

CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

1-TUI: MOBILE USER INTERFACE FOR BLIND USERS

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

by

Won Chang

May 2015

Copyright © 2015 Won Chang

All rights reserved

The thesis of Won Chang is approved:

Dr. G. Michael Barnes	Date
-----------------------	------

Dr. George Youssef	Date
--------------------	------

Dr. Ani Nahapetian, Chair	Date
---------------------------	------

Dedication

I would like to thank my parents for supporting me throughout my academic pursuit. Without their support, I would never have dared to change my career to Computer Science. Now I do what I love for a living, which I will always be grateful for.

I also like to thank Prof. Nahapetian, Prof. Barnes, and Prof. Youssef for their input. Prof. Nahapetian guided me through this project and kept me in track. I would not have been able to finish my thesis in time without her. Prof. Barnes revealed errors in my project early on, which I failed to see. Without his insight, these errors would have cost me greatly later on. Prof. Youssef exposed me to a new technology which would have made this project far superior than its current form. I only wish that I had more experience in circuits to apply it to the project.

Table of Contents

Copyright.....	ii
Signature Page.....	iii
Dedication.....	iv
ABSTRACT.....	xi
1 INTRODUCTION.....	1
2 BACKGROUND.....	3
2.1 Braille ^{[3][4]}	3
2.2 User Interface.....	4
2.2.1 Braille Display.....	4
2.2.2 Text to Speech.....	5
2.2.3 Screen Reader.....	6
3 RELATED WORK.....	7
3.1 Haptic Feedback.....	7
3.2 Read Speed.....	7
3.3 Possible 1-TUID's.....	8
4 USER INTERFACE OVERVIEW.....	9
4.1 1-TUID.....	9
4.2 1-TUI: Reading.....	10
4.3 1-TUI: Interactive Objects.....	11
4.4 1-TUI: Writing.....	12
5 SYSTEM OVERVIEW.....	15
6 IMPLEMENTATION.....	16

6.1 1-TUID.....	16
6.2 Braille Library.....	18
6.2.1 APOS Machine: Apostrophe.....	21
6.2.2 DBLQT Machine: Double Quotation.....	22
6.2.3 PNCT Machine: Punctuation.....	22
6.2.4 LTR Machine: Letter Indicator.....	22
6.2.5 NUM Machine: Number Digits.....	23
6.2.6 CAP Machine: Capital Letters.....	24
6.2.7 CNTR Machine: Contraction(Grade 2).....	24
6.3 1-TUI API.....	27
6.3.1 1-TUI Objects.....	27
6.3.2 1-TUID Objects.....	29
6.3.3 Event Protocols.....	30
6.4 Android Applications.....	30
6.4.1 TUIReader.....	31
6.4.2 TUICaller.....	32
6.4.3 TUISMS.....	34
6.4.4 TUILauncher.....	36
7 VALIDATION AND RESULTS.....	37
7.1 1-TUI Validation.....	37
7.1.1 System Testing: Printed Braille vs 1-TUI.....	37
7.1.2 System Testing: Screen Reader vs. 1-TUI.....	39
7.2 Braille Translator Benchmark.....	41

8 CONCLUSION.....	44
REFERENCES.....	45

List of Tables

Table 2-1 . UEB Alphabets.....	3
Table 2-2 . UEB Grade 2 Examples.....	3
Table 2-3 . Apostrophe in Braille.....	4
Table 7-1 . Braille Read Comparison.....	38
Table 7-2 . TalkBack vs TUISMS.....	39

List of Figures

Figure 2-1 . A Braille Display.....	5
Figure 4-1 . 1-TUI App Use Case Overview.....	9
Figure 4-2 . 1-TUID General Design.	10
Figure 4-3 . Reading in 1-TUI.....	11
Figure 4-4 . Writing in 1-TUI.....	14
Figure 5-1 . Project Overview.....	15
Figure 6-1 . 1-TUID Test Unit Design.....	16
Figure 6-2 . 1-TUID Test Unit Circuit.....	18
Figure 6-3 . 1-TUID Test Unit.....	18
Figure 6-4 . Braille Library Overview.....	19
Figure 6-5 . APOS Machine.....	21
Figure 6-6 . LTR Machine.....	23
Figure 6-7 . CAP Machine.....	24
Figure 6-8 . Partial CNTR Machine.....	26
Figure 6-9 . 1-TUI Objects.....	27
Figure 6-10 . TUICellView.....	28
Figure 6-11 . 1-TUI Message Protoco.....	30
Figure 6-12 . TUIReader.....	31
Figure 6-13 . Making Call.....	33
Figure 6-14 . Incoming Call.....	34
Figure 6-15 . TUISMS.....	35
Figure 6-16 . TUILauncher.....	36
Figure 7-1 . Word Count vs Translation Speed.....	42

Figure 7-2 . Word Count vs Size Increase..... 43

ABSTRACT

1-TUI: MOBILE USER INTERFACE FOR BLIND USERS

By

Won Chang

Master of Science in Computer Science

With the rise of iPhone and Android, computing devices became more accessible and mobile for the general users. In an attempt to help the blind users, few User Interfaces (UI) such as text to speech, screen reader, and Braille display are developed. However, these UI's are either unsuitable for technical reading/writing or too expensive. This thesis solves these issues by combining screen reader and Braille display to create a new UI. Named 1-Cell Tactile User Interface (1-TUI), the proposed UI retains the key features of both Braille display and screen readers such as technical read/write, read-flow control, 2-dimensional navigation, reliable user input, and lower cost.

This project implements 1-TUI by developing a Braille Translator Library, an 1-TUI API for Android, an 1-TUID test unit, and Android applications using 1-TUI API. The Android applications are used for testing so the usability of 1-TUI can be compared against the original UI's. Due to the scarcity of Braille users, a system test is done with a beginner-level Braille user. The test reveals that reading in 1-TUI is 72.17% faster and 25.97% more accurate than reading in printed Braille when used by Braille beginners. The test also indicates that 1-TUI is 150.00% slower than screen readers, but further

analysis suggests that this difference would be reduced to 9.99% if the subject is fluent in Braille.

1 INTRODUCTION

According to Cornell University, there are approximately 6,670,300 people with visual disability in U.S. as of 2012 (including those with low visions)^[1]. Based on a 2014 report by American Printing House for the Blind, at least 17.68% of them rely on tactile or auditory readers^[2]. For them, the current mobile devices are less approachable because these devices use Graphical User Interface (GUI), which depends on visual coordination.

To make the mobile devices more accessible, many mobile Operating Systems (OS) provide features such as text to speech and screen reader. The best of the two, screen reader provides 2-dimensional control with reliable input methods. Both iOS and Android offer it for free, and it is faster than tactile reading. However, it lacks read-flow control as the users cannot easily read back and forth between certain parts of a text. Also, these audio solutions cannot describe technical expressions well. Such issues become more than an inconvenience if the user is reading for education or work, which require constant back-and-forth reading and accurate description of the content.

Another available UI is Braille display, which outputs Braille characters so the user can read by touching. It provides text-based navigation with a reliable input via keyboard, and since the user is in charge of reading, s/he has much more read-flow control than with screen reader. Most importantly, it is capable of describing technical expressions through Braille. However, Braille displays cost from \$500 to \$3000 and are too large to be used for mobile purposes.

As a solution, this project proposes a new method named 1-Cell Tactile User Interface (1-TUI). 1-TUI is a mixture of screen reader and Braille display that seeks to strengthen their weakness. Similar to a screen reader, 1-TUI outputs whatever the user

touches on a touchscreen. But instead of using audio output, 1-TUI uses tactile outputs via a tactile device named 1-TUI Device (1-TUID). With this method, the tactile device can achieve the same read speed as the other Braille displays using just one cell, because the reading process in Braille is cell-by-cell. 1-TUI also uses haptic feedback to help the user to identify objects on the screen and navigate. As a result, 1-TUI inherits 2D navigation and reliable input methods from screen reader. Since 1-TUI uses tactile output, 1-TUI is also capable of technical reading and writing via Braille. Most importantly, 1-TUID is much cheaper than the other tactile displays since it requires only one cell instead of many. By partially sacrificing the screen reader's speed and cost, 1-TUI achieves to retain technical reading/writing and 2D navigation while reducing the cost and size of the extra hardware.

This project implements 1-TUI by using Braille, such that 1-TUID displays a Braille character that the user is touching on screen. It develops an 1-TUID test unit, a Braille Translator, an 1-TUI API, and four 1-TUI applications. It then performs tests to verify the usability of 1-TUI.

2 BACKGROUND

2.1 Braille^{[3][4]}

Braille is a tactile reading and writing system created by Louis Braille in 1824. It either uses 6 or 8 dots for each symbol, but 6-dot Braille is the more predominately used of two. As of 2013, a survey by UNESCO reveals that 142 countries use Braille in 133 different languages^[5]. To unify some of these languages, ICEB standardizes English Braille for all English-speaking countries in 2013, resulting in The Rules of Unified English Braille(UEB). As such, this thesis uses UEB instead of the regular English Braille.





























 a	 b	 c	 d	 e	 f	 g	 h	 i	 j
 k	 l	 m	 n	 o	 p	 q	 r	 s	 t
 u	 v	 w	 x	 y	 z	 NUM	 CAP		

Table 2-1. UEB Alphabets

Shown on Table 2-1 is UEB table for Grade 1 alphabet. In Grade 1, English alphabets are mapped one-to-one, while letters ‘a’ to ‘j’ are also used for number digits 0 to 9 by following ‘NUM’ indicator. UEB also uses ‘CAP’ indicator in front of letters if the letters are in uppercase. Although such conversion is fast and easy, the translated Braille sequence takes large space due to the size of each cell. In order to minimize the space, shorten symbols called contractions are used by Grade 2.











 and	 ch	 but	 wh	 mother	 afternoon	 day	 here	 be	 in
--	---	--	---	---	--	---	---	---	---

Table 2-2. UEB Grade 2 Examples

Shown on Table 2-2 are few of UEB Grade 2 contractions. UEB defines about 160 contractions, but there are more that are informally used by smaller groups or individuals (Grade 3). UEB defines different rules for different contractions, but its most consideration is on whether a contraction is a stand-alone or not. For example, UEB dictates that contraction for ‘but’ be used only if it is stand-alone, or not as a part of a word. On the other hand, contraction for ‘day’ can be used as non-stand-alone to form longer words such as ‘Monday’.





 single quot open	 single quot close	 prime	 possessive
--	---	--	---

Table 2-3. Apostrophe in Braille

Also, note that Braille is contextual. For many punctuation, UEB uses different Braille symbols for a same English symbol based on its context. Such case is best demonstrated by the use of apostrophe symbol, as shown in Table 2-3. Regular English uses apostrophe symbol for open/close single quotation, prime sign, and possessive mark. In contrast, UEB uses different symbols for open single quotation, close single quotation, prime, and possessive mark.

2.2 User Interface

As of today, the blind users have three options to use mobile devices: Braille display, text to speech, and screen reader.

2.2.1 Braille Display

The oldest of the three options, Braille display has been used with desktops before extending itself to mobile devices. As shown on Figure 2-1, a Braille display typically has 20 to 40 cells, and also acts as Braille keyboard. It requires special software to display the texts on screen and output commands. Such software for mobile device are

VoiceOver(iOS) and BrailleBack(Android). They both support various Braille display models, and set up special commands that the user can quickly enter using the Braille display. Compared to other two options, Braille Display is capable of describing technical expressions and has more read-control. However, these devices costs from \$500 to \$3000 due to having 20 to 40 cells tightly packed together, where each cell requires 8 mechanical pins. In addition, the size of Braille displays are at least 10 inches long. Given the nature of the mobile devices, having to carry such a large device is unattractive.



Figure 2-1. A Braille Display. A typical 8-dot Braille display with 24 Braille cells. Six buttons on either side of the board are used for typing. Braille displays like this cost between \$500 and \$3000.

2.2.2 Text to Speech

Used by both blind and non-blind users, text to speech focuses in providing a quick access to desired information. The basic feature of text to speech is for the device to output the text as an audio. For example, it can read a new text message or email, and it can also write the text through voice recognition. However, if the user wants to read a middle portion of the text only, s/he has to start from the beginning of the text. The user also has to disable other audios such as music while using text to speech. Also, the voice recognition is not a reliable technology, especially with names and jargon. According to a

research by Google in 2013, voice recognition still has word error rate(WER) of 15.1%^[6], which is not reliable for work or educational purposes.

2.2.3 Screen Reader

The last option, screen reader, is an extension of text to speech that reads any object that the user touches. Using this feature, the user can physically locate where the object is on screen. A survey by WebAIM in 2014 reveals that 82% of the blind web users use mobile screen readers, and that the most used mobile screen readers is VoiceOver(iOS)^[7]. Compare to text to speech, the user can type and navigate without speaking. Still, it does not give the user read-flow control if the text is long. Also, it is unsuitable for describing technical expressions because they involve scientific symbols and jargon.

1-TUI can be described as a product of the advantages of Braille display and screen reader. Taking a similar approach to screen readers, 1-TUI inherits 2D navigation which outputs what the user is touching. By incorporating this method, 1-TUI only requires 1 cell on its Braille display, which significantly reduces the cost of the hardware. By using a Braille display, 1-TUI provides more read-control and technical reading/writing to a mobile device.

3 RELATED WORK

3.1 Haptic Feedback

Haptic feedback is an important part of 1-TUI because it is one of few ways for an UI to interact with the blind users. For their wearable Optical Character Recognition device, Shilkrot and his team test different haptic feedback to help the users navigate through texts. Their result shows that 100% of test subjects find the strength and the pattern of vibration to be more helpful than using multiple vibrators^[10]. Based on their results, 1-TUI uses different haptic feedback patterns to identify different objects, while using different strength to differentiate vertical and horizontal finger movements.

