitry in order to be integrated with a microcontroller so it is actually bigger than mentioned in the datasheet.
Cost	0.5USD	7.5 USD	Both are considered very cheap.
Sensitivity Type	Reflection	1 RGB Pixel, Ambient Light	APDS-9960 provides more parameters to study
SNR	<p>Good</p> <ul style="list-style-type: none"> <li>• Highly susceptible to noise due to sudden head movements</li> <li>• Affected by Skin color (due to different reflectance ratios) [125]</li> <li>• Affected by temperature</li> </ul>	<p>Good</p> <p>Noise is in the form of visible light.</p>	Although both sensors are sufficient for the task, the type of induced noise is different.
Initialization	Not required	Required	Both require calibration after being placed on the user's cheek

The following table summarizes the features of all the solutions that we have mentioned in terms of Comfort, Privacy, Training Requirement, Accuracy, Calibration, and Cost:

*Table 4 Feature Summary of Different Input Methods*

	Comfort	Privacy	Training	Accuracy	Calibration
Facial Expressions (Camera) - Low Cost	Contactless, very comfortable	Low, users and surrounding may be monitored	Depends on algorithms used	High, affected by lighting conditions	Not required
Thermal Imaging - Low to moderate cost	Contactless, very comfortable	Low, users and surrounding may be monitored	Depends on algorithms used	High, affected by temperature and infrared	Not required
EMG - Low Cost	In contact with muscle, irritating, may require gel	High	Not required	High	May be required
PANDA Resonator - Unknown Cost	In contact with muscle, irritating	High	Not required	Unknown	May be required
EEG - High Cost	In contact with muscle, irritating, may require gel	High	Required	Moderate to High	Required, usually very slow
Piezoelectric Sensors - Low Cost	In contact with muscle, may be irritating	High	Depends on multiple pressure levels support	High	Usually not required
Breath Pressure Sensors - Unknown Cost	In contact with nose, may be irritating	High	Required	High	Required

	Comfort	Privacy	Training	Accuracy	Calibration
TCRT5000 (reflectivity) - Very Low Cost	Contactless, very comfortable	High	Not Required	High	Required, but usually quick
APDS-9960 (ambient light and RGB) - Very Low Cost	Contactless, very comfortable	High	Not Required	High	Required, but usually quick

Since TCRT5000 and APDS-9960 do not interfere with each other, we propose a multi-modality system that utilizes both sensors instead of choosing one solely over the other. This is important for getting the best of both worlds, especially that weights can be assigned to each sensor's features and a voting system may be used for the final decision on whether the detected input is valid or not.

## 5.2 STEPHEN HAWKING'S SYSTEM: ASSISTIVE CONTEXT-AWARE TOOLKIT (ACAT)

For the purpose of comparison, we are most interested in ACAT: Assistive Context-Aware Toolkit which was developed by Intel over the period of 3 years, and was the result of research of several years where Hawking was a major contributor. The system was developed specifically to help Hawking control his PC, check his Email, and edit documents to name a few. The system was released as open-source in 2015 and can be found via the Github repository [22]: <https://github.com/intel/acat>

An FAQ document along with a user's manual were published, and developers were encouraged to add new modules and modify their own versions that suit different users.

Figure 63 shows the default input method of ACAT, which utilizes a camera to detect cheek movements.

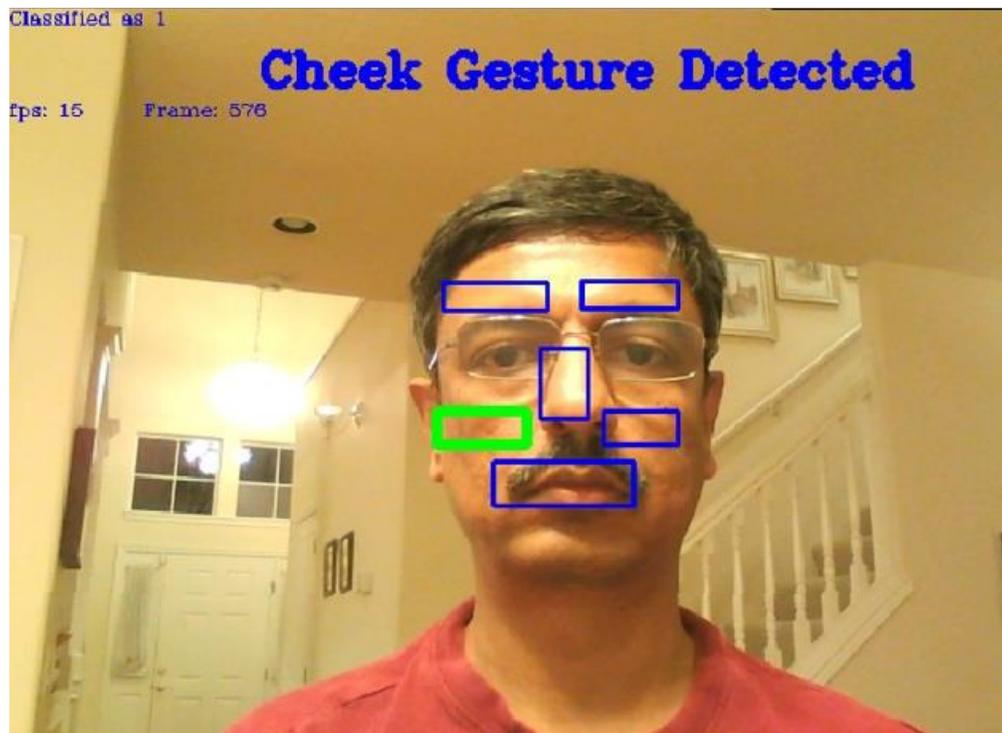
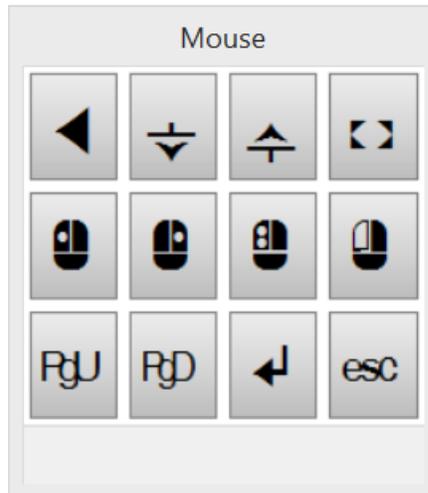


Figure 63 Facial Expression Detection in ACAT [22]

ACAT is a complete PC-control platform for people in need of such accessibility capabilities. It does not only provide means for typing text and reading it loudly, but also provides the possibility to control the pointer and the mouse buttons using an accessibility focused window as shown in Figure 64.



*Figure 64 ACAT's Scanning Grid Keyboard for Pointer Controls [22]*

ACAT separates between different functionalities as each functionality is provided in a separate window. For example, speech synthesis activates its own new window, mouse scanner - shown in Figure 64 - activates a new window, and the same is done for the other features as well. While this shows reasonable modularity, further research is required to determine whether separate windows help the users or add to their distractions and pose as a hurdle in their way of getting used to different pages.

Figure 65 presents a full QWERTY keyboard featured in ACAT that also features additional functional buttons.

