논문 21-46-01-21 The Journal of Korean Institute of Communications and Information Sciences '21-01 Vol.46 No.01
https://doi.org/10.7840/kics.2021.46.1.181

시각장애인을 위한 점자키보드 구현

권 종 훈•, 임 완 수°

Implementation of Braille Keyboards for the Visually Impaired

Jonghun Kwon•, Wansu Lim°

요 약

본 논문은 시각장애인과 비장애인과의 정보격차를 해소하고, 시각장애인의 스마트 기기에 대한 접근성 및 편리성 향상을 위한 국문/영문 역점역 알고리즘을 탑재한 점자키보드를 구현하였다. 국문 역점역 알고리즘은 시각장애인이 사용하는 점자를 키코드로 변환하여 국문으로 출력하는 알고리즘이고, 국립국어원의 역점역 규정을 기반으로 구현하였다. 국문 역점역은 자모음과 약자와 약어까지 표현 가능한 Grade2로 개발하였고, 영문 역점역은 약자와 약어를 제외한 알파벳을 표현 가능한 Grade1으로 개발하였다. 또한, 숫자와 문장부호, 특수 명령어 등을 구현하여 시각장애인의 스마트 기기에 대한 접근성 및 편의성을 향상시켰다. 키코드는 USB HID 키보드 키코드를 사용하여 블루투스를 키보드 모드로 설정하였고, Android, iOS, Windows, MAC 등 다양한 운영체제를 지원할 수 있도록 하였다.

키워드 : 점자키보드, 국문 역점역, 영문 역점역, 스마트 기기, 시각장애인

Key Words : Braille keyboard, reverse Korean braille translation, reverse English braille translation, Smart device, visually impaired

ABSTRACT

In this paper, a reverse Korean/English braille translation algorithm is developed to solve the information gap of the visually impaired, improving the smart device accessibility and convenience. This algorithm converts braille characters into keycode, and output Korean and English, numbers or punctuation marks to the smart device. In order to recognize the received input braille on the smart device, keycode conversion was used to work on Android, iOS, Windows, and MAC operating system. In addition, Korean consonants, vowels, abbreviations, English alphabets, numbers, punctuation marks, etc. have been implemented in accordance with reverse Korean braille translation regulations, making it easier for the visually impaired to use them.

Ⅰ. 서 론

스마트 기기는 언제 어디서나 인터넷에 연결할 수 있어 생활하는 데 필수 기기가 되고 있으며, 많은 사람에게 다양한 콘텐츠 및 서비스를 제공하는 주요기기로 자리 잡고 있다[1-3]. 이러한 변화에 맞춰 국내에서도 스마트 기기의 활용이 확대되고 시장도 급격히 증가하는 상황이다. 그러나 빠르게 성장하는 스마트 기기 분야와는 별개로 시각장애인의 경우에는 그 활용 범위가 한정되어 있다. 이는 대부분의 스마트 기기

※ 본 연구는 금오공과대학교 학술연구비로 지원되었음(202001010001)

• First Author : Kumoh National Institute of Technology, Department of Electronic Engineering, jonghun.kwon@kumoh.ac.kr, 학생회원

° Corresponding Author: Kumoh National Institute of Technology, Department of Aeronautics, Mechanical and Electronic Convergence Engineering, wansu.lim@kumoh.ac.kr, 정회원

논문번호 : 202008-205-D-RN, Received August 20, 2020; Revised September 20, 2020; Accepted October 26, 2020

가 터치 형식의 스크린을 사용해 시각장애인의 UI/UX를 고려하지 않고 있어 이용환경에 대해 높은 접근장벽이 있기 때문이다. 이로 인해 시각장애인은 빠르게 진화하는 스마트 정보화 시대에서 소외되어 일반인과의 정보격차가 증가하고 있다[4,5].

이러한 격차를 줄이기 위해 시각장애인을 위한 다양한 점역 입력 기법이 제안되었다. [6]은 점역사 면담을 통해 점역과 역점역 알고리즘 개선점을 제안하였으나 구현하지는 못했다. [7]은 모바일에서 편리하게 점자를 입력하기 위한 SBraille 기법을 제안하고 관련된 어플리케이션을 소개하였다. 그러나 손가락 피로도를 줄이는 방안에 집중하였기에 시각장애인이 사용하는 점역을 충분히 반영하지 않았다. [8]은 한글, 영어, 숫자 표현이 점자키보드를 구현하였으나, 프로토타입 수준으로 개발하여 방대한 점자규정을 표현하기에는 한계가 있었다. [9]는 요철 필름을 스마트폰 화면에 붙여서 점자를 입력하는 기법을 제안하였다. 수 많은 점자 중 일부 입력이 가능하나, 시각장애인이 사용하는 점자 입력을 완벽히 반영하지 못하였다.

본 논문은 시각장애인과 비장애인의 정보격차를 해소하고, 시각장애인의 스마트 기기 접근성 및 사용 편의성을 향상하기 위한 국문/영문 역점역 알고리즘을 탑재한 점자키보드를 구현하고자 한다. 또한, 시각장애인의 입력 보조 장치 및 스마트 기기와 연동하기 위한 인터페이스를 개발하여 시각장애인의 편의를 도모하고, 활용성을 극대화하여 비장애인과의 정보격차를 줄이고자 한다. 국문 역점역 알고리즘은 기본적으로 국립국어원에서 제정한 점자규정의 규칙을 따르고, 문장부호 및 명령어, 운영체제 변경 등을 포함하여 다양한 기기에서도 사용가능하도록 구현한다. 본 논문의 결과를 활용하면 시각장애인의 정보통신 의사소통을 통해 스마트 정보화 시대의 장벽을 낮출 수 있을 것으로 기대한다.

Ⅱ. 점자키보드 구현

2.1 키코드(keycode) 변환

본 논문의 역점역 알고리즘은 시각장애인이 사용하는 점자를 스마트 기기에 국문 및 영문으로 출력하는 알고리즘이다. 제안한 알고리즘은 점자를 국문 및 영문으로 변환하는 과정에 키코드(Keycode)를 사용하였는데, 이는 스마트 기기가 점자를 인식할 수 없기 때문에, 점자로 입력된 값을 키코드로 변환하여 스마트 기기가 인식할 수 있도록 하였다. 키코드는 일반적인 키보드에서 사용되는 키코드를 사용하였으며, USB HID usage table의 Section 7. 키보드를 위한 USB HID Keyboard Standard Code[10]를 이용하였다. 키코드로 인해 국문 자모음 외에 영문 알파벳, 숫자, 특수문자 등도 변환할 수 있으며, Android, iOS, Windows, MAC 운영체제에서 역점역 알고리즘이 동작하도록 구현했다.

