

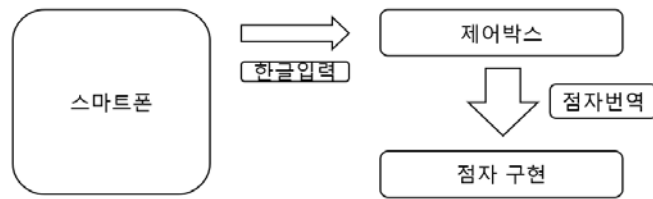
제19회 임베디드SW경진대회 개발완료보고서
[자유공모]

□ 개발 요약

팀 명	3행2열
	
작품명	BRED(: BRaille Education)
작품설명 (요약)	스마트폰을 이용하여 원하는 글자 혹은 무작위 글자를 입력하여 해당 글자를 점자로 변환하여 읽고 쓸 수 있는 휴대성이 높은 점자 교육기기이다.
소스코드	https://github.com/ok701/2021ESWContest_free_1022
시연동영상	https://youtu.be/PNk0jNRVqwc

□ 개발 개요

○ 개발 작품 개요



해당 작품은 시각 장애인분들이 점자교육을 받기 힘든 환경에 놓였을 때 언제 어디서든 쉽게 교육이 가능하게끔 제작되었다. 점자교육 관련 기관과 교육자 수가 적을 뿐 만 아니라 코로나로 인해 기관에 가 교육을 받기가 더더욱 어려워졌다. 따라서 점자교육을 언제 어디서든 쉽게 하기위해 만든 작품으로 점자교육의 접근성을 높이는 것을 최종 목표로 한다.

기존 점자 교육기는 처음 점자를 접하는 사람들, 특히 어린아이들에게 많이 쓰여진다. 하지만 크기가 커 실제 점자와는 차이가 크고 휴대성 또한 좋지 않다. 또한 점자 교육을 하기 위해 점자교육을 받는 비 장애인들(강사,교육자)은 크기가 큰 제품을 굳이 들고 다니지 않아도 눈으로 점자를 확인할 수 있으므로 크기가 큰 교육기기는 비효율적이다. 우린 단어장을 들고 다니면서 단어를 외우는 것처럼 소형화를 통해 언제 어디서든 간편하고 휴대성이 좋은 교육기를 만드는 것을 목표로 한다.

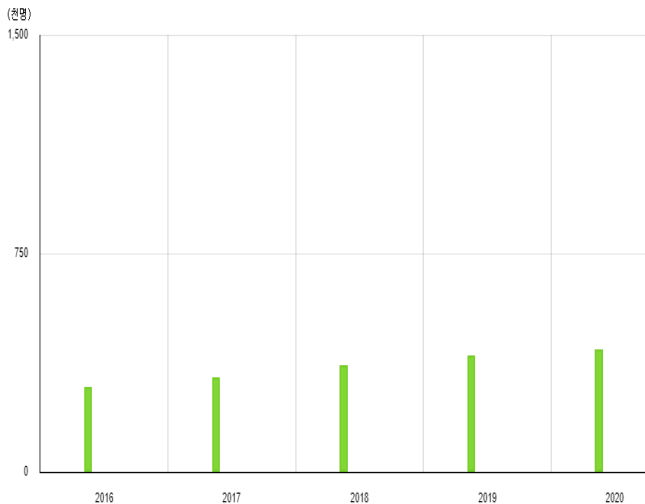
○ 개발 목표

- 점자 교육기기의 상용 보급화
- 점자교육을 스스로 하거나 비전문가의 도움을 받아 원활한 교육을 받을 수 있게끔 제작
- 경제적 부담을 줄일 수 있도록 제작
- 언제 어디서든지 사용할 수 있게끔 제품의 소형화

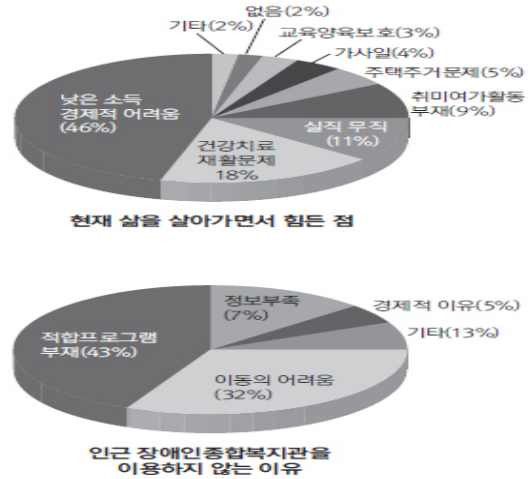
○ 개발 작품의 필요성

-실제사례

김창수 지회장은 고등학교 졸업 이후 시력을 잃은 중도시각장애인이다. 갑자기 빛을 잃은 김 지회장은 한동안 무력감을 겪었다. 그러던 중 점자 책에 손을 뻗었고, 점자로 익힌 지식으로 업을 찾았고, 장애인 인권을 향한 문제의식도 키웠다. 그러면서 "시각장애인들에게 점자를 배우는 행위는 다시 세상에 눈을 뜨는 일과 같다"라고 말했다.



시각장애인들이 겪는 불편함 (자료 경기도시각장애인 연합회 조사)



대한민국 시각 장애인 수는 약 252,000명(2020년 기준, 출처-e 나라지표)으로 전 국민의 약 2%를 차지하고 있고 그 수는 점점 증가하고 있다. 과거부터 지금까지 시각 장애인을 위한 여러 관련 기관 및 시설, 교육 등이 이루어졌지만 아직까지 부족한 점이 많다는 의견이 대다수이다. 기관에서 하는 프로그램이 자신이 원하는 프로그램이 아닌 경우도 많은 경우도 있으며 점자 관련 교육을 받고자 하여도 관련 기관과 점자 교육자의 수가 적어 교육을 받을 기회가 적다. 또한 시각이 불편하여 동반자 없이는 이동이 어려워 정작 장애인분들을 위한 시설임에도 불구하고 시설의 활발한 이용이 이루어지지 않는다. 뿐만 아니라 낮은 소득으로 인해 경제적으로도 부담스러워 프로그램이나 교육을 받기 어려운 사람들도 많다. 이 때문에 점자교육을 받기 어려워하는 사람들이 많다.

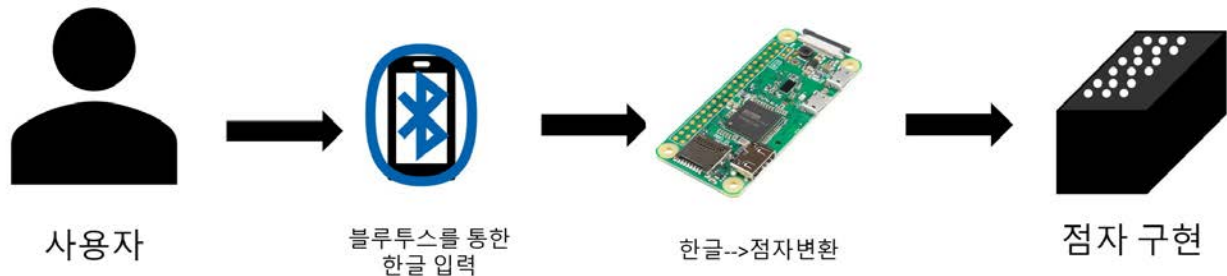
또한, 기존의 제품은 어린아이들 및 점자를 처음 배우는 사람들을 대상으로 하여 다소 부피가 크다. 처음 배우는 입장에서는 분명 크기가 큰 제품이 배우기도 쉽겠지만 익숙해진다면 실제 크기를 바탕으로 한 점자 교육의 필요성을 느낀다. 또한 비장애인이 점자교육을 받는 경우 점자를 눈으로 보고 확인할 수 있음으로 크기가 큰 것은 비효율적이다. 더구나 크기가 크면 휴대성 또한 낮아지기 때문에 소형화된 교육기기가 필요하다.

우리는 이러한 단점들을 보완하고자 경제적으로 값이 싸면서 언제 어디서든지 교육기기를 용이하게 사용할 수 있게끔 하고자 제품을 구상하였다.

□ 개발 환경 설명

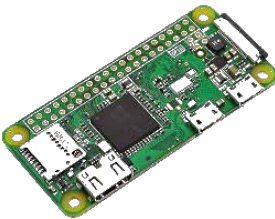

○ Hardware 구성

- 전체 H/W 구동 방식



전체적인 Hardware는 스마트폰, Raspberry pi zero W, 코일 및 스위치, 진동모터로 구성된다. 스마트폰의 앱을 통하여 원하는 단어 혹은 랜덤으로 주어지는 한글을 입력한 후 Raspberry pi zero W에서 한글을 점자로 변환한다. 그 후 변환된 점자는 코일을 통해 구현된 후 사용자는 그에 맞는 버튼을 입력하여 점자를 학습한다.