3.2 Read Speed

Since 1-TUI's foundation is in tactile reading, the speed of tactile reading has a great influence on its usability. In their paper, Legge, Madison, and Mansfield report a median Braille read speed as 124 words per minute(wpm)^[8]. According to their data, reading in Grade 2 increases the speed by 41% because Grade 2 reduces the number of characters. For this reason, 1-TUI implementation uses Grade 2 Braille for long texts. Going further, Legge, Madison, and Mansfield report 28% speed increase when the readers use two hands for a quicker relocation of their finger. For a similar effect, 1-TUI outputs a haptic feedback whenever the user moves to a different line. By adding such feature, 1-TUI lets its user to easily move between lines.

On the other hand, a study by Asakawa, Takagi, Ino, and Ifukube suggests that the auditory reading can be as fast as 827 Morae per minute (Mpm) with at least 90% Recall Rate (accuracy)^[9]. The paper also converts 1300 Mpm(Japanese) to 500 wpm(English), which suggests that 827 Mpm converts to 318 wpm. This is far faster than the speed of

reading in Braille, which suggests that 1-TUI will be much slower than screen readers. Because the goal of 1-TUI is to provide a cheaper tactile solution with a better read-flow control and technical reading/writing, 1-TUI does not need to be faster than screen readers. Still, the speed and accuracy of screen reader and 1-TUI will be compared for better understanding of the two.

3.3 Possible 1-TUID's

The usefulness of 1-TUI depends on the development of 1-TUID. Fortunately, various researches have been made to create a portable Braille display. One of those researches is a paper by Saadeh^[11] in which he performs an extensive research in optimal tactile forces for a Braille display. His data shows that reading Braille without sliding requires around 0.786N, while reading Braille by sliding requires 0.431N in normal force and 0.418N in tangential force. On the other hand, Sekitani and his team develops a Braille display that is 1mm thick in 2007 by using Electroactive Polymer (EAP)^[12]. The display is capable of pushing up each Braille dot by 0.2mm using 10V. Using a same material an improvement is made by Koo and his team in 2008, where they manage to increase the displacement of dot to 0.4mm, but by using 1~3.5kV^[13].

There exists more researches and patents on portable 1-cell Braille displays. However, these works only focus in the hardware without any UI. Without UI, these devices do not provide the functions such as reading, writing, or navigating. 1-TUI provides the UI that they need to work with mobile devices, and thus, makes them available for public use.

4 USER INTERFACE OVERVIEW

The user interaction in 1-TUI involves two devices: the mobile device, and 1-TUID. The user uses the touchscreen on mobile device to touch the Braille character that s/he wants to read, and uses 1-TUID to actually read the Braille character. Shown on Figure 4-1 is an use case diagram for a general 1-TUI application.

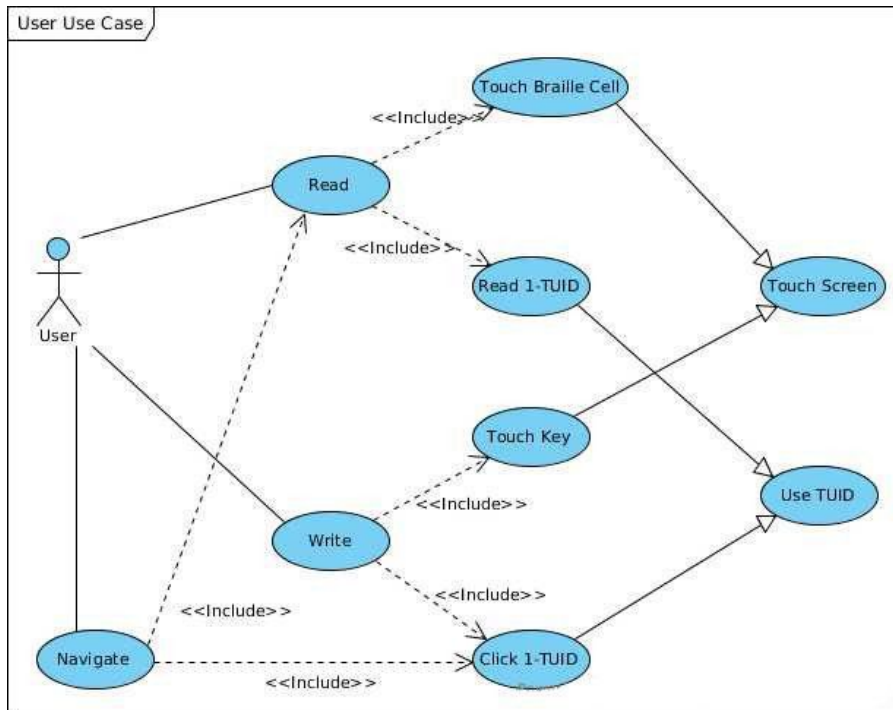


Figure 4-1. 1-TUI App Use Case Overview. Every user activity such as reading and writing involves using touchscreen and 1-TUID.

4.1 1-TUID

Shown on Figure 4-2 is an example of 1-TUID. 1-TUID is an 1-cell Braille display that can output haptic feedback and input button clicks. While the other Braille displays use small Braille cells in order to pack 20 to 40 cells into the hardware, 1-TUID has a larger cell since it only uses 1. Such design not only makes the hardware cheaper, but also makes the tactile reading much easier, and thus, faster.

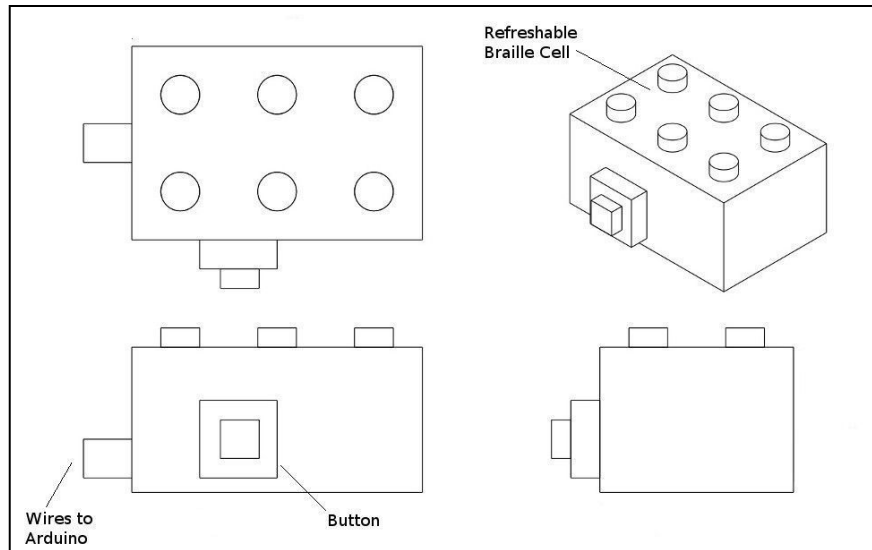


Figure 4-2. 1-TUID General Design. Any device can be used as an 1-TUID as long as it has: 1 Braille display, button, and vibrator. The device does not need to look like the one in figure.

4.2 1-TUI: Reading

Figure 4-3 shows how 1-TUI handles the reading process. Reading in 1-TUI works by touching a Braille character on screen, then tactile reading the output on 1-TUID. The user can continue reading by moving his/her finger on the screen.

A problem with such method is that the user cannot tell if s/he has moved to a next character or not if there is a sequence of a same character. Also, the user tends to slide down to other line without realizing it, causing a confusion. As shown on Figure 4-3(c)&(d), this issues are solved by using haptic feedback on the mobile device. Whenever the user moves to an another character, the mobile device gives a short vibration. As for the lines, each line has a boundary line at the bottom which causes a continuous vibration while the user touches it. Consequently, this solution also helps user to locate his/her finger on the screen.

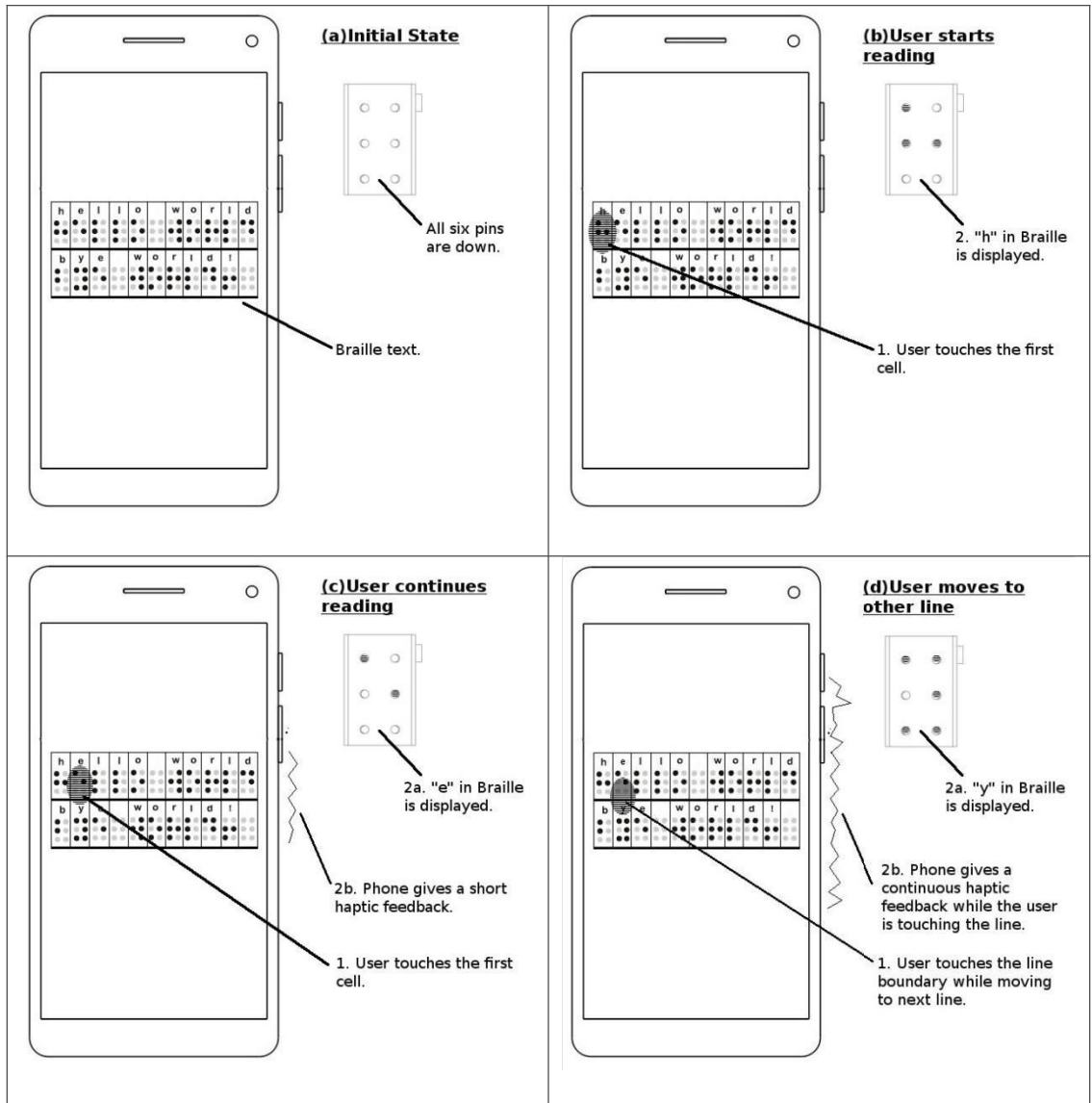


Figure 4-3. Reading in 1-TUI. (a) Initial state. (b) Reading a character. (c) Continuous reading, each character notified by a short haptic-feedback. (d) Moving to other line, mobile device vibrates while the user's finger is touching the line boundary.

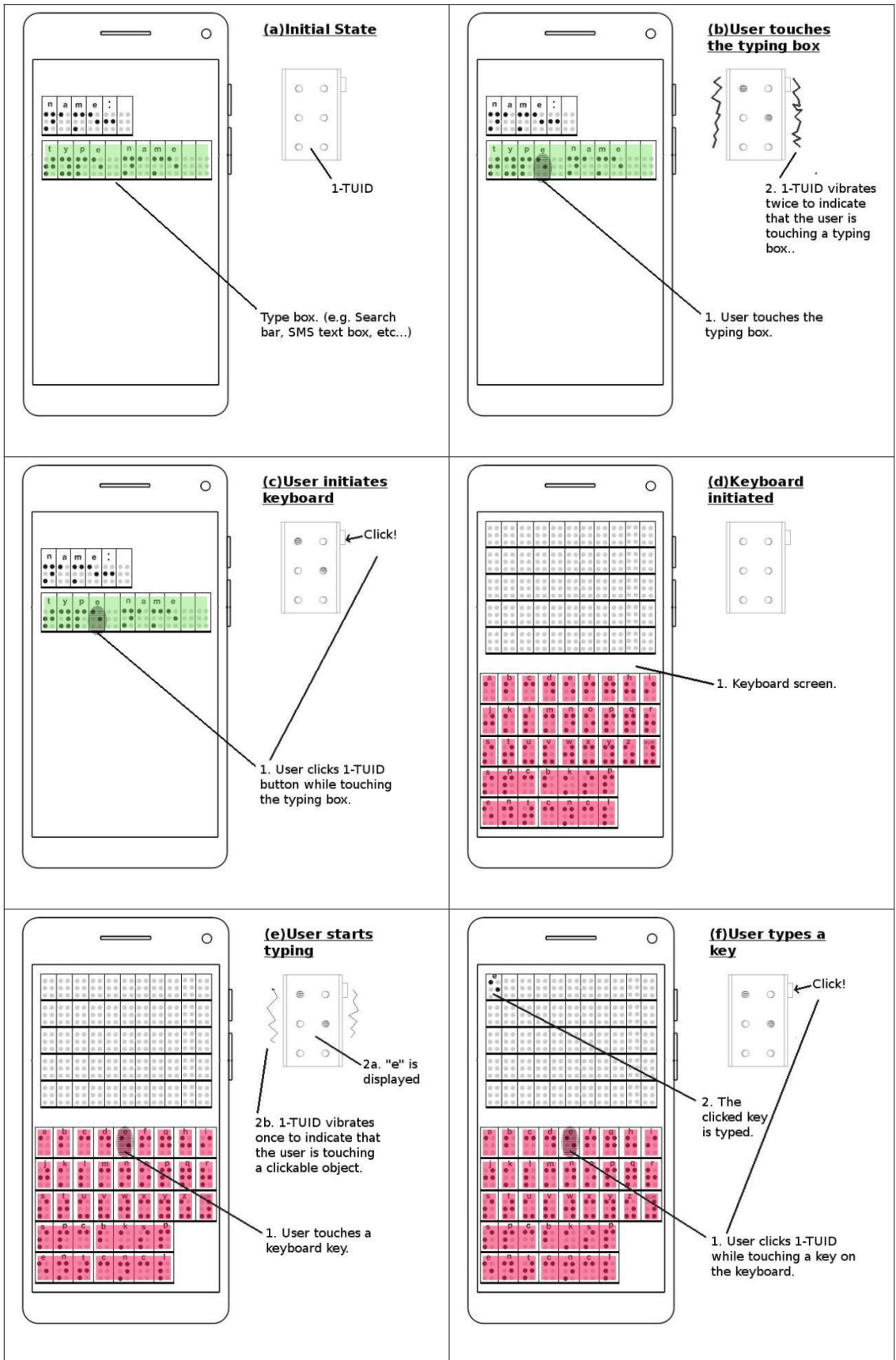
4.3 1-TUI: Interactive Objects

Because 1-TUI's intended users are blind, they need to be able to move around the screen without clicking or dragging objects. Therefore, 1-TUI disables interactive objects from reacting to touch events. Instead, they respond to the button click on 1-TUID, allowing the user to freely explore the screen without losing the interactivity.