그림 1. 키코드 변환과정 예시
Fig. 1. Example of keycode conversion process

그림 1은 점자로 받은 입력 '김'을 국문 자모음 각각에 해당하는 키코드로 변환하여 스마트 기기에 출력하는 과정을 나타낸 예시이다. 초성 'ㄱ'의 점자 4점과 모음 'ㅣ'의 점자 1-3-5점, 종성 'ㅁ'의 점자 2-6점이 입력되면, 본 논문의 알고리즘을 통해 각 키에 해당되는 키보드 값 KEY_R, KEY_L, KEY_A를 키코드로 각각 변환하여 16진수 데이터 0x15, 0x0F, 0x04를 스마트 기기로 전송한다. 그리고 스마트 기기는 수신한 16진수 데이터를 키코드로 변환하여 화면에 '김'을 출력한다.

2.2 역점역 규칙 구현

한국점자규정은 1997년 국립국어원에서 처음 제정되었으며, 2006년 1차 개정을 거친 뒤 2017년에 2차 개정되었다[11]. 본 연구는 2017년 개정판을 참고하여 자음, 모음, 숫자 및 약자와 약어 등의 점자를 한글로 변환하는 역점역 알고리즘을 구현하였다.

2.2.1 자음

자음의 초성과 종성은 서로 다른 점형을 사용한다. 초성 'ㅇ'은 첫소리 자리에 쓰일 때에는 이를 표기하지 않고 바로 모음을 적으며, 약자 '군(g)'과 동일하여 실제로는 사용되지 않기 때문에 본 연구에서는 구현하지 않았다. 된소리표의 점형은 초성 복자음 'ㄲ, ㄸ, ㅃ, ㅆ, ㅉ'와 같은 쌍자음을 표현할 때 사용되는 것이며, 각 자음 앞에 된소리표(,, 6점)를 붙여서 사용하도록 하였다.

그림 2는 초성과 된소리 알고리즘의 의사코드로, 입력부와 변환부, 그리고 전송 및 출력부의 세 파트로 구성된다. 입력부인 Part 1은 점자 입력을 기다리며 대기하다 점자가 입력되면 버퍼 배열에 저장한다. 이

```
1.  Wait for key press
2.  Key press is key input?
3.  if didn't key input then
4.    go back line 1.
5.  else store inside buffer array // size of buffer:(25x10)
6.    Press space button?                                  Part 1
7.    if didn't press space then
8.      go back line 1.
9.    else check buffer inside
10.     if buffer is empty then
11.       go line 23.
12.   else convert buffer array to keycode array
13.       if there is a 된소리표 before 초성 then
14.         if 초성 is one of 'ㄱ, ㄷ, ㅂ, ㅅ, ㅈ' then
15.           Modifier keycode is Shift
16.         else put 초성 'ㅅ' in the keycode array          Part 2
17.         end
18.       else Modifier keycode is None and put 초성
19.       end
20.     Is buffer all converted?
21.     if still remaining convert work then
22.       go back line 12.
23.     else send keystrokes from keycode array
              to the Bluetooth module via SPI
              Bluetooth module sends keystrokes              Part 3
              to user device
24.     end
25.   end
26.  end
27.  end 초성/된소리 Algorithm
```

그림 2. 초성과 된소리 의사코드
Fig. 2. Consonant pseudo code

후 스페이스를 입력하면 버퍼 배열 내부를 체크하고, 스페이스가 입력되지 않으면 다음 점자 입력을 기다리며 대기한다. 스페이스가 입력되어 버퍼 내부를 체크할 때 버퍼 내부가 비어있다면, 알고리즘은 Part 3 전송 및 출력부의 블루투스를 통해 스마트 기기로 공백(스페이스)을 전송한다. 이후 나오는 모든 알고리즘에서 Part 1 입력부는 동일한 방식을 사용한다.

변환부인 Part 2는 스페이스를 입력했을 때 버퍼 내부가 채워져 있다면, 버퍼 배열을 키코드 배열로 변환하고, 변환이 완료되면 Modifier keycode를 체크한다. 만약 초성 앞에 된소리표가 있고, 초성이 'ㄱ, ㄷ, ㅂ, ㅅ, ㅈ' 중 하나라면 Modifier keycode는 Shift, 두 조건을 만족하지 않는다면 Modifier keycode는 None이 된다. Modifier keycode가 Shift라면 블루투스를 통해 사용자 기기로 전송될 때 shift + 초성을 전송하며 'ㄲ, ㄸ, ㅃ, ㅆ, ㅉ' 중 하나를 출력한다. Modifier keycode가 None이라면 사용자 기기로 초성만 전달하는 형식이다.

전송 및 출력부인 Part 3은 변환이 다 되었는지 확인 후, 키코드 배열로부터 SPI 통신을 통해 블루투스 모듈로 키스트로크를 보내고, 블루투스 모듈은 키스트로크를 사용자 기기로 전송하여 출력하도록 한다. 이후 설명하는 모든 알고리즘의 Part 3 전송 및 출력부는 동일한 방식을 사용한다. 종성 알고리즘은 초성과 된소리의 알고리즘과 동일한 방식의 Part 1, Part 2, Part 3로 동작하며, 점자 규정에 따라 초성 점형과 종성 점형을 다르게 사용한다. 종성 복자음을 사용할 시에는 된소리표를 쓰지 않고 종성 자음을 두 번 입력하는 것으로 사용할 수 있도록 하였다.

2.2.2 모음

'ㅏ, ㅐ, ㅑ, ㅓ, ㅔ, ㅕ, ㅖ, ㅗ, ㅘ, ㅙ, ㅚ, ㅜ, ㅝ, ㅞ, ㅟ, ㅡ, ㅢ, ㅣ' 17가지의 모음은 한 칸의 점자로 표현하고 'ㅒ, ㅙ, ㅞ, ㅢ' 4가지 모음은 두 칸의 점자로 표현한다. 'ㅇ'은 자음 초성으로 사용하지 않으므로 모음 앞에 자음이 없으면 자동적으로 'ㅇ'을 붙여 출력하도록 하였다. 예를 들어 'ㅏ(<)', 'ㅣ(o)'를 점자로 입력했을 때, '아이'를 출력하도록 하였다. 그림 3은 모음 알고리즘의 의사코드로, 입력부인 Part 1과 전송 및 출력부인 Part 3은 초성과 된소리의 알고리즘과 동일한 방식을 사용한다. 변환부인 Part 2에서는 점자로 모음을 입력하였을 때 모음 앞에 자음이 이미 입력이 되어 있다면 자음과 모음을 같이 어울러서 키코드 배열로 변환하고, 모음만 입력이 되었다면 모음 앞에 자음 'ㅇ'을 추가하여 키코드 배열로 변환한다. 그리고 점자로 모음이 입력되었을 때 앞에 이미 입력된 모음

```
1.  Wait for key press
2.  Key press is key input
3.  if didn't key input then
4.    go back line 1.
5.  else store inside buffer array // size of buffer:(25x10)
6.    Press space button?                                  Part 1
7.    if didn't press space then
8.      go back line 1.
9.    else check buffer inside
10.     if buffer is empty then
11.       go line 22.
12.   else convert buffer array to keycode array
13.       if there is a 자음 before 모음 then
14.         put 자음 then 모음 in keycode array
15.       else if only 모음 input then                     Part 2
16.         put 'ㅇ' then 모음 in keycode array
17.       else special case*
18.       end
19.     Is buffer all converted?
20.     if still remaining convert work then
21.       go back line 12.
22.     else send keystrokes from keycode array
              to the Bluetooth module via SPI
              Bluetooth module sends keystrokes              Part 3
              to user device
23.     end
24.   end
25.  end
26.  end 모음 Algorithm
```

그림 3. 모음 의사코드
Fig. 3. Vowel pseudo code

이 있다면 이중모음이 되는 특수한 경우도 존재하며, 이는 그림 4에서 설명하도록 한다.