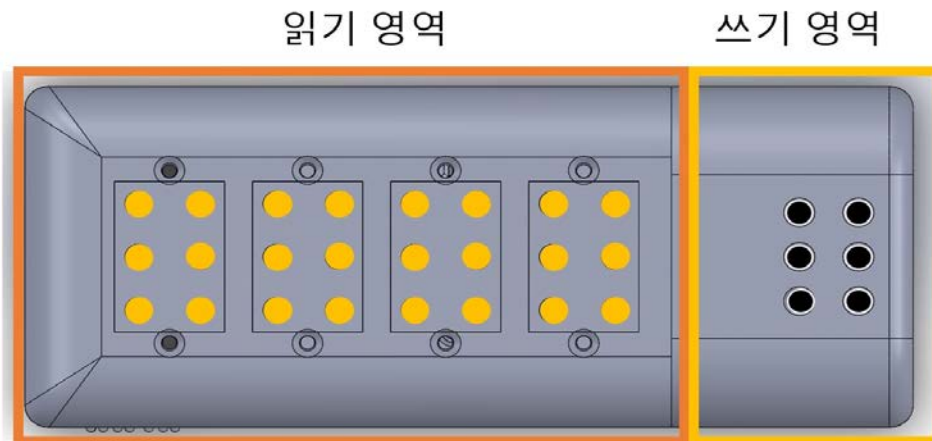
-재료 선정

재료	선정 이유
 Raspberry pi zero W	<p>오픈소스(파이썬 코드)를 사용하기 위해 선택된 MCU.</p> <p>초창기 Raspberry pi zero W의 CPU보다 안정적인 성능을 보이는 CPU-BCM2835, 1GHz ARM11를 사용한다.</p> <p>기존의 Raspberry pi zero W에서 몇 가지 기능을 제거한 대신 크기를 줄인 버전으로 소형화 및 휴대성이 용이해질 뿐만 아니라 경제적인 측면에서도 부담이 적어 선정하였다.</p>
 HC-06	<p>TX/RX 양방향 통신이 가능한 블루투스 모듈로 쉽게 구할 수 있으며 전송 속도 또한 2.1mbps로 준수하다고 판단하였다.</p> <p>또한 비싸지 않은 가격에 스마트폰과 Raspberry pi의 통신을 안정적으로 연결할 수 있다</p>

	<p>Raspberry pi zero의 부족한 Gpio port를 보충하기 위한 시프트 레지스터이다. 가장 많이 범용화 된 소자이며 소형화 및 제어에도 용이하다</p>
<p>74HC-595</p>	
	<p>외부전원을 사용하여 코일을 제어하기 위한 소자. (Max)Ic가 600mA, Vce가 40V로 코일에 필요한 전류(Ic) 200mA를 고려하여 적합하다고 판단. 가격 또한 저렴하다.</p>
<p>2N2222A</p>	
	<p>가장 많이 쓰이는 Tact-Switch로, 연결이 쉽고 누르는 시점에서의 반응도 준수하다.</p>
<p>Tact-Switch</p>	
	<p>트랜지스터의 Base, Collector 에 연결되는 저항으로 2.2k, 10옴 이 사용되었다.</p>
<p>Register</p>	
	<p>쉽게 사용이 가능하며 크기가 작고 필요 전류도 작아 선정하였다.</p>
<p>초소형 진동 모터</p>	
	<p>코일에 역기전력 발생 시 회로에 문제가 생길 수 있으므로 이를 방지하기 위한 다이오드이다.</p>
<p>다이오드</p>	

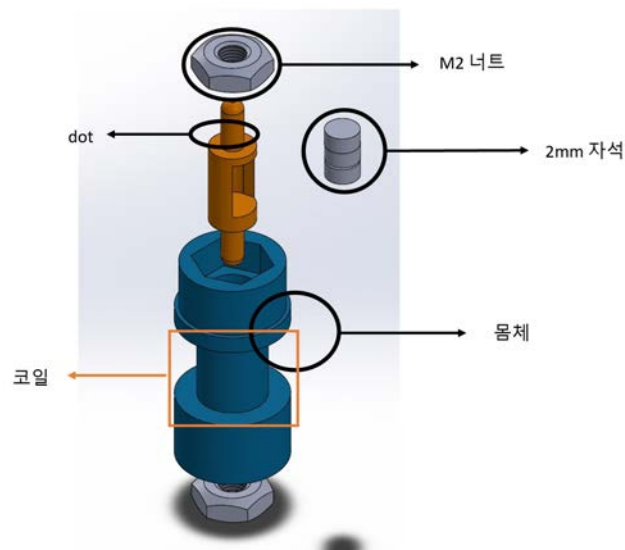
-점자 구현부

하드웨어는 크게 점자를 읽기 영역과 쓰기 영역으로 이루어져 있다.



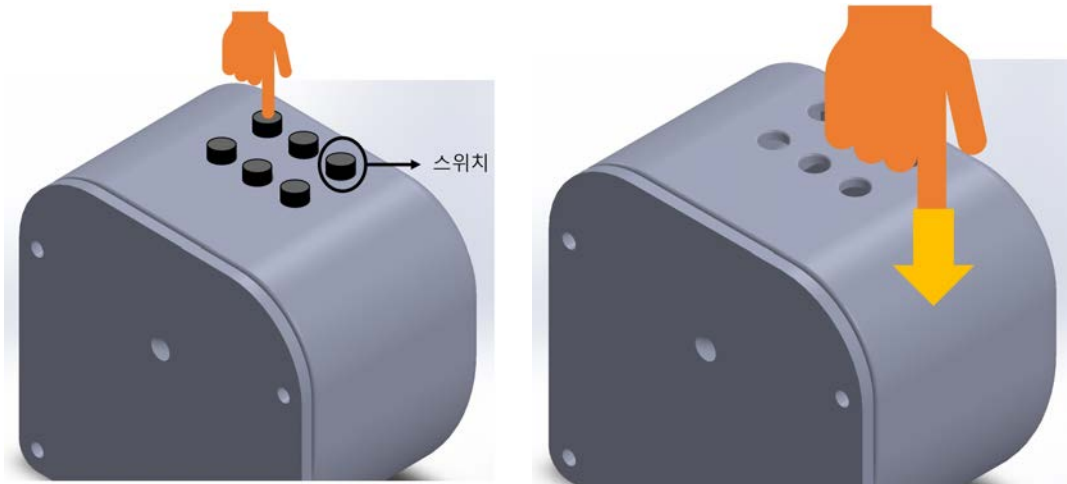
-읽기 영역

읽기 영역은 총 4세트의 6점으로 구성된 점자로 이루어져 있다. 한 단어를 표현하기 위해서는 최소 4세트 이상의 점자가 필요해 4세트로 구성하였고 한 점마다 점자를 표현하였으며, 이를 통해 사용자가 점자를 읽을 수 있게 구성하였다.



점자를 표현해 주는 구동체는 M2 너트, 점자를 표현해주는 부분 dot(이하 dot으로 표현하겠다.), 자석, dot의 움직임을 가이드 해주는 몸체로 이루어져 있다. 코일을 몸체에 감아 솔레노이드로 만들어 전류의 흐름을 제어하여 dot이 상승과 하강을 통해 점자를 표현할 수 있다.

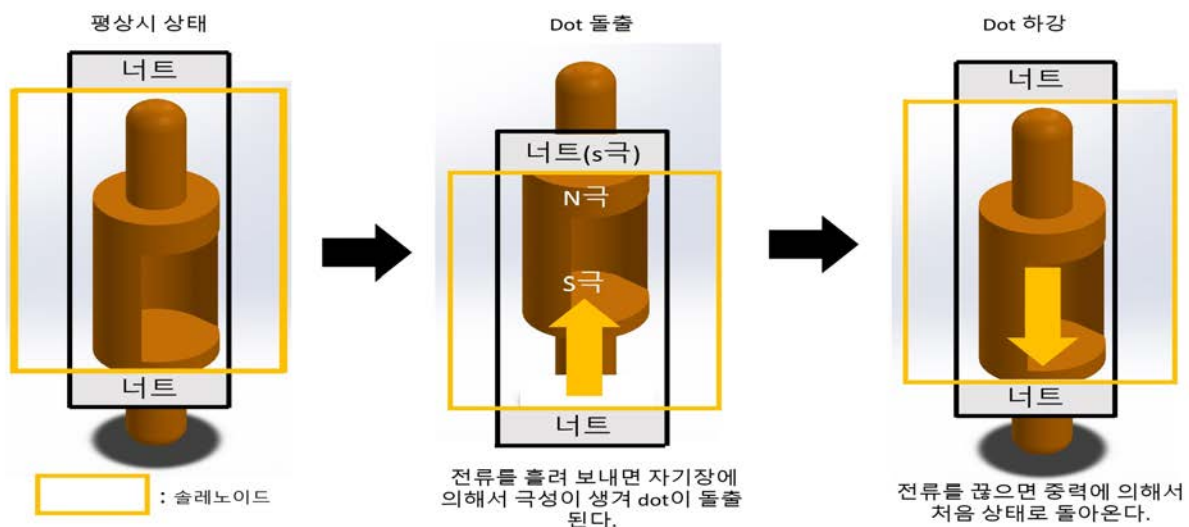
-쓰기 영역



쓰기 영역은 택트 스위치를 이용하여 사용자가 버튼을 누름으로써 점자를 입력할 수 있게 설계하였고 다음 세트를 넘어가기 위해 쓰기 부분을 회전시키는 모션을 통해 다음 단계로 넘어갈 수 있도록(이하 회전버튼이라 명명한다.) 디자인하여 점자를 입력하는 행위와 다음 세트로 넘어가는 행위를 구분하였다. 또한 사용자에게 피드백을 명확히 주기 위하여 내부에 진동모터를 내장시켰다.

○ Hardware 기능 (제어 방법 등 서술)

-dot의 구동방법



-쓰기 영역 Spring 설계



회전 버튼은 가운데 스프링 형태의 설계로 인해 힘을 주어 회전을 하고 난 다음, 탄성의 의해 다시 원래 형태로 돌아올 수 있도록 설계하였다. 이와 같이 설계한 이유는 다음 세트로 넘어가기 위해 버튼을 누르는 행위와 점자를 입력할 때의 행위가 유사하여 혼동을 줄 수 있다고 생각하여 각 행위를 명확히 구분하기 위한 설계가 필요하다고 생각하였다. 따라서 단순히 버튼을 누르기보단 스프링형태로 설계를 하였다. 회전을 인식하는 방법은 내부의 스위치가 회전을 할 때 눌리도록 설계하였다.

