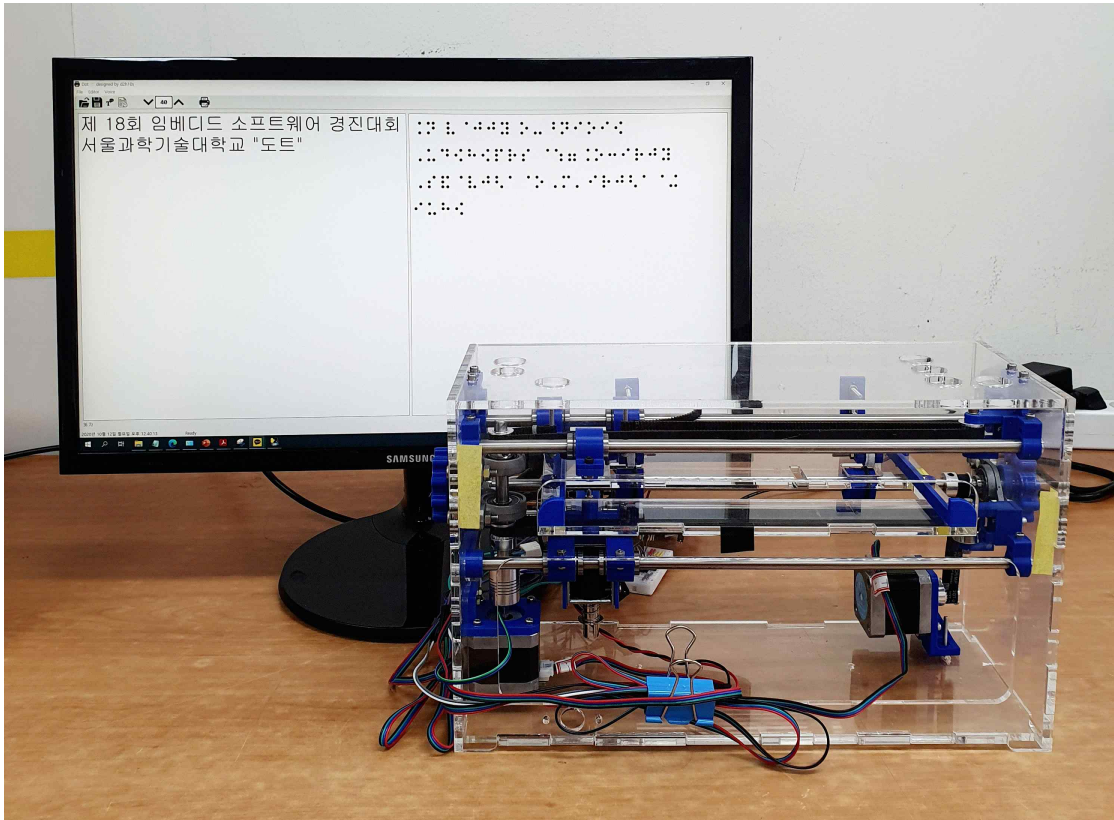


제18회 임베디드SW경진대회 개발완료보고서

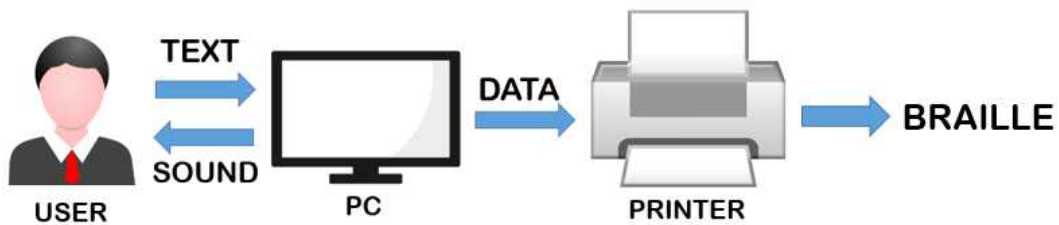
[자유공모]

□ 개발 요약

팀 명	도트
	
작품명	시각장애인을 위한 보급형 텍스트 자동 번역 프린터
작품설명 (요약)	PC를 이용하여 짧은 글 또는 텍스트를 음성 또는 점자로 출력하여 저장할 수 있는 프린터로, 처음 사용하는 사용자도 손쉽게 텍스트를 원하는 종류의 데이터로 번역한 후 저장하거나 출력할 수 있다.
소스코드	https://github.com/d2h10s/pybraille (PC) https://github.com/d2h10s/braille_mbed (MBed)
시연동영상	https://www.youtube.com/watch?v=mGELOXtkAHk&feature=youtu.be

□ 개발 개요

○ 개발 작품 개요



- 이 작품은 시각장애인들이 원하는 문서를 점역하고자 할 때 빠른 시간, 저렴한 비용으로 자료를 제공받을 수 있도록 한다. 점역을 원하는 자료를 PC에 입력하면 이를 음성파일 또는 점역한 결과를 프린터를 통해서 점자 문서로 출력하여 시각장애인들에게 유용한 자료로 변환한다.

- 해당 작품을 기획한 배경은 비대면 수업이 진행되면서 시각장애인들은 온라인으로 올라온 강의자료를 어떻게 활용할 수 있을지 궁금증을 느껴 시작했다. 간단한 강의자료라도 이것을 읽을 수 없고 심지어 점역하려는 과정은 점역 전문가의 도움을 받아야 하고 매우 복잡하다. 더불어 대학 도서관에 시각장애인을 위한 점자 자료가 한 건도 없다는 걸 알게 되면서 시각장애인들이 큰 정보 격차를 겪고 있다고 느꼈고 이를 해소하고자 하였다.

○ 개발 목표

- 1) 시각장애인들이 원하는 자료를 빠르고 간편한 과정으로 받을 수 있도록 한다.
- 2) 저렴한 제작비용으로 학교 도서관, 개인 수준까지 공급할 수 있도록 한다.
- 3) 점역 과정을 자동화하여 비전문가도 손쉽게 다룰 수 있도록 한다.

○ 개발 작품의 필요성

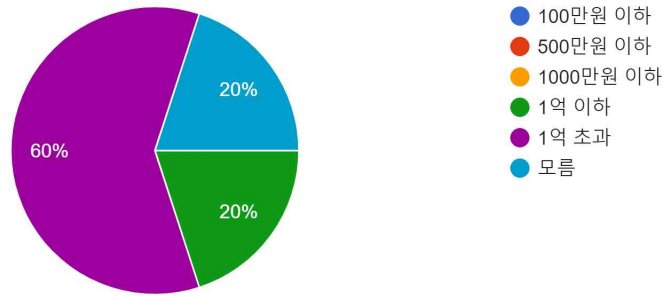
-실제사례 : 일반적으로 수능특강(학습자료)은 매년 1월 달에 발행한다. 그러나 시각장애를 가진 학생 A군은 수능으로부터 4개월밖에 남은 않은 7월이 되어서야 점자 번역된 학습 자료를 받을 수 있었다.

사례와 같이 시각장애인, 특히 학생은 학습 자료를 시기적으로 제때에 받지를 못한다. 왜 그런가? 시각장애인을 위한 자료의 공급 여건이 늘 부족하기 때문이다.

점자프린터 인쇄 과정은 전문기관에서 의뢰를 받아 '목자 입력 - 교정 및 점역 - 출력 - 절지 - 제본 - 표지' 작업 순으로 진행된다. 이 과정은 책 한권을 번역하는데 길게는 3달까지도 걸리는 과정이다. 점자번역을 의뢰하고 무료로 도서자료를 열람할 수 있는 전국의 점자도서관수는 약 30여 곳. 대체로 대도시 위주로 위치해 있다. 이렇게 한정된 공급의 이유는 기관에서 사용하는 점자프린터가 약 1억 원에 달하는 매우 고가의 인쇄기계를 사용하고 있기 때문에 작은 도서관이나 복지관에서는 설비를 갖추 여건이 되지 않는다. 하단의 도표는 일반 도서관 및 점자 도서관 15곳에 설문 조사를 요청하여 응답받은 8곳의 도서관 중 프린터를 보유하고 있는 5곳에서 온 답변이다.



프린터 1대를 구매하는데 평균 어느 정도의 비용이 소모되었나요?

응답 5개



점자프린터 구매비용 그래프

위의 자료를 보면 과반수 이상의 기관들이 1억 이상의 비용이 들었음을 알 수 있고 점자프린터가 얼마나 고가의 장비인지 체감할 수 있었다. 이런 고가의 장비가 필요한 점역 외에도 기관의 사회봉사자들은 vrailleur라는 간이 점자 라벨기로 아이들을 위한 점자 스티커를 만들어 붙여주는 작업을 자주 한다. 이 작업은 도구의 구멍 속에 핀들을 일일이 넣고 스티커를 덮어 물리적 압력을 주면서 점자를 만들어 낸다. 관계자들은 핀셋으로 핀을 하나하나 집어 작은 구멍 속에 넣는 일은 많은 인내심과 집중력이 필요한 일이라 꽤나 힘들다고 말한다. 게다가 이는 점자 체계에 대한 정보를 알아야 할 수 있는 작업으로 전문성이 요구된다.