그림 4는 이중모음 알고리즘 의사코드로, 입력부인 Part 1과 전송 및 출력부인 Part 3은 모음 알고리즘과 동일하다. 하지만 이중모음 중 'ㅐ, ㅒ, ㅖ, ㅟ,'는 두 칸의 사용하는 점자로 'ㅒ'는 'ㅑ'와 'ㅐ'가 어울러서 된 글자이고, 'ㅟ'는 'ㅜ'와 'ㅐ'가 어울러서 된 글자, 'ㅒ'는 'ㅑ'와' ㅐ'가 어울러서 된 글자, 'ㅖ'는 'ㅕ'와 'ㅐ'가 어울러서 된 글자이다. 때문에 'ㅑ', 'ㅘ', 'ㅝ', 'ㅜ' 다음에 'ㅐ'가 연결되는 문자가 온다면 의미가 변형되어 전달될 수 있다. 이와 같은 이유로 변환부인 Part 2에서는 각 경우에 따라서 변환되도록 하였고 'ㅑ', 'ㅘ', 'ㅝ', 'ㅜ' 다음에 'ㅐ'가 연결되는 문자를 쓸 때에는 붙임표(o, 3-6점)를 이용하여 구분한다. 예를 들어 '소화액'(,ujv-ra)을 입력할 때 붙임표를 넣지 않을 시 '소확'(,ujvra)을 출력하고, 구애(@m-r)를 입력할 때 붙임표를 넣지 않으면 '귀'(@mr)를 출력한다.

쌍시옷 받침 'ㅆ'과 모음 'ㅖ'은 동일한 점형(/, 3-4점)을 사용하여, 이를 구분하기 위해 붙임표(-, 3-6점)를 이용한다. 점형 3-4점 앞에 붙임표가 있다면 '예', 점형 3-4점 앞에 붙임표가 없다면 쌍시옷 받침 'ㅆ'으로 변환된다. 예를 들어 '서예(,s-/)'에서 붙임표를 넣지 않으면 '섰(,s/)'을 출력한다. 하지만, 붙임표는 붙일 때 혼동을 일으키지 않는 단어라면 붙임표를 쓰지 않는다. 예를 들어 '노예(cu-/)'나 '도예(iu-/)'는 붙임표를 쓰지 않으면 '놌', '돘'이 되지만, 'ㅙ'을 사용하는 단어는 존재하지 않기 때문에 자동으로 '노예', '도예'로 변환하여 출력한다.

```
1.  Wait for key press
2.  Key press is key input
3.  if didn't key input then
4.    go back line 1.
5.  else store inside buffer array // size of buffer:(25x10)
6.    Press space button?                                  Part 1
7.    if didn't press space then
8.      go back line 1.
9.    else check buffer inside
10.     if buffer is empty then
11.       go line 29.
12.   else convert buffer array to keycode array
13.       if there is a 모음 before 'ㅐ(1-2-3-5점)' then
14.         if 모음 is 'ㅑ(3-4-5점)' then
15.           put 'ㅒ' in keycode array
16.         else if 모음 is 'ㅘ(1-2-3-6점)' then
17.           put 'ㅙ' in keycode array
18.         else if 모음 is 'ㅝ(1-2-3-4점)' then
19.           put 'ㅞ' in keycode array                    Part 2
20.         else if 모음 is 'ㅜ(1-3-4점)' then
21.           put 'ㅟ' in keycode array
22.         else put 'ㅐ' in keycode array
23.         end
24.       else go line 29.
25.       end
26.   Is buffer all converted?
27.   if still remaining convert work then
28.     go back line 12.
29.   else send keystrokes from keycode array
            to the Bluetooth module via SPI
            Bluetooth module sends keystrokes                Part 3
            to user device
30.   end
31.  end
32. end
33. end 이중모음 Algorithm
```

그림 4. 이중모음 의사코드
Fig. 4. Double vowel pseudo code

2.2.3 약자와 약어

한국점자규정은 자주 사용하는 단어 및 자모음에 대해 약자와 약어를 제공한다. 즉, "가"를 표현할 때 "ㄱ(@)"과 "ㅏ(<)"로 표현하지 않고, 약자 표현인 "가($)"로 표현한다.

그림 5는 약자 '나, 다, 마, 바, 자, 카, 타, 파, 하' 등 9개 표현에 관한 규정을 구현한 의사코드이다. 약자 '나, 다, 마, 바, 자, 카, 타, 파, 하' 는 초성 자음 'ㄴ, ㄷ, ㅁ, ㅂ, ㅈ, ㅋ, ㅌ, ㅍ, ㅎ' 과 동일한 점형을 사용한다. 따라서 변환부인 Part 2에서는 초성 자음 'ㄴ, ㄷ, ㅁ, ㅂ, ㅈ, ㅋ, ㅌ, ㅍ, ㅎ' 이 단독으로 쓰일 때, 모음 'ㅏ'를 자동적으로 추가하여 변환하도록 하였다. 그리고 약자 '나, 다, 마, 바, 자, 카, 타, 파, 하'가 낱말 또는 문장의 끝에 사용되거나, 약자 '나, 다,

```
1.  Wait for key press
2.  Key press is key input
3.  if didn't key input then
4.    go back line 1.
5.  else store inside buffer array // size of buffer:(25x10)
6.    Press space button?                                  Part 1
7.    if didn't press space then
8.      go back line 1.
9.    else check buffer inside
10.     if buffer is empty then
11.       go line 19.
12.   else convert buffer array to keycode array
13.     if there is no 모음 after 초성* then
14.       put 'ㅏ' after 초성 in keycode array**           Part 2
15.     end
16.   Is buffer all converted?
17.   if still remaining convert work then
18.     go back line 12.
19.   else send keystrokes from keycode array
            to the Bluetooth module via SPI
            Bluetooth module sends keystrokes                Part 3
            to user device
20.   end
21.  end
22. end 약자 'ㅏ' Algorithm
```

* 가, 라, 사, 차는 제외

** 초성 다음에 받침은 없으나 다음 글자가 모음일 때

그림 5. 약자 의사코드
Fig. 5. Shortcut word pseudo code

마, 바, 자, 카, 타, 파, 하' 다음에 받침이 있거나, 약자 '나, 다, 마, 바, 자, 카, 타, 파, 하' 다음에 받침은 없지만 다음에 오는 글자가 자음으로 연결될 때도 모음 'ㅏ'의 입력 없이 자동적으로 모음 'ㅏ'를 추가하여 변환한다. 예를 들어 '달밤(i1^5)', '바다(^i)'라는 단어는 모음 'ㅏ'점자가 전부 생략된 단어이다. 하지만 약자 '나, 다, 마, 바, 자, 카, 타, 파, 하'에 받침이 없고 이어서 나오는 글자가 모음으로 연결된다면, 모음 'ㅏ'의 입력을 생략할 수 없다. 예를 들어 '나이(c<o)'에서 모음 'ㅏ'가 생략되면 '니(co)'로 의미가 변형되기 때문이다.