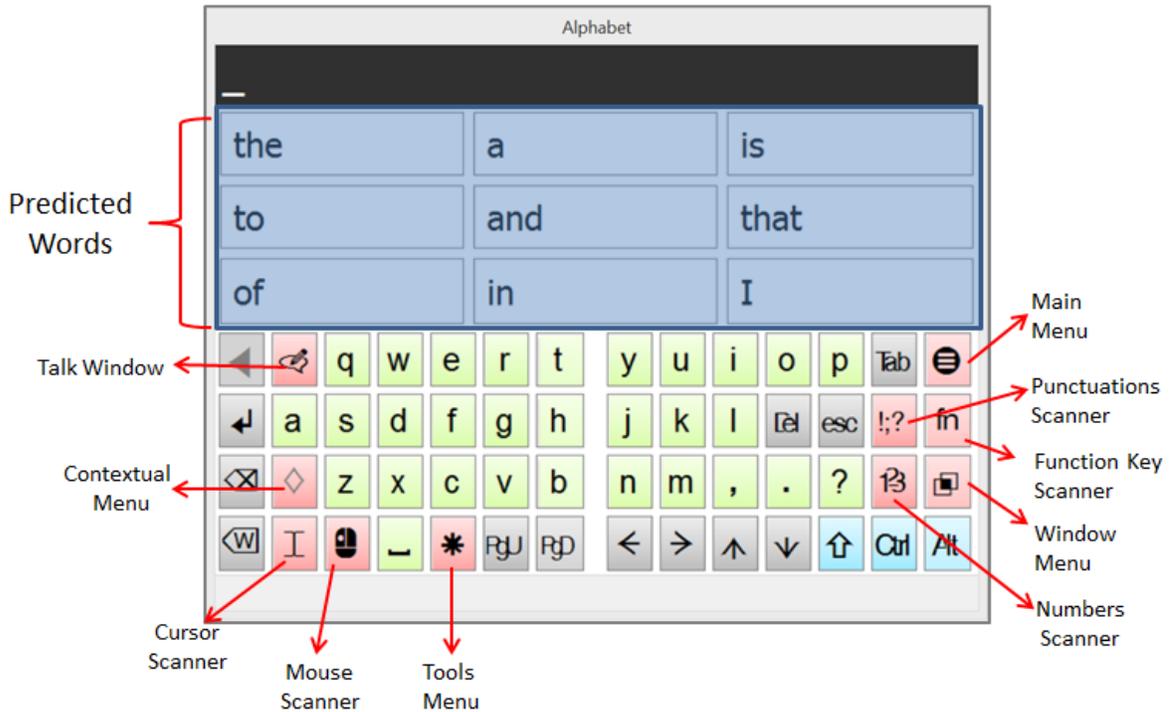


Figure 65 Full Scanning Grid Keyboard With Predicted Words in ACAT [22]

In Figure 66, the scan speed configuration window is presented. The user is requested to type a word and modify the speed of the scanner as desired.

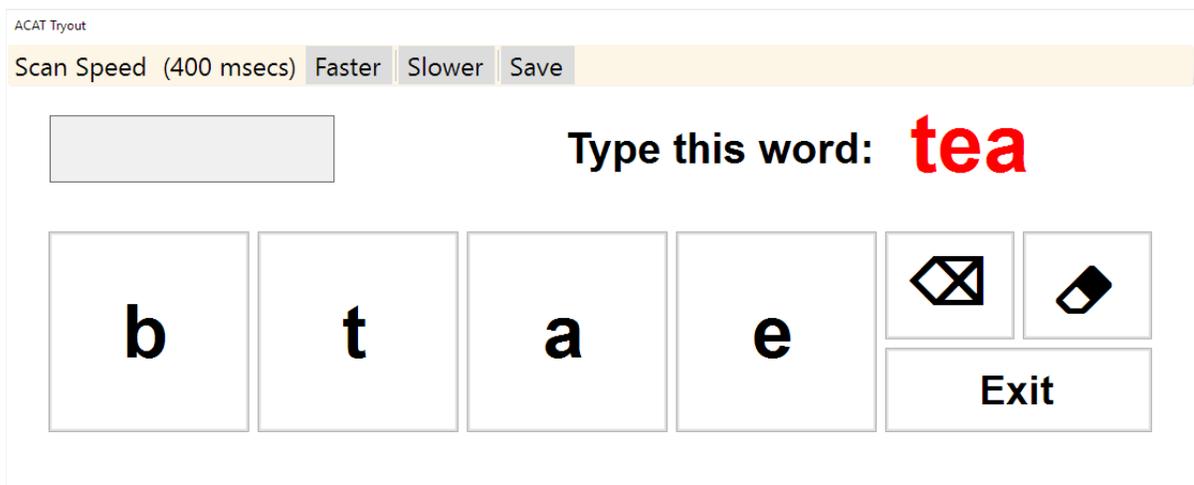


Figure 66 Scan Speed Configuration Window in ACAT [22]

As discussed in the literature review, cluster ambiguous keyboards are a good fit for applications similar in nature to ACAT; however, ACAT uses a full-blown keyboard that supports alphabetically ordered letters and QWERTY layouts.

The scanner starts by scanning 3 main blocks: the suggestions block, the left side of the keyboard, and the right side as shown in Figure 67.

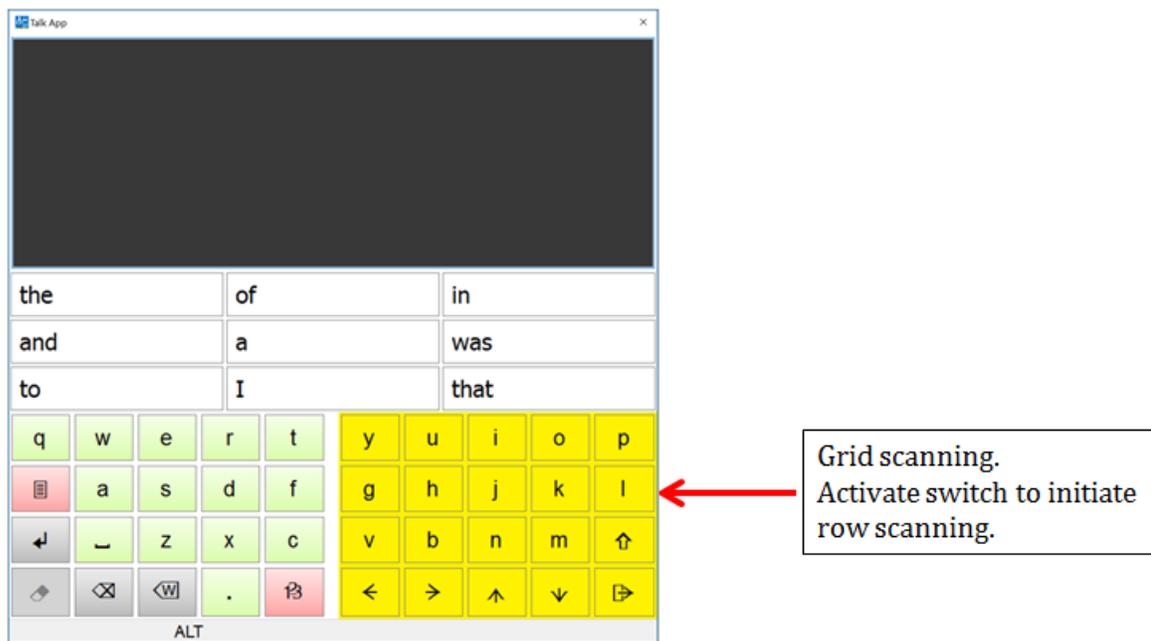


Figure 67 Step 1: Block Selection In ACAT's Scanning Grid Keyboard [22]

Once user input is detected, the next step of the scanner is activated. In the next step, the scanner starts scanning the rows of the selected block. After that, column scanning starts, where each button in the selected row is scanned independently as shown in Figures 68, 69, and 70. The user needs three input signals to select a single character; in the case of a mistake, the user needs three input signals to reach the backspace button which deletes the last character, and - bearing in mind the 3 wasted input signals for selecting the wrong character and the time consumed to reach the corresponding rows and columns - we can say that a mistake is very time consuming.