-제어 박스 구성 요소

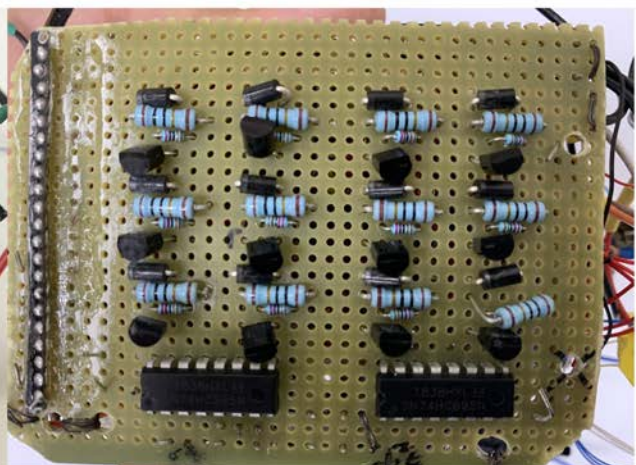
Raspberry pi zero W, 트랜지스터(2N2222a), 저항(2.2k, 10옴), Diode(1N4001), 외부전원 3.7V(18650)

<실제 사진>

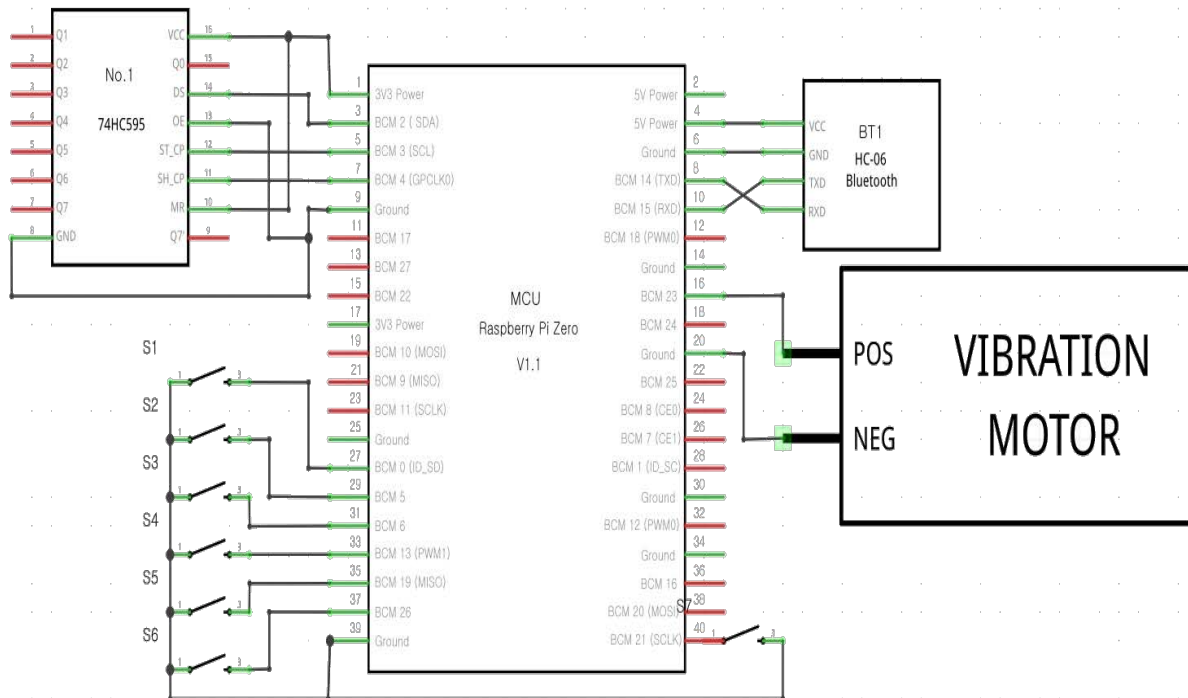
코일 회로 배선(뒤)



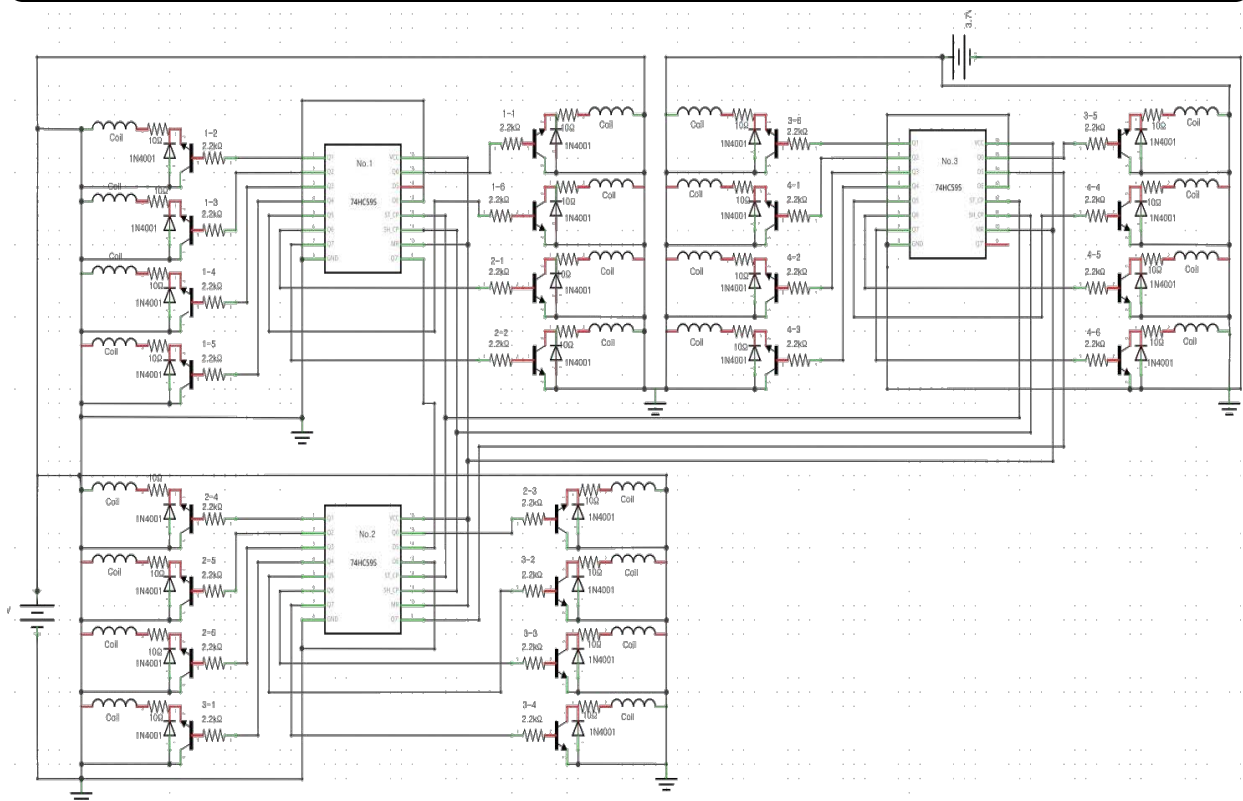
코일 회로 배선(앞)



Raspberry pi zero – 74HC595(No.1) Circuit



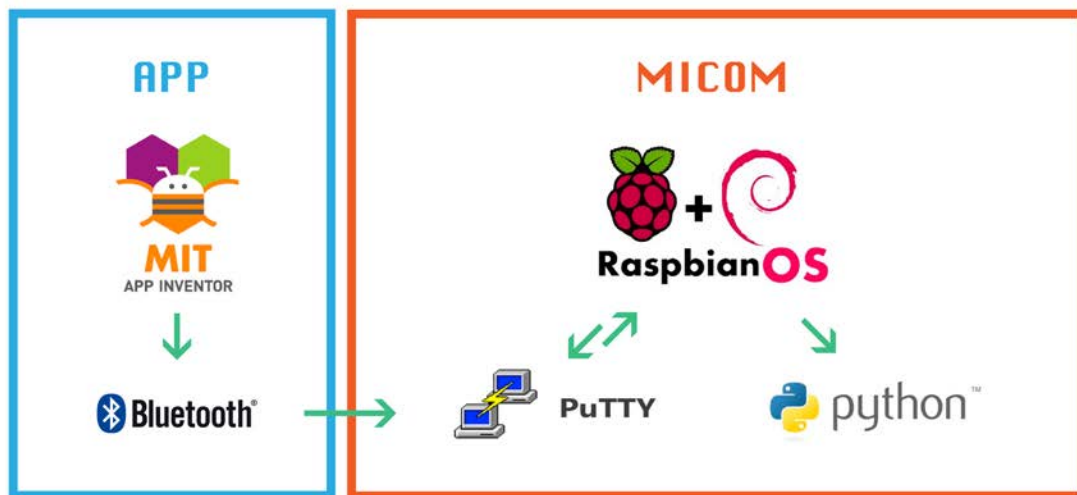
74HC595(No.1) - 74hc595(No.3) Circuit



점자를 표현 할 6개의 점자 기준으로 신호를 줄 포트 6개, 4세트의 점자를 구현하기 위해선 총 24개의 포트가 필요하며 버튼을 제어할 포트 또한 6개의 포트를 사용한다. 또한 블루투스 포트 및 외부전원 그라운드, 회로의 그라운드 등을 포함하면 40개 가까운 포트를 사용한다. 하지만 이 모두를 직접 Raspberry pi zero W에 연결하면 케이블의 부피가 커져 소형화하기 힘들고 핀의 개수 또한 부족하였다. 따라서 가급적 MCU에 연결될 케이블 수를 줄이기 위해 시프트 레지스터 74HC595를 사용하였고 그 결과 부피를 줄일 뿐 만 아니라 20개가 안되는 케이블연결로 제어가 가능하게끔 회로를 설계하였다.