2.2.4 숫자

점자에서 숫자를 사용할 때는 숫자 앞에 수표(#, 3-4-5-6점)를 사용한다. 이는 숫자 점형과 초성 자음 중 몇 가지는 같은 점형이기 때문이다. 초성 자음 'ㄴ, ㅍ, ㅁ, ㅋ, 약자 ㄱ, ㅌ, ㄷ, ㅎ'은 숫자 '3, 4, 5, 6, 7, 8, 9, 0'과 같은 점형이기 때문에 숫자 다음에 초성 자음 'ㄴ, ㅍ, ㅁ, ㅋ, 약자 ㄱ, ㅌ, ㄷ, ㅎ'이 나올 때는 숫자 다음에 붙임표(-)를 사용해서 국문과 숫자를 구분한다. 예를 들어 '1년'은 #ac* 이 아니라 #a-c* 와 같이 붙임표를 넣어 사용한다.

그림 6은 숫자를 구현한 의사코드로, 변환부인 Part 2는 수표(3-4-5-6점)를 입력한 상태에서 붙임표(3-6점)를 입력하기 전까지 숫자와 동일한 초성 자음을 입력하여도 숫자로 변환하도록 한 코드이다. 숫자 다음에 모음 21자와 약자(ㅐ, ㄴ, ㄹ, ㄶ, ㄻ, ㅇ, ㅜ, ㄷ, ㅎ, ㄳ, ㄷ, ㄹ, ㄴ), 초성 자음 'ㄱ, ㄹ, ㅂ, ㅅ, ㅈ, ㅊ'과 약자 '가, 사, 것'이 올 때는 숫자 다음에 붙임표(-)를 사용하지 않는다.

```
1.  Wait for key press
2.  Key press is key input
3.  if didn't key input then
4.    go back line 1.
5.  else store inside buffer array // size of buffer:(25x10)
6.    Press space button?                                   Part 1
7.    if didn't press space then
8.      go back line 1.
9.    else check buffer inside
10.     if buffer is empty then
11.       go line 24.
12.     else convert buffer array to keycode array
13.       if 수표(3-4-5-6점) then
14.         All following input is number?
15.         while input is number do
16.           if next character is NOT 초성자음 'ㄴ,
                ㅍ, ㅁ, ㅋ, ㅌ, ㄷ, ㅎ' or 약자 'ㄴ' then   Part 2
17.           else need 붙임표 to stop number
18.           end
19.         end
20.       end
21.     Is buffer all converted?
22.     if still remaining convert work then
23.       go back line 12.
24.     else send keystrokes from keycode array
            to the Bluetooth module via SPI
            Bluetooth module sends keystrokes               Part 3
            to user device
25.   end
26. end
27. end 숫자 Algorithm
```

그림 6. 숫자 의사코드
Fig. 6. Number pseudo code

2.3 문장부호 구현

문장부호는 규정이 없어서, 점자책, 점자입력기기마다 서로 다른 문장부호 표현을 사용한다. 따라서 본 연구는 시각장애인이 가장 많이 사용하는 시각장애인용 점자정보단말기 한소네 U2의 문장부호를 인용하였고, 국문 문장부호만 구현하였다. 표 1은 본 연구에서 설정하고 구현한 문장부호로 7점은 백스페이스(Backspace) 버튼, 8점은 엔터(Enter) 버튼을 나타내

표 1. 문장부호
Table 1. Punctuation

문장부호	점형	문장부호	점형
소괄호 열기 (	3-6-7점	콜론 :	5-8점, 2-8점
소괄호 닫기)	3-6-8점	세미콜론 ;	5-6-8점, 2-3-8점
중괄호 열기 {	2-3-6-8점, 2-3-8점	물결 ~	3-6-8점, 3-6-8점
중괄호 닫기 }	5-6-8점, 3-5-6-8점	샵 #	3-4-5-6-8점
대괄호 열기 [	2-3-6-8점, 3-8점	달러 $	1-2-4-6-8점
대괄호 닫기]	6-8점, 3-5-6-8점	퍼센트 %	1-4-6-8점
대시 -	3-6-7-8점	엠퍼센트 &	1-2-3-4-6-8점
슬래시 /	3-4-8점	등호 =	2-5-8점, 2-5-8점
점 .	2-5-6-8점	더하기 +	2-6-8점
콤마 ,	5-8점	빼기 -	3-5-7점
별표 *	3-5-8점, 3-5-8점	보다작다 <	1-2-6-8점
느낌표 !	2-3-5-8점	보다크다 >	3-4-5-8점
물음표 ?	2-3-6-7점	백슬래시 ＼	1-2-5-6-7점
큰따옴표 “	2-3-6-8점	버티컬 바 |	1-2-5-6-8점
	3-5-6-8점	캐럿 ^	4-5-7점
	5-7점	언더라인 _	4-5-6-7점
작은따옴표 ‘	6-8점, 2-3-6-8점	액센트 ‘	4-7점
	3-5-6-8점, 3-8점	At @	4-8점
	3-7점		

며, 일반 키보드에서 사용하지 않는 문장부호는 제외하고 개발하였다.

2.4 운영체제 모드 변경

스마트 기기는 다양한 운영체제를 사용하며, 각 운영체제는 서로 다른 키코드를 사용한다. 따라서 본 역점역 알고리즘은 다양한 운영체제를 지원하기 위해 운영체제 모드 설정이 필요하며, Android, iOS, Windows, MAC 4가지 운영체제를 지원하도록 하였다. 기본 설정은 Android로 설정하였다.