 <p>Braillo 400 SR</p>	
<p>점자 도서관에서 사용하는 Braillo SR모델 가격대 : 1억 3천만원</p> <ul style="list-style-type: none"> - 점역 시스템에 입력하면 점자를 찍어내고 낱장으로 뜯어져 나오는 기능 수행 	<p>Vrailer : 간이 점자 라벨기 가격대 : 50 \$ (약 6만원)</p> <ul style="list-style-type: none"> - 가격은 저렴하나 한정된 글자 수 - 매 작업마다 손이 많이 감 - 점자체계를 모르면 사용하기 어렵

그렇다면 '좀 더 저렴하면서 자동화적으로 점역 및 점자프린팅 작업을 할 수 없을까?' '점자에 대해 공부하지 않은 사람들도 점역작업을 손쉽게 할 수 없을까?' 하는 의문을 품게 되었다. 현존하는 고가의 프린터보다 품질은 떨어지더라도 저가 상업용 프린터가 존재한다면 작은 도서관, 관공서, 복지관에서 시각 장애인을 위한 콘텐츠나 자료, 심지어 조그마한 점자 스티커에 이르기까지 다양한 용도를 더 쉽고 편하게 제작할 수 있을 것이다.

□ 개발 환경 설명

○ Hardware 구성

1) 전체 Hardware 구성

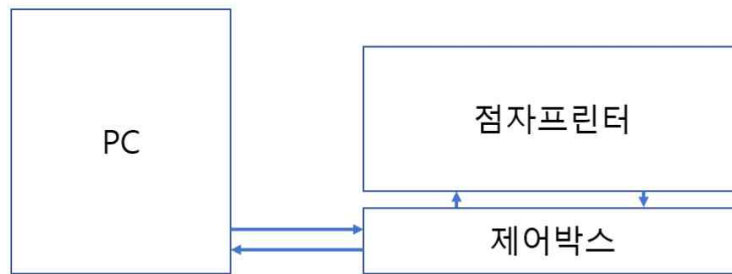


그림 6 - 전체 하드웨어 구성 (요약도)

하드웨어는 PC, 점자프린터, 제어박스로 구성하였다. PC는 사용자가 점자프린터를 작동할 명령어를 제어박스로 전송하고 점자프린터를 작동시킨다.

2) 점자프린터

점자프린터는 크게 점을 찍는 타점부와 종이를 이송하는 이송부로 나뉜다. 타점부는 보드에서 수신한 데이터를 기반으로 액추에이터를 가동시켜 점을 찍는다. 이송부는 종이를 공급하거나 점을 찍을 때 다음 줄을 찍을 수 있도록 종이를 밀어준다.

① 점자프린터 - 타점부

구성 부품

- 점을 찍기 위한 부품 : 솔레노이드 액추에이터, 핀, 리니어 베어링
- 액추에이터 위치 조정 : 스텝 모터, 리미트 스위치, 타이밍 풀리, 벨트, 베어링

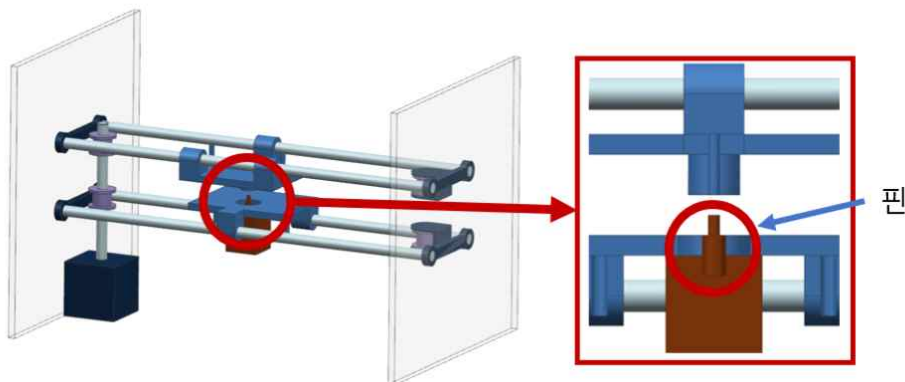


그림 7 - 타점부 구성도

점자프린터의 타점부는 액추에이터를 좌우로 움직일 수 있는 스텝모터와 점을 찍는 액추에이터를 사용한다. 점자프린팅은 점 간격이 일정해야한다. 따라서 위치를 빠르고 정확하게 제어할 수 있는 스텝모터를 액추에이터의 위치를 조절하는데 사용하였다. 스텝모터는 타이밍풀리와 벨트를 이용하여 액추에이터를 좌우로 움직일 수 있다.

점 하나를 빠르고 정확하게 찍을 수 있도록 액추에이터는 솔레노이드를 채택하였다. 하단 레일에는 액추에이터를 연결하였다. 액추에이터의 상단에는 점의 깊이와 너비를 결정하는 핀이 연결되어 상단 레일의 틀과 함께 점의 모양을 만든다.

② 점자프린터 - 이송부

구성 부품

- 종이 위치 조정 : 스텝 모터, 리미트 스위치, 타이밍 폴리, 벨트, 고무링 등

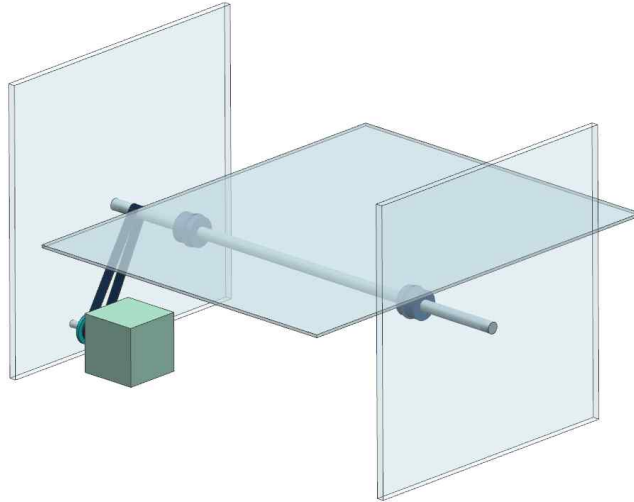


그림 8 - 이송부 구성도

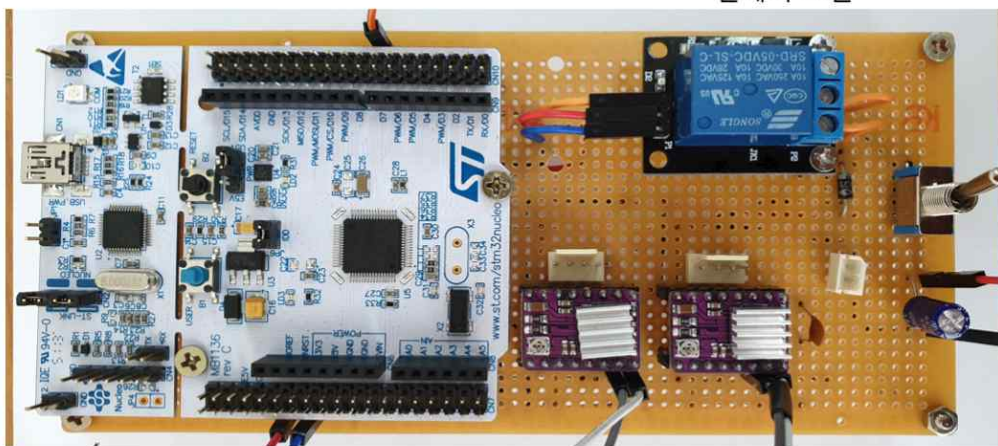
점자프린터의 이송부 또한 종이를 앞뒤로 빠르고 정확하게 움직여야 한다. 따라서 타점부와 마찬가지로 스텝모터를 폴리와 연결하여 축을 움직여 종이를 공급할 수 있도록 하였다. 타점부와 다른 점은 축에 고무링을 연결하여 마찰력으로 종이가 움직이도록 설계하였다.

3) 제어박스

구성 부품

- 회로 구성 : Mbed(F401RE), SMPS, 릴레이 모듈, 스텝모터 드라이버 (DRV8825)

릴레이 모듈



토글스위치

Microcontroller
STM32 F401RE

모터드라이버
DRV 8825

그림 9 - 제어박스 내부 사진

점자프린터를 구동하는데 필요한 제어박스의 회로도 및 실물은 그림 8과 같다. Microcontroller로는 ARM사의 Mbed(F401RE)를 사용하였다. 이는 32bit 프로세서로 기존 8bit 마이크로컨트롤러보다 빠른 계산속도를 가지고 있으며 회로 구성에 드는 비용이 동급의 아두이노보다 더 저렴하다. 회로는 스텝 모터와 솔레노이드 액추에이터의 전원은 12V SMPS를 사용하였으며 솔레노이드를 제어하기 위한 릴레이 모듈 하나와 스텝모터를 제어하기 위한 스텝모터 드라이버 (DRV8825) 두 개를 탑재하였다.