Also, 1-TUI uses haptic feedback on 1-TUID to inform the user that s/he is touching an interactive object. Each object outputs different vibration so the user can identify what s/he is touching. It also uses different primary colors on these objects to support the users with low vision. Current implementation defines three different interactive objects: clickable object, textbox, and draggable object. A clickable object is colored red (see Figure 4-4(d)), and 1-TUID outputs a short vibration. A textbox is colored green (see Figure 4-4(a)), and 1-TUID outputs two short vibration. A draggable object is colored blue, and 1-TUID outputs a long vibration upon touch, and a continuous vibration while the object is being dragged.

4.4 1-TUI: Writing

Shown on Figure 4-4 is the writing process in 1-TUI. Writing in 1-TUI follows a same process as regular mobile typing. The user locates a textbox when 1-TUID gives two short vibrations, clicks 1-TUID to call a keyboard, types a text, and presses enter. The textbox in 1-TUI is shown in Figure 4-4(a), and behaves as described in the previous section. Shown in Figure 4-4(d)-(g), 1-TUI keyboard works similar to regular soft keyboards, except that all the interactive actions are handled by 1-TUID. In fact, it is just a collection of clickable buttons that works as described in previous section. Unfortunately due to the Braille using large spaces, 1-TUI keyboard only supports full screen keyboard.



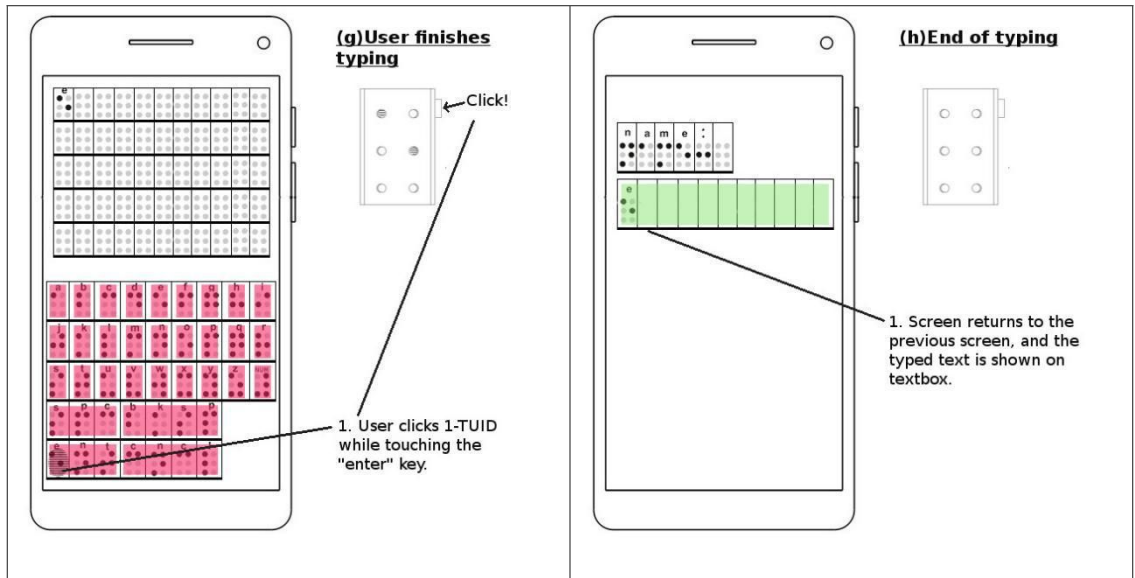


Figure 4-4. Writing in 1-TUI. (a) Initial state. (b) User finds the textbox. (c)~(d) User calls keyboard. (e)~(g) User types in a character. (h) User is back to the previous screen, and the textbox now displays the character that s/he typed in.

5 SYSTEM OVERVIEW

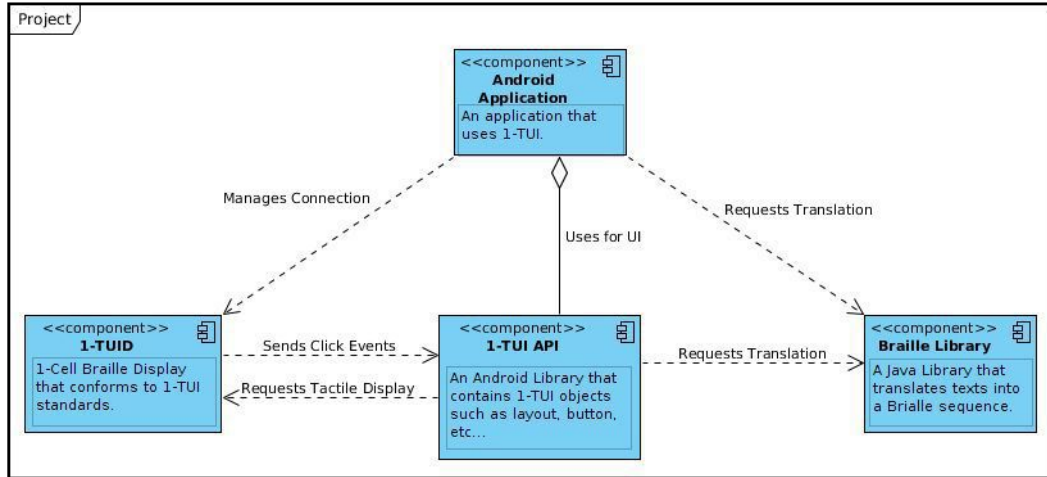


Figure 5-1. Project Overview

This project is divided into four parts: Braille Translator Library, 1-TUI API, 1-TUID test unit, and 1-TUI Android applications(See Figure 5-1).

In Figure 5-1, 1-TUID represents an actual 1-TUID hardware. It uses wireless communication to exchange messages with the 1-TUI objects from the 1-TUI Package. The hardware is programmed and controlled by Arduino Uno controller.

Braille Library is a Java library that performs Braille translation. It also defines data types that represent Braille cell and Braille language type. Since it is a stand-alone library, the Braille library is available for any other Java applications.

1-TUI API is an Android library for the developers who wish to create an application with 1-TUI. It contains basic UI objects that are converted to 1-TUI standard. It also provides a Bluetooth communicator interface that lets the 1-TUI objects to relay information with Android application and 1-TUID.

The last component, Android application, is an application that uses 1-TUI as its UI. For this project, there are Text Reader, Caller, SMS, and Launcher. In the most of cases, an application needs to manage the connection between 1-TUID and 1-TUI objects.

6 IMPLEMENTATION

6.1 1-TUID

Since the focus of this project is in developing an UI, only a test unit is built for 1-TUID (shown in Figure 6-1). The test unit composes of Arduino Uno, HC-05 Bluetooth module, 6 micro motors, a micro vibrator, and a button. Although much smaller and cheaper micro controllers exist, Arduino Uno is used because it is much easier to load the program. On the other hand, HC-05 module is chosen purely due to its lower cost, so it can be replaced by a smaller module if needed.

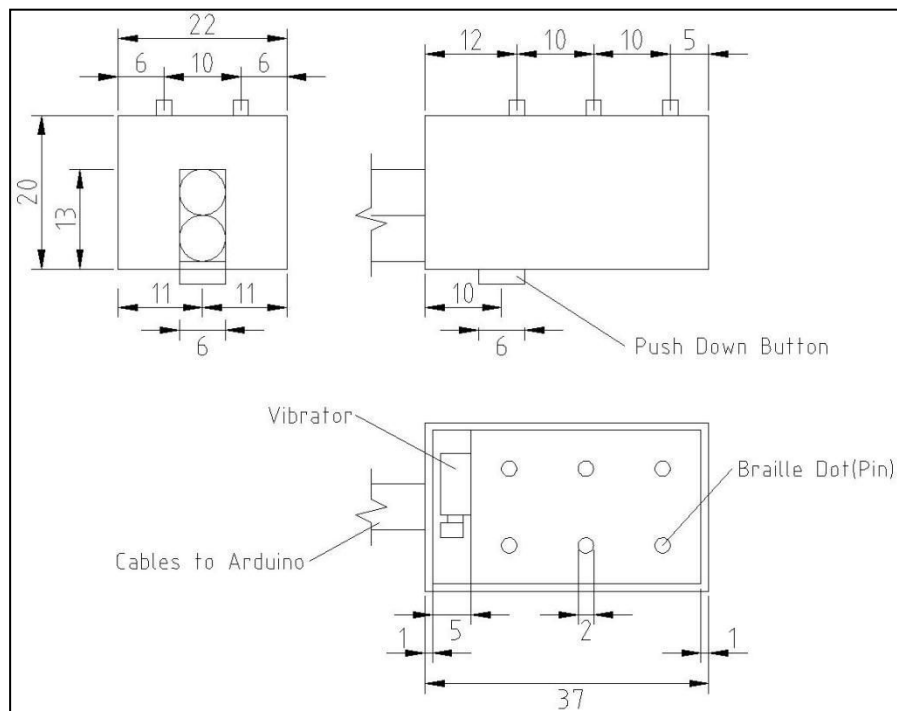


Figure 6-1. 1-TUID Test Unit Design.

For the tactile display mechanism, various methods were experimented with. The first mechanism was piezo discs, which can generate small vibration for each dot. It was deemed an idealistic solution because it requires less power and is very small. However, the experiment with the disc revealed that each disc requires 200Hz to 300Hz frequency

with 35V to 45V. Since Arduino Uno cannot generate such wave with high voltage, the piezo disc was dropped.

As a result, the vibration source was changed to micro vibrators, which are bigger but easy to control. Various motor strength were tested with various materials to isolate each vibration from one another. However, each trial failed because the vibrators were either too strong or too weak against the materials. Also, a continuous vibration on finger caused numbness and distraction. Therefore, the attempt of using vibration for a Braille dot stopped.

The final decision was made with micro motors. The mechanism works by using motor shaft as a Braille dot. The user can determine which Braille dots are up by checking which motors are on. Various experiments confirmed that this method can be used to represent a Braille character. Although not as refined as the other mechanisms used by commercial Braille displays, this mechanism was chosen since the goal is to make a test unit.

Figure 6-2 shows the circuit schematics of the final 1-TUID test unit. Note that the labels “P#” refer to Arduino in/out pins. P0, P1, and P10 are used as signal in/out, whereas the controller uses the other pins as power on connected components. Also, the power source shown in the figure is actually 5V pin and ground pin from Arduino. Figure 6-3 shows the actual 1-TUID test unit.

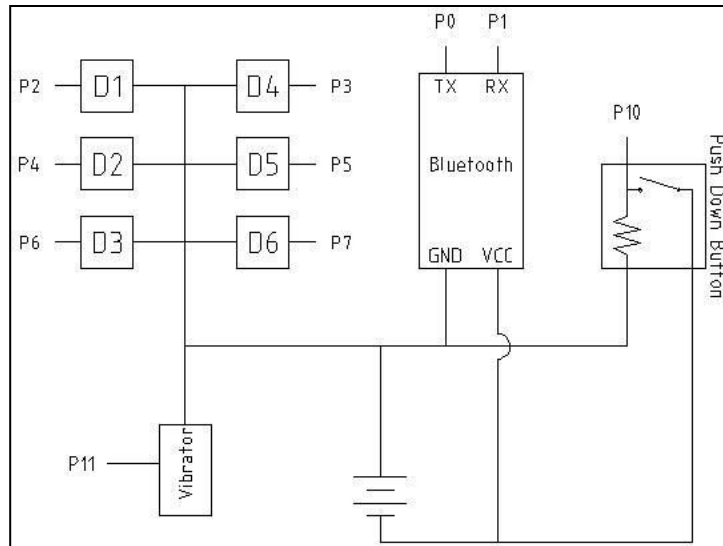


Figure 6-2. 1-TUID Test Unit Circuit. 'P#' represents a pin from Arduino Uno. P0, P1, and P10 acts as signal input/output, whereas the other pins control its components by acting as the power source.