To type the word "hello" assuming that the right word is going to be the first suggestion after typing the letter "h", the user needs 3 input signals to select h and two others to select the first suggestion, and that is considered a very optimistic example.

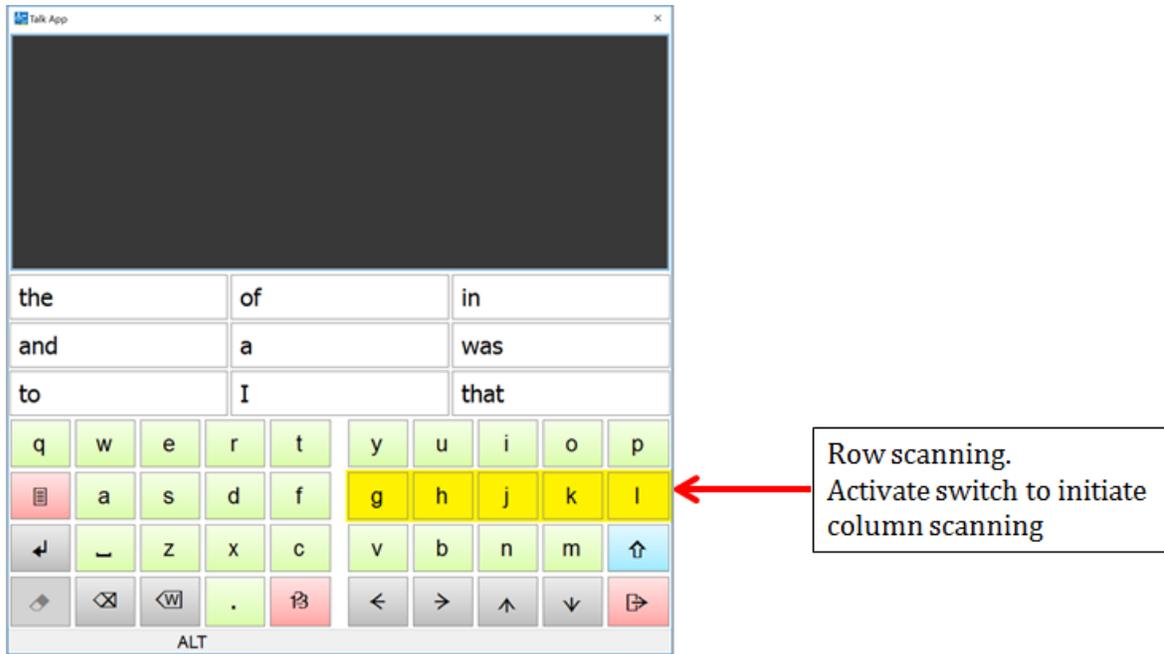


Figure 68 Step 2: Row Selection In ACAT's Scanning Grid Keyboard [22]

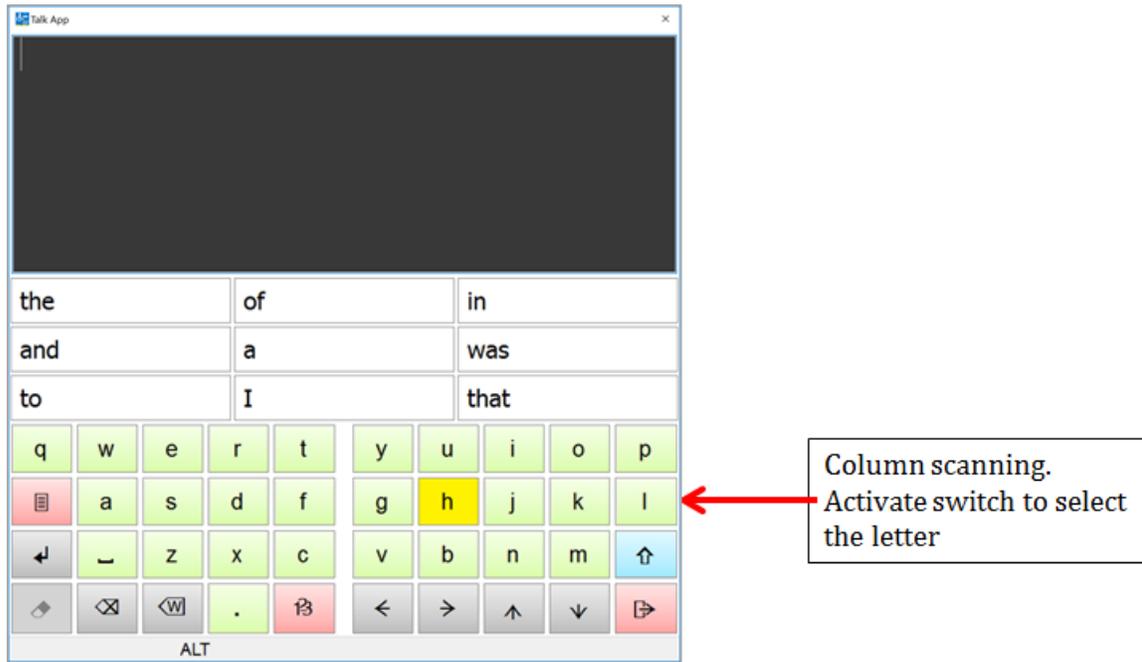


Figure 69 Step 3: Column Selection In ACAT's Scanning Grid Keyboard [22]

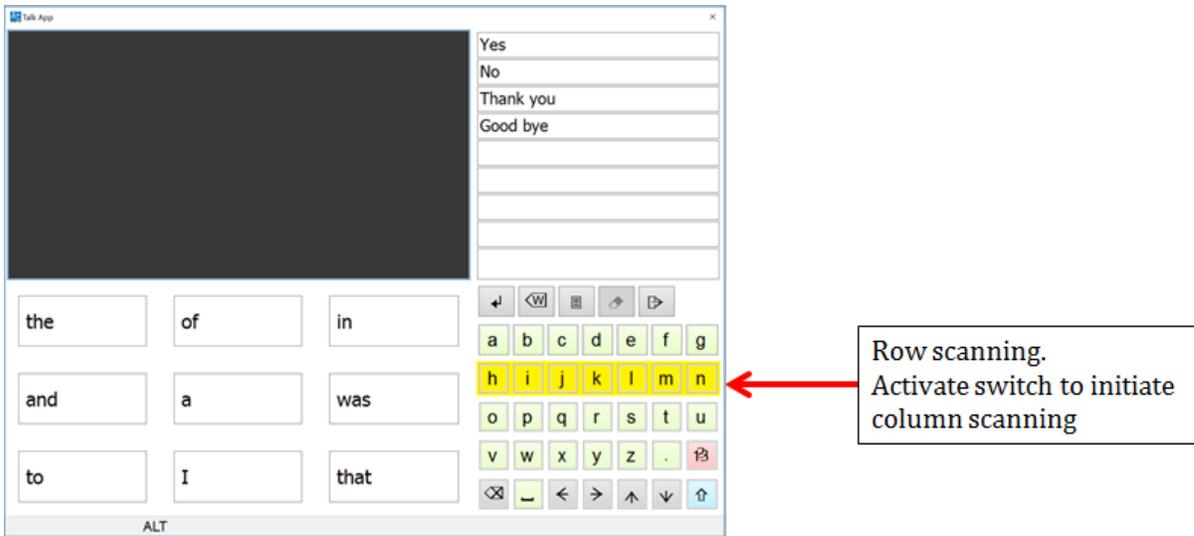


Figure 70 Alphabetic Layout Scanning Grid Keyboard in ACAT [22]

ACAT provides different keyboards for different tasks. Some keyboards do not change the scanning block until a special key is pressed as shown in Figure 71.