○ Hardware 기능

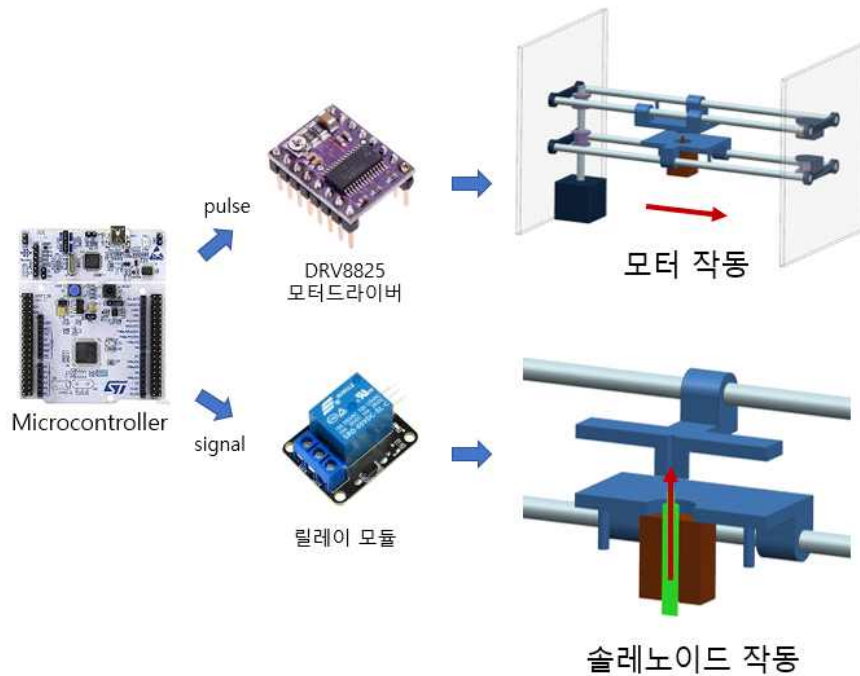


그림 10
타점부 작동 방법

1. 타점부

PC에서 점을 찍을 위치에 대한 정보를 얻으면 마이크로컨트롤러가 데이터를 처리하여 모터와 액추에이터를 작동시킨다. 이 때, 마이크로컨트롤러가 모터드라이버에 펄스를 보내면 모터가 작동하여 풀리를 회전시켜 액추에이터를 좌우로 이동시킨다. 마이크로컨트롤러에서 릴레이모듈에 신호를 보내면 액추에이터가 작동하여 점을 위로 찍는다.

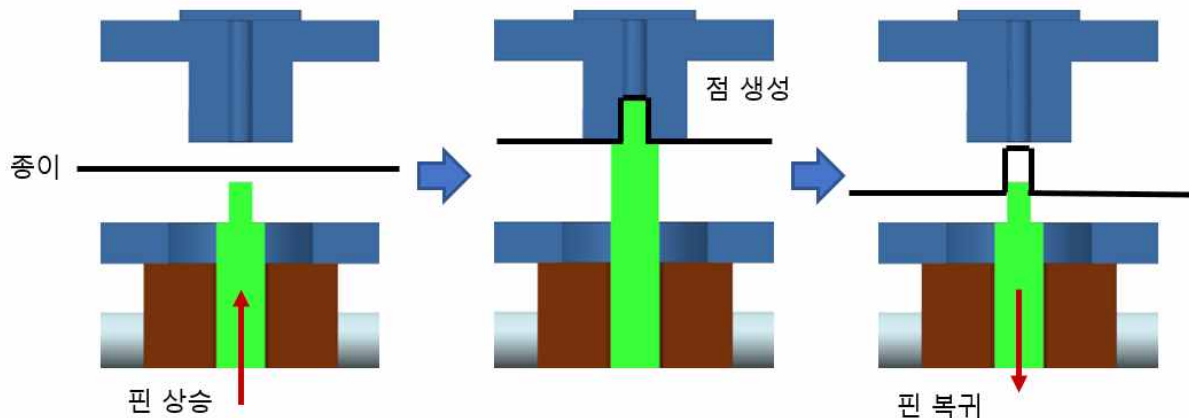


그림6 은 액추에이터가 점을 찍는 과정을 순서대로 묘사하였다.

- 1) 마이크로컨트롤러에 연결된 핀이 High가 되면 액추에이터 내부 핀이 급속도로 상승한다.
- 2) 솔레노이드 내부 핀이 상단 레일에 연결된 틀 안으로 종이를 밀어 넣는다.
- 3) 마이크로컨트롤러에서 LOW의 신호를 주면 솔레노이드 내부 핀이 하강한다.
- 4) 종이에 점이 생성되고 솔레노이드의 핀은 본래 위치로 돌아간다.

위 4가지 과정을 반복하여 액추에이터의 위치를 옮겨 점자를 생성한다.

2. 이송부

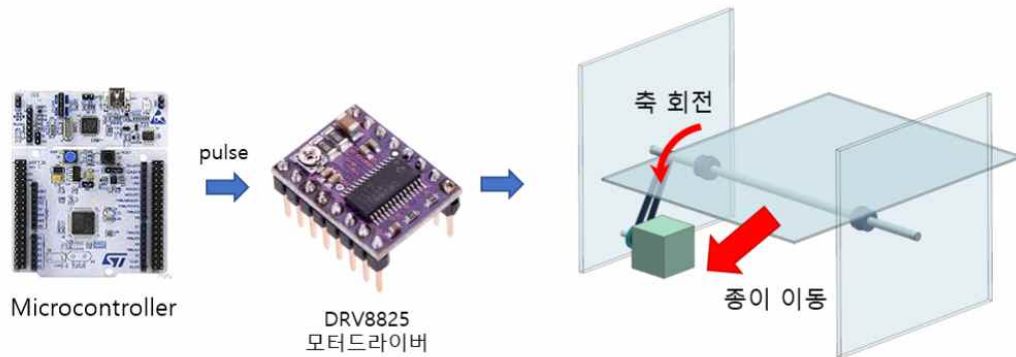


그림 12 이송부 작동 방법

타점부의 모터와 작동방식이 동일하다. 마이크로컨트롤러에서 모터드라이버로 펄스를 보내면 스텝모터가 회전하여 연결된 고무링이 회전하며 종이를 밀어낸다.