표 2는 운영체제 변경키를 정리한 것으로, OS Mode 버튼과 1~4점 버튼을 입력하면 모드가 변경된다. OS Mode 버튼과 1점은 Android, OS Mode 버튼과 2점은 iOS, OS Mode 버튼과 3점은 Windows, OS Mode 버튼과 4점은 MAC을 지원하도록 하였다. 그림 7은 운영체제 변경에 관한 의사코드이고, 입력부인 Part 1과 전송 및 출력부인 Part 3은 3.2절 기본 역점역 규칙 구현의 알고리즘들과 동일한 방식을 사용한다. 하지만 변환부인 Part 2에서는 Android와 Windows, iOS와 MAC 간의 특수명령어 키코드가 서로 다르기 때문에 운영체제를 변경할 때마다 버퍼 내부를 초기화 시켜 문제가 발생하지 않도록 하였다.

```
1.  Wait for key press
2.  Key press is key input
3.  if didn't key input then
4.    go back line 1.
5.  else store inside buffer array // size of buffer:(25x10)
6.    Press space button?                                  Part 1
7.    if didn't press space then
8.      go back line 1.
9.    else check buffer inside
10.     if buffer is empty then
11.       go line 21.
12.     else convert buffer array to keycode array
13.       if 1점 with OS Mode button then
14.         clear buffer and change to Android key command
15.       else if 2점 with OS Mode button then
16.         clear buffer and change to iOS key command       Part 2
17.       else if 3점 with OS Mode button then
18.         clear buffer and change to Windows key command
19.       else if 4점 with OS Mode button then
20.         clear buffer and change to MAC OS key command
21.       else send keystrokes from keycode array
                to the Bluetooth module via SPI
                Bluetooth module sends keystrokes
                to user device                               Part 3
22.     end
23.   end
24.   end
25. end OS Mode Algorithm
```

그림 7. 운영체제 변환 의사코드
Fig. 7. OS mode pseudo code

표 2. 운영체제 모드
Table 2. Operation system mode

Keys	Mode
OS Mode + 1점	Android(Default)
OS Mode + 2점	iOS
OS Mode + 3점	Windows
OS Mode + 4점	MAC

2.5 특수명령어 구현

2.5.1 커서 이동

시각장애인은 문서 내에서 문장의 편집이나 TTS(Text to Speech) 기능을 활용하기 위해 커서 이동 기능이 필수적이다. Andorid와 Windows 상에서 작동 가능한 커서 이동 명령어는 MS Word와 한컴오피스 한글의 명령어를 기반으로 구현하였고, iOS와 MAC은 다른 키코드 명령어를 사용하였다. 표 3은 커서 이동에 관한 명령어이다.

표 3. 커서 이동 명령어
Table 3. Cursor movement commands

Function	Keys
이전 글자로 이동	Space + 3점
다음 글자로 이동	Space + 6점
이전 단어로 이동	Space + 2점
다음 단어로 이동	Space + 5점
줄의 처음으로 이동	Space + 13점
줄의 끝으로 이동	Space + 46점
이전 줄로 이동	Space + 1점
다음 줄로 이동	Space + 4점
이전 문단으로 이동	Space + 23점
다음 문단으로 이동	Space + 56점

2.5.2 한/영 전환

본 역점역 알고리즘의 국문과 영문 점자규정은 국립국어원의 2017년 개정판과 UEB Standard Second Edition 2013 개정판을 각각 이용하였다. 영문은 국문과의 문자 사용법이 다르기 때문에 간단히 알파벳 정도만 사용가능하도록 구현하였다. 영문은 UEB Standard Second Edition 2013 개정판에서 Section 6, 7, 8을 구현하였으며, Section 6 부분은 numeric passage, line continuation(numeric mode), fraction line을 제외하였고, Section 7 부분은 문장부호 파트로 ‘ , ’, ‘ ; ’, ‘ : ’, ‘ ... ’, ‘ ! ’, ‘ ? ’, ‘ ‘ ’, ‘ - ’를 구현하였다. Section 8 부분은 대문자 파트로 UEB 규

```
1.  Wait for key press
2.  Key press is key input
3.  if didn't key input then
4.    go back line 1.
5.  else store inside buffer array // size of buffer:(25x10)
6.    Press space button?                                  Part 1
7.    if didn't press space then
8.      go back line 1.
9.    else check buffer inside
10.     if buffer is empty then
11.       go line 17.
12.   ------------------------------------------------------------
13.     else convert buffer array to keycode array
14.       if space button with backspace button then
15.         clear buffer and change to English             Part 2
16.       else if space button with enter button then
17.         clear buffer and change to Korean
18.     ------------------------------------------------------------
19.       else send keystrokes from keycode array
                to the Bluetooth module via SPI
                Bluetooth module sends keystrokes
                to user device
                                                           Part 3
20.     end
21.   end
22. end 한/영 전환 Algorithm
```

그림 8. 국문/영문 전환 의사코드
Fig. 8. Korean/English conversion pseudo code

정에 따라 구현하였고, 영문 약자와 약어는 구현하지 않았다.

그림 8은 국문과 영문의 전환 알고리즘 의사코드로, 국문에서 영문으로 전환 시에는 스페이스와 엔터를 입력하고, 영문에서 국문으로 전환 시에는 스페이스와 백스페이스를 입력한다. 또한, 국문과 영문으로 전환될 때 버퍼 내부를 초기화하여 국문과 영문이 뒤섞여 출력되는 일이 없도록 하였으며, 국문상태에서 국문으로, 또는 영문상태에서 영문으로 전환 버튼이 입력되었을 때는 코드가 동작하지 않도록 하였다.

2.6 입력 전송 방식

그림 9는 입력 전송 방식을 시각화한 그림으로 점자를 입력하면 InputBuffer에 저장되고, 이후 스페이스를 입력하면 InputBuffer의 점자는 키코드로 변환된 후 SendBuffer를 거쳐 블루투스를 통해 사용자 기기로 전송되어 국문 및 영문이 출력된다.

InputBuffer는 2차원 배열 형태로 가로 25, 세로 8의 크기를 가지는 행렬이다. 세로 8개의 행에서 1행부터 6행까지는 각각 점자의 1점부터 6점까지의 값을 저장하며, 7행은 백스페이스, 8행은 엔터 값을 저장한다. 예시로 점자 '김'을 입력하면 4점(ㄱ), 1-3-5점(ㅣ), 6점(ㅁ)이 InputBuffer에 저장되고, 이후 스페이스를 입력하면 InputBuffer 내부의 값을 [표 3.1]의 USB HID Keyboard Standard Code에 따라 키코드로 변환하여 SendBuffer에 저장한다.

SendBuffer는 2차원 배열 형태로 가로 25, 세로 2의 크기를 가지는 행렬이다. 1행은 Shift, Ctrl, Alt 등의 특수키를 저장하고, 2행은 국문 자모음, 영문 알파벳, 숫자, 문장부호 등의 키코드를 저장한다. 예시로 점자 '김'을 입력하였을 때 1행의 특수키는 None, 2행은 키코드로 변환된 값 'key_r, key_l, key_a'가 저장되었다. 그리고 SendBuffer 내부 키코드에 해당되는 16진수 데이터 값을 블루투스를 통해 사용자 기기로 전송시켜 출력한다. 예시, '김'은 키코드 'key_r(ㄱ), key_l(ㅣ), key_a(ㅁ)'에 해당되는 16진수 데이터 값 0x15, 0x0F, 0x04를 블루투스를 통해 사용자 기기로 전송하여 국문 '김'을 출력하였다.

2.7 하드웨어 제작

점자키보드는 Perkin's layout keyboard를 기반으로 제작하였다. Perkin's layout 스타일은 왼손 검지, 중지, 약지가 각각 1점, 2점, 3점, 오른손 검지, 중지, 약지가 각각 4점, 5점, 6점을 쓰며 왼손 소지가 7점 백스페이스, 오른손 소지가 8점 엔터키, 가운데 하단에 양손 엄지가 놓이는 부분에 스페이스바와 기타 기능키를 넣은 스타일을 뜻한다.