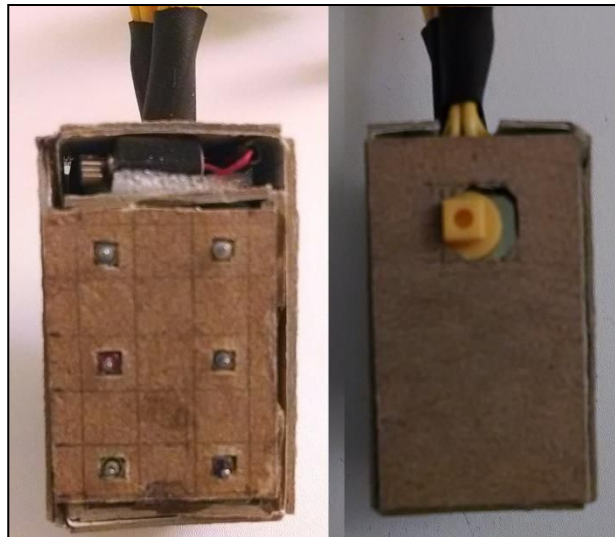


Figure 6-3. 1-TUID Test Unit. On left, top view of 1-TUID test unit. The six pins are motor shafts that are used for tactile reading. On right, bottom view of 1-TUID test unit. The yellow extrusion is a push down button.

6.2 Braille Library

As different languages use different symbols and different grammars, Braille behaves differently on each language. Furthermore, some languages have Grad 2 Brailles

in order to shorten the length of the translation, making it difficult to provide wider language support. Braille Library addresses this issue by providing two different methods to add new languages: Factory Pattern and Turing machine.

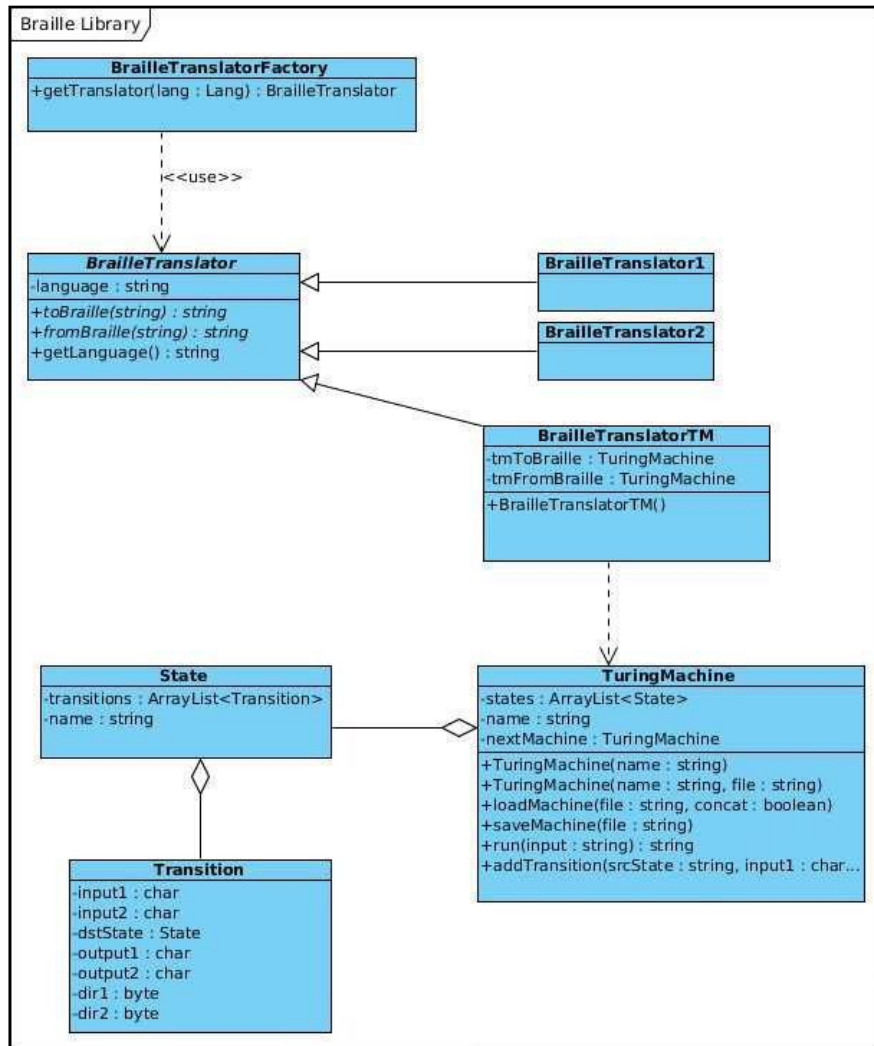


Figure 6-4. Braille Library Overview. Braille library uses Factory Pattern, as shown on the top half of the figure. However, it also provides BrailleTranslatorTM so that the users can store/load translators as file.

Using Factory Pattern allows developers to add new classes without having to modify dependent components. Braille Library uses an abstract class named BrailleTranslator as a base for all other translator classes. By extending from

BrailleTranslator, developers can easily add new languages. Figure 6-4 shows BrailleTranslator being extended to different translator classes, while BrailleTranslatorFactory instantiates the translators.

Another method is adding a file which contains Turing machine table for the new translator. As seen in bottom half of Figure 6-4, Braille Library contains a class, TuringMachine, which interprets a machine from this file. It also provides a basic method for the developers to create the automata file. Although much more difficult than Factory Pattern, using automata file allows the users to store only the needed translators to desired locations(such as external sdcard). Given that there exists almost 135 languages in Braille, the ability to choose which translator will be stored in device is desirable.

For this project, a 2-tape Turing machine is built with few special range inputs, because there are too many letters which require a transition for each. These inputs are: LW, CAP, N, PNCT, ANY, and OTHER. LW, CAP, N, and PNCT represent any lower case letters, any upper case letters, any numbers, and any punctuation respectively. ANY('~' on automata diagrams) indicates that a transition accepts any input, while OTHER('c' on automata diagrams) indicates that a transition accepts any other inputs that is not specified in source state. By using this special inputs, the library user does not need to create transition for every letter and symbols. In order to prevent these range inputs from overlapping with actual input symbols, a range of unicodes from 0x2600 to 0x26F0 are chosen. These unicodes are chosen because they only contain visual icons which do not exist in Braille, and their wide range provides more room for expansion when new languages are added.

As an implementation, a Turing machine for Grade 2 UEB translator is created. Due to the complexity of the translation, the machine is broken into 7 sub-machines which runs in a series.

6.2.1 APOS Machine: Apostrophe

APOS machine is shown in Figure 6-5. APOS machine scans for any apostrophe symbols and determines whether it is used as opening single quot, closing single quot, prime, or apostrophe. Once the context of the symbol is determined, the machine replaces the symbol with corresponding Braille sequence.

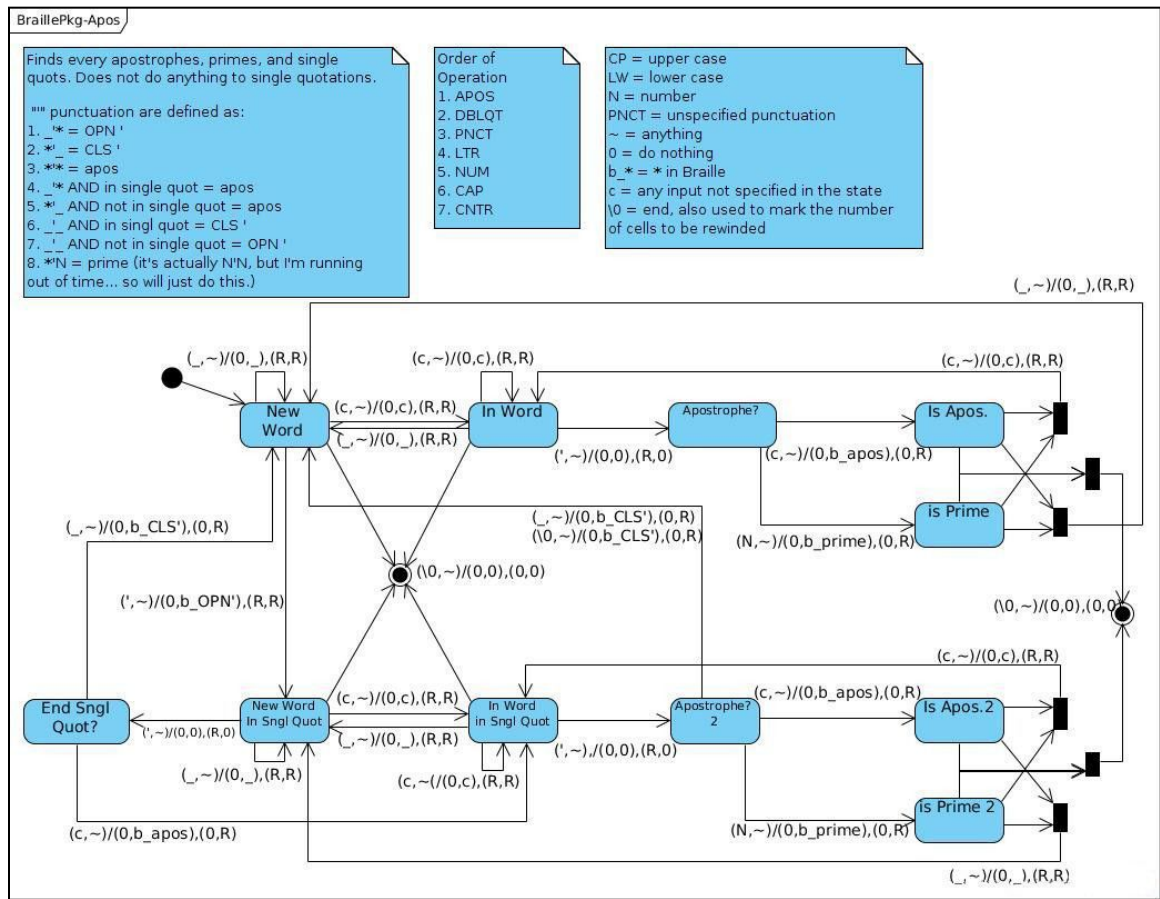


Figure 6-5. APOS Machine. This machine scans for apostrophe, determines the usage of it, then translates it to Braille.

6.2.2 DBLQT Machine: Double Quotation

DBLQT scans for any double quotation symbols, and determines whether it is used as opening quot, closing quot, or non-directional quot (punctuation used in Braille). Once the use of the symbol is determined, the machine replaces the symbol with corresponding Braille sequence. Since double quotation symbol is used similar to single apostrophe symbol, DBLQT works similar to APOS(shown in Figure 6-5).

6.2.3 PNCT Machine: Punctuation

PNCT translates the other punctuation that are not covered by APOS or DBLQT. Current implementation only covers the punctuation on regular keyboard. Unlike apostrophe and double quotation symbols, these symbols do not depend on context. Therefore, PNCT machine uses simple table search for its operations.

6.2.4 LTR Machine: Letter Indicator

Because Grade 2 UEB uses single alphabet letters as a contraction for words, a letter can be mistaken for a word when it is not. UEB uses letter indicator(also called Grade 1 indicator) to prevent such confusion. Another confusion is when letters follow a number because Braille uses letters ‘a’ to ‘j’ as numbers. UEB prevents confusion by using letter to declare end of a number. LTR machine scans for such occurrences, and inserts the letter indicator whenever needed (shown in Figure 6-6).

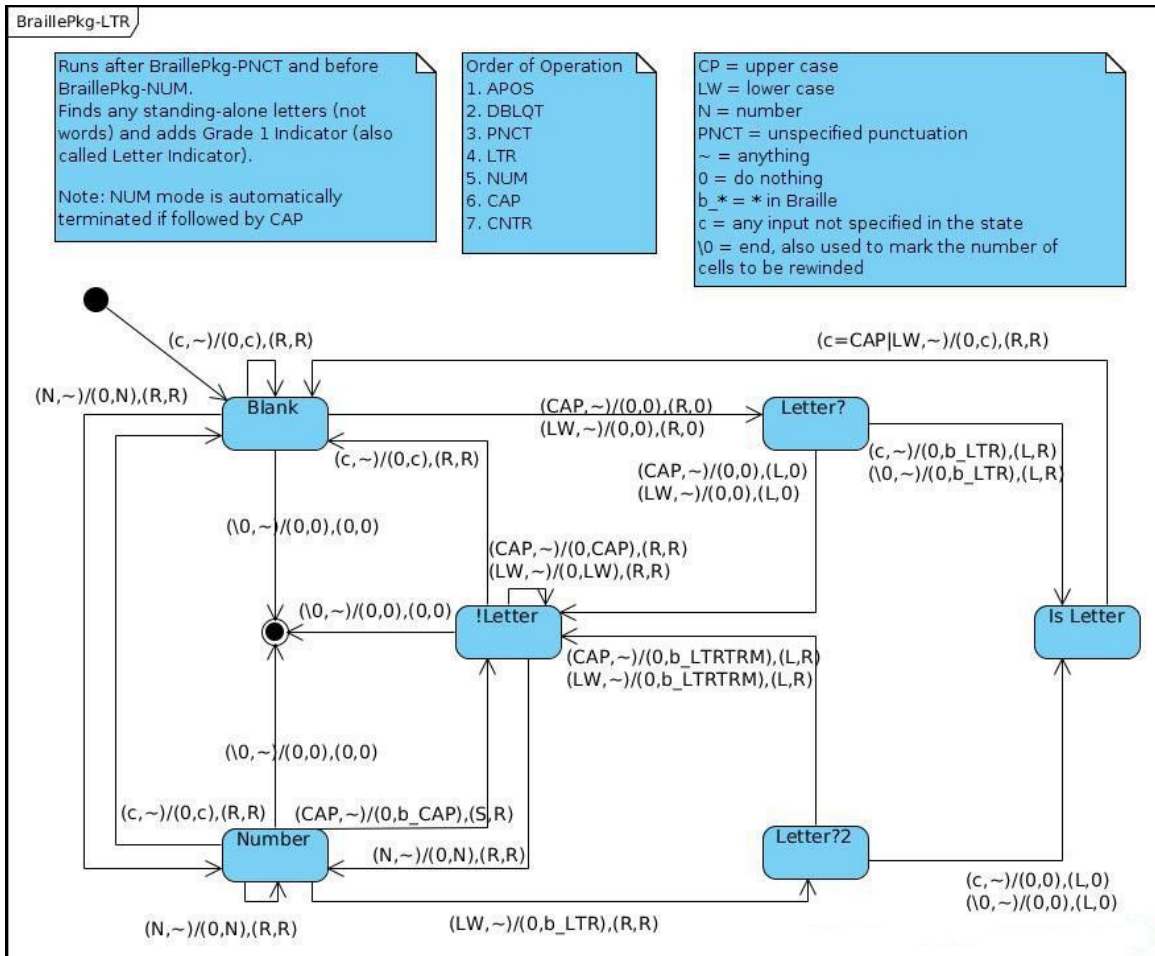


Figure 6-6. LTR Machine. This machine searches for any stand-alone letters, and adds Letter Indicator.