3. 제어박스

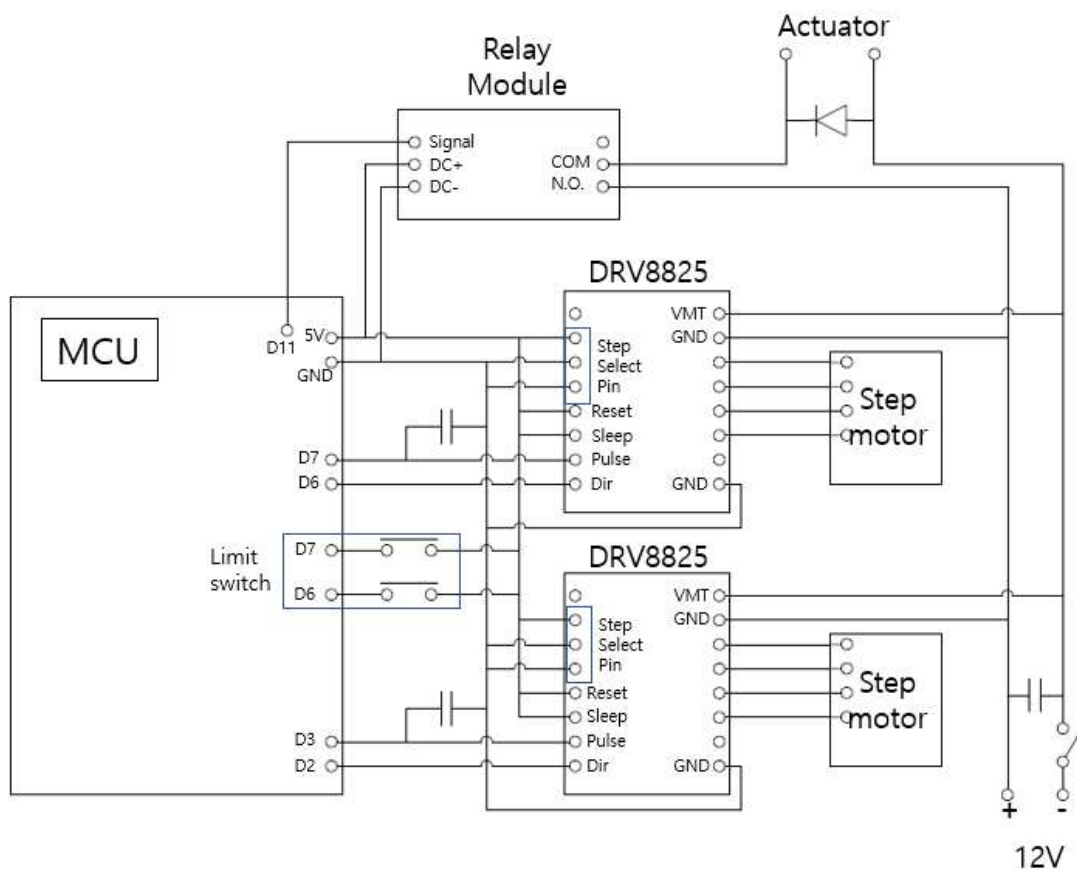


그림 13 - 제어박스 내부 회로도

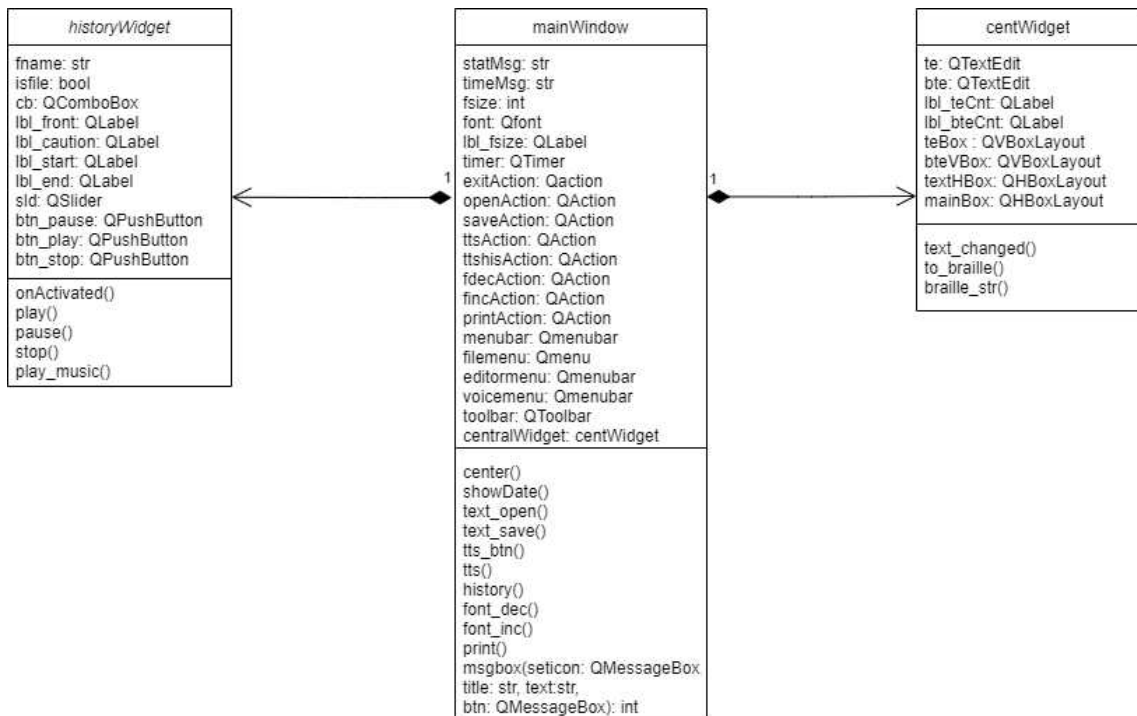
제어박스에 릴레이 모듈, 스텝모터 드라이버, SMPS를 추가하여 프린터 작동에 필요한 회로를 구축하였다. 릴레이 모듈은 솔레노이드를 가동할 때, 보드의 회로와 액추에이터의 회로를 분리하여 보드를 보호하고 액추에이터를 조절할 수 있다. 릴레이가 작동할 때 스텝모터 드라이버에 영향을 주는 것을 막기 위해 세라믹 커패시터를 추가하였다. 또한 전압의 안정적인 공급을 위해 SMPS와 커패시터를 병렬로 연결하였다.

○ Software 구성

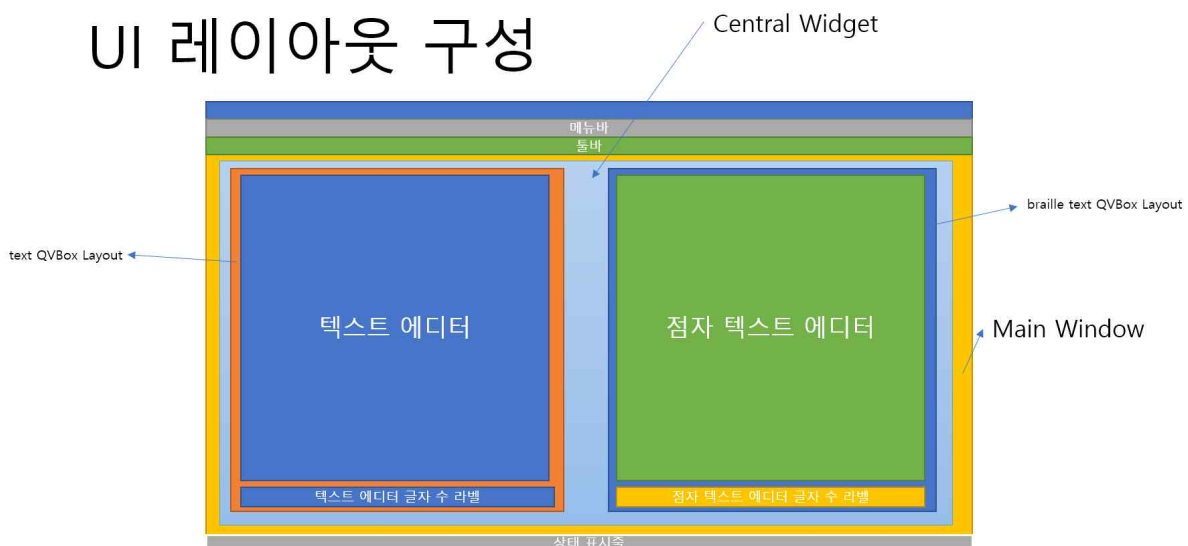
- 점자 프로젝트의 SW는 크게 PC와 microController(이하 micom)로 나누어져 있다. micom은 ARM社 Architecture인 Cortex M4 프로세서를 사용하였으며 c++기반의 mbed-os가 내장되어 있다. micom은 HW의 solenoid actuator와 step motor 및 센서들을 제어하는 SW를 담고 있다. PC는 사용자 접근성을 높이기 위해 GUI 프로그램으로 구성되어 있고, 사용자의 입력을 받아 데이터를 프린터에 보내기 위해 pre-processing과 고성능의 연산 또는 많은 크기의 memory를 필요로 하는 작업들을 담당하고 있다. 완전히 가공된 데이터를 micom으로 Serial 통신을 통해 전송하기 때문에 micom은 효율적으로 작업이 가능하도록 구성하였다.

○ Software 설계도 (흐름도 및 클래스 다이어그램 등 / 개발언어에 따라 선택)