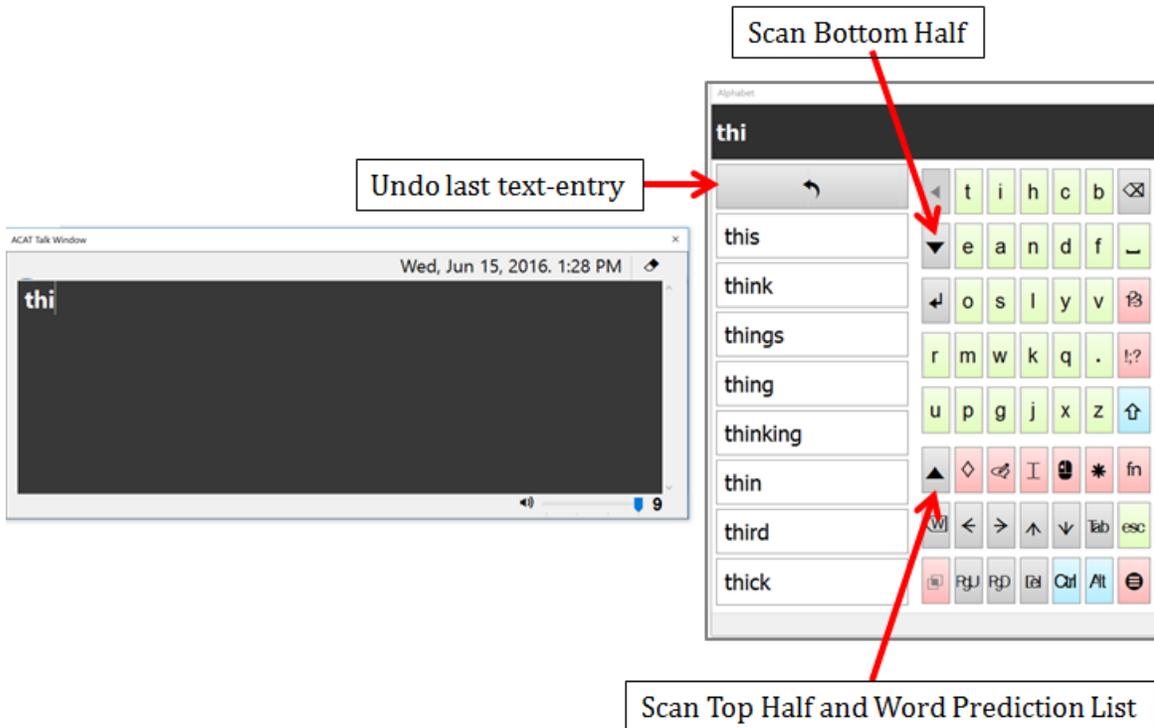


Figure 71 ACAT's Talk Window and Talk-Related Scanning Grid Keyboard [22]

## **ACAT vs EMPWRD Comparative Overview**

ACAT is an open-source tool that provides accessibility tools for users suffering from motor impairments. It was developed by Intel and improved over a long period of time and was primarily used by one of the greatest theoretical physicists of our age, Stephen Hawking.

However, some of the major downsides of ACAT are that it was developed in C# specifically for the Windows operating system, that it uses too many windows, that its main dashboard depends on using a mouse before an accessibility window starts, that it is resource intensive, and that it does not make use of cluster ambiguous keyboards.

EMPWRD, on the other hand, is a portable platform that is developed in JavaScript and can run on any browser-enabled device regardless of the operating system, does not require intensive resources, utilizes a dock to connect different modules instead of activating new windows, and the whole platform is directly accessible to the user (in case the sensor is already calibrated).

Moreover, some of the most important features in EMPWRD are the cluster ambiguous keyboard that utilizes an enhanced input-adaptive scanner (section 4.2) and an improved auto-complete and auto-correct algorithm for cluster ambiguous keyboards (section 4.1), the GSM live calling module, and the binary-input calibration module which makes sure that the sensor/s is/are working properly. All which are features that ACAT lacks.

To showcase the importance of the cluster ambiguous keyboard along the enhanced algorithms, we consider typing the whole word “hello” as an example; each letter requires 3 input signal in ACAT, making a total of 18 signal inputs for that word alone including the spacebar; whereas the whole word requires only 5 signal inputs for the whole word in EMPWRD, and 2 for the selection of the word from among the disambiguated ones, making that a total of 7 signal inputs.

As another example, let’s assume that the word “hello” is suggested after typing the second letter; that would take 6 signal inputs in ACAT for the two letters, and 2 for the selection of the suggestions, making a total of 8 signal inputs. While it would take 2

signal inputs for the letters and 2 for the suggestions in EMPWRD. In both cases, EMPWRD requires at most two times fewer signal inputs. Taking into consideration the time wasted for every additional signal input, especially for the wait until the scanner reaches the desired block, we can say that EMPWRD features a major improvement in time and effort over ACAT's current solution. Figure 72 presents EMPWRD's cluster ambiguous keyboard.



*Figure 72 Cluster Ambiguous Keyboard Featured in EMPWRD*

It is also worth mentioning that since ACAT is dependent on the operating system because it is a native Windows app, it is more susceptible to errors and may be considered less stable than EMPWRD if the latter is provided as a service (online

SaaS). It is also worth mentioning that even though EMPWRD doesn't support mouse functionality or native document editing, it is easy to integrate a new module that would cater for that. It's also possible to use ACAT's own mouse control window (Figure 64) to work with EMPWRD.

The following table presents a quick comparison between ACAT and EMPWRD.

*Table 5 Comparative Overview of ACAT vs EMPWRD*

	ACAT	EMPWRD
Programming Language	C#	JavaScript
Compatibility	Specific versions of Windows only	Any browser-enabled device
Resource Intensity	High	Low
Feature-set	Extensive, requires training, and is not for everyday users	Simple, built for all users who may benefit from it
IoT Control	N/A	Featured
Typing rate	Slow	Presumably faster (benchmark suggested in future work)
Keyboards	Alphabetic and QWERTY	Cluster Ambiguous
Modularity	Each module is open in a separate window	All-in-one solution, powered by a universal dock
Live calls compatibility	Can control third party modules using mouse controls	Built-in with extended features such as common phrases
Native apps control such as word editing and file browsing	Built-in	Can be added as a module
Software as a service (SaaS)	Needs installation of software and its dependencies	Native and SaaS compatible

## CHAPTER 6 CONCLUSIONS AND FUTURE WORK

### Conclusion

It is amazing how technologically advanced we have become. We have the means to empower people in every way possible. Even those who have lost control over their whole body and are considered to be in a “locked-in syndrome” - where the brain is fully functional and they are aware of their surroundings but are unable to make any movement to express themselves - may benefit from the latest solutions meant to address these cases.