6.2.5 NUM Machine: Number Digits

As mentioned in section 2.1, Braille uses letters ‘a’ to ‘j’ for digits 0 to 9 by adding number indicator in front of the number sequence. NUM machine determines where to add the number indicator and converts numbers into corresponding Braille letters. Note that the machines following NUM machine, CAP and CNTR, are designed to ignore any Braille symbols. This design ensures that the numbers translated to Braille alphabet do not interfere with other alphabetic translations.

6.2.6 CAP Machine: Capital Letters

Shown in Figure 6-7, CAP scans for uppercase letters, and determines whether the occurrence is a capital letter, all capital word, or all capital passage. Depending on the occurrence, CAP adds appropriate number of capital indicators (1 for letter, 2 for word, and 3 for passage). During this process, CAP also converts every uppercase letters to lowercase, so the next sub-machine only needs to handle lowercase letters.

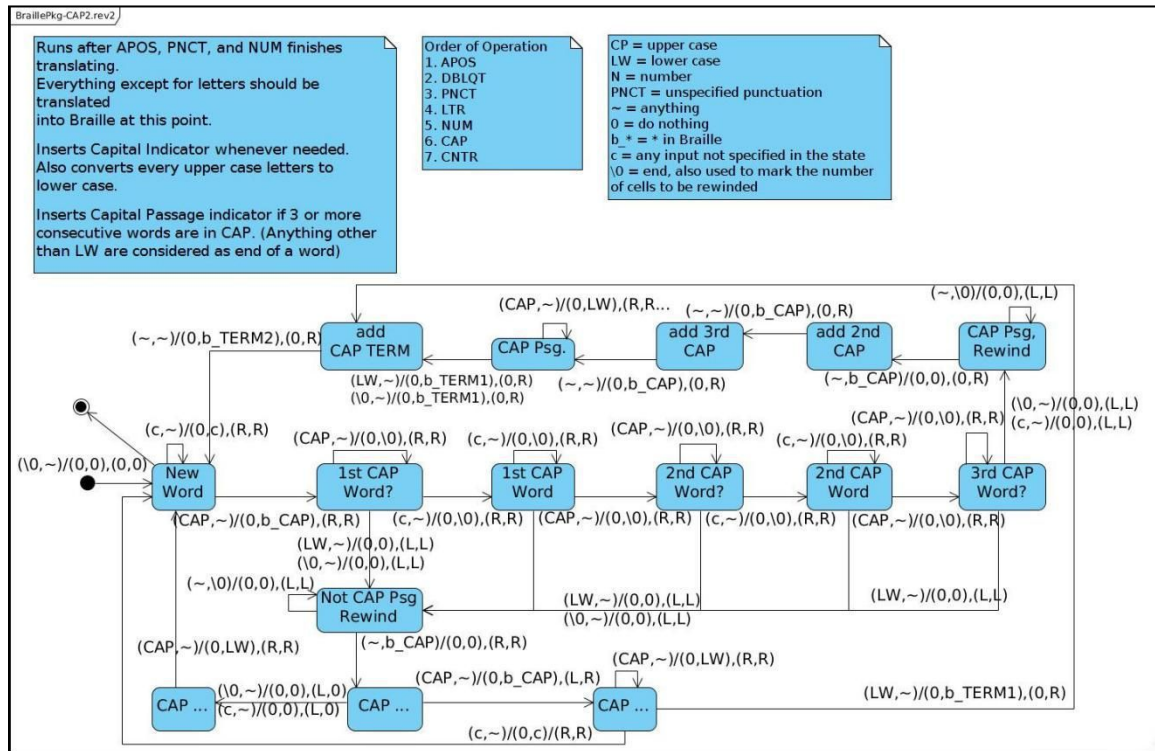


Figure 6-7. CAP Machine. This machines scans for uppercase letters. Upon encountering one, it determines whether the whole word/passage is uppercase. It then adds appropriate number of Capital Indicator in front of the letter. In addition, it converts all uppercase letters into lowercase, so the next machine only needs to handle lowercase letters.

6.2.7 CNTR Machine: Contraction(Grade 2)

Partially shown in Figure 6-8, CNTR is the final sub-machine that performs the actual translation. It uses two search trees: a tree for standing-alone contractions and a

tree for non-standing-alone contractions. The machine will first search the input word in standing-alone tree. If the the word is not found, it searches in non-standing-alone tree to check if the word contains one of the contractions. If no contraction is found, the machine translates the first character of the word, then repeats the process with the rest of the word.

Note that the Figure 6-8 is a showing only a portion of CNTR machine due to its massive size. In fact, the machine in the figure is only handling contractions “and” and “do,” whereas UEB has about 160 contractions.

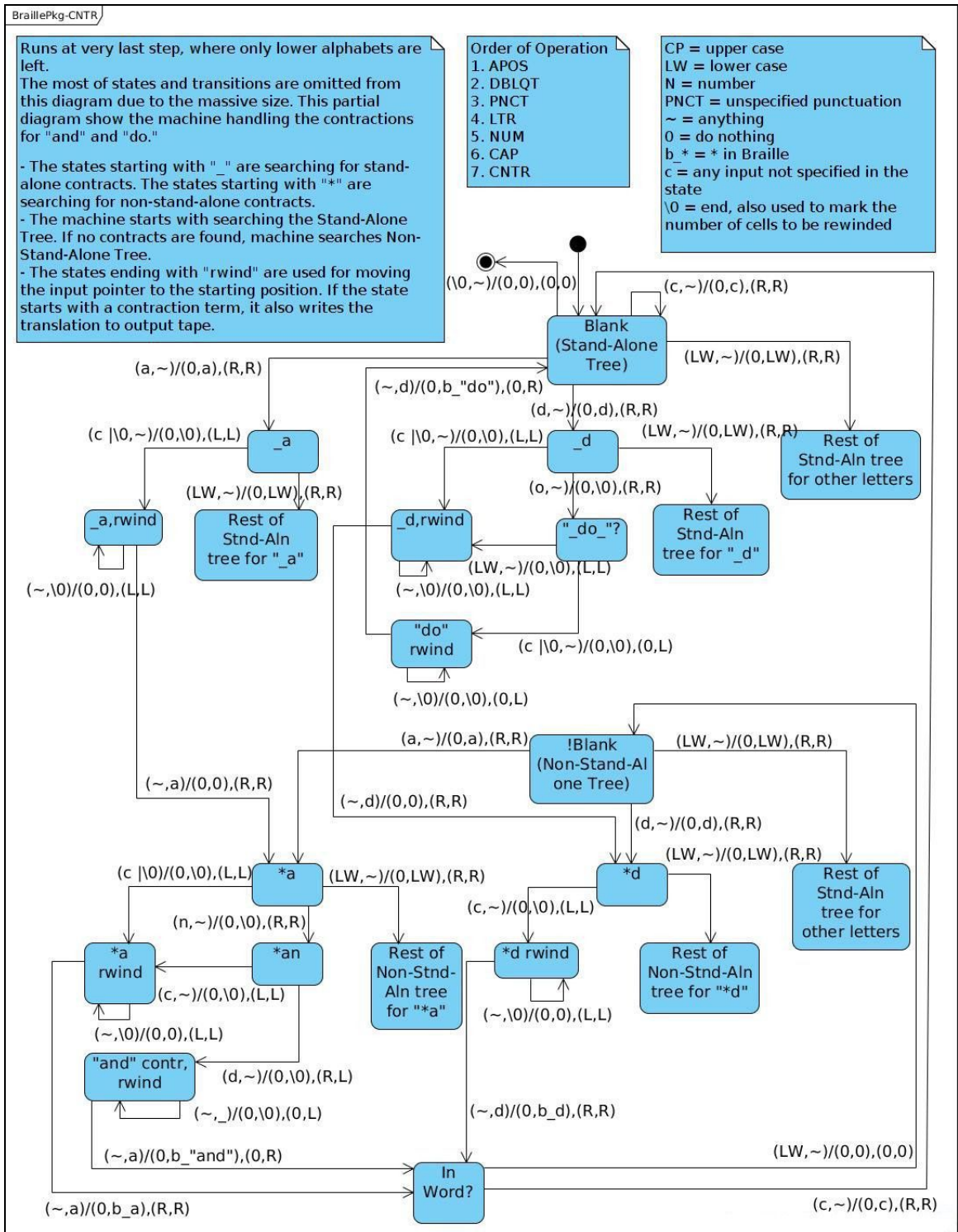


Figure 6-8. Partial CNTR Machine. This diagram shows only a part of an actual CNTR machine due to the immense size of the original. Note that the first half of the machine is Stand-Alone Tree, while the other half is Non-Stand Tree.

6.3 1-TUI API

For this project, 1-TUI API is created for Android. The components of the API are categorized into: 1-TUI objects, 1-TUID objects, and event protocols. The 1-TUI objects are the objects that the users can see and interact with. The 1-TUID objects are the communications objects that the application uses to communicate with the 1-TUID hardware. The event protocols are the event interfaces and message formats that the 1-TUI objects and 1-TUID objects use to communicate. The current implementation is created to develop prototype applications, so it lacks stability and organization.

6.3.1 1-TUI Objects

The 1-TUI objects are designed to be used just like the regular Android View objects, and to be used on XML layout as well. Therefore these objects are created by extending from the existing Android objects, as shown in Figure 6-9.

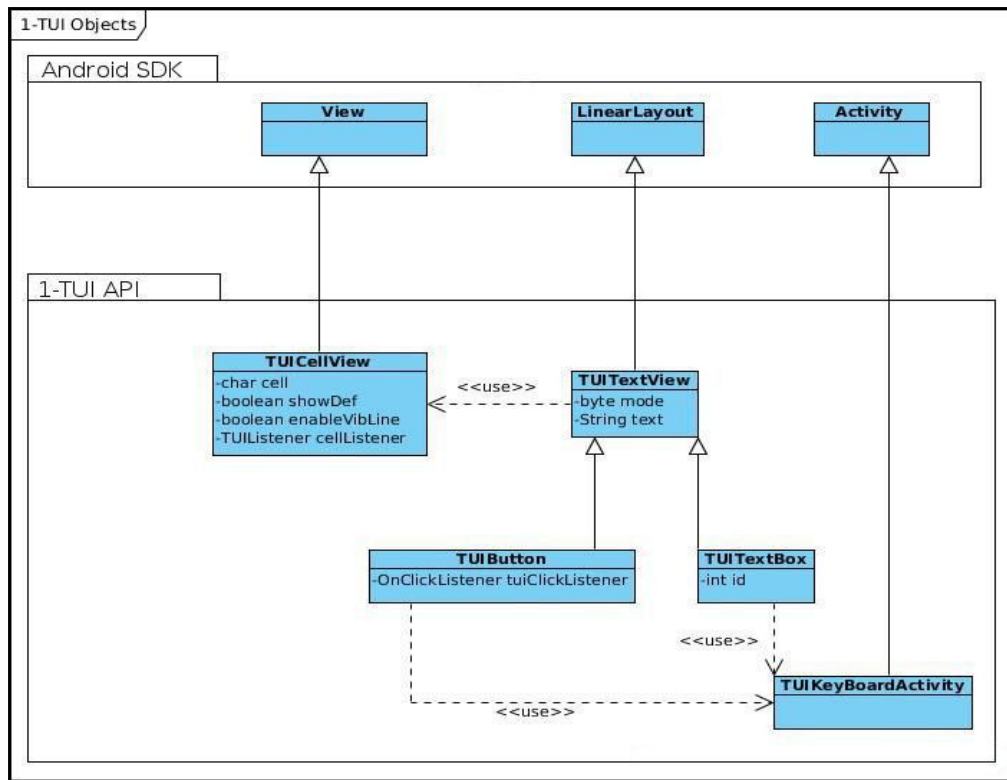


Figure 6-9. 1-TUI Objects. The most of 1-TUI objects are extensions of basic Android objects.

Per 1-TUI design, the graphical objects need to send signal to 1-TUID whenever the user touches or swipes onto them. Such behavior is enforced by implementing `onTouch()` and `onDrag()` events to detect the user's finger movement and invoke relevant 1-TUI events. But because Android does not start drag event on empty spaces, the 1-TUI objects do not recognize the drag event if the event starts on an empty space. In order to prevent this, 1-TUI API for Android requires that all 1-TUI application have `TUILayout` as the root layout. `TUILayout` is an extension of `RelativeLayout` that fills the whole background, and initiates drag mode for the other 1-TUI objects. Having `TUILayout` to fill the whole screen on background removes these empty spaces, so the 1-TUI objects responds to the users' movements consistently.

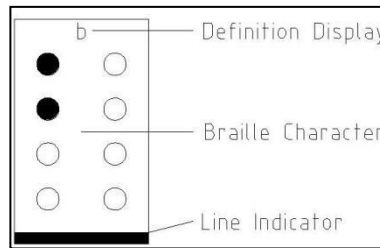


Figure 6-10. `TUICellView`. The most basic part of 1-TUI API that displays Braille on screen. It has 2 options: `Definition Display` and `Line Indicator`. `Definition Display` displays the definition of the displayed Braille if enabled. `Line Indicator` outputs a continuous vibration while touched.