- main.py



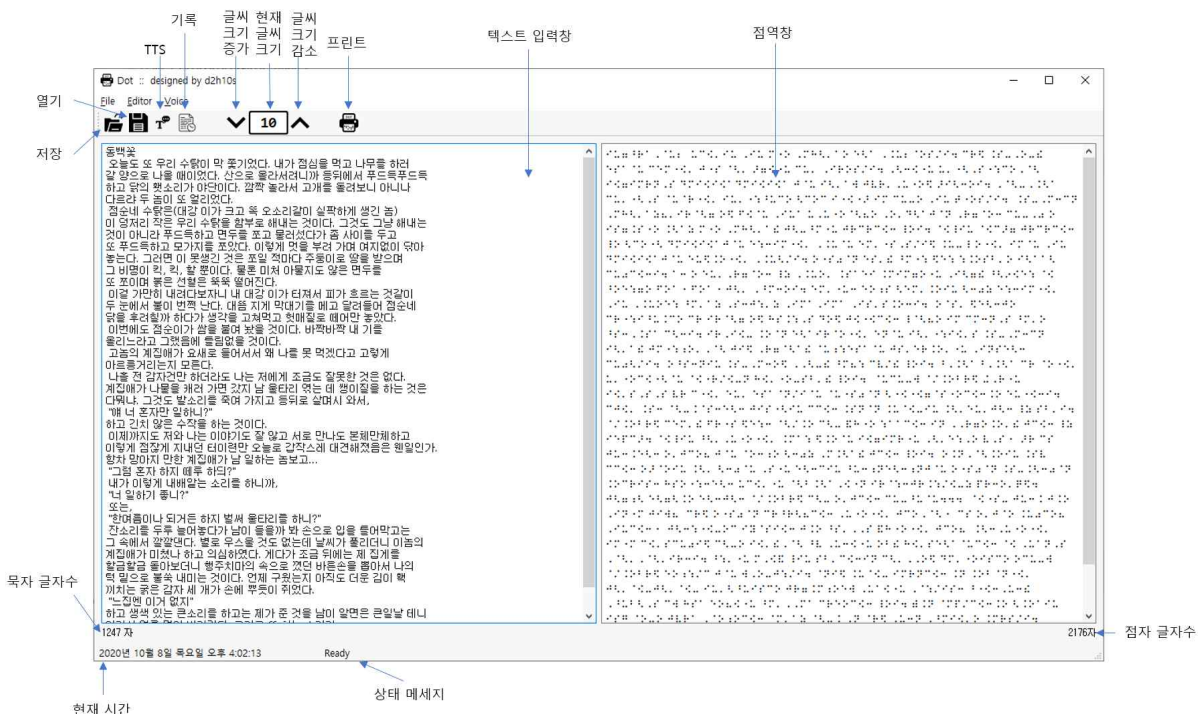
UI 레이아웃 구성



○ Software 기능 (필요 시 알고리즘 설명 포함)

- 열기: 텍스트 파일을 불러와서 바로 점역 및 사용이 가능하다.
- 저장: 텍스트 에디터에 쓰인 글을 저장할 수 있다.
- TTS: 텍스트 에디터에 한/영문으로 입력한 글자를 재생한다.
- 히스토리: 가장 최근 실행한 10개의 TTS 기록을 보고 재생할 수 있다.
- 폰트 사이즈 증가: 텍스트 에디터와 점자 번역 창이 글꼴 크기를 2단위로 증가한다.
- 폰트 사이즈 감소: 텍스트 에디터와 점자 번역 창이 글꼴 크기를 2단위로 증가한다.
- 프린트: 프린터와 통신을 통해 현재 입력한 텍스트를 점자로 출력한다.

○ 프로그램 사용법 (Interface)



- 텍스트 입력창에 글을 입력하면 실시간으로 오른쪽에 번역되어 나타나게 된다.
- 글자 수가 텍스트창 아래에 표시되므로 어느정도 크기로 프린트가 될 지 가늠할 수 있다.
- 툴바의 열기 아이콘 또는 'ctrl + o' 단축키로 입력한 텍스트 파일을 저장할 수 있으며 기본 경로는 바탕화면으로 되어 있다.
- 툴바의 저장 아이콘 또는 'ctrl + s' 단축키로 저장되어 있는 텍스트 파일은 불러올 수 있다.
- 툴바의 TTS 아이콘 또는 'ctrl + t' 단축키로 입력한 글을 음성으로 들을 수 있다.
- 툴바의 기록 아이콘 또는 'ctrl + h' 단축키로 이전에 실행하였던 최근 10개의 tts 기록을 보고 다시 재생할 수 있다.
- 툴바에 적혀진 숫자를 보고 현재 글씨 크기를 확인할 수 있으며 툴바의 위/아래 화살표 또는 'ctrl + 1' / 'ctrl + 2' 단축키를 통해 현재 글씨의 크기를 증감할 수 있다.
- 상태 메시지를 통해 최근에 실행한 프로그램 기능의 성공/실패 여부 또는 현재 프로그램의 상태를 알 수 있다.

○ 개발환경 (언어, Tool, 사용시스템 등)

<p>windows</p> 	<p>일반적인 사용 환경이 가정집 혹은 공기관이고, 일반인이 사용하는 것으로 고려하여 리눅스 환경으로 하지 않고 windows 기반으로 환경을 구축하였다.</p>
<p>python</p> 	<p>framework가 필요하지 않고 cross-platform을 지원하는 인터프리터 언어이며, 다목적 프로그램을 쉽게 내장할 수 있는 python을 선택하였다.</p>
<p>anaconda</p> 	<p>쉽게 가상환경을 구축하여 프로젝트간 파이썬 환경을 간단하게 분리할 수 있는 anaconda를 사용하였다.</p>
<p>pycharm</p> 	<p>통합개발환경(IDE)은 일반 목적의 디버깅과 GNU 디버깅 등 다양한 편의 기능을 제공하는 pycharm을 사용하였다.</p>
<p>pyqt5</p> 	<p>cross-platform ui 개발이 가능한 c++ open source framework이다. python으로 개발이 가능한 pyqt5를 사용하였다. 고급 기능 구현을 위해 qt designer를 사용하지 않고 파이썬 만으로 구현하였다.</p>
<p>gTTS</p> 	<p>Google의 딥러닝 기반의 TTS(Text to Speech) API이다. 무료로 공개되어 있고, api key가 기본으로 제공된다. pyttsx3와 같이 간편하게 tts를 제공하는 프로그램도 있지만, 자연스러운 음성과 프로그램을 가볍게 만들어 하나의 설치파일로 간단하게 쓸 수 있도록 하기 위해서 온라인 api를 사용하였다.</p>
<p>pyinstaller</p> 	<p>원래 파이썬은 인터프리터 언어로 바이트 코드로 컴파일하여 exe 파일로 불가능하기 때문에, 프로젝트를 exe 파일로 생성하는 pyinstaller를 사용하였다.</p>
<p>Mbed</p> 	<p>Mbed는 ARM-cortex기반의 MCU를 사용하여 IoT제품이나 여러 전자제품의 프로토타이핑을 쉽게 제작하기 위한 플랫폼으로 아두이노만큼 접근성이 좋고 높은 성능을 갖고 있다. 스텝모터를 효과적으로 제어하기 위해 Mbed중 NUCLEO-F401RE를 선정했다.</p>

□ 개발 프로그램 설명

○ 파일 구성

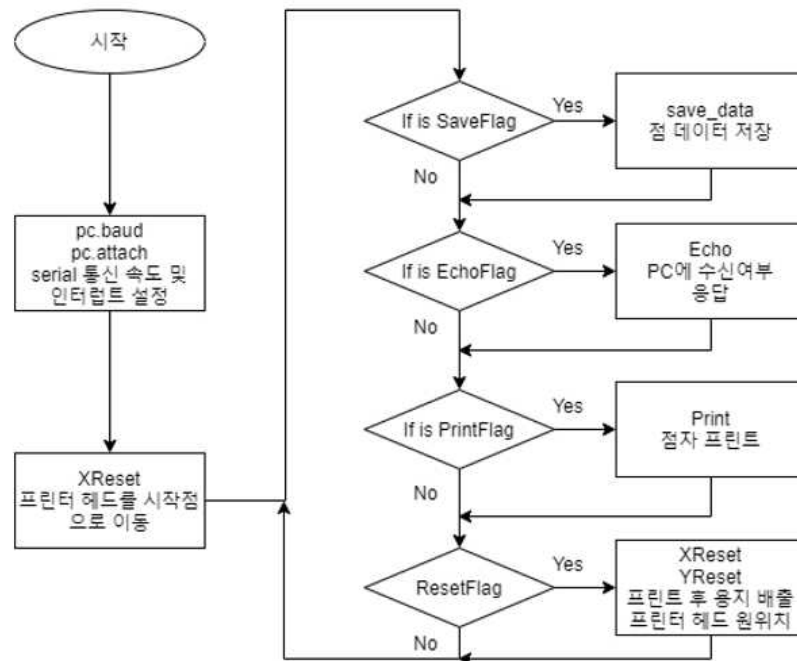
- PC: pybraille

```
├ main.py      - main widget, central widget, histoty widget과 같은 QWidget 클래스들의 집합으로 GUI의 주요 상호작용 기능들을 구현한다.
├ libpkg       - 커스텀 라이브러리의 집합이다.
│   ├── Communication.py - 데이터 전처리 및 checksum과 같은 micro controller와의 Serial 통신을 위한 함수들의 집합이다.
│   │   │
│   │   └ tts.py        - google의 tts api를 사용하기 위한 함수가 정의되어 있다.
├ translator   - 목자를 점자로 번역하는 알고리즘들의 집합이다.
│   ├── kor_to_braille.py - 입력된 문자열을 점자데이터로 점역하는 파일이다.
│   │   │
│   │   └ map_kor_to_braille.py - 한글,영어,기호 등의 점역을 위한 점자 유니코드 리스트가 담겨있는 파일이다.
├ icon         - gui 프로그램의 아이콘 이미지 파일들을 보관한다.
│   ├── down.png      - 아래 방향 화살표 아이콘 이미지 파일이다.
│   │   │
│   │   └ printer.png - 위 방향 화살표 아이콘 이미지 파일이다.
└ data         - 이전 tts의 기록을 저장하는 폴더이다.
    ├── tts1.mp3      - 가장 최근의 tts 파일이다.
    │   │
    │   └ tts10.mp3   - 가장 마지막에 실행한 tts 파일이다.
```