Such solutions feature multiple techniques and each technique has its own advantages and disadvantages. For example, while breath pressure detection may have a higher typing rate than other systems, it may be tiresome to control breath for a prolonged time, especially in the case of users with severe motor disabilities. Similarly, for other methods such as facial expression recognition, while the technique is wireless and comfortable in general, it may pose a threat on privacy. Finding a good balance between comfortability, privacy, training required, cost, accuracy, and calibration requirement is not an easy task, and while our results show that the use of reflectivity index along with ambient light measurement and a single pixel image may be optimal, it is still a matter of what the user actually has control over. For example, a user who does not have control over any muscles nor control their breath pressure, is bound to use EEG as it is the only current solution that relies solely on the brain alone.

Moreover, most common sensors used in such solutions require calibration. Finding a good balance between comfort, training required, and accuracy for calibration is not an easy task either; each user is different and thus the calibration must be adaptable to different needs. In this report, we have presented a calibration framework with realistic examples using TCRT5000 to address this issue. In addition, we are proposing the use of APDS-9960 alongside TCRT5000 in cases where only TCRT5000 is currently being utilized.

We have also presented a solution that enables live phone calls by utilizing TTS using a self-hosted GSM client. The TTS system works based on the custom keyboard and scanning grid developed. Common phrases help save time and make a live call a smoother errand, whereas confirmation dialogs prevent accidental calls and actions. In order to improve the accuracy of the suggestions, we extended state-of-the-art NLP algorithms with a technique for considering typos and automatic correction. In addition, we proposed an input-adaptive scanner to improve users' comfort.

Finally, we combined all of our work in a modularly scalable framework so as to cater for different users' needs and upcoming technology advancements. This solution ensures that people with special needs are not left out when it comes to telecommunications. After all, finding the best solutions to cater for the different needs of challenged people is still a user-by-user situation rather than a one solution fits all approach.

## **Future work**

### **Multi-user collaboration**

Corpora can be merged from different users. Different corpora can be created and grouped based on users' background, geographic location, culture, and experience.

### **Speech to text Module**

In a live phone or online call, speech of the other party can be converted to text, and AI/Machine learning can be used to suggest phrases in the "common phrases" section. This makes it possible to have a natural conversation rather than one with predefined responses.

### **Command control in text box**

The current implementation enables controlling nearby devices using the dock (as explained in section 3.2.1.2). By enabling commands in the text box right where users can express themselves, we can connect the system to voice-based control hubs such as Amazon Alexa and Google Home.

### **Long press utilization**

The current solution utilizes the light-based sensors (namely TCRT5000 and APDS9960) as single binary inputs. By taking long muscle actions into consideration, the same sensor can act as dual binary inputs instead, and thus the user will be able to control two switches instead of one using the same setup.

### **Internet access, file management, and document editing**

The current system only allows the user to control nearby devices and express themselves. It does not allow them to manage files or read emails for example. However, as the system is modular, new modules can be developed where control over the whole pc, and even the mouse pointer can be granted.

### **Benchmarks of the proposed cluster ambiguous keyboard algorithms**

We proposed an algorithm that integrates text correction with word predictions in section 4.1. As our proposed solution can be integrated into any existing prediction algorithm, a benchmark can showcase the best situations where text correction is most relevant.

### **Benchmarks of the adaptive scanning grid**

Since we have introduced an input-adaptive scanning grid in section 4.2, different scanning orders and multi-input benchmarks can showcase the strengths and weaknesses of the proposed solution.

### **Re-Calibration Warnings**

It would be helpful to be able to automatically predict when a sensor recalibration is needed. That can be done by extending the sensor calibration module.

## REFERENCES

- [1] Elecrow, "A7 GPRS+GSM+GPS Shield," 2017, [Online], Available: [https://www.elecrow.com/wiki/index.php?title=A7\\_GPRS%2BGSM%2BGPS\\_Shield](https://www.elecrow.com/wiki/index.php?title=A7_GPRS%2BGSM%2BGPS_Shield) [Accessed: 07-May-2019].
- [2] Sparkfun, "APDS-9960," 2013, [Online], Available: [https://cdn.sparkfun.com/assets/learn\\_tutorials/3/2/1/Avago-APDS-9960-datasheet.pdf](https://cdn.sparkfun.com/assets/learn_tutorials/3/2/1/Avago-APDS-9960-datasheet.pdf) [Accessed: 07-May-2019].
- [3] "What is an API? (Application Programming Interface)," [Online], Available: <https://www.mulesoft.com/resources/api/what-is-an-api> [Accessed: 07-May-2019].
- [4] "Introduction to AT Commands," [Online], Available: <https://www.developershome.com/sms/atCommandsIntro.asp> [Accessed: 07-May-2019].
- [5] N. Klarlund and M. Riley, "Word n-grams for cluster keyboards," in *Proceedings of the 2003 EACL Workshop on Language Modeling for Text Entry Methods*, vol. TextEntry, Association for Computational Linguistics, 2003, pp. 51–58.
- [6] "5 General Purpose Input/Output," [Online], Available: <https://docs.oracle.com/javame/8.0/me-dev-guide/gpio.htm> [Accessed: 07-May-2019].
- [7] M. Rouse, "GPRS (General Packet Radio Services)," 2007, [Online], Available: <https://searchmobilecomputing.techtarget.com/definition/GPRS>.
- [8] TechTerms, "GPS Definition," [Online], Available: <https://techterms.com/definition/gps>.
- [9] A. A. Huurdeman, *The worldwide history of telecommunications*. John Wiley & Sons, 2003.

- [10] JIMBLON, "Integrated Circuits," [Online], Available: <https://learn.sparkfun.com/tutorials/integrated-circuits/all> [Accessed: 07-May-2019].
- [11] M. Rouse, "internet of things (IoT)," [Online], Available: <https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT> [Accessed: 07-May-2019].
- [12] M. Rouse, "microcontroller," 2017, [Online], Available: <https://internetofthingsagenda.techtarget.com/definition/microcontroller>.
- [13] "MQTT," [Online], Available: <http://mqtt.org/> [Accessed: 07-May-2019].
- [14] W. G. Lehnert and M. Ringle, *Strategies for natural language processing*. Psychology Press, 1982.
- [15] "Node.js," 2019, [Online], Available: <https://en.wikipedia.org/wiki/Node.js> [Accessed: 07-May-2019].
- [16] "nodejs/node," [Online], Available: <https://github.com/nodejs/node> [Accessed: 07-May-2019].
- [17] L. Orsini, "What You Need To Know About Node.js," 2013, [Online], Available: <https://readwrite.com/2013/11/07/what-you-need-to-know-about-nodejs/> [Accessed: 07-May-2019].
- [18] R. Cellan-Jones, "A 15 pound computer to inspire young programmers," 2011, [Online], Available: [https://www.bbc.co.uk/blogs/thereporters/rorycellanjones/2011/05/a\\_15\\_computer\\_to\\_inspire\\_young.html](https://www.bbc.co.uk/blogs/thereporters/rorycellanjones/2011/05/a_15_computer_to_inspire_young.html).
- [19] "GPIO," [Online], Available: <https://www.raspberrypi.org/documentation/usage/gpio/> [Accessed: 07-May-2019].