The most basic graphical component of 1-TUI API is `TUICellView` (see Figure 6-10), which represents a Braille character. On screen, a `TUICellView` displays an 8-dot Braille cell. To give a better sense of screen location, touching a `TUICellView` returns a haptic feedback so the user can determine the location of his/her finger. The class supplies few options such as `Definition Display` and `Line Indicator`. When `Definition Display` is enabled, the definition of the Braille character(given by developer) is

displayed on top of the View. Line Indicator, which is enabled by default, creates an underline which returns a longer haptic feedback when touched. This indicator is used to let the user know that s/he has moved to an another line. Without this line, the blind user can easily stray onto other lines of text without realizing it.

TUITextView is a graphical component that manages a sequence of TUICellView's, representing a Braille text. Because 1-TUI represents every object as a Braille text, it is designed to work as the base for the most of 1-TUI objects, . This class handles the cell size and organization, so developers can easily extend it for their own use. TUITextView has two modes: Cell Mode and Grid Mode. Cell Mode handles the Braille text similar to a regular text. In this mode, the TUICellView's are given a size, and are stacked left to right. Cell Mode is recommended when the size of cells needs to be fixed (e.g. reading a long text on large screen). Grid Mode puts the TUICellViews in given numbers of rows and columns, then scales them to fit the whole view. This mode is recommended when a simple layout with dynamic size is desired.

6.3.2 1-TUID Objects

1-TUID object maintains the two-way communication between the 1-TUID hardware and the 1-TUI application. Because the prototype 1-TUID uses Bluetooth module for communication, the current API has 1-TUID object for Bluetooth only (called TUIDBluetooth).

Once an 1-TUID object is created, developer must call start() method to initiate the connection. After the connection is made, developer can call sendBraille() or sendBytes() to send data to 1-TUID. Another way to send data to 1-TUID is by registering its instance as an onTUITouch event handler for the 1-TUI objects. By doing

so, any 1-TUI events will automatically be sent to 1-TUID. As for receiving data from 1-TUID, the listener need to be registered to the 1-TUID object as onTUIDClick event handler. This event will tell whether the user pressed or released the 1-TUID button.

6.3.3 Event Protocols

In 1-TUI, the communication between the 1-TUI objects and 1-TUID is crucial. 1-TUI provides an interface called TUIListener which defines two events: onTUITouch and onTUIDClick. onTUITouch is invoked when a TUICellView is touched and sends signal to 1-TUID. onTUIDClick is invoked when 1-TUID button is pressed or released.

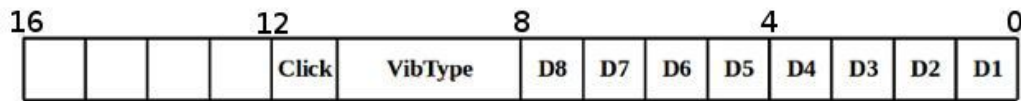


Figure 6-11. 1-TUI Message Protocol. 1-TUI objects and 1-TUID communicate by sending data in above form. The first byte is the dot configuration for a Braille character. The second byte contains options and status values.

1-TUI API also provides a messaging protocol class named TUIProtocol, as shown in Figure 6-11. The protocol uses a 2-bytes message where the first byte stores the Braille character and the second byte stores vibration commands and click status. The vibration command tells 1-TUID what kind of haptic feedback that it needs to generate. Click status is a boolean value set by 1-TUID, so 1-TUI objects can handle 1-TUID click events. TUIProtocol class also includes few static methods that the developers can use to easily retrieve or set data from or to a raw message bytes.

6.4 Android Applications

Four basic Android applications are developed to showcase 1-TUI. These applications are developed using the before mentioned Braille library and 1-TUI API.

6.4.1 TUIReader

Shown in Figure 6-12, TUIReader is a text file reader which implements 1-TUI. It lets the user to navigate within a designated folder and select a “.txt” or “.brl” file. “.brl” file is a Braille text file, so it is immediately loaded if the user selects it. On the other hand, the selected file needs to be translated to Braille if “.txt” is chosen. TUIReader handles this process by creating a separate thread which translates the “.txt” file to a “.brl” file. Since this thread is running in background, the user can start reading the new “.brl” file without waiting for the whole file to be translated.

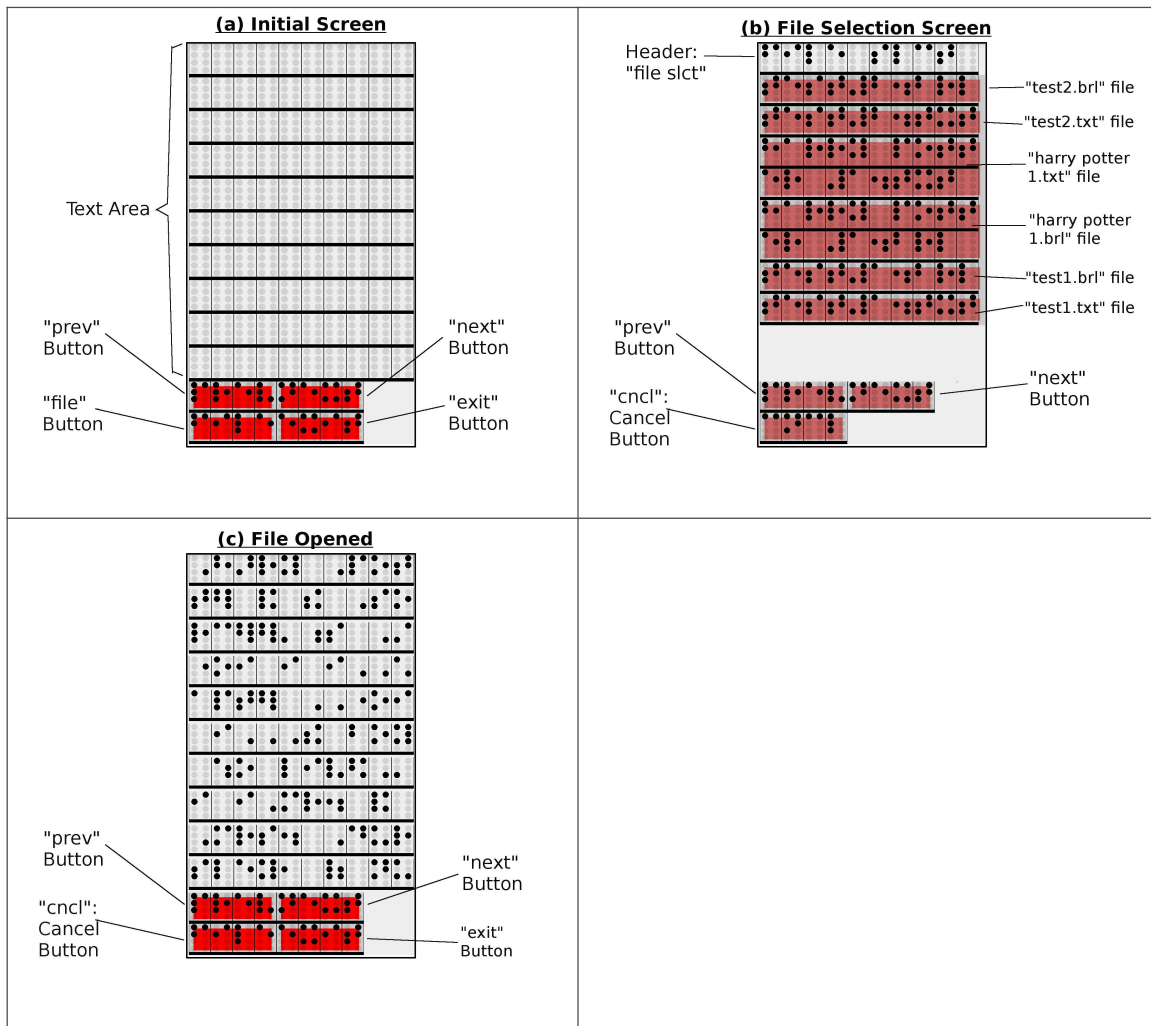


Figure 6-12. TUIReader. (a) Initial screen. User can click “file” button to open file selection screen. (b) File selection screen. User can use “prev” and “next” button to see more list of files. User can

then click a file to open it. (c) File opened. Once the user selects a file, it is loaded and s/he can navigate it by clicking “prev” and “next” button.

6.4.2 TUICaller

TUICaller is a caller applications which implements 1-TUI. Unfortunately, Android does not allow developers to have call control due to the security reasons. Java reflection on Android source code is tried to access the ITelephony call control, but it only works on older Android devices. Killing the Telephony service is tried to hang up a call, but it also fails because the service is system protected. Finally, sending a fake headset signal works, but only for answering an incoming call. Only the stock Android caller application is allowed to make an actual call action, so TUICaller works alongside it.

For making an outgoing call, TUICaller sends a call intent to Android OS, which then calls the stock application. After sending the intent, TUICaller waits approximately 2 seconds until the stock application shows up. When the stock caller starts, TUICaller starts an activity of its own on top of the stock caller. Since the TUICaller’s activity is on top, the user can use 1-TUI interface to read the call information. However, this activity does not have ability to hangup for the reasons explained before. The outgoing call operation is shown in Figure 6-13.

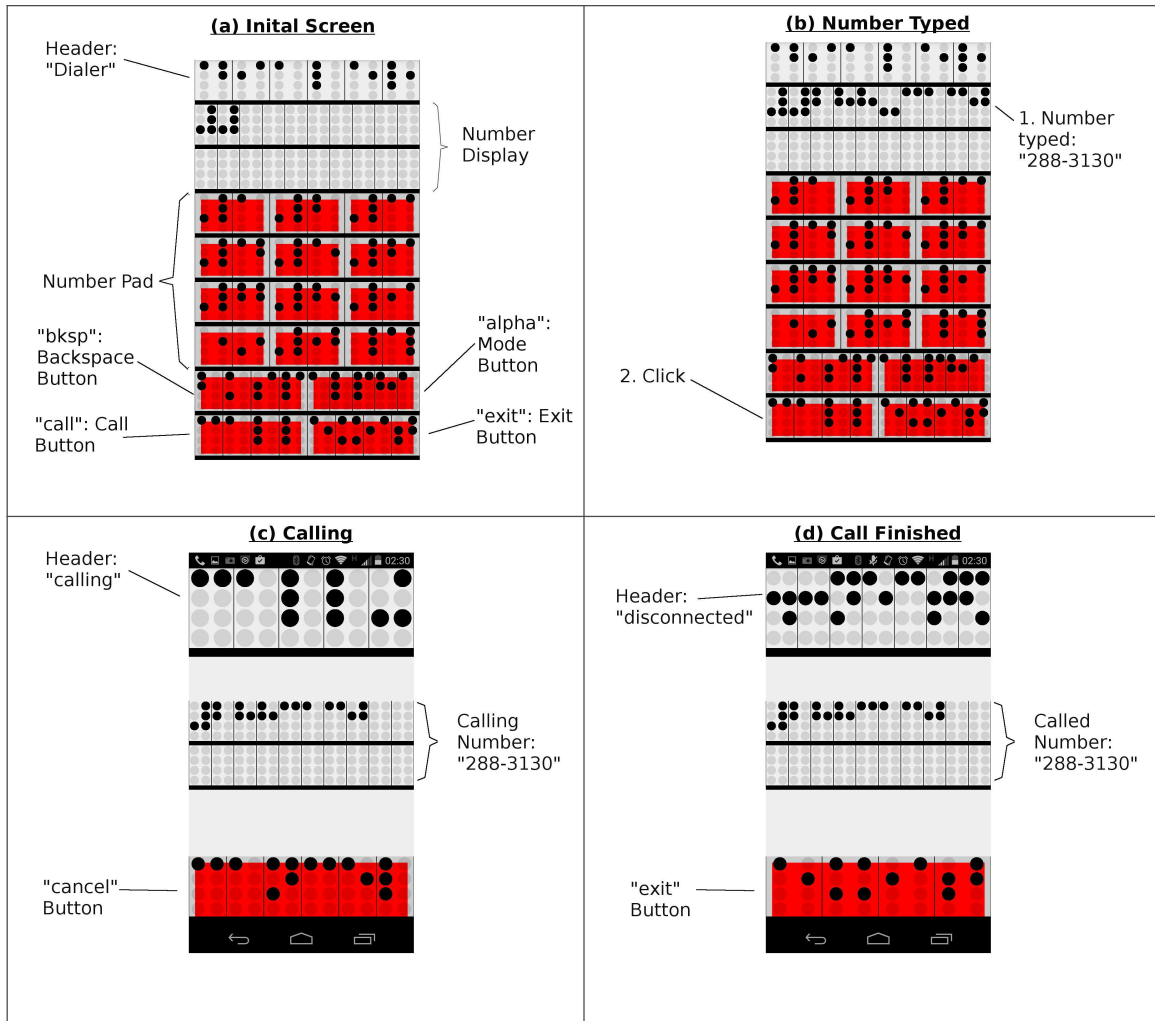


Figure 6-13. Making a Call. (a) Initial screen, TUICaller uses similar layout as other callers. Note that “alpha” button just switches the pad labels between numbers and alphabets. (b) Number typed. (c) Calling screen. This screen is shown on top of Android stock caller. “cancel” button does not work due to not having call control. (d) Call ended.

Incoming calls are handled similarly. The moment it is installed, TUICaller starts listening for any change in phone status. When an incoming call is detected, TUICaller waits approximately 2 seconds while Android OS starts the stock caller. Once the stock caller starts, TUICaller starts its own activity on top. If the user chooses to answer the call, TUICaller sends a fake headset signal. Android OS receives the signal and then notifies the stock caller app to answer the call. Incoming call handling is shown in Figure 6-14.