- MBed

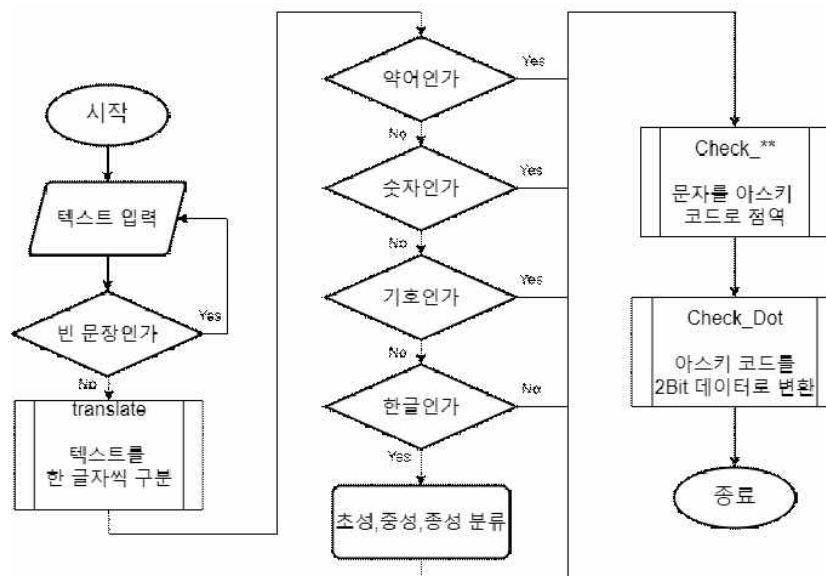
```
├ main.cpp     - MBed의 메인함수. 기능별 flag 상태에 따라 데이터 수신 및 저장, 프린터 타점, 액추에이터 원위치 등 프린터를 작동하는 함수를 호출한다.
├ motor.h      - 스텝모터 구동에 필요한 클래스를 선언하는 헤더파일이다.
├ motor_setup.cpp - 점자를 프린트하는 함수를 내장한다. 수신된 점자를 불러와 액추에이터를 가동시켜 1줄에 해당하는 점자를 찍거나 프린트의 완료를 알린다.
└ Serial_setup.cpp - pc에서 serial 통신으로 데이터를 수신하는 함수와 pc에 응답하는 echo를 선언한다.
```

- Mbed Main 알고리즘 Flowchart



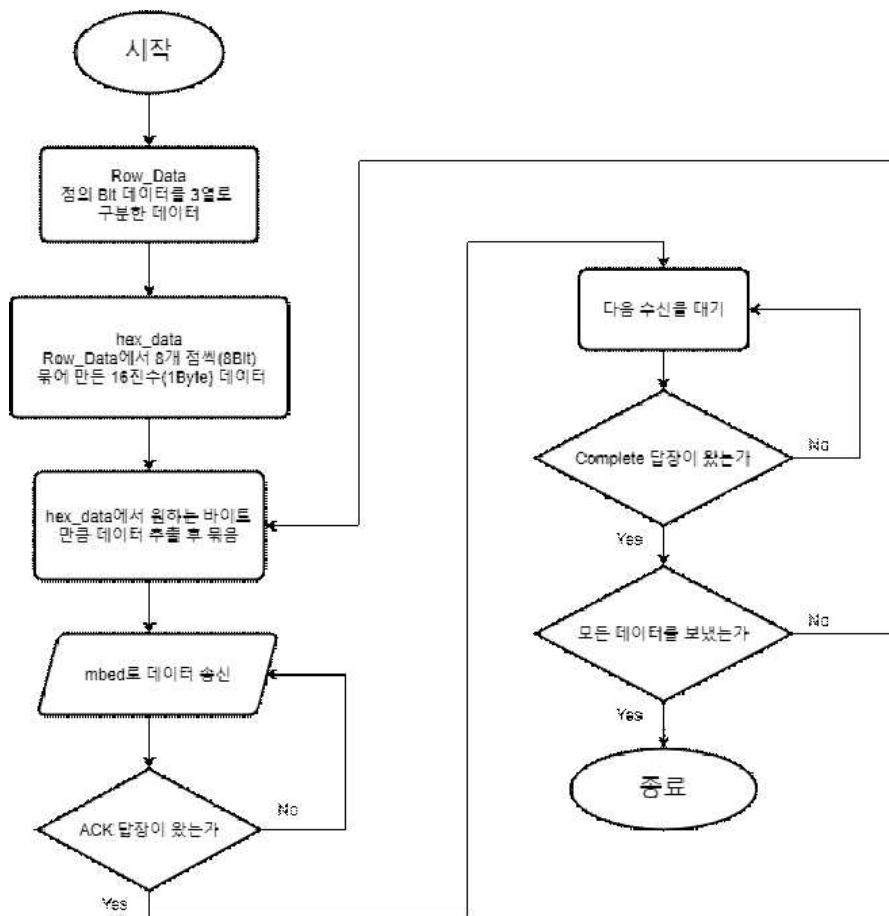
- Mbed가 작동하면 우선 시리얼 통신 설정(통신속도, 수신인터럽트)을 지정한다. 그 후 프린터의 타점부 액추에이터를 시작지점으로 이동한다. 수신 인터럽트가 완료되면 점자에 해당하는 데이터를 옮겨온 뒤 PC에 데이터 수신여부로 Acknowledged(ACK, 0x06), Not Acknowledged(NAK, 0x16) 보낸다. 정상 수신된 뒤, 프린트가 시작되며 모든 점자를 출력하면 프린터 타점부가 원위치로 이동하고 용지를 프린터 밖으로 배출한다.

- 점자 점역 알고리즘 Flow chart



- 점역할 문장을 받으면 문장을 한 글자씩 구분하고 해당 문자가 어떤 문자인지 구별한다. 이후 점자 표기를 위해 6점자 유니코드로 점역하고 점자 데이터를 프린터 제어 보드(mbed)에 보내기 위해 유니코드 데이터를 2Bit 형식으로 변환한다.

- PC 통신 알고리즘 Flow chart



- 이전 점역 과정을 통해 2bit 점자 데이터가 만들어지면 이를 보내기 위해 데이터를 정리하는 과정이 필요하다. 먼저 점자는 두 점 씩 3줄로 이루어져 있는데 1줄씩 데이터를 보내기 위해서 전체 데이터를 3줄로 나누었다. 그리고 바이트 통신을 위해서 Bit 데이터를 묶어 1Byte로 만들고 보낼 데이터만큼 나누어 hex_data로 정리한다. 만든 데이터를 mbed로 송신하면 보드가 잘 받았는지 답장을 기다린다. NAK가 답장으로 오면 데이터를 재송신하고 ACK를 받으면 보드의 다음 메시지를 기다린다. 점자 프린터에서 다음 데이터를 바라면 Complete를 보내고 이를 받으면 Pc에서 다음 데이터를 보내고 모든 데이터를 보내고 나면 통신을 완료한다.

○ 기술적 차별성

- 기존 점자 프린터는 점 간격의 크기가 약 10mm 이상으로 실제 점의 크기(한 점 1.5mm, 점간 간격 2.5mm)를 구현하지 못했다. 하지만 우리팀은 실제 점자 사이즈를 출력할 수 있는 프린터를 구현하였다.
- 또한 하드웨어 설계는 3D 프린터를 참고하였으나, 프린팅 정보를 G-code에서 Byte로 변환하여 선택하였다. 이는 하나의 좌표(3Byte 이상)에서 점 하나의 정보를 얻어오는 기존 방식에서 1 Byte에 8bit, 즉 8개의 점 데이터를 담을 수 있도록 하였다. 이는 실시간으로 짧은 문구를 프린트로 프린트할 수 있는 장점을 가진다.