- [20] M. Rouse, "RESTful API," [Online], Available: <https://searchmicroservices.techtarget.com/definition/RESTful-API> [Accessed: 07-May-2019].
- [21] N. Gohring, "Microsoft describes software plus services," 2007, [Online], Available: <https://www.infoworld.com/article/2642618/microsoft-describes-software-plus-services.html> [Accessed: 07-May-2019].
- [22] Intel, "intel/acat," 2015, [Online], Available: <https://github.com/intel/acat/releases> [Accessed: 07-May-2019].
- [23] Vishay, "Reflective Optical Sensor with Transistor Output," 2017, [Online], Available: <https://www.vishay.com/docs/83760/tcrt5000.pdf> [Accessed: 07-May-2019].
- [24] J. Baart and V. Van Heuven, "From text to speech; The MITalk system: Jonathan Allen, M. Sharon Hunnicutt and Dennis Klatt (with Robert C. Armstrong and David Pisoni): Cambridge University Press, Cambridge, 1987. xii," *Lingua*, vol. 81, pp. 265–270, 1990.
- [25] "Basics Of UART Communication," 2016, [Online], Available: <http://www.circuitbasics.com/basics-uart-communication/>.
- [26] "What is Voip," [Online], Available: <https://www.epiknetworks.com/what-is-voip/> [Accessed: 07-May-2019].
- [27] S. Thornton, "Watchdog Timer: what is it?," 2018, [Online], Available: <https://www.microcontrollertips.com/whats-watch-dog-timer-wdt-faq/> [Accessed: 07-May-2019].
- [28] "WebRTC," [Online], Available: <https://webrtc.org/> [Accessed: 07-May-2019].
- [29] D. J. McFarland, W. A. Sarnacki, and J. R. Wolpaw, "Electroencephalographic (EEG) Control of Three-Dimensional Movement," *Journal of Neural Engineering*, vol. 7, no. 3, p. 36007, 2010.

- [30] O. Hurlbatt, "How does Stephen Hawking's Voice Machine Work?," 2016, [Online], Available: <https://takenototyping.com/stephen-hawking-voice-machine-work/>.
- [31] A. Dix, "Human-Computer Interaction," in *Encyclopedia of Database Systems*, Boston, MA: Springer US, 2009, pp. 1327–1331.
- [32] S. Arjunan and D. K. Kumar, "Recognition of Facial Movements and Hand Gestures Using Surface Electromyogram(sEMG) for HCI Based Applications," in *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*, 2007, pp. 1–6.
- [33] D. Kim and J. Sung, "Facial Expression Recognition," in *Automated Face Analysis*, IGI Global, 2009, pp. 255–317.
- [34] M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, "Recognizing Facial Expression: Machine Learning and Application to Spontaneous Behavior," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, pp. 568–573.
- [35] Z. Al-Kassim and Q. A. Memon, "Designing a low-cost eyeball tracking keyboard for paralyzed people," *Computers & Electrical Engineering*, vol. 58, pp. 20–29, Feb. 2017.
- [36] H. A. J. Manikandan, S. Sajeev, and J. Abraham, "Assistive Technology for People with Complete Paralysis," *International Research Journal of Engineering and Technology (IRJET)*, vol. 05, no. 05, pp. 2747–2750, 2018.
- [37] B. Chambayil, R. Singla, and R. Jha, "Virtual keyboard BCI using Eye blinks in EEG," in *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, 2010, pp. 466–470.
- [38] D. DeMasters and D. Peterson, "Healthcare Privacy Considerations of Body-Worn Cameras," *Journal of AHIMA*, vol. 87, no. 2, p. 32–33, Feb. 2016.

- [39] S. Jarlier *et al.*, “Thermal Analysis of Facial Muscles Contractions,” *IEEE Transactions on Affective Computing*, vol. 2, no. 1, pp. 2–9, Jan. 2011.
- [40] F. Pittaluga, A. Zivkovic, and S. J. Koppal, “Sensor-level privacy for thermal cameras,” in *2016 IEEE International Conference on Computational Photography (ICCP)*, 2016, pp. 1–12.
- [41] G. E. Robertson, G. E. Caldwell, J. Hamill, G. Kamen, and S. N. Whittlesey, *Research methods in biomechanics*. 2014.
- [42] S. Mohammad, P. Firoozabadi, and M. Oskoei, “A Human-Computer Interface based on Forehead Multi Channel Bio-signals to Control a Virtual Wheelchair,” *Proceedings of the 14th Iranian Conference on Biomedical Engineering (ICBME)*, pp. 272–277, 2008.
- [43] M. B. I. Raez, M. S. Hussain, and F. Mohd-Yasin, “Techniques of EMG signal analysis: detection, processing, classification and applications.,” *Biological procedures online*, vol. 8, pp. 11–35, 2006.
- [44] C.-N. Huang, C.-H. Chen, and H.-Y. Chung, “The Review of Applications and Measurements in Facial Electromyography,” *Journal of Medical and Biological Engineering*, vol. 25, no. 1, pp. 15–20, 2004.
- [45] N. Meziane, J. G. Webster, M. Attari, and A. J. Nimunkar, “Dry electrodes for electrocardiography.,” *Physiological measurement*, vol. 34, no. 9, pp. R47-69, Sep. 2013.
- [46] I. Kotsia and I. Pitas, “Facial Expression Recognition in Image Sequences Using Geometric Deformation Features and Support Vector Machines,” *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 172–187, Jan. 2007.
- [47] B. Fasel and J. Luetttin, “Automatic facial expression analysis: a survey,” *Pattern Recognition*, vol. 36, no. 1, pp. 259–275, Jan. 2003.
- [48] K. Tamee, K. Chaiwong, K. Yothapakdee, and P. P. Yupapin, “Muscle sensor model using small scale optical device for pattern recognitions.,” *TheScientificWorldJournal*, vol. 2013, p. 346047, Oct. 2013.

- [49] S. B. Ryan, K. L. Detweiler, K. H. Holland, M. A. Hord, and V. Bracha, "A long-range, wide field-of-view infrared eyeblink detector," *Journal of Neuroscience Methods*, vol. 152, no. 1–2, pp. 74–82, Apr. 2006.
- [50] K. Yothapakdee, K. Tamee, and P. P. Yupapin, "Muscle Sensing Device Design Using a PANDA Ring Resonator System," *International Journal of Electronics and Electrical Engineering*, vol. 4, no. 3, pp. 273–276, 2016.
- [51] L. Mayaud, S. Filipe, L. Pétégnef, O. Rochecouste, and M. Congedo, "Robust Brain-computer Interface for virtual Keyboard (RoBIK): Project results," *IRBM*, vol. 34, no. 2, pp. 131–138, Apr. 2013.
- [52] M. Brown *et al.*, "ISCEV Standard for Clinical Electro-oculography (EOG) 2006.," *Documenta ophthalmologica. Advances in ophthalmology*, vol. 113, no. 3, pp. 205–12, Nov. 2006.
- [53] A. B. Usakli and S. Gurkan, "Design of a Novel Efficient Human–Computer Interface: An Electrooculogram Based Virtual Keyboard," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 8, pp. 2099–2108, Aug. 2010.
- [54] H. Zeng and Y. Zhao, "Sensing Movement: Microsensors for Body Motion Measurement," *Sensors*, vol. 11, no. 1, pp. 638–660, Jan. 2011.
- [55] G. Gautschi, *Piezoelectric Sensorics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.
- [56] T. Bodaghi, M. R. Karami, and H. Jazayeriy, "VIWA: Computer Interface Device For Paralyzed People Using Breath Pressure," *Biomedical Engineering: Applications, Basis and Communications*, vol. 28, no. 01, p. 1650004, Feb. 2016.
- [57] T. Bodaghi, M. R. Karami, and H. Jazayeriy, "Turning breath into words – new device unveiled to give paralysis victims a voice," 2015, [Online], Available: <https://phys.org/news/2015-08-words-device-unveiled-paralysis-victims.html> [Accessed: 11-May-2019].