그림 10은 테스트용 점자기기의 하드웨어 블록다이어그램으로, 일반 키보드와 같이 다양한 키는 없지만, Perkin's layout keyboard의 인터페이스와 같이 10개의 점자 입력 버튼으로 다양한 문자를 입력할 수 있

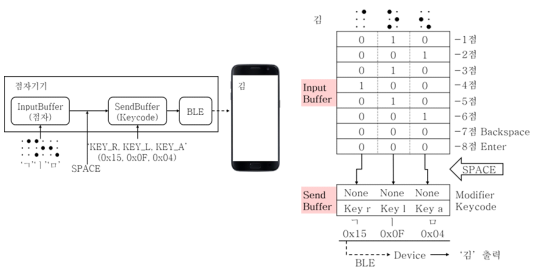


그림 9. 입력 전송 방식 블록다이어그램
Fig. 9. Input transmission method block diagram

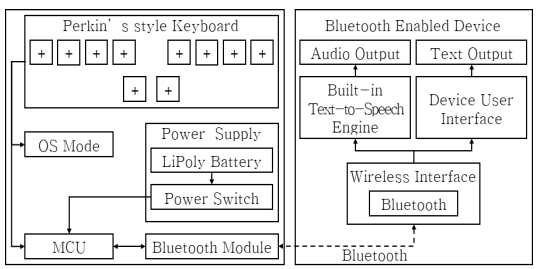


그림 10. 하드웨어 블록다이어그램
Fig. 10. Hardware structure

도록 설계하였다. 8개의 버튼을 상단에 놓고 하단부에 스페이스 버튼 하나, 운영체제 변경을 위한 버튼 하나로 총 10개의 버튼으로 인터페이스를 구성하였다. 또한, 스마트 기기와는 블루투스를 사용하여 쉽게 연결되도록 하였다. 대부분의 스마트 기기가 TTS기능을 지원하기 때문에 점자키보드에서 오디오 출력부는 구현하지 않았다.

그림 11은 하드웨어 블록다이어그램을 바탕으로 제작한 점자키보드이다. 6점자를 입력하기 위한 6개의 1점(①), 2점(②), 3점(③), 4점(④), 5점(⑤), 6점(⑥) 버튼과 백스페이스(ⓑ)버튼, 엔터(ⓔ)버튼, 운영체제변경(ⓜ)버튼, 스페이스(ⓢ)버튼, MCU, 저전력 블루투스(BLE), 리튬 배터리, 전원 스위치로 구성되어 있다. MCU는 Atmel사의 ATSAMD21G18, BLE 모듈은 MDBT40-256RV3, 배터리는 리튬배터리로 400mAh, 3.7V, 1.48Wh 제원의 제품을 사용하였다.

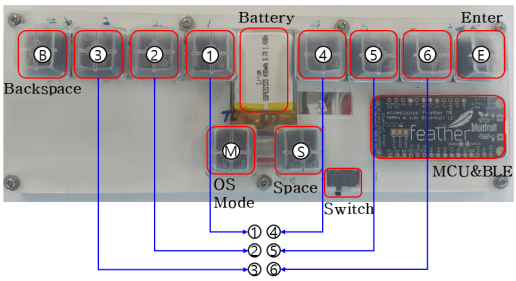


그림 11. 구현한 점자키보드
Fig. 11. Implemented Braille Keyboard

Ⅲ. 점자키보드 동작 테스트

3.1 스마트 기기 연동 실험

본 논문의 점자키보드는 블루투스 키보드를 지원하는 모든 기기(스마트폰, 태블릿, 노트북 등)와 연결할 수 있도록 하였다. 블루투스를 한번 연결한 후에는 서로를 자동으로 인식하며 페어링 된다. 그림 12는 안드로이드 스마트폰과 테스트 플랫폼을 블루투스로 연결하는 사진이다. 스마트폰 화면에 출력된 "블루투스 키보드를 연결합니다."를 통해 테스트 플랫폼에 업로드된 알고리즘이 키보드 형식으로 인식되며 작동됨을 알 수 있다.

표 4는 각 운영체제와 블루투스 연동 테스트를 실시한 것으로 운영체제변경버튼(Ⓜ)과 1점(①)을 눌러 Android, Ⓜ과 2점(②)을 눌러 iOS, Ⓜ과 3점(③)을 눌러 Windows, Ⓜ과 4점(④)을 눌러 MAC으로 변환

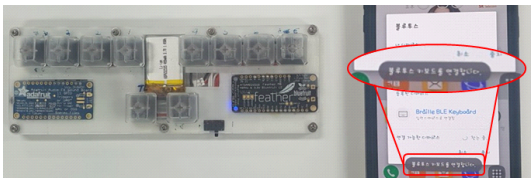


그림 12. 블루투스 연결화면
Fig. 12. Bluetooth connection

한 후 테스트하였다. 블루투스 페어링 후 운영체제마다 국문, 영문, 숫자, 문장부호 등 간단히 테스트하였고, 모든 과정에서 정상적으로 동작하는 것을 확인하였다.

표 5는 안드로이드 스마트폰에서 국문 자모음 및 약자와 약어, 숫자, 문장부호를 테스트한 것으로 국문 사용 시, 긴 문장도 자음과 모음, 약자와 약어를 섞어 사용하여 어려움 없이 쉽게 작성할 수 있었다. 숫자도 수표를 사용하여 1부터 0까지의 출력하였다. 국문과 숫자는 국립국어원에서 제정한 점자규정을 바탕으로

표 4. 운영체제 변환 테스트
Table 4. OS mode test

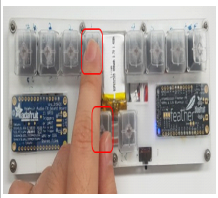
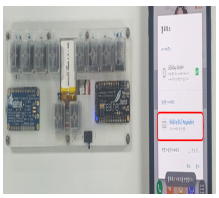
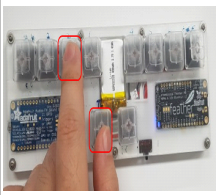
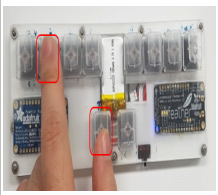
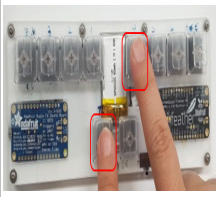
구분	OS Mode 변경	테스트
Android (Ⓜ+①)		
iOS (Ⓜ+②)		
Windows (Ⓜ+③)		
MAC (Ⓜ+④)		