코일을 제어하기 위해선 각 코일마다 200mA의 전류가 필요하다. Raspberry pi zero W의 각 포트에서 200mA를 사용할 수 없으므로 외부전원(3.7V)을 사용하였다. 이를 위해 트랜지스터에 신호를 줘 필요할 때에만 전류를 공급하게끔 설계를 하였으며 코일의 역기전력을 막기 위한 다이오드를 장착하였다.

○ Software 구성



1. Application

MIT App Inventor : 스마트폰 애플리케이션 개발을 위해 MIT App Inventor 를 사용하였다.
스마트폰과 기기를 (Master – Slave 관계) 블루투스로 연결하고, 문자열을 Serial 통신한다.
스마트폰 애플리케이션 안에서 읽기 연습과 쓰기 연습의 작업들을 구분해서 통신하기 때문에,
Micro Processor 에서는 그 기준에 따라 다음 행동을 하게 된다.

2. Micro Processor – Raspberry Pi Zero

한글 낱말은 초성, 중성, 종성으로 나눌 수 있다. 이를 원소라고 하자. 한글 점자는 각각의 원소에 대하여 그 데이터 값이 존재한다. 따라서 점자 번역을 하기 위해서는 들어온 한글 문자를 각각의

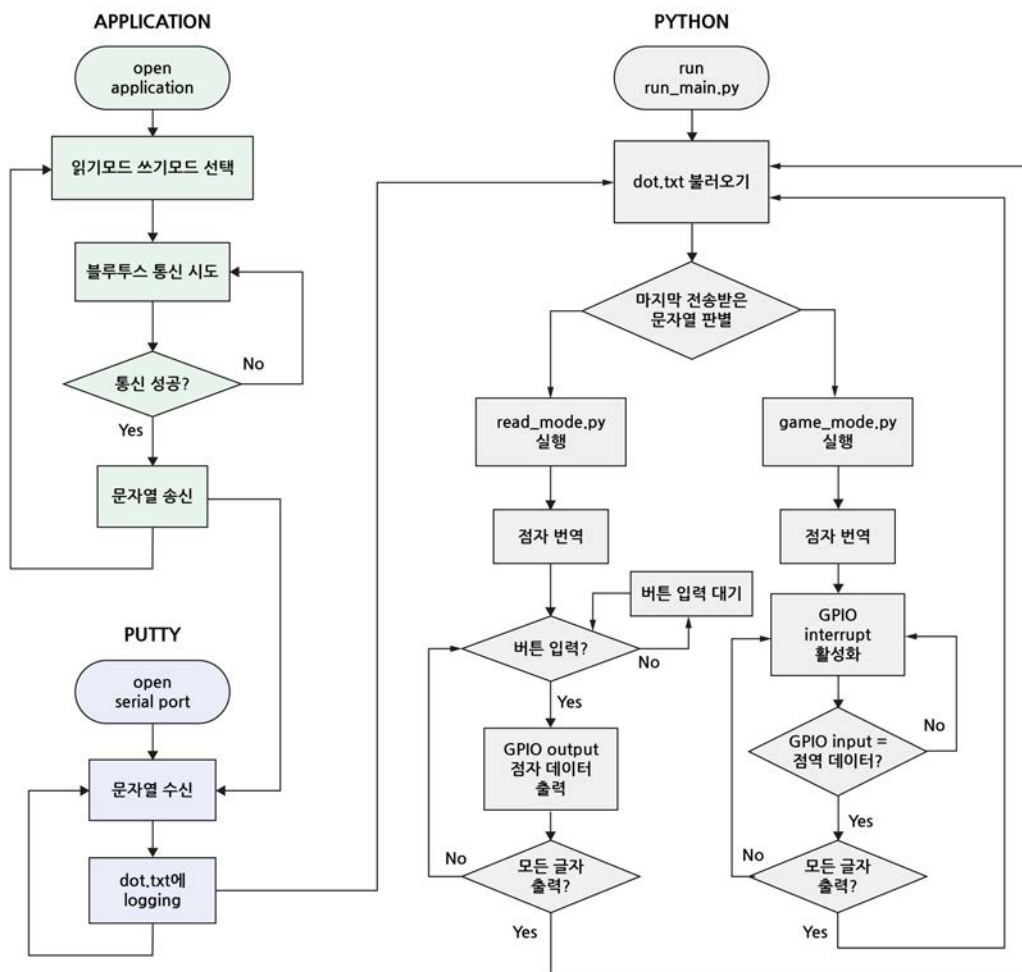
원소에 대해서 나눠 줄 필요가 있다. C 언어로는 다음 작업을 수행하는 코드를 작성하기 어려움이 있기 때문에 Python 을 사용했고, 이를 내장하는 Raspberry Pi 를 이용한다.

PuTTY : Raspberry Pi 에 PuTTY 라는 프로그램이 내장되어 있다. 일반적으로 SSH 통신을 할 때 많이 사용하지만, 우리는 PuTTY 로 Serial 통신을 하였다. 애플리케이션에서 보내준 단어를 받고, 받은 값을 실시간으로 Raspberry Pi txt 파일에 logging(기록) 해주도록 설정하였다.

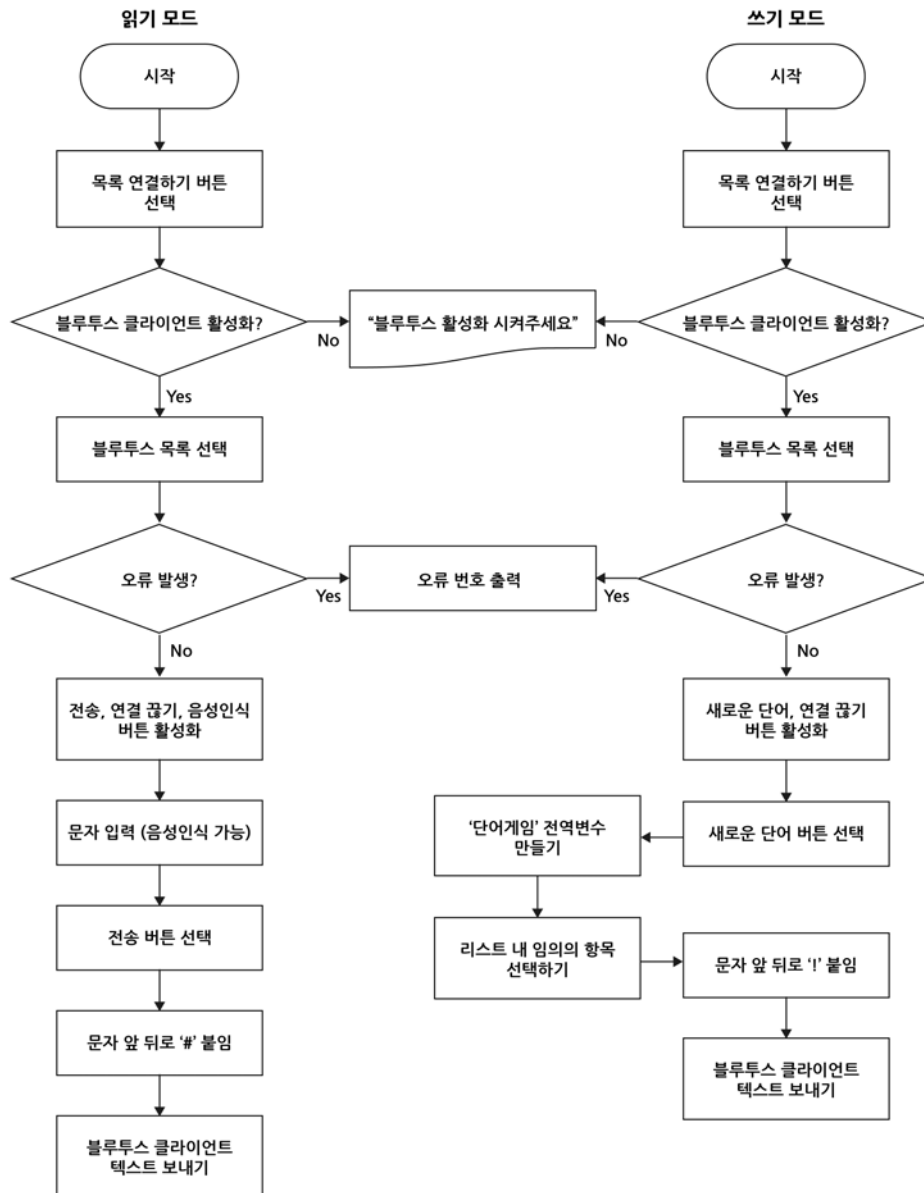
Python : Raspberry Pi 에 내장된 Python 으로 txt 파일에 저장된 문자열을 읽어오는 코드를 성하였다. 이를 초성, 중성, 종성으로 나누고 점자 데이터로 번역한다. 읽기 연습과 쓰기 연습으로 구분한 후 BRED 의 행동을 정해준다. Raspberry Pi 의 GPIO pin 제어도 전부 Python 코드로 작성하였다.