○ 함수별 기능

- Mbed

└ main.cpp	
└ Serial_setup.cpp	
└ RX_ISR()	- Serial 통신에 사용할 수신 인터럽트를 선언한다. 정해진 프로토콜에 맞추어 PC에서 수신된 데이터를 Start Sequence Byte(STX), Command(CMD), Length(LEN), Data, Received Checksum, End Sequence Byte(ETX)로 분류한다. PC에서 데이터를 모두 수신한 후, Command에 따라 출력, 응답, 긴급정지를 실행한다.
└ save_data()	- RX_ISR()에서 PC로부터 수신한 데이터 중 점자에 해당하는 정보를 추출하여 Data에 저장한다.
└ Echo()	- RX_ISR()에서 데이터를 모두 수신한 후, 데이터의 오류 여부를 Checksum()을 이용하여 확인하여 PC로 전송한다. 데이터에 오류가 없을 시, Acknowledged (ACK, 0x06)을 전송한다. 데이터에 오류가 있을 경우 Not Acknowledged (NAK, 0x16)을 전송한다. Command(CMD) 값이 1일 경우에 데이터에 오류가 없으면 Acknowledged를 송출한 뒤 print를 실행하도록 준비한다.
└ Flush_array(volatile uint8_t*)	입력된 배열을 모두 초기화한다.
└ CheckSum()	- Checksum은 intel 방식을 간소화한 알고리즘을 채택하였다. pc에서 받은 ReceivedCS값과 Data를 알고리즘에 대입한 결과값을 비교하여 serial 통신의 오류 여부를 확인한다.
└ motor.h	
└ DRV8825.move(float)	- 클래스로 선언된 객체의 이동거리(mm)만큼 모터를 회전한다. 스텝모터에 연결된 액추에이터나 종이를 원하는 거리만큼 이동시킬 때 사용한다.
└ motor_setup.cpp	
└ Act()	- 액추에이터를 1회 가동시킨다. solenoid 내부 핀을 짧은 시간 동안 상승시켰다가 하강시킨다.
└ Print()	- save_data()에서 저장된 데이터를 불러온 뒤, 점을 찍는다. 프린트 헤드가 좌->우로 이동(정방향)할 때 Data 내부의 Byte를 정방향으로 읽고 Act를 호출하여 점을 찍는다. 우->좌로 이동(역방향)일 때 Data 내부의 Byte를 역방향으로 읽고 Act를 호출하여 점을 찍는다. Byte 내부 bit를 읽고 나서 모터를 움직여 옆으로 한 칸 또는 아래로 한 줄 이동한다. 프린트가 완료되면 Complete()을 호출한다.
└ Complete()	- 프린트 완료를 알리는 EM(End Media, 0x19)를 PC에 전송한다.
└ XReset()	- 프린터 헤드를 X축 처음 위치로 이동한다. 액추에이터가 좌로 반복 이동하는 중 Limit 스위치가 감지하면 모터가 정지한다.
└ YReset()	- 종이를 완전히 내보낸다. Y축 모터를 종이를 50cm 이동시킬 때까지 반복하여 모터를 작동시킨다.

```

└─ main.py
    │
    │ histotyWidget: QWidget
    │ │
    │ │ └─ onActivated(text:str) - 콤보 박스에서 tts 파일을 선택하면 해당 파일의 이름을 저장
    │ │ │ 해둔다.
    │ │
    │ │ └─ play() - 데몬쓰레드를 생성하고 play_music()을 attach하여 tts파일을 재생한다.
    │ │
    │ │ └─ pause() - 재생을 일시 정지한다.
    │ │
    │ │ └─ stop() - 재생을 중단한다.
    │ │
    │ │ └─ play_music() - playsound모듈을 이용하여 mp3를 재생한다.
    │
    │ centWidget: QWidget
    │ │
    │ │ └─ text_changed() - 텍스트창의 텍스트에 변경이 일어나면 발생하는 이벤트랑 연결하
    │ │ │ 여 실시간 점역과 텍스트 글자 수를 계산하여 표기한다.
    │ │
    │ │
    │ │ └─ to_braille(json:str) - 6점자로 점역된 결과를 알려준다
    │ │
    │ │
    │ │ └─ braille_str(json:str) - 텍스트를 점역 파일을 통해 점역한다
    │
    │
    └─ mainWindow: QMainWindow
        │
        │ └─ center() - 화면의 사이즈를 계산하여 중앙점으로 화면을 이동시켜 주는 함
        │ │ 수이다.
        │
        │
        │ └─ showDate() - QTimer를 이용하여 상태표시줄에 900ms의 interval로 시간을 업
        │ │ 데이트한다.
        │
        │
        │ └─ text_open() - 컴퓨터 내에 있는 텍스트 파일(.txt)을 불러와 텍스트 창에 띄워준
        │ │ 다.
        │
        │
        │ └─ text_save() - 텍스트 창에 입력된 텍스트를 .txt 파일로 만들어 지정 경로에 저
        │ │ 장하는 역할을 한다. 기본 경로는 바탕화면이다.
        │
        │
        │ └─ tts_btn() - tts 버튼과 연결된 이벤트 함수이다. tts함수와 연결된 데몬 쓰레
        │ │ 드를 생성하는 역할을 하고 있다.
        │
        │
        │ └─ tts() - tts 쓰레드와 연결된 tts 함수이다. 텍스트 입력값이 없으면 도입
        │ │ 부에서 return을 하고 그렇지 않으면 text2speech를 실행한다.
        │
        │
        │ └─ history() - 히스토리 위젯을 생성하여 호출한다.
        │
        │
        │ └─ font_dec() - 폰트 사이즈 감소 버튼과 연결된 이벤트 함수이다. 텍스트 창과
        │ │ 점역 창의 폰트 사이즈를 2 감소시킨다. 최소값은 10이다.
        │
        │
        │ └─ font_inc() - 폰트 사이즈 증가 버튼과 연결된 이벤트 함수이다. 텍스트 창과
        │ │ 점역 창의 폰트 사이즈를 2 증가시킨다. 최댓값은 40이다.
        │
        │
        │ └─ print() - 프린트 버튼과 연결된 이벤트 함수이다. 통신 함수를 호출하여
        │ │ 데이터 전처리와 micom과 통신을 한다.
        │
        │
        └─ msgbox(seticon: QMessageBox, title: str, text: str, btn: QMessageBox)
            │
            │ └─ 제목과 내용, 알림을 설정하여 Message를 띄운다.

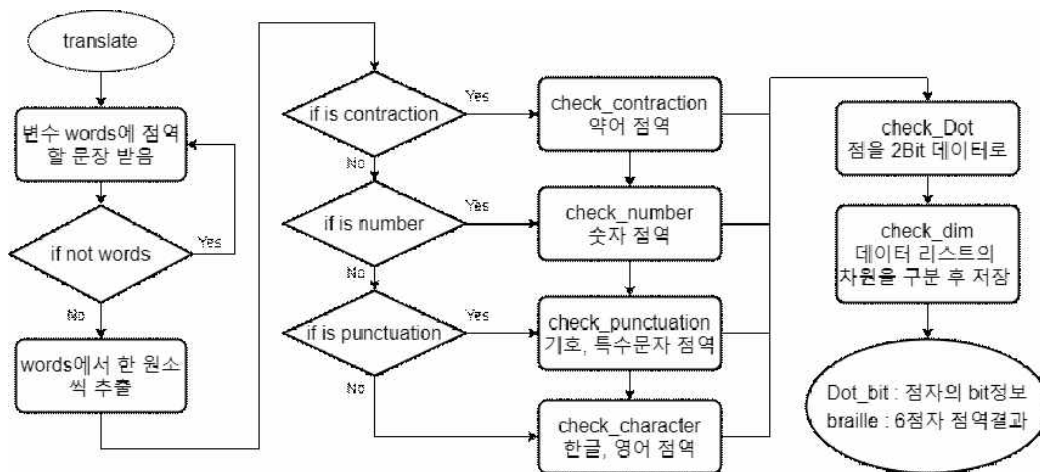
```

- PC: translator

└─ kor_to_braille.py	
└─ extract_words()	- 입력한 문장을 받아오는 역할로 translate 함수에서 사용한다.
└─ check_contraction()	- 약어나 특수한 단어를 점역하는 역할을 한다.
└─ check_number()	- 숫자를 점역하는 역할을 한다.
└─ check_punctuation()	- 기호, 특수문자를 점역하는 역할을 한다.
└─ check_character()	- 입력된 문자형을 점역하는 역할을 한다.
└─ check_Dot()	- 유니코드로 점역된 결과를 2Bit로 저장하기 위한 함수
└─ check_dim()	- 점역 결과 리스트의 차원을 확인하여 리스트에 저장하는 함수
└─ translate()	- 입력한 문장을 받아와 한 원소씩 점역하고 점역 결과를 반환한다.
└─ Communication.py	
└─ CS()	- mbed와 주고받은 데이터의 유효성을 검사하는 역할을 한다.
└─ bit2byte()	- 8개의 점(8bit)의 정보를 모아 1Byte로 조합하고 이를 7번 반복하여 총 7Byte의 데이터 리스트를 반환한다
└─ dot_debugging(dot_data: list)	- 데이터 전송 직전의 바이트 데이터를 다시 점데이터로 만들어 콘솔에 출력하는 디버깅용 함수이다.
└─ spread(dot: list)	- 점자 유니코드를 [1,0,1,1,0,0]과 같은 리스트로 만드는 함수이다.
└─ Data_Send()	- 만들어진 데이터를 mbed로 보내고 답장을 받으며 모든 데이터를 보내는 작업을 한다.