- [58] S. M. Hosni, H. A. Shedeed, M. S. Mabrouk, and M. F. Tolba, "EEG-EOG based Virtual Keyboard: Toward Hybrid Brain Computer Interface," *Neuroinformatics*, Oct. 2018.
- [59] "Peripheral Neuropathy Fact Sheet | National Institute of Neurological Disorders and Stroke," 2018, [Online], Available: <https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Peripheral-Neuropathy-Fact-Sheet> [Accessed: 11-May-2019].
- [60] "Fundamental Physics Prize Foundation Inaugural Prize Ceremony," 2013, [Online], Available: <https://www.gettyimages.ca/detail/news-photo/stephen-hawking-laureate-of-2013-physics-frontiers-prize-news-photo/164169007> [Accessed: 07-May-2019].
- [61] D. L. Grover, M. T. King, and C. A. Kushler, "Reduced keyboard disambiguating computer," 26-Jul-1995.
- [62] L. Boggess, "Two Simple Prediction Algorithms to Facilitate Text Production," in *Second Conference on Applied Natural Language Processing*, 1988.
- [63] T. Masui and Toshiyuki, "An efficient text input method for pen-based computers," in *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '98*, 1998, pp. 328–335.
- [64] S. Inverso, N. Hawes, J. Kelleher, R. Allen, and K. Haase, "Think and Spell: Context-Sensitive Predictive Text for an Ambiguous Keyboard Brain-Computer Interface Speller," 2019.
- [65] J. Li and G. Hirst, "Semantic knowledge in word completion," in *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility - Assets '05*, 2005, p. 121.
- [66] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

- [67] J. Gong and Jun, "Semantic & syntactic context-aware text entry methods," in *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility - Assets '07*, 2007, p. 261.
- [68] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, Mar. 1980.
- [69] I. S. MacKenzie and K. Tanaka-Ishii, "Chapter 5 - Text Entry Using a Small Number of Buttons," in *Text Entry Systems*, Morgan Kaufmann, 2007, pp. 105–121.
- [70] R. William Soukoreff and I. Scott Mackenzie, "Theoretical upper and lower bounds on typing speed using a stylus and a soft keyboard," *Behaviour & Information Technology*, vol. 14, no. 6, pp. 370–379, Nov. 1995.
- [71] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement.," *Journal of Experimental Psychology*, vol. 47, no. 6, pp. 381–391, 1954.
- [72] R. Hyman, "Stimulus information as a determinant of reaction time.," *Journal of Experimental Psychology*, vol. 45, no. 3, pp. 188–196, 1953.
- [73] Y.-C. Huang and F.-G. Wu, "Visual and manual loadings with QWERTY-like ambiguous keyboards: Relevance of letter-key assignments on mobile phones," *International Journal of Industrial Ergonomics*, vol. 50, pp. 143–150, Nov. 2015.
- [74] J. Arnott and M. Javed, "Probabilistic character disambiguation for reduced keyboards using small text samples," *Augmentative and Alternative Communication*, vol. 8, no. 3, pp. 215–223, Jan. 1992.
- [75] G. W. Lesh, B. J. Moulton, and D. J. Higginbotham, "Optimal character arrangements for ambiguous keyboards.," *IEEE transactions on rehabilitation engineering: a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 6, no. 4, pp. 415–23, Dec. 1998.

- [76] S. H. Levine, G. C. Trepagnier, C. O. Getschow, and S. L. Minneman, "Multi-character key text entry using computer disambiguation," in *RESNA 10th Annual Conference, San Jose, California*, 1987, pp. 177–178.
- [77] R. Foulds, M. Soede, and H. van Balkom, "Statistical disambiguation of multi-character keys applied to reduce motor requirements for augmentative and alternative communication," *Augmentative and Alternative Communication*, vol. 3, no. 4, pp. 192–195, Jan. 1987.
- [78] A. Pavlovych and W. Stuerzlinger, "Less-Tap: A Fast and Easy-to-learn Text Input Technique for Phones," in *Proceedings - Graphics Interface*, 2003, pp. 97–104.
- [79] K. C. F. Tan, E. Ng, and J. J. S. Oh, "Development of a Qwerty-Type Reduced Keyboard System for Mobile Computing: Tengo," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 47, no. 6, pp. 855–859, Oct. 2003.
- [80] I. S. Mackenzie and T. Felzer, "SAK: Scanning ambiguous keyboard for efficient one-key text entry," *ACM Transactions on Computer-Human Interaction*, vol. 17, no. 3, pp. 1–39, Jul. 2010.
- [81] J. Miró and P. A. Bernabeu, "Text Entry System Based on a Minimal Scan Matrix for Severely Physically Handicapped People," in *Computers Helping People with Special Needs*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1216–1219.
- [82] R. C. Simpson and H. H. Koester, "Adaptive one-switch row-column scanning," *IEEE Transactions on Rehabilitation Engineering*, vol. 7, no. 4, pp. 464–473, 1999.
- [83] R. I. Damper, "Text composition by the physically disabled: a rate prediction model for scanning input.," *Applied ergonomics*, vol. 15, no. 4, pp. 289–96, Dec. 1984.

- [84] J. A. Doubler, D. S. Childress, and J. S. Stryzik, "A microcomputer-based control and communication system for the severely disabled," *ACM SIGCAPH Computers and the Physically Handicapped*, no. 24, pp. 43–46, Oct. 1978.
- [85] M. Baljko and A. Tam, "Indirect text entry using one or two keys," in *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility - Assets '06*, 2006, p. 18.
- [86] Y.-L. Lin, T.-F. Wu, M.-C. Chen, Y.-M. Yeh, and H.-P. Wang, "Designing a Scanning On-Screen Keyboard for People with Severe Motor Disabilities," in *Computers Helping People with Special Needs*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1184–1187.
- [87] T. Felzer, R. Nordmann, and S. Rinderknecht, "Scanning-Based Human-Computer Interaction Using Intentional Muscle Contractions," Springer, Berlin, Heidelberg, 2009, pp. 509–518.
- [88] S. Bhattacharya, D. Samanta, and A. Basu, "User errors on scanning keyboards: Empirical study, model and design principles," *Interacting with Computers*, vol. 20, no. 3, pp. 406–418, May 2008.
- [89] S. Bhattacharya, D. Samanta, and A. Basu, "Performance Models for Automatic Evaluation of Virtual Scanning Keyboards," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, no. 5, pp. 510–519, Oct. 2008.
- [90] P. E. Jones, "Virtual keyboard with scanning and augmented by prediction," in *2nd International Conference on Disability, Virtual Reality and Associated Technologies*, 1998, pp. 45–51.
- [91] F. Shein, G. Hamann, N. Brownlow, J. Treviranus, M. Milner, and P. Parnes, "WiViK: A visual keyboard for Windows 3.0," *Proc. RESNA'91*, pp. 160–162, 1991.