○ Software 설계도

1. 전체 시스템 Flow Chart

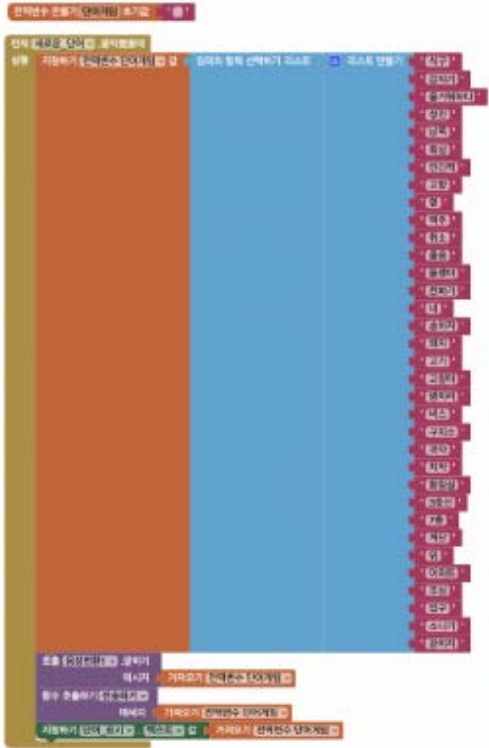


2. Application Flow Chart



3. App Inventor Programming

Code Block	Description
	읽기 연습에서 전송하려는 메시지(단어) 앞, 뒤로 '#'을 붙여주어, Raspberry Pi에서read_mode.py를 실행해야 한다는 것을 알려준다.
	쓰기 연습에서 전송하려는 메시지(단어) 앞, 뒤로 '!'을 붙여주어, Raspberry Pi에서game_mode.py를 실행해야 한다는 것을 알려준다.

	<p>쓰기 모드에서는 점자로 자주 쓰이는 단어들을 모아둔 리스트가 저장되어 있다.</p> <p>이 가운데 임의의 항목을 선택하여 '단어 게임'이라는 변수 안에 넣는다. 애플리케이션 안에서 '새로운 단어' 버튼을 선택하였을 때, 변수 내에 있는 임의의 단어를 음성으로 출력하고 Raspberry Pi와 통신한다.</p> <p>쓰기 연습을 게임과 같이 진행하게 된다. 추 후에 애플리케이션 내에서 단어를 직접 저장할 수 있게 하여, 나만의 단어장을 만들고 게임처럼 쓰기를 연습할 수 있는 방법을 고안하려고 한다.</p>
---	--

○ Software 기능

1. Application

- **튜토리얼** : 처음 애플리케이션 사용자를 위해 작동 방식을 알려준다.
- **블루투스 송신** : 애플리케이션에서 Raspberry Pi 와 블루투스 연결을 시도한다. Wi-fi 통신보다 장소의 구애가 적고, 전력소모가 낮아서 오래 사용할 수 있다.
- **글자 입력 / 음성 인식** : 핸드폰 키보드를 통해서 원하는 단어를 입력할 수 있다. 시야가 안 좋은 사람들은 음성 인식 기능을 통해 단어를 입력할 수 있다.
- **새로운 단어** : 쓰기 연습에서 단어장 내에 있는 임의의 단어를 출력해 사용자가 재미있게 점 자를 학습할 수 있게 했다.

2. Device

- **자동실행** : autostart 를 이용하여 전원이 켜지면 자동으로 원하는 프로그램이 실행될 수 있게 하였다. 추후에 수정/보완할 수 있기 때문에 rc.local 파일을 직접 수정하지는 않았다.
- **블루투스 수신** : 스마트폰과 블루투스로 연결한다.
- **인터럽트** : 쓰기 연습에서 스위치를 누르는 순간 interrupt 되어 그 데이터를 기록한다.
- **힌트** : 쓰기 연습에서 메인스위치를 1 초이상 누르면 해당 단어의 점자가 올라오면서 힌트가 1 초동안 주어진다.
- **종료** : 전원공급을 강제로 제거하면 Raspberry Pi 고장의 원인이 될 수 있기 때문에, 메인 스위치를 10 초이상 누르면 전원이 꺼지게 하였다.


○ 프로그램 사용법 (Interface)

1. Application UI

메인 화면	읽기 연습	읽기 연습
		
애플리케이션을 실행 시 볼 수 있는 메인 화면이다. 버튼 그림을 누르면 간략한 튜토리얼이 제공된다.	연결이 안되어 있으면 '연결하기' 버튼만 활성화 되어있다. 블루투스 목록에서 BRED에 해당하는 블루투스를 선택한다.	키보드 또는 음성인식을 통해 단어를 입력/수정할 수 있다. 블루투스 연결이 성공적으로 되었다면 전송 버튼이 활성화된다.

메인 화면	쓰기 연습	쓰기 연습
		
뒤로 가기를 누르면 메인 화면으로 돌아갈 수 있다. 버튼 그림을 누르면 튜토리얼이 제공된다.	읽기 연습 때와 마찬가지로 연결하기 버튼을 눌러 BRED와 연결한다.	'새로운 단어'를 선택하면 단어장에 저장된 임의의 단어가 음성과 함께 출력된다.

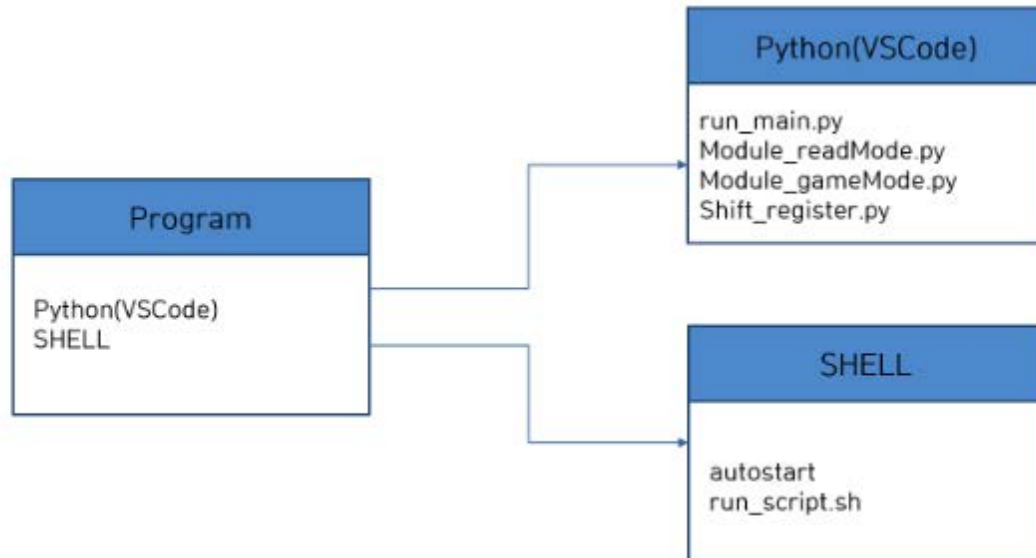
○ 개발환경

 Python	단어를 원소로 쉽게 바꿀 수 있고, 많은 데이터를 저장/처리 할 수 있는 객체 지향 프로그래밍 언어 Python을 사용하였다. 다량의 점자 데이터를 NumPy로 연산하여 효율적인 작업을 추구한다.
 Raspbian	Raspberry Pi에 최적화된 데미안 계열의 운영체제인 Raspbian을 사용하였다.
 Thonny	Serial 통신을 안정적으로 수행하기 위해서 Raspbian에 내장된 Thonny 통신 프로그램을 이용하였다.
 VSCode	컴파일이 필요하지 않은 Python이기 때문에 가볍고 접근성이 좋은 VSCode를 이용하여 코드를 개발하였다.
 MIT AppInventor	원하는 애플리케이션의 기능과 레이아웃을 간편하고 깔끔하게 구성하고 기기와 통신하는 프로그래밍을 하기 위해 MIT AppInventor를 사용하였다.
 xRDP / VNC	Raspberry Pi 중 특히 Zero 모델은 저전력 프로세서이기 때문에 매우 느린 편이다. 따라서 내부에서 직접 코드를 짜기보다는 xRDP를 통해 원격제어하여 컴퓨터에서 작성한 코드를 옮겼다.
 Bash	PuTTY와 Python을 구동하기 위해서는 셸이 필요한데 이때 bash를 사용해 셸 스크립트를 구성하였다.

□ 개발 프로그램 설명

○ 파일 구성

1. File Map



1) Python

- run_main.py : 애플리케이션으로부터 데이터를 전달받고 다른 모듈 파일로 이동할 수 있는 메인 파일이다.
- Module_readMode.py : run_main.py로부터 문장을 전달받아 읽기 연습을 실행 시키는 파일. 점역 프로그램이 포함되어 있다.
- Module_gameMode.py : run_main.py로부터 문장을 전달받아 쓰기 연습을 실행시키는 파일. 마찬가지로 쓰기 연습 전용 점역 프로그램이 포함되어 있다.
- Shift_register.py : shift register에 관한 데이터를 관리하는 함수가 내장된 파일. 읽기/쓰기 연습 시 이 파일을 모듈로 불러와 점자를 출력시킨다.