표 5. 국문, 숫자, 문장부호 테스트
Table 5. Korean, number, puctuation test

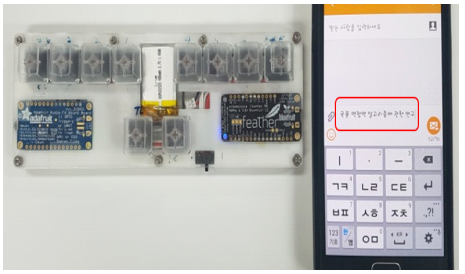
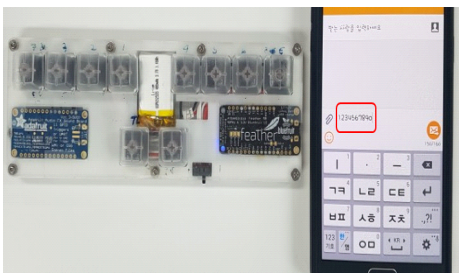
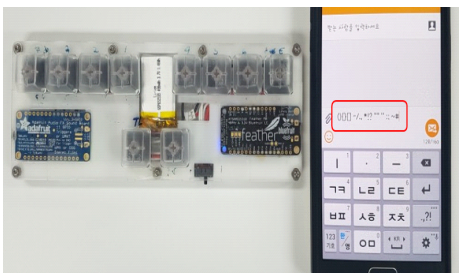
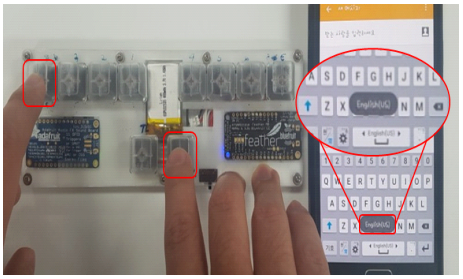
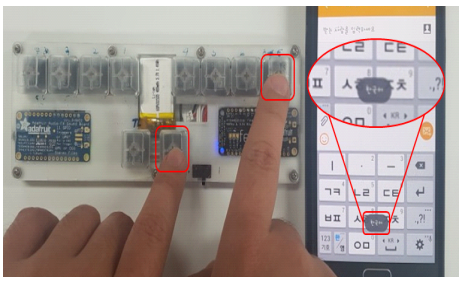
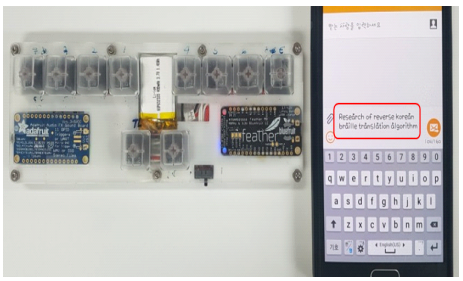
구분	테스트
국문	
숫자	
문장부호	

표 6. 국문/영문 전환 및 영문 테스트
Table 6. Korean/English conversion, English test

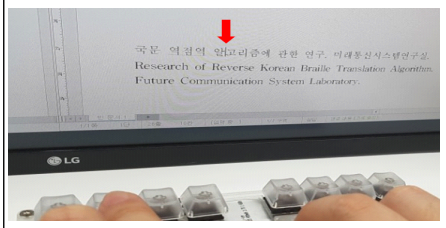
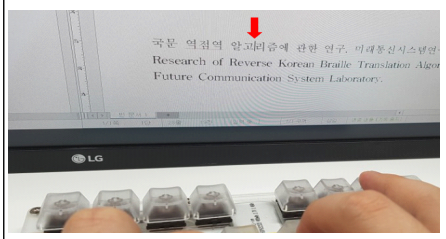
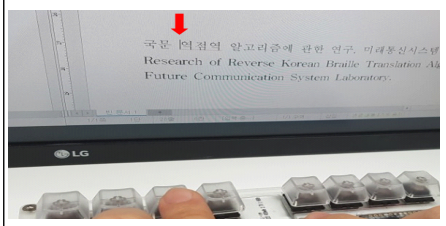
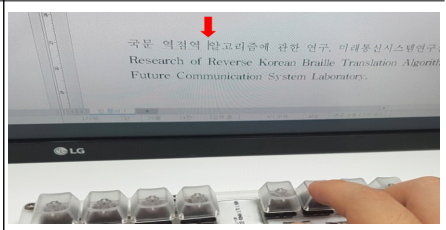
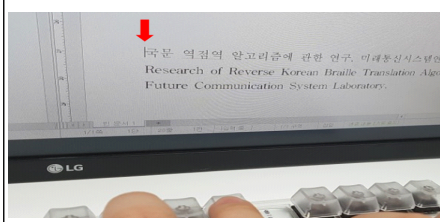
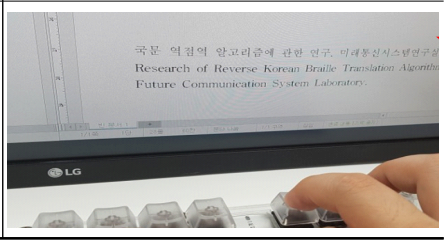
구분	테스트
영문 전환 (Ⓑ+Ⓔ)	
국문 전환 (Ⓢ+Ⓔ)	
영문	

구현하였기에 기본적인 규정은 모두 정상 동작하는 것을 확인하였다. 문장부호는 임의로 선택한 부호를 무작위로 입력하였으며, 입력한 점자와 출력된 문자가 일치하는 것을 확인하였다.

표 6은 안드로이드 스마트폰에서 국문, 영문 전환과 영문 입력을 테스트 하였다. 본 논문의 알고리즘에서 디폴트 설정을 국문으로 하여 영문으로 전환 시에는 영문 전환 명령키인 스페이스(ⓢ) 버튼과 백스페이스(ⓑ) 버튼을 눌러야 한다. 영문에서 국문으로 돌아오고자 할 때는 스페이스 버튼과 엔터(ⓔ) 버튼을 눌러야 하며, 국문, 영문 전환이 정상적으로 작동됨을 확인하였다. 이어서 영문으로 문장을 작성하여 테스트 하였다. 알파벳은 입력한 점자에 맞게 정확히 출력되지만 영문 약자를 구현하지 않아서 긴 문장을 입력할 때에는 불편함이 있을 것으로 예상된다.

표 7은 커서 이동 명령어를 테스트한 것이며 Windows 운영체제 PC에서 한컴오피스 한글로 진행하였다. 테스트는 글자와 단어 단위의 이동과 줄의 처음과 끝으로 이동하는 것을 실시하였다. (a)와 (b)는 각각 '이전 글자로 이동', '다음 글자로 이동'을 테스트한 것으로 '알고리즘'이라는 단어에서 '고'를 기준으로 앞뒤로 커서가 움직이는 것을 확인하였다. (c)와 (d)는 각각 '이전 단어로 이동', '다음 단어로 이동'을 테스트 한 것으로 '알고리즘' 단어에서 이전 단어로 이동하여 '역점역' 앞으로 커서가 이동하였고, '역점역' 앞에서 '알고리즘' 앞으로 커서가 이동하는 것을 확인하였다. (e)와 (f)는 줄 단위로 앞뒤로 움직이는 테스트로 스페이스버튼과 1점, 3점을 눌렀을 때 줄의 시작점으로 커서가 이동하였고, 스페이스버튼과 4점, 6점을 눌렀을 때 줄의 끝으로 커서가 이동하는 것을 확인하였다. 이외 다른 커서 이동 명령어도 정상 작동하는 것을 확인하였다.