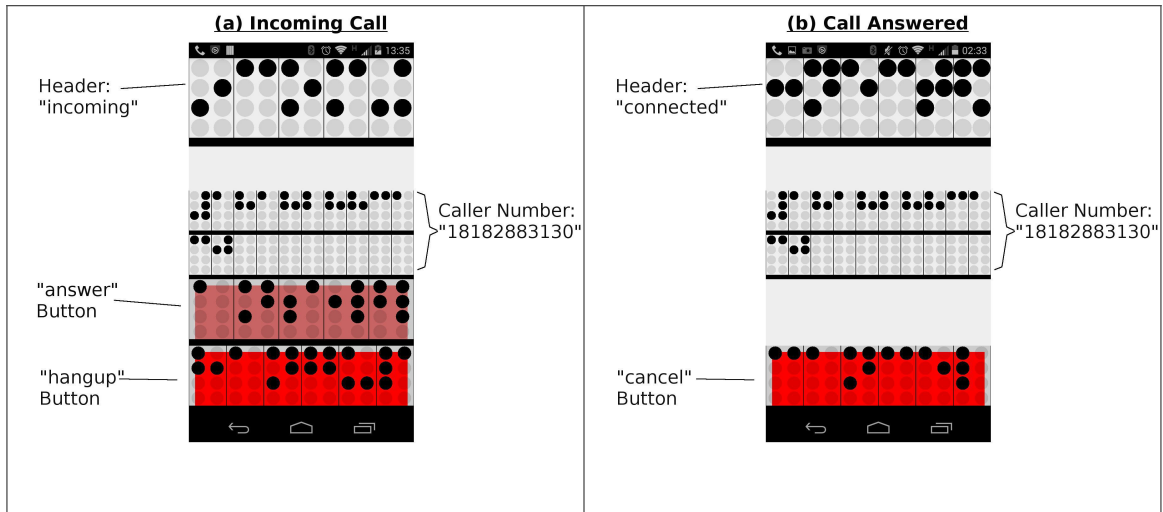
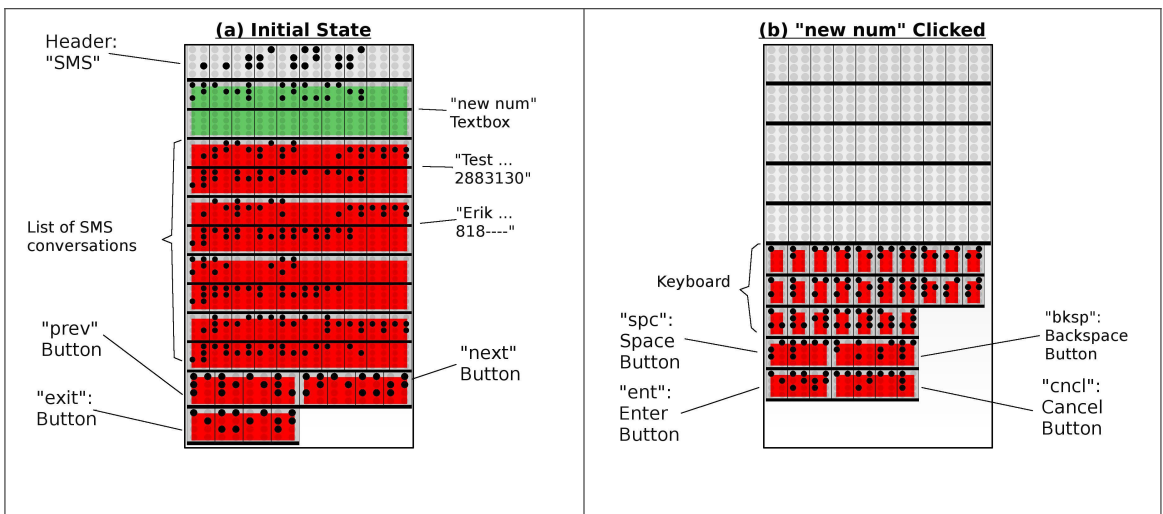


Figure 6-14. Incoming Call. (a) Incoming call screen. This screen is shown on top of Android stock caller. Like in outgoing call, "hangup" button does not work. (b) Call answered.

6.4.3 TUISMS

TUISMS is a SMS messenger which implements 1-TUI. Its behaviors are based on the other messenger applications, so the user can send, receive, and navigate SMS messages. Each message is labeled with the sender's name. If the sender's number is not found in contacts database, the last 4 digits of the number is displayed. Every incoming messages are translated to Grade 2 Braille, while every outgoing messages are translated to English. TUISMS usage is shown in Figure 6-15.



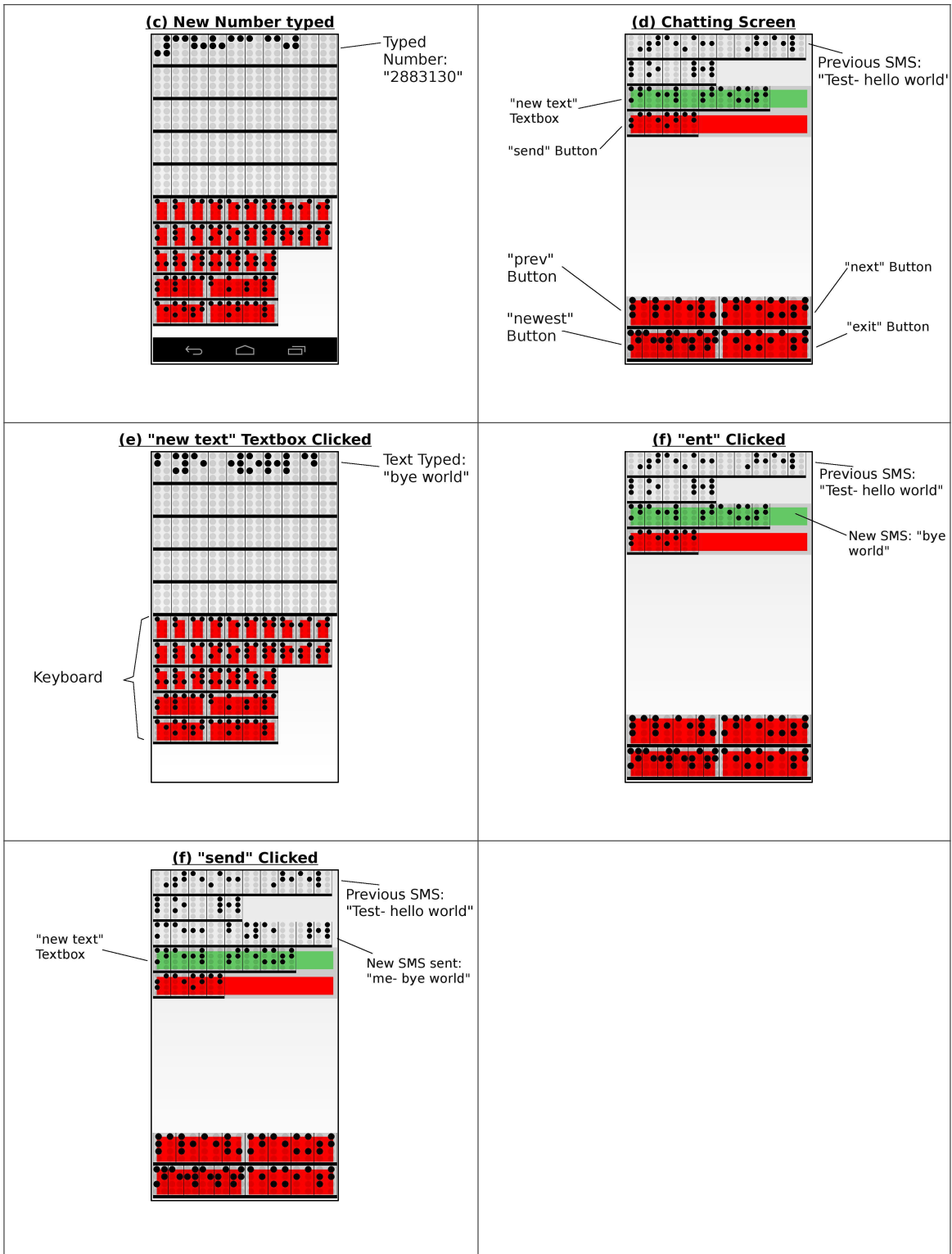


Figure 6-15. TUISMS. (a) Initial screen. This screen shows a list of previous conversations, so the user can select one to continue. The user can also type in a phone number in “new num” textbox to start a new conversation.(b)~(c) Keyboard screen. (d)~(f) Conversation screen. The user can

read previous SMS messages by using “prev” and “next” button. S/he can also send a new message by typing in “new text” textbox and clicking “send” button.

6.4.4 TUILauncher

Show in Figure 6-16, TUILauncher is a device home screen that displays a list of TUI applications. Being a device home screen, TUILauncher screen is shown whenever the user exits from a 1-TUI application.

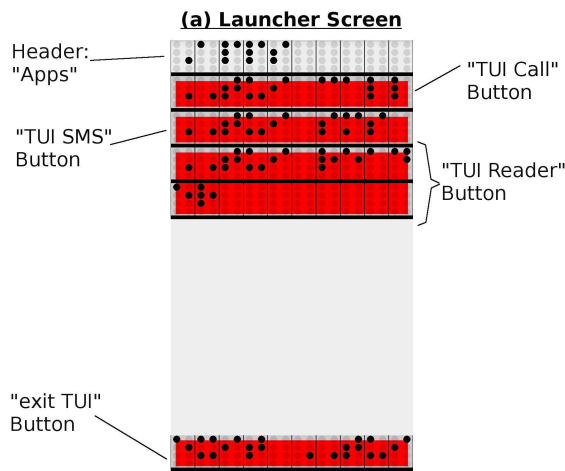


Figure 6-16. TUILauncher. Each 1-TUI application is shown as a button.

7 VALIDATION AND RESULTS

The validity of 1-TUI is evaluated by inspecting the quantifiable attributes of User Interfaces(UI). The data on these attributes are collected by system testing and researching on other works, and then analyzed. In addition, data on Braille Library are collected for benchmark purposes.

7.1 1-TUI Validation

The validation of 1-TUI involves quantifying the usability of the UI. Two of the quantifiable attributes of an UI are the speed and the accuracy at which the blind users handle common activities. A few system test are performed to collect data on these attributes for 1-TUI and the other existing UI's, so the performance differences between 1-TUI and the other UI's can be measured. The gauged differences are then analyzed with the data from the other researches to derive the theoretical range of the speed and the accuracy of 1-TUI.

Since the tester (the researcher of this project) is not blind, he was blindfolded during the testing procedure. For a full disclosure, the tester has never used Braille or screen readers prior to this research, and thus, produced much slower and less accurate results than the experienced Braille readers would have.

7.1.1 System Testing: Printed Braille vs 1-TUI

The first system test compares 1-TUI against a printed Braille. On each trial, the tester records himself reading aloud from a printed Braille text or TUIReader(an 1-TUI text file reader) for 1 minute. After 3 trials on both printed Braille text and TUIReader, the record is used to determine how many characters were read, and how many of them were correct.

	Printed Braille			TUIReader		
Trial #	Cell Read	Correct	Accuracy	Cell Read	Correct	Accuracy
1	7	6	85.71%	9	9	100%
2	6	4	66.67%	12	11	91.67%
3	5	4	80.00%	10	10	100%
Average	6	4.67	77.46%	10.33	10	97.22%

Table 7-1. Braille Read Comparison. This table compares the reading done on a printed Braille and on TUIReader App. Each trial took 1 minute, during which the tester attempted to read a random line.

As seen in Table 7-1, 1-TUI produces much faster and more accurate reading. With a printed Braille, the tester achieves an average speed of 6 characters per minute(cpm) at 77.46% accuracy. On the other hand, the tester achieves an average speed of 10.33 cpm at 97.22% accuracy with TUIReader. In this test, 1-TUI produces a far better result because it is easier to identify each character. For a novice reader, the printed Braille characters are too small to recognize, to the point that two Braille characters are confused for one. Such problem does not exist in 1-TUI since it only displays one character, in a larger size than the printed Braille characters. Therefore, 1-TUI provides a better reading experience to the inexperienced Braille readers.

Certainly, an expert Braille reader will result in faster and more accurate reading. In their paper Legge, Madison, and Mansfield, converts 124 wpm (the average read speed of Braille) to 7.5 cps, which equals to 450 cpm^[8]. Given that the speed on TUIReader was 1.7 times faster than the speed on printed Braille, the theoretical average speed of 1-TUI would be 775 cpm. Still, the Braille readers depend on sliding motion for reading, which

1-TUI do not, so the actual average speed of 1-TUI may be lower. Considering these, an experience Braille user is expected to read between 450 cpm and 775 cpm with 1-TUI.

7.1.2 System Testing: Screen Reader vs. 1-TUI

The second system test compares 1-TUI against a screen reader (TalkBack). On each trial, tester records himself reading and answering questions via SMS for 3 minutes. Each questions is limited to a simple arithmetic problem involving 1-digit numbers, so the tester does not spend time on figuring out the answer. After 3 trials on both TalkBack and TUISMS, the record is used to determine how many questions are read correctly, and how many correct answers are sent.

	TalkBack		TUISMS	
Trial #	Read/Answered	Total	Read/Answered	Total
1	4/4	8	2/1	3
2	6/6	12	2/2	4
3	5/5	10	3/2	5
Average	5/5	10	2.33/1.67	4

Table 7-2. TalkBack vs TUISMS. This table compares screen reader and 1-TUI on SMS messaging. Each trial took 3 minutes, during which the tester attempted to read and answer simple math questions via SMS. Note that “Answered” refers to the number of questions that the tester managed to reply back within time limit. “Total” is the total number of SMS activities that the subject completed.

The Table 7-2 compares the results from screen reader and 1-TUI. Note that the “Read” refers to the number of questions that the tester read correctly, while “Answered” refers to the number of replies that the tester managed to send. “Total” column is the sum

of “Read” and “Answered,” representing the total number of SMS activities that the subject completed.