2) Shell

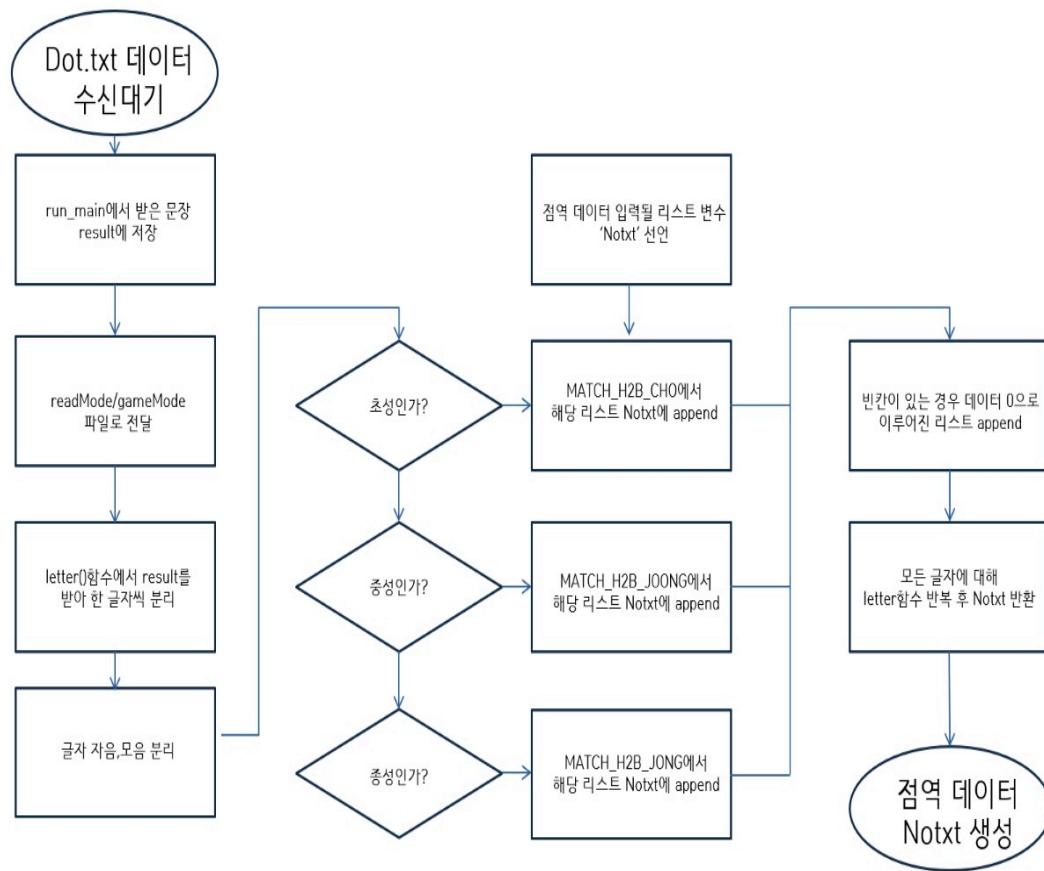
- Autostart : Raspberry 부팅 시 run_script.sh를 자동실행 해주는 명령어.
- run_script.sh : Putty와 run_main.py를 실행 시켜주는 Shell script.

○ 주요 함수 기능

- 1) find_index() : dot.txt 의 내용에서 특정부호의 모든 index 들을 역순으로 정렬하여 index_list 에 반환한다.
- 2) letter() : run_main 에서 수신한 문장의 글자 하나를 점자 데이터로 변환한다. 한글의 경우 초성/중성/종성을 구분한다.
- 3) text() : letter()를 통해 점역한 데이터들을 전달받아 readMode/gameMode 를 실행한다.
- 4) all_off() : 점자 출력을 초기화한다.
- 5) button_call() : 메인 스위치 감지 시간을 판별하여 각기 다른 행동을 취하게 한다.
- 6) hint() : gameMode 실행 도중 힌트를 점자로 출력시켜주는 역할을 한다.
- 7) vibe() : 메인 스위치 조작에 따라 점자 기기에 진동을 주는 역할을 한다.

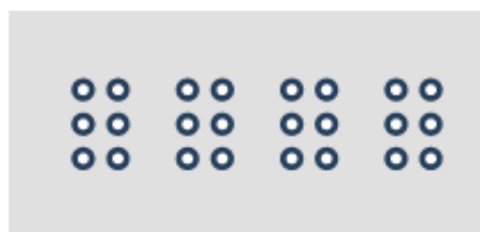
주요 변수 이름	역할
symbol	readMode/gameMode 구분하는 객체 선언
index	애플리케이션에서 수신한 문장 위치 저장
index_list	Index 집합
input_filr	txt 파일 저장
symbol1	readMode 에 대한 index 저장
symbol2	gameMode 에 대한 index 저장
input_text	txt 파일에서 불러온 문장 저장
result	Input_text 문장 string 형식으로 저장
MATCH_H2B_CHO	초성 점역 데이터 보관
MATCH_H2B_JOONG	중성 점역 데이터 보관
MATCH_H2B_JONG	종성 점역 데이터 보관
MATCH_H2B_ALPHABET	알파벳 및 특수기호 점역 데이터 보관
zerosix	[0] x 6 공란 데이터
zerotwv	[0] x 12 공란 데이터
zeroeigth	[0] x 18 공란 데이터
zerotwofour	[0] x 24 공란 데이터, all_off()에 사용
index1	초성 점역 데이터 개수
index2	중성 점역 데이터 개수
index3	종성 점역 데이터 개수
Notxt	문장 점역 데이터
hangul	글자 저장 변수
ledResult	Notxt 를 [1 x 24 x n] 행렬로 변환한 값 저장 readMode 전용
braResult	Notxt 를 [1 x 6 x n] 행렬로 변환한 값 저장 gameMode 전용
row	글자 개수 index
a	readMode/gameMode 진행상황 판단 index, row 와 비교
start	메인 스위치 ON 시점 저장 변수
end	메인 스위치 OFF 시점 저장 변수
call, shutdown	Raspberry pi 종료시 사용

○ 주요 함수의 흐름도



1. 점역 프로그램 Flowchart

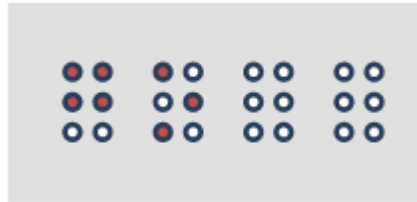
- 1) App 에서 수신한 문장을 run_main 에서 result 변수에 저장한다.
- 2) 읽기 연습에서 수신한 경우 readMode, 쓰기 연습일 경우 gameMode 라는 모듈 파일로 result 를 전달한다.
- 3) letter() 함수를 통해 문장-글자-자음/모음 순으로 분해한다.
- 4) 분해한 원소의 초성/중성/종성을 판단하고 해당 점역 데이터를 불러온다.
- 5) 'Notxt'라는 list 변수에 순차적으로 점역 데이터를 덧붙인다.
- 6) 3 행 2 열 팀이 제작한 'BRED'는 아래 그림과 같이 6 점자 4 세트이다.



한 글자씩 점자를 출력할 때, 사용하지 않는 세트가 발생하는 경우 공란 데이터로 Notxt 의 공란을 채운다.

ex) '이' : 'ㅇ'([1,1,0,1,1,0]) + 'ㅣ'([1,0,1,0,1,0]) + '공란'([0,0,0,0,0,0]) x 2

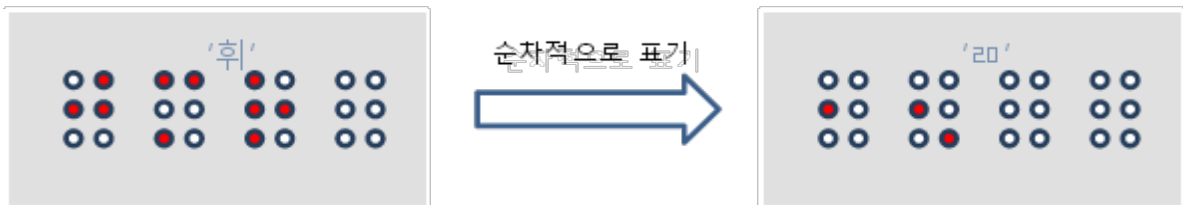
->[1x24] 1 차원 배열



'흠' : 'ㅎ'([0,1,0,1,1,0]) + 'ㅌ'([1,0,1,1,0,0,1,1,0,1,0]) + '공란' ([0,0,0,0,0,0])

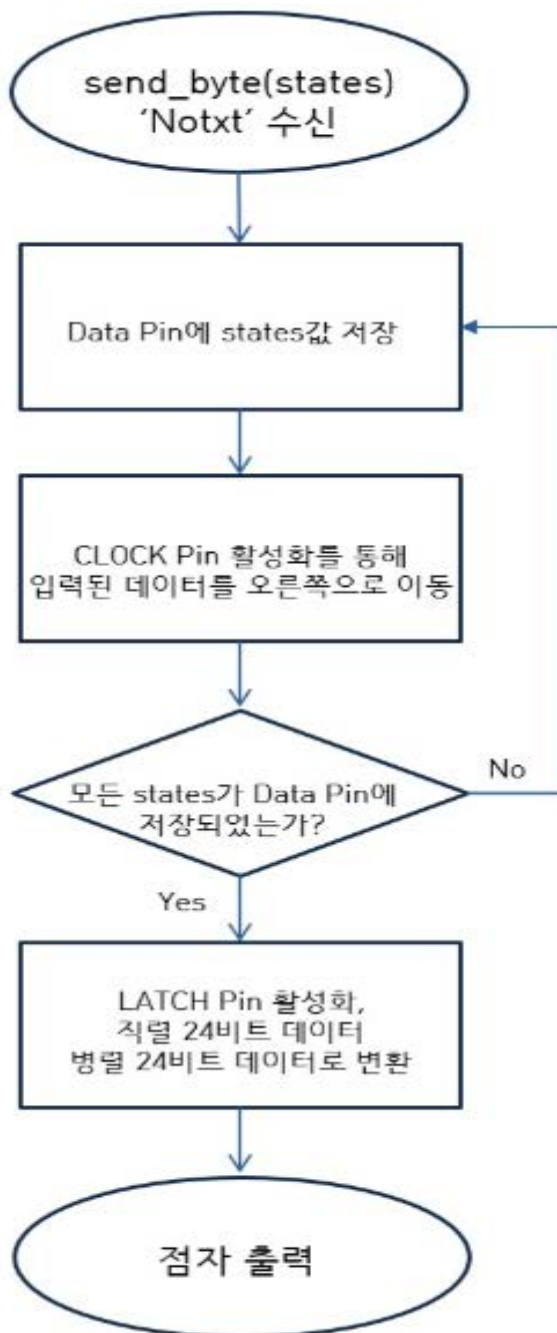
'ㄹ'([0,1,0,0,0,0,0,1,0,0,0,1]) + '공란'([0,0,0,0,0,0]) x 2

4 칸을 초과하는 경우 종성만 따로 표기 ->[1 x 24 x 2] 2 차원 배열



7) 모든 글자에 대한 list 를 덧붙여 최종적으로 [1 x 24 x n] 2 차원 배열 Notxt 를 반환한다.