표 7. 커서 이동 명령어 테스트
Table 7. Cursor movement command test

구분	테스트
(a) 이전 글자로 이동 (Ⓢ+③)	
(b) 다음 글자로 이동 (Ⓢ+⑥)	
(c) 이전 단어로 이동 (Ⓢ+②)	
(d) 다음 단어로 이동 (Ⓢ+⑤)	
(e) 줄의 처음으로 이동 (Ⓢ+①③)	
(f) 줄의 끝으로 이동 (Ⓢ+④⑥)	

3.2 시각장애인 시험 평가

본 절은 구현한 점자키보드를 시각장애인이 사용한 후 평가한 역점역 알고리즘의 장점과 단점을 정리하였다. 평가는 점자 사용경력 8년 이상, 점자정보단말기 및 점자 키보드 사용경력 3년 이상의 남성 6명, 여성 2명으로 이루어진 8명이 실시하였다. 평가자를 대상으로 점자키보드와 스마트폰을 블루투스로 연결하여 국문, 영문, 숫자, 문장부호의 문자입력과 국문/영문 전환, 커서이동 등의 특수 명령어를 활용한 스크린리더 제어를 수행함으로써 역점역 알고리즘의 효율성과 정확성을 조사하였다.

그림 13은 시각장애인이 점자키보드를 평가하는 사진이고, 국문, 영문, 숫자, 문장부호의 문자입력과 국문/영문 전환, 커서이동 등의 특수 명령어에서 높은 만족도를 나타내었다. 표 8은 국문/영문 입력, 약어/약

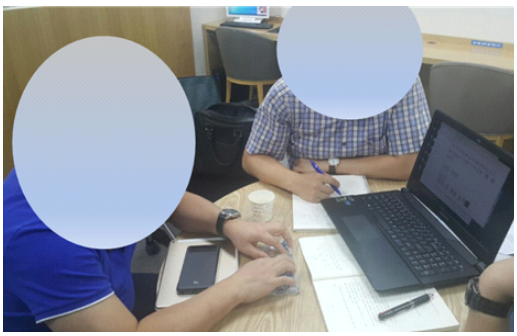


그림 13. 시각장애인 평가
Fig. 13. Visually impaired evaluation

표 8. 입력오류율
Table 8. Input error rate

입력 종류	입력 값	시행 횟수	오류 횟수
국문	자음, 모음, 이중모음	100회	0회
약어/약자	ㄱ, ㄷ, 카, 타, 그리고, 그러면 등	50회	0회
영문	영문 알파벳과 단어	100회	0회
숫자	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	50회	0회
문장부호	(,), {, !, #, $, ‘, “, @ 등	30회	0회
영문 전환	국문→영문, 영문→국문	10회	0회
OS 변경	안드로이드, iOS, 윈도우, MAC OS	10회	0회
커서 이동	이전 글자, 다음 글자, 줄 처음 등	20회	0회

자 입력을 포함하여 구현한 점자 입력에 대한 시행 횟수와 오류 횟수를 정리한 것이며, 시행 방법은 표 4에서 표 7과 같이 실시하였다. 평가 결과 모든 입력에 대하여 오류는 발생하지 않았다.

Ⅳ. 결 론

본 논문은 시각장애인과 비장애인과의 정보격차를 해소하고, 시각장애인의 스마트 기기에 대한 접근성 및 편리성 향상을 위한 국문/영문 역점역 알고리즘을 탑재한 점자키보드를 구현하고 그 성능을 평가하였다. 점자키보드는 아크릴, 아두이노, 기계식 키보드 스위치 등으로 구성하여 제작하였고, 시각장애인에게 익숙한 Perkin 스타일로 버튼을 배치하여 시각장애인이 사용할 때에 불편함이 없도록 접근성 및 편의성을 향상시켰다. 점자를 사용한 경력이 8년 이상 된 시각장애인이 국립국어원에서 제정한 점자규정을 바탕으로 국문, 영문, 숫자, 문장부호와 커서이동 등의 특수명령어를 활용한 스크린리더 제어 등 사용성을 평가하였다.

References

[1] S. Kim and J. Jung, “Proposal of home multimedia device reflecting the behavior of video content consumption trend for binge-watchers of generation Z,” *Ind. Design*, vol. 14, no. 1, pp. 13-24, Jan. 2020.

[2] J. Song and D. Kim, “Factors affecting attitude to use devices in watching video through smart devices,” *J. Korea Contents Assoc.*, vol. 20, no. 5, pp. 46-57, May 2020.

[3] H. Kim, Y. Lee, and H. Lee, “Negative transition of smart device utility: empirical study on it-enabled work flexibility, after hours work connectivity, and work-life conflict,” *Informatization Policy*, vol. 26, no. 4, pp. 36-61, Apr. 2019.

[4] S.-W. Kim, J.-K. Lee, and C.-W. Lee, “Implementation of information access embedded system for the blind people,” *J. KICS*, vol. 33, no. 2, pp. 167-172, Feb. 2008.

[5] J. Song and D. Kim, “A study on ability and utilization of smart devices for the disabled: Focusing on the effect of education for smart device utilization,” *Informatization Policy*, vol. 21, no. 2, pp. 67-88, Feb. 2014.

[6] J. Yook, S. Park, and S.-B. Lim, “A study on improvements of braille and reverse translation programs,” *J. Rehabilitation Res.*, vol. 23, no. 4, pp. 47-59, Apr. 2019.

[7] S. Lee, J. Park, K. Kim, and J. Shon, “A mobile braille input method for reducing hand fatigue,” *J. KIIT*, vol. 16, no. 2, pp. 119-124, Feb. 2018.

[8] M. Koo, B. Kim, and H. Shin, “Efficient braille keyboard of smart phone for the blind,” *Convergence Secur. J.*, vol. 15, no. 2, pp. 11-18, Feb. 2015.

[9] T. Eom, J. Lee, and B. Kim, “Design of smart phone-based braille keyboard system for visually impaired people,” *Convergence Secur. J.*, vol. 12, no. 1, pp. 63-70, Jan. 2012.

[10] USB HID usage table, USB HID Keyboard Standard Code, Section 7.

[11] National Institute of Korean Language.

권 종 훈 (Jonghun Kwon)
2016년 2월 : 금오공과대학교 전자공학부 학사
2018년 2월~현재 : 금오공과대학교 전자공학부 석사
<관심분야> 지능 제어, 임베디드 시스템

임 완 수 (Wansu Lim)
2006년 : 한국항공대학교 정보통신공학과 학사
2007년 : GIST 정보통신공학과 석사
2010년 : GIST 정보통신공학과 박사
2014년 9월~현재 : 금오공과대학교 전자공학부 교수
<관심분야> 지능 제어, 임베디드 시스템
[ORCID:0000-0003-2533-3496]