As expected in Section 3.2, screen reader performs far better than 1-TUI, based on the researches on read speed of Braille^[8] and screen readers^[9]. The test subject manages to complete an average of 10 SMS activities with screen reader, while only 4 SMS activities are completed with TUISMS. The performance difference reveals that 1-TUI is 150% slower than screen reader. While the faster speed on auditory reading is the main cause of the performance difference, 1-TUI fails to achieve its potential speed because the tester is not fluent in Braille. On each character being read or written, the tester had to spend few seconds in translating between Braille and English.

Therefore, it is possible that the performance difference may be reduced if 1-TUI is tested by a subject who is fluent in Braille. For a further analysis, we assume that the writing speed is 33.33% of the reading speed. Therefore, writing speed in 1-TUI is assumed to be 150 cpm (given the average read speed of 450 cpm). In Braille, each question had an average of 6 characters, while each answer had an average of 3 characters. Also, the average SMS delivery delay was 5 sec. Using this numbers, we estimate the performance by an experienced Braille reader to be:

$$\begin{aligned} & (1 \text{ question}) * (6 \frac{\text{char}}{\text{question}}) * (\frac{\text{min}}{450 \text{ char}}) + (1 \text{ answer}) * (3 \frac{\text{char}}{\text{answer}}) * (\frac{\text{min}}{150 \text{ char}}) \\ & + (2 \text{ SMS}) * (5 \frac{\text{sec}}{\text{SMS}}) * (\frac{\text{min}}{60 \text{ sec}}) \\ & = \underline{0.20 \text{ min per Q/A}} \end{aligned}$$

Therefore, the number of SMS per min is

$$\begin{aligned} & \frac{1}{0.20 \text{ min per Q/A}} = 5 \text{ Q/A per min} \\ & = \underline{10 \text{ SMS Activities per min}} \end{aligned}$$

Applying the wpm-to-cps ratio used by Legge, Madison, and Mansfield, the auditory read speed of 318 wpm^[9] is converted to 1154 cpm. Also, the writing speed in

screen reader is assumed to be 384.67 cpm (33.33% of the read speed). With screen reader, each question had an average of 4 characters, while each answer had an average of 2 characters. Using this numbers, we estimate that a test with an experienced screen reader user will produce:

$$\begin{aligned}
 & (1 \text{ question}) * (4 \frac{\text{char}}{\text{question}}) * (\frac{\text{min}}{1154 \text{ char}}) + (1 \text{ answer}) * (2 \frac{\text{char}}{\text{answer}}) * (\frac{\text{min}}{384.67 \text{ char}}) \\
 & + (2 \text{ SMS}) * (5 \frac{\text{sec}}{\text{SMS}}) * (\frac{\text{min}}{60 \text{ sec}}) \\
 & = \underline{0.18 \text{ min per Q/A}}
 \end{aligned}$$

Therefore, the number of SMS per min is

$$\begin{aligned}
 & \frac{1}{0.18 \text{ min per Q/A}} = 5.56 \text{ Q/A per min} \\
 & = \underline{11.11 \text{ SMS Activities per min}}
 \end{aligned}$$

Therefore, a testing with a subject who is fluent in Braille and experienced with screen reader will reduce the performance difference from 150% to 9.99%. Although this reduction is drastic, it is because the first test was done by an inexperienced Braille user which resulted in a large difference in speed.

7.2 Braille Translator Benchmark

The benchmark data on Braille translator library is collected by performing translations on various text files. Ranging from “Bible” to “Origin of Species,” 24 books on public domain are used. The test is automated by a simple Java program, which translates each book into Grade 1 UEB and Grade 2 UEB. The program records the time it took to translate a file and write the translation to a new file.

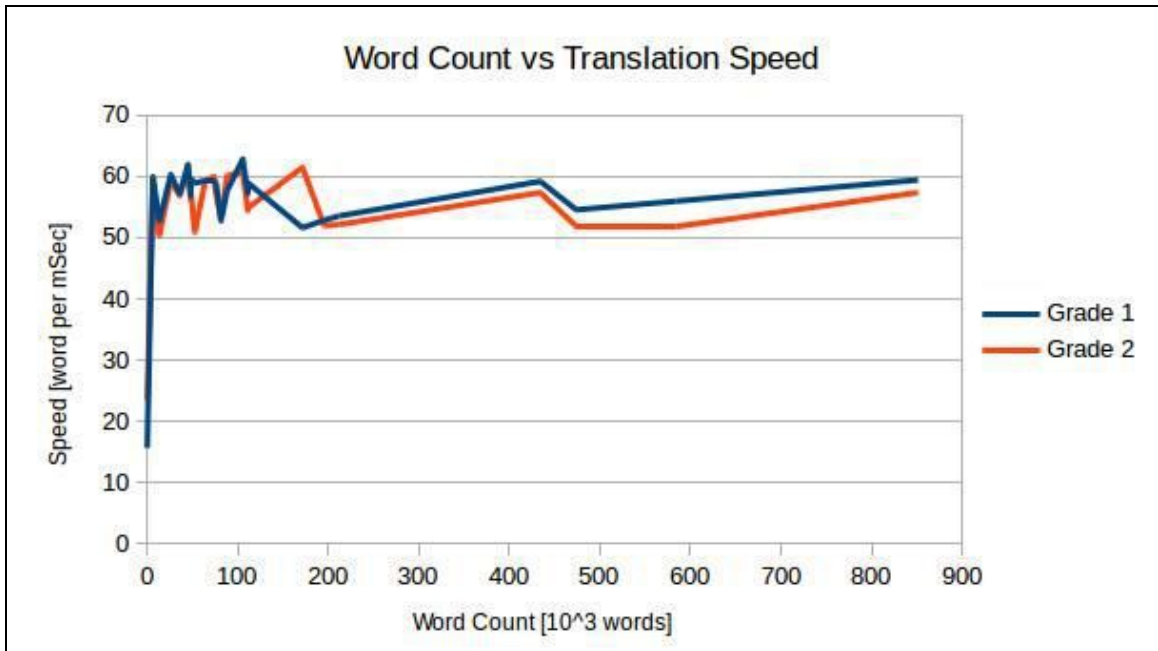


Figure 7-1. Word Count vs Translation Speed. The collected data shows that the Grade 1 Braille translation is generally faster than the Grade 2 Braille translation. Note that the data at very left only has word count of 47, so the large portion of the translation time is writing time, which significantly reduces word per mSec value.

The analysis of the speed of translation calculates the number of words translated per millisecond (shown in Figure 7-1). With a median speed of 58.28 words per millisecond, the Grade 1 translation proves itself to be faster than the Grade 2 translation, which has median speed of 56.97 words per millisecond. The higher speed by the Grade 1 Braille translation is explained by its use of one-to-one mapping, which is much faster than the tree search used by the Grade 2 Braille translation. However, this explanation is inconsistent against the speed difference of 2.3%, which is small considering that the tree search is much slower. The reason for this inconsistency is found by analyzing the size change (as shown in Figure 7-2).

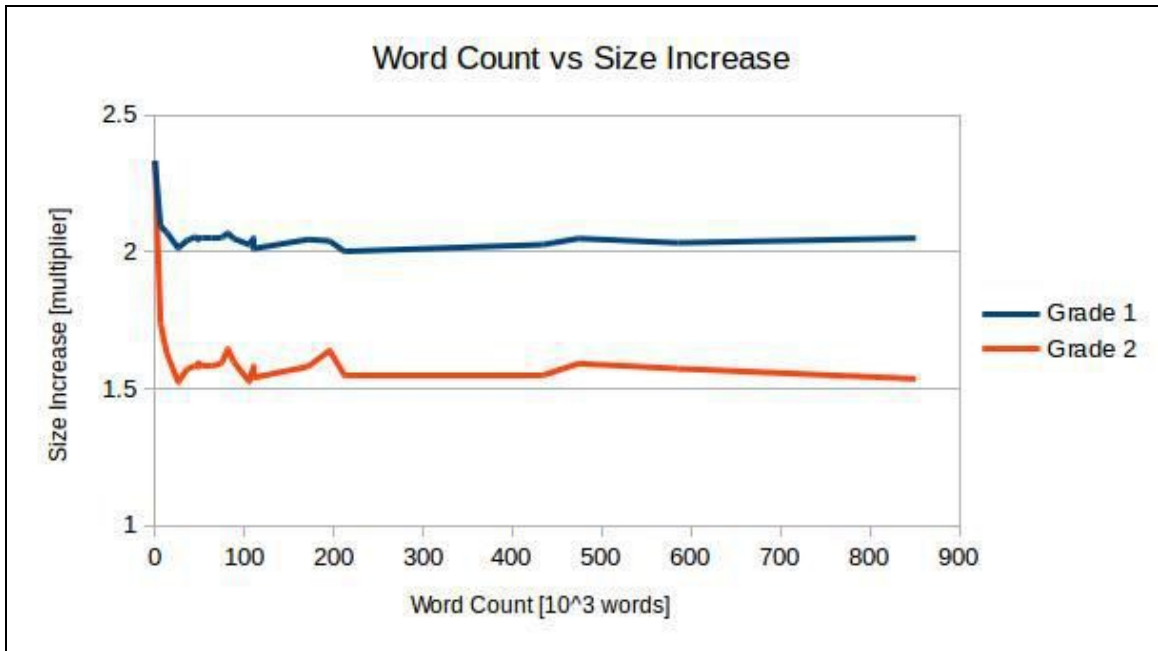


Figure 7-2. Word Count vs Size Increase. The size increase is measured by the ratio of final size to original size. Note that the data at very left only has word count of 47, so the large portion of the translated text is Braille punctuation instead of the original text. As a result, it has much higher size increase than the others.

The size analysis reveals that, at median, the Grade 1 Braille translation increases file size by 2.05 times while the Grade 2 Braille translation increases file size by 1.58 times. This large difference explains the less-than-expected difference in translation speeds. Since the Grade 2 Braille uses much less space than the Grade 1 Braille, the Grade 2 Braille translation needs much less time for writing the new file. In effect, the overall translation time is reduced, which increases the value of word per mSec.

As for the size increase, Braille adds its own punctuation for every uppercase letters, numbers, and stand-alone letters. Furthermore, many Braille punctuation use two symbols instead of one. For these reasons, Braille texts uses more characters than the regular texts, and thus, requires more space.

8 CONCLUSION

This research studied various methods with which the blind users can access mobile devices. The inspection of these methods revealed that they either lack user control and technical read/write, or too expensive. 1-TUI was proposed as a solution to these issues by combining screen reader and Braille display. 1-TUI was implemented into a suit of basic Android applications, which were used to validate its usefulness.

A system test by the researcher proved that reading on 1-TUI can be better than reading on printed Braille. Still, 1-TUI was much slower when compared to screen readers, because the tester was not fluent in Braille. To properly validate 1-TUI, this paper suggests further testing with subjects who are fluent in Braille. In fact, further analysis suggests that the future test can reduce the performance difference between screen reader and 1-TUI to 9.99%. To validate 1-TUI's usability, these tests should result in read speed of 450 cpm or more, and at least 10 questions answered via SMS in 1 minute.

Although the analysis concludes that 1-TUI is slower than the screen readers, it does not mean that 1-TUI is not useful. Keep in mind, that 1-TUI is capable of technical reading/writing, which the screen readers are not capable of. For the general blind users, 1-TUI is not as useful as the screen readers. But for the blind students and professionals, 1-TUI offers a better way to learn and communicate within their fields.

REFERENCES

- [1] W. Erikckson, C. Lee, & S. von Schrader. “2012 Disability Status Report,” Cornell Univ. EDI, Ithaca, NY, 2014.
- [2] American Printing House for the Blind. (2014). *Annual Report 2014*. [Online]. Available Web: <http://www.aph.org/federal-quota/dist14.html>.
- [3] Wikipedia. (2015, Apr. 9). *Braille*. [Online]. Available Web: Wikipedia.org.
- [4] *The Rules of Unified English Braille 2nd Ed.* ICEB Standard, 2013.
- [5] UNESCO, “World Braille Usage 3rd Ed.,” Perkins, Watertown, MA, 2013.
- [6] X. Lei, A. Senior, A. Gruenstein, & J. Sorensen, “Accurate and Compact Large Vocabulary Speech Recognition on Mobile Devices,” InterSpeech, Lyon, France, 2013.
- [7] WebAIM. (Jan. 2014). *Screen Reader User Survey #5*. [Online]. Available Web: <http://webaim.org/projects/screenreadersurvey5/>
- [8] G. E. Legge, C. Madison, & J. S. Mansfield, “Measuring Braille Reading Speed with the MNREAD Test,” Dept. Psych., Univ. of Minnesota, 2000.
- [9] C. Asakawa, H. Takagi, S. Ino, T. Ifukube, “Maximum Listening Speeds for the Blind,” Int’l Conference on Auditory Display, Boston, MA, July 6-9, 2003.
- [10] R. Shilkrot, J. Huber, C. K. Liu, P. Maes, S. C. Nanayakkara, “FingerReader: A Wearable Device to Support Text Reading on the Go,” CHI, Toronto, Ontario, Canada, 2014.
- [11] M. Y. Saadeh, “A Refreshable and Portable E-Braille System for the Blind and Visually Impaired,” Ph.D disseration, Dept. Mech. Eng., Univ. of Nevada, Las Vegas, 2012.

- [12]Y. Kato, T. Sekitani, M. Takamiya, M. Doi, K. Asaka, T. Sakurai, & T. Someya, “Sheet-Type Braille Displays by Integrating Organic Field-Effect Transistors and Polymeric Actuators,” IEEE Transaction on Electron Devices, vol.54 no.2, pp.202-209, Feb., 2007.
- [13]I. M. Koo, K. Jung, J. C. Koo, J. D. Nam, Y. K. Lee, H. R. Choi, “Development of Soft-Actuator-Based Wearable Tactile Display,” IEEE Transactions on Robotics, vol.24, no.3, pp.549-558, Jun., 2008.