2. shift register.py_send_byte()함수 Flowchart



점자 출력기는 6점자 4세트로 구성되어 있어 24(6x4)개의 출력 데이터 핀이 요구된다.

부족한 출력 핀은 74HC595(이하 shift register)라는 레지스터를 이용하여 핀을 확장시킨다.

DATA, CLOCK, LATCH 출력 Pin 3개를 이용하여 24개의 출력을 제어할 수 있다는 이점이 있다.

Shift register 1개로 8개의 출력을 제어할 수 있으므로 3개의 shift register를 서로 연결하여 24개의 점자를 한번에 제어할 수 있도록 한다.

1. 읽기 연습/쓰기 연습에서 전송한 'Notxt' 수신

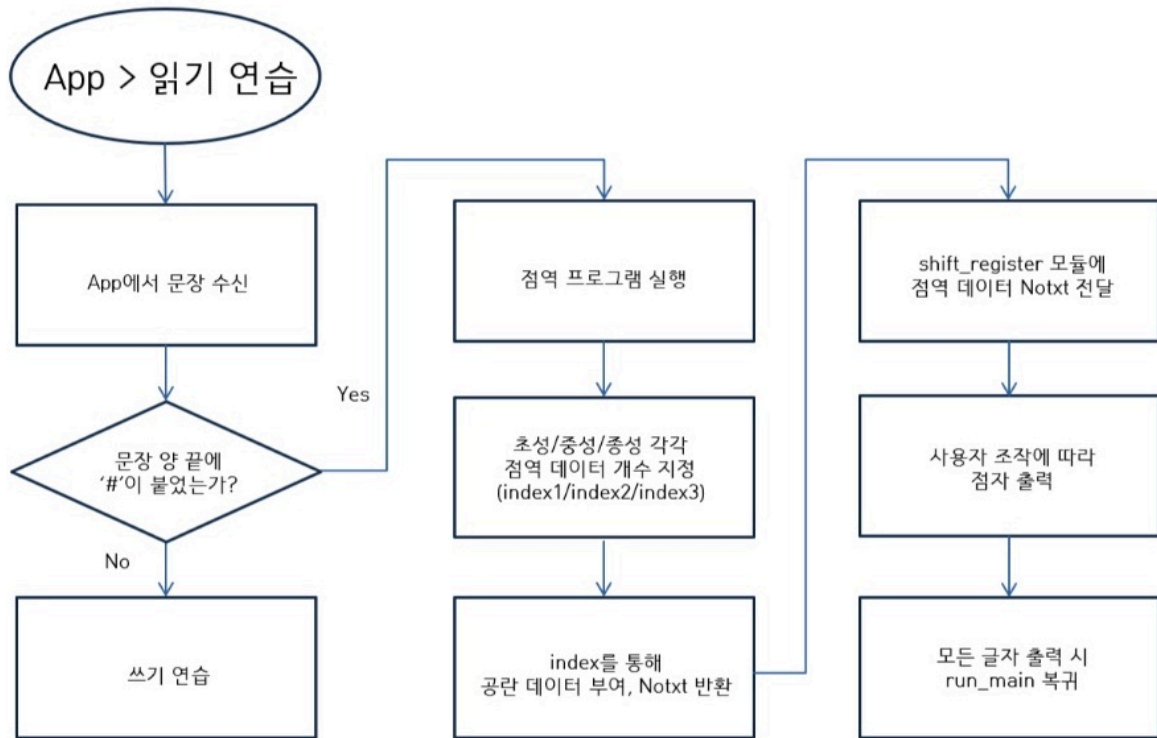
2. Q에 순차적으로 Notxt 원소들 저장

3. CLOCK Pin (이하 CLK)에 상승 엣지가 발생하면 $Q_0 > Q_1$, $Q_1 > Q_2$... 순차적으로 Q값 이동

4. $Q_0 \sim Q_{23}$ 에 Notxt가 저장 후 값을 유지

5. LATCH Pin에 상승 엣지 발생 시 직렬 비트 데이터를 병렬 비트 데이터로 변환하여 OUTPUT으로 출력

3. 읽기 연습 Flowchart



1) App 에서 수신한 문장을 판별한다.

문장 양 끝에 '#'이 붙은 경우 Module_readMode 로,
'!'이 붙은 경우 Module_gameMode 로 문장을 전달한다.

2) 점역 프로그램을 실행하여 'Notxt'를 생성한다.

3) 글자 초성,중성,종성 각각의 점역 데이터 개수를 지정한다.

ex) 'ㄱ' : [0,0,0,1,0,0] -> index1 = 6

'내' : [1,1,1,0,0,1,1,1,1,0,1,0] -> index2 = 12

4) index 를 통해 'Notxt'의 한 글자에 대한 점역 데이터 크기가 24 가 되도록 공란 데이터를 append 한다. 공란 데이터는 앞으로 이하와 같이 표기한다.

-zerosix = [0,0,0,0,0,0]

-zerotwv = [0,0,0,0,0,0] x 2

-zeroeigth = [0,0,0,0,0,0] x 3

Ex) '괘' 글자 분해 > index1 = 6, index2 = 12, index3 = 0

크기가 24 가 되려면 [0]이 6 개 필요, Notxt 에 zerosix append

5) 공란 데이터 append 알고리즘

(1) index1 + index2 + index3 = 12 : Notxt + zerotwv

(2) index1 + index2 + index3 = 18 : Notxt + zerosix

(3) index1 = 6, index2 = 12, index3= 12 : 중성 점역 데이터 뒤에 zerosix append 후 종성 뒤에 zerotwv append

(4) index1 = 12, index2 = 6, index3= 12 : 중성 점역 데이터 뒤에 zerotwv append 후 중성 뒤에 zerosix append

(5) index1 = 12, index2 = 12, index3= 6 : Notxt + zeroeighth

(6) index1 = 12, index2 = 12, index3= 12 : Notxt + zerotwv

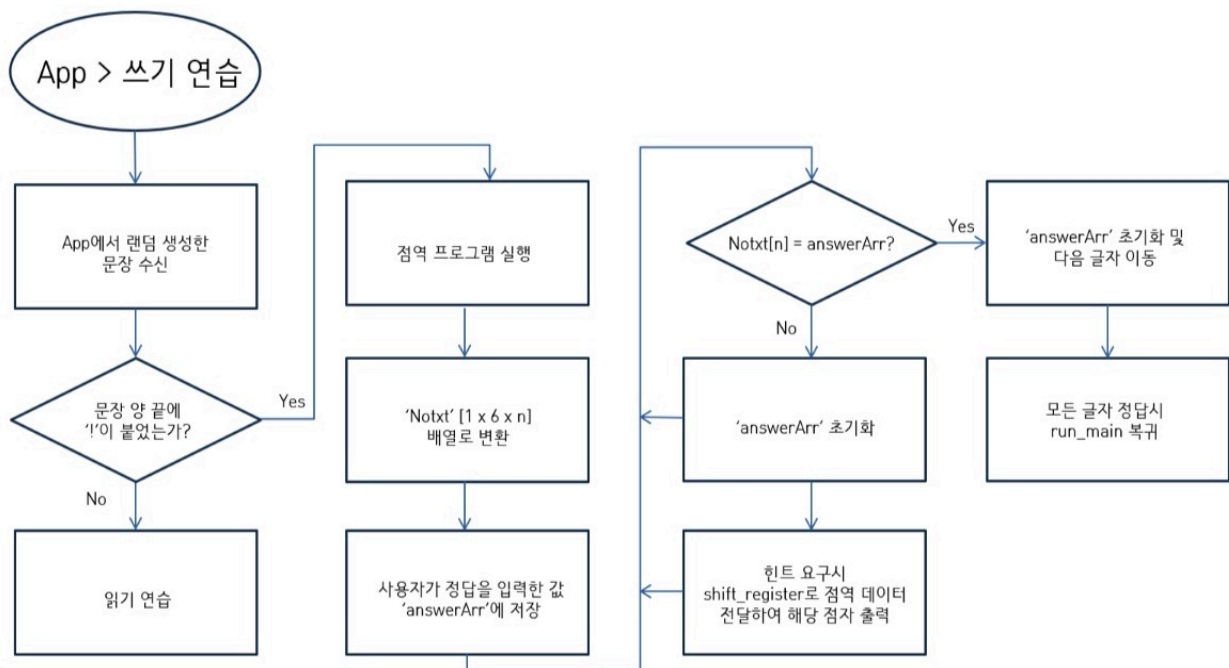
※ 한 글자의 index 합이 24 가 넘는 경우 초성과 중성만 먼저 출력하고 중성만 뒤에 따로 출력되도록 한다.

ex) '괘'은 5-(3) 경우에 해당 : '괘'+zeroWwsix+'리'+zerotwv

6) 완성된 점역 데이터를 shift_register 모듈에 전달하여 점자를 출력한다.

7) 모든 글자 출력 시 다시 다른 문장을 수신할 수 있도록 run_main 파일로 복귀한다.

4. 쓰기 연습 Flowchart



1) 쓰기 연습은 App 에서 랜덤으로 생성하는 문장을 사용자가 점자 모양 스위치를 눌러 맞추는 게임 형식이다.