- [92] I. S. MacKenzie, H. Kober, D. Smith, T. Jones, and E. Skepner, "LetterWise: prefix-based disambiguation for mobile text input," in *Proceedings of the 14th annual ACM symposium on User interface software and technology - UIST '01*, 2001, p. 111.
- [93] J. Gong and P. Tarasewich, "Alphabetically constrained keypad designs for text entry on mobile devices," in *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '05*, 2005, p. 211.
- [94] X. Bi, B. A. Smith, and S. Zhai, "Quasi-qwerty soft keyboard optimization," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010, pp. 283–286.
- [95] M. D. Dunlop and M. Montgomery Masters, "Investigating five key predictive text entry with combined distance and keystroke modelling," *Personal and Ubiquitous Computing*, vol. 12, no. 8, pp. 589–598, Nov. 2008.
- [96] S.-W. Yang, C.-S. Lin, S.-K. Lin, and C.-H. Lee, "Design of virtual keyboard using blink control method for the severely disabled," *Computer Methods and Programs in Biomedicine*, vol. 111, no. 2, pp. 410–418, 2013.
- [97] B. Martin and I. Pecci, "État de l'art des claviers physiques et logiciels pour la saisie de texte State of art of physical and software keyboards for text entry."
- [98] I. Schadle, B. Le Pévédic, J.-Y. Antoine, and F. Poirier, "Prédiction de lettre pour l'aide à la saisie de texte."
- [99] K. Trnka, J. McCaw, D. Yarrington, K. F. McCoy, and C. Pennington, "User Interaction with Word Prediction," *ACM Transactions on Accessible Computing*, vol. 1, no. 3, pp. 1–34, Feb. 2009.
- [100] Francis Leboutte, "Clavier Dvorak-fr: Accueil," [Online], Available: <http://www.algo.be/ergo/dvorak-fr.html> [Accessed: 07-Jun-2019].
- [101] R. Cassingham, "The Dvorak Keyboard," [Online], Available: <https://www.dvorak-keyboard.com/> [Accessed: 13-Jun-2019].

- [102] M. N. Ltd., “XPeRT Keyboard,” [Online], Available: <http://www.xpertkeyboard.com/>.
- [103] Y. Guerrier, C. Kolski, and F. Poirier, “Étude comparative entre clavier virtuel de type AZERTY et K-Hermès, destinés à des utilisateurs souffrant d’une Infirmité Motrice Cérébrale Uniwatch View project HCI engineering integrated with capability maturity models View project,” in *2ème Conférence Internationale Sur l’Accessibilité et les Systèmes de Suppléance aux personnes en situations de Handicap*, 2011, pp. 148–155.
- [104] S. Zhai, M. Hunter, and B. A. Smith, “The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design,” in *Proceedings of the 13th annual ACM symposium on User interface software and technology - UIST ’00*, 2000, pp. 119–128.
- [105] T. Wandmacher, J.-Y. Antoine, and F. Poirier, “SIBYLLE: a system for alternative communication adapting to the context and its user,” in *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility - Assets ’07*, 2007, p. 203.
- [106] D. J. Ward, A. F. Blackwell, and D. J. C. MacKay, “Dasher: A Gesture-Driven Data Entry Interface for Mobile Computing,” *Human–Computer Interaction*, vol. 17, no. 2–3, pp. 199–228, 2002.
- [107] “Inference Group: Dasher Project: How does Dasher work?,” 2016, [Online], Available: <http://www.inference.org.uk/dasher/DasherSummary2.html> [Accessed: 13-Jun-2019].
- [108] M. Raynal, “KeyGlasses : Des touches semi-transparentes pour optimiser la saisie de texte,” 2019.
- [109] A. Grange and Aristide, “L’interface du clavier virtuel Chewing Word,” in *Conference Internationale Francophone sur l’Interaction Homme-Machine on - IHM ’10*, 2010, p. 237.

- [110] Y. Guerrier, M. Baas, C. Kolski, and F. Poirier, “Comparative Study between AZERTY-Type and K-Hermes Virtual Keyboards Dedicated to Users with Cerebral Palsy,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6766 LNCS, no. PART 2, Springer, Berlin, Heidelberg, 2011, pp. 310–319.
- [111] M. Baas, Y. Guerrier, C. Kolski, and F. Poirier, “Système de saisie de texte visant à réduire l’effort des utilisateurs à handicap moteur,” in *Proceedings of the Ergonomie et Informatique Avancee Conference on - Ergo’IA ’10*, 2010, p. 19.
- [112] Y. Guerrier, M. Baas, C. Kolski, and F. Poirier, “Étude comparative entre clavier un virtuel AZERTY et un clavier multitap pour des utilisateurs souffrant d’une Infirmité Motrice Cérébrale de type tétraplégique ...,” in *Conférence ASSISTH 2011*, 2011.
- [113] D. Sauzin, F. Vella, and N. Vigouroux, “SOKEYTO V2: A toolkit for designing and evaluating virtual keyboards,” *Assistive Technology Research Series*, vol. 33, pp. 939–945, 2013.
- [114] D. McDuff, A. Mahmoud, M. Mavadati, M. Amr, J. Turcot, and R. el Kaliouby, “AFFDEX SDK: A Cross-Platform Real-Time Multi-Face Expression Recognition Toolkit,” in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA ’16*, 2016, pp. 3723–3726.
- [115] Newegg, “Raspberry Pi 3 Model B+,” [Online], Available: <https://www.newegg.ca/Product/Product.aspx?Item=N82E16813142011> [Accessed: 07-May-2019].
- [116] “Noyokere External USB AUDIO Sound Card Adapter Virtual 7.1 ch USB 2.0 Mic Speaker Audio Headset Microphone 3.5mm Jack Converter,” [Online], Available: <https://www.aliexpress.com/item/External-USB-AUDIO-SOUND-CARD-ADAPTER-VIRTUAL-7-1-ch-USB-2-0-Mic-Speaker-Audio/32727980558.html> [Accessed: 07-May-2019].

- [117] "Audio Streaming Server and Software | Wowza," [Online], Available: <https://www.wowza.com/audio-only-streaming> [Accessed: 11-May-2019].
- [118] "12V 1 Channel Red Color Relay Module," [Online], Available: [http://bdspeedytech.com/index.php?route=product/product&product\\_id=2515](http://bdspeedytech.com/index.php?route=product/product&product_id=2515) [Accessed: 07-May-2019].
- [119] A. J. Sporka, T. Felzer, S. H. Kurniawan, O. Poláček, P. Haiduk, and I. S. MacKenzie, "CHANTI: Predictive Text Entry Using Non-verbal Vocal Input," in *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, 2011, p. 2463.
- [120] Q. Yuan, G. Cong, and N. M. Thalmann, "Enhancing naive bayes with various smoothing methods for short text classification," in *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*, 2012, p. 645.
- [121] F. Peng and D. Schuurmans, "Combining Naive Bayes and n-Gram Language Models for Text Classification," Springer, Berlin, Heidelberg, 2003, pp. 335–350.
- [122] Z. Xiaojin and R. Rosenfeld, "Improving trigram language modeling with the World Wide Web," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 1, pp. 533–536.
- [123] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Computer Speech & Language*, vol. 13, no. 4, pp. 359–394, Oct. 1999.
- [124] A. Hazem and E. Morin, "A Comparison of Smoothing Techniques for Bilingual Lexicon Extraction from Comparable Corpora," in *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, 2013, pp. 24–33.
- [125] M. J. Mendenhall, A. S. Nunez, and R. K. Martin, "Human skin detection in the visible and near infrared," *Applied Optics*, vol. 54, no. 35, p. 10559, Dec. 2015.