2) App 에서 랜덤으로 생성한 문장을 수신한다.

3) 단어에 대한 점역 데이터 'Notxt'를 생성한다.

4) 사용자가 스위치로 입력한 값을 인터럽트를 사용하여 'answerArr'에 저장한다.

n 번 스위치 입력 시 my_callback[n] 인터럽트 함수 실행, answerArr[n]에 '1'을 저장한다.

Ex) 1,3 번 스위치 입력 시 answerArr = [1,0,1,0,0,0]

5) 메인 스위치의 입력 시간에 따라 다른 행동을 수행하도록 한다.

time 함수를 이용해 스위치가 눌린 시점의 시간을 'start' 변수에 저장, 떼어진 시점의 시간을 'end' 변수에 저장한다.

(end-start)를 계산하여 스위치가 눌린 시간을 도출한다.

6) 스위치가 눌린 시간 판별

- (1) 1 초 미만 : 해당 글자의 점역 데이터와 입력값인 'answerArr'를 비교하여 정답인지 판별
- (2) 1 초 이상 8 초 미만 : 해당 글자 힌트 출력
- (3) 8 초 이상 : Raspberry pi 종료

7) 정답이면 짧은 진동 2 번 출력(0.2s)과 다음 정답을 입력할 수 있도록 answerArr 초기화

8) 오답이면 긴 진동 1 번 출력(0.8s)과 다시 입력할 수 있도록 answerArr 초기화.

9) 메인 스위치 1 초 이상 입력 시 힌트가 출력된다.

초성/중성/종성에 따라 출력되는 위치가 다를 수 있도록 공란 데이터를 append 한 Notxt[n]를 shift_register 파일로 전송하여 해당 점자를 3 초 동안 출력시키고 all_off() 함수로 다시 내린다.

10) 모든 글자가 정답일 시 Module_gameMode 이 종료되고 run_main 파일로 복귀한다.

○ 기술적 차별성

- 1) 회전 버튼에 기존 1 차원 운동으로 힘을 전달하는 철사 스프링을 보완하여 회전 운동으로 힘을 전달할 수 있는 회전 스프링을 고안하였다. 이를 통해 메인 스위치는 원래 자리로 돌아 올 수 있으면서 공간을 절약하고 내구성을 유지할 수 있다.
- 2) 기존 솔레노이드는 실제 크기의 점자를 구현하기 어렵다. 'BRED'는 소형 코일, 전자석, 레진 3D 프린터로 제작한 소형 부품을 활용해 실제 점자 크기와 유사한 dot 구동부를 구현해낼 수 있었다.
- 3) 통신과 프로그램을 순차적으로 진행하지 않고 병행함으로써 비효율적인 통신 프로세스 (통신프로그램 on/off 에 요구되는 연산처리 능력)를 개선하였다.

□ 개발 중 발생한 장애요인과 해결방안

○ 코일 개발

코일을 감아 원하는 크기의 힘과 낮은 전류에서 작동할 수 있도록 설계하는 것을 목표로 설계하고 있었지만, 크기의 제한이 있어서 코일을 무수히 많이 감을 수 없었고, 낮은 전류에서 코일이 낼 수 있는 힘은 한계가 있었다. 그래서 서칭을 통해 한 코일 설계를 참고하였다. 무게로 인해 공중으로 띄지 못했던 점을 레진 프린터를 이용해 작고 가벼운 부품을 만들 수 있게 되어 낮은 전류에서도 점자 표현을 가능할 수 있도록 하였다.

○ 블루투스 설정

초기의 목표는 Raspberry Pi W에 내장된 블루투스를 사용하는 것이었다. 하지만 이 제품은 일반 블루투스가 아닌 저전력 블루투스 BLE를 사용하였고, 이는 사용성, 사용 난이도면에서 어려웠다. 이 후 Arduino에 hc-06을 연결하여 이중통신을 시도하였지만 통신속도와 안전성이 떨어져 불만족스러웠다. 원격제어/SSH에 주로 사용되는 PuTTY를 Serial 통신 하는데 사용해 보았더니, 추가적인 Arduino나 유니코드 변환 필요 없이 만족스러운 결과를 얻을 수 있었다.

○ 프로그램 실행 알고리즘

통신 프로그램 PuTTY를 실행하고 단어를 입력하면, '원할 때'마다 Python 코드를 실행해주는 알고리즘을 구현하고자 하였다. 문제점은 쉽게 통신할 수 있도록 PuTTY로는 '원할 때'를 정해주기 어렵다는 데에 문제가 있었다. 해결 방법으로 두개를 동시에 실행하다가 버튼이 눌렸을 때 PuTTY를 끊는 방법을 생각해보자 하지만 이 코드는 연산 처리 기능이 많아져 작업 속도가 현저히 느려 지고, 시간이 조금 어긋나면 프로그램이 정상적으로 실행되지 않았다.

통신과 처리를 꼭 구분 지어야 하는가에 의문이 들었고, PuTTY를 종료하지 않고 Python코드 내에서 원하는 단어를 구분하는 코드를 구성해 보았다. 처리 시간이 훨씬 줄어들었고 프로그램이 안정화될 수 있었다.

○ Raspberry pi 간섭

초기 'BRED'의 프로토 타입을 제작할 당시 Raspberry pi는 외부로부터의 간섭이 없는 상태에서 테스트를 진행했었다. 문제가 없는 것을 확인 후 모든 설계 부품과 조립을 했을 때 PuTTY에서 수신한 문자가 깨지는 현상이 발생하였고 진행이 불가능 했었다. UTF-8 ENCODING와 Bluetooth 송수신 에러를 의심하여 조치를 취했지만, 조립만 하면 같은 문제가 발생하는 것을 확인하고 Raspberry pi에 물리적/노이즈 간섭을 최소화해보기로 하였다. Pi에 케이스를 장착하고 타 부품과의 간격을 조정한 결과 같은 에러의 발생을 막을 수 있었고, 회로적으로도 안정성을 확보할 수 있었다.

□ 개발결과물의 차별성

1) 경제적 부담이 적다.

탭틸로의 가격은 150만원으로 개인이 소지하기엔 경제적인 부담이 크다.

BRED의 제작비용은 다음과 같다.

제품	개수	가격
Raspberry pi zero	1EA	30,800
HC-06	1EA	3,600
Register	48EA	7,500
Transistor	24EA	1,500
Diode	24EA	1,800
Tact Switch	6EA	1,800
구리선(0.1mm)	40g	8,000
초소형 진동모터	1EA	1,100
블루베리 레진	24EA	15,000
Shift-Register	3EA	600
		총 비용 : 71,600

이는 기존 제품보다 훨씬 더 낮은 가격임으로 비용적인 면에서 부담이 적다.

2) 휴대성의 용이함

탭틸로는 크기가 커서 전용 가방을 사용해야 가지고 다닐 수 있다. 이는 교육자가 교육을 하기 위해서는 이 큰 물건을 들고 다녀야 하는 불편함이 있다.

탭틸로의 크기(51x22x13cm)는 주머니나 일반 가방에 넣기 힘든 크기이고 그 크기에 비례해 무게 또한 상당히 나가 상대적으로 힘이 약한 어린아이나 노인들은 휴대하기에 부담이 크다. 하지만 이번 제품(18x7.7x8.6cm)은 소형화 제품임으로 일반 가방에 넣고 다니기 편하고 또한 무게 또한 가볍다.

3) 실제 점자크기의 교육기기

점자교육기중 가장 많이 상용화 된 탭틸로는 실제 점자보다 큰 크기로 구성되어 있다.

이는 처음 배울 시 쉽게 점자를 배울 수 있지만 점자에 익숙해지면 실제 크기의 점자를 바탕으로 한 교육이 필요하다. BRED는 이러한 이유에 바탕으로 실제 점자크기를 구현하여 사용자가 실생활에서의 점자사용을 원활하게 사용할 수 있도록 한다.

□ 개발 일정

내용	2021年											
	6月			7月			8月			9月		
주제선정 및 자료조사												
H/W 제작(한성민, 김근태)												
점자구현 방식 테스트												
H/W 설계												
H/W 제작 및 테스트												
회로 설계 및 테스트												
S/W 제작(홍태주, 이진우)												
MCU 및 언어 선정												
APP제작 및 테스트												
APP-MCU 통신 구현												
한글-점자 변환												
읽기, 쓰기모드 제작												
S/W 통합 및 수정												
최종 작업 및 조립(한성민, 홍태주, 이진우, 김근태)												
H/W 조립												
H/W, S/W 통합												
제품 수정 및 최종 테스트												

※H/W 제작 시 점자구현 방식의 변경으로 설계 및 제작기한이 연장됨.

□ 팀 업무 분장

No	성명	구분	업무 분담 파트
1	홍태주	팀장	-Python 프로그래밍 담당 -점역 프로그램 알고리즘 제작 -readMode / gameMode 알고리즘 제작 -점자 출력 제어 -dot 내부 부품 제작 및 실험
2	김근태	팀원	-BRED 외관 설계 -내부 디자인 -BRED 부품 3D 프린팅 -점자 구동부 선정을 위한 실험 -제품 조립 및 보조
3	한성민	팀원	-회로 설계 및 제작, 보완 -점자 구동부 선정을 위한 실험 -소자 납땜 -3D 프린트 -제품 조립 및 보조
4	이진우	팀원	-BRED 메인 스프링/스위치 설계 및 제작 -3D 프린트 -App Inventor 알고리즘 제작/프로그래밍 -PuTTY 이용한 블루투스 통신 -Raspberry Pi 기본 세팅, 모듈/프로그램 설치, 문제점 해결 -Python run_main.py 알고리즘 제작/프로그래밍