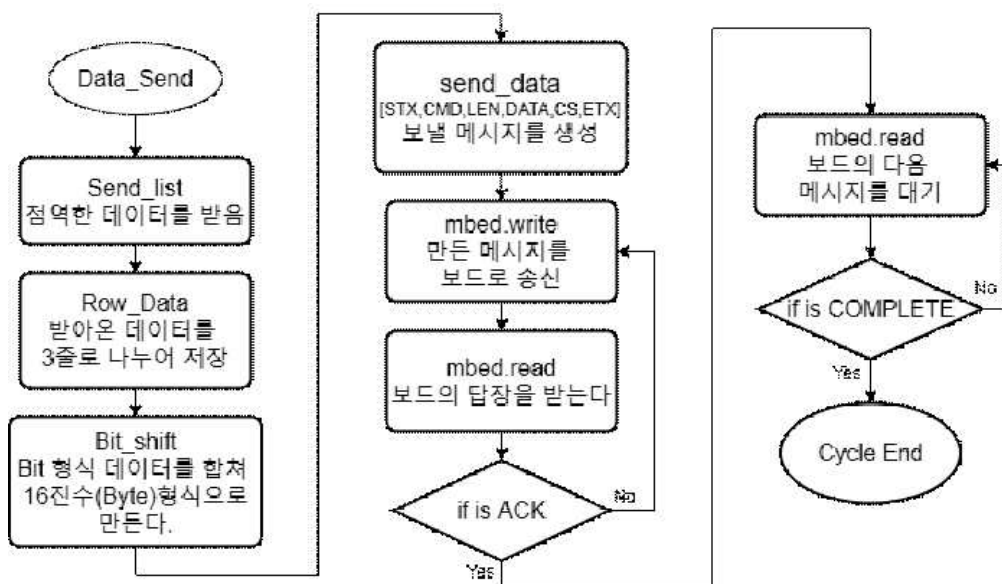
○ 주요 함수의 흐름도

- 점역 프로그램의 주요 함수 Flow chart



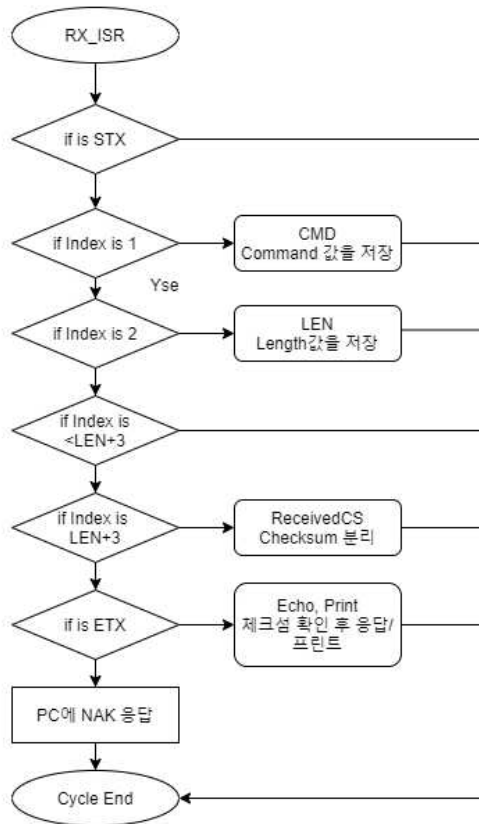
- 1) 점역할 문장을 변수 words에 저장하고 빈 문장의 경우 되돌아간다.
- 2) 변수 words에 저장된 문장에서 한 글자씩 추출한다.
- 3) 추출한 글자의 종류를 찾는다 -> 약어, 숫자, 기호, 한글 등
- 4) 종류에 해당하는 점자 리스트와 비교하여 6점자 유니코드로 점역한다.
- 5) 유니코드로 점역된 결과는 GUI에 점자를 표기하는데 사용된다.
- 6) 결과를 2Bit 형식의 데이터로 변환한다. ex) 'ㄱ': [1,0,0,0,0,0], '것': [[0,0,0,1,1,1], [0,1,1,1,0,0]]
- 7) Bit 형식의 데이터를 하나의 리스트로 저장한다.

- Pc 쪽 통신 프로그램의 주요 함수 Flow chart




- 1) Bit 형식의 점자 데이터를 받아오면 이를 Byte 형식으로 묶어준다.
- 2) 원하는 Byte만큼의 데이터를 포함한 메시지를 만든다. 형식은 STX,CMD,LEN,DATA,CS,ETX 순서로 시작과 끝을 알리는 STX,ETX와 프린트 명령인 CMD, 데이터 길이를 알려주는 LEN, 데이터 유효성을 확인하는 CS가 포함된다.
- 3) 송신 후 보드가 잘 받았으면 ACK를 보내고 못받았으면 NAK를 보낸다. 만약 NAK를 받으면 데이터를 재송신한다.
- 4) 이후 프린트가 COMPLETE를 보낼 때까지 대기하고 받으면 다음 메시지 과정을 반복한다.

MBed 수신 인터럽트 Flow chart

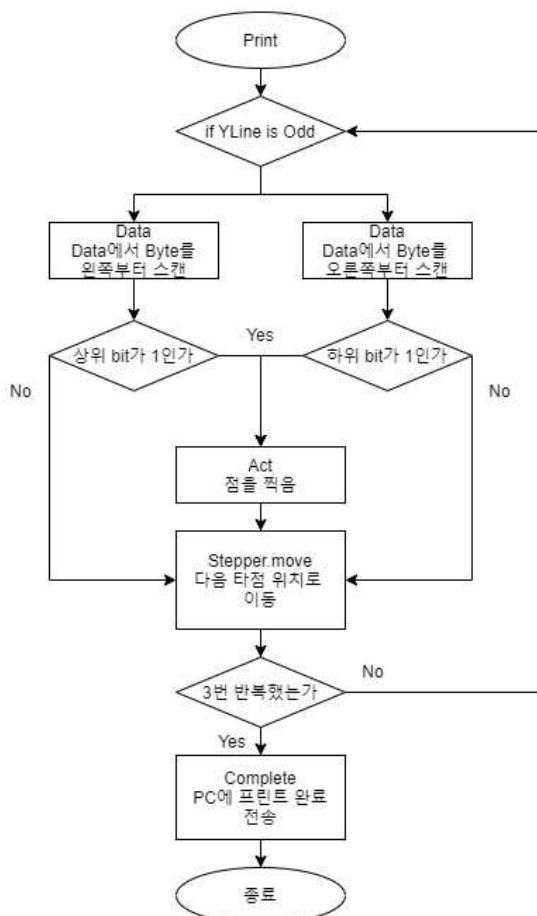


- 1) Start Sequence Byte가 들어오면 저장을 시작한다.
- 2) 1번 Byte는 보낸 데이터의 종류로 Command에 저장한다.
- 3) 2번 Byte는 Data의 길이에 대한 정보로 Length에 저장한다.
- 4) 3~LEN+2의 Data는 buffer에 임시로 저장한다.
- 5) LEN+3의 Data는 Received Checksum에 저장한다.
- 6) 마지막 Byte가 ETX인지 비교 후, 프로토콜 형식을 갖추었으면 ACK를 전송하고 그렇지 않으면 NAK를 전송하도록 한다.

※ 전송된 Data 중 1bit에 1개의 점 정보가 담겨 있다. 따라서 Byte 통신을 이용하여 Byte에 8bit의 정보, 즉 8개의 점 정보를 Byte에 담아서 PC에서 마이크로컨트롤러로 전송한다.

e.g) 점 데이터 -> 
Byte로 변환된 데이터 -> 10101111

MBed Print 함수 Flow chart




- 1) 입력하는 점자가 홀수행이면 Data의 Byte를 왼쪽부터 차례대로 불러온다. 짝수행이면 Byte를 오른쪽부터 왼쪽으로 Byte를 불러온다.

- 2) 불러온 Byte를 홀수행이면 큰 bit부터 비교한다. 짝수행이면 작은 bit부터 비교한다.

※ 헤드 이동방향에 맞춰 찍는 순서를 바꾼다.

헤드 이동방향 좌->우  점을 찍는 순서 10101111

헤드 이동방향 우->좌  점을 찍는 순서 10101111

- 3) 비교한 bit가 1이면 점을 찍는다.
- 4) 다음 점 위치로 프린터의 헤드가 움직인다.
- 5) 모든 Byte를 비교하였으면 프린트를 종료한다.

□ 개발 중 발생한 장애요인과 해결방안

○ PC 및 Mbed 통신

- PC에서 파이썬으로 통신할 경우, 유니코드 문자열 형식으로 encode 후에 보내야했다. 이 방식은 복잡한 형식의 데이터를 보내는 것이 아니므로 적합하지 않고 많은 통신 데이터를 차지하기 때문에 적합하지 않다. 따라서 점자의 정보가 Bit 형식으로 저장되므로 데이터를 Byte 단위인 8bit로 slice하여 데이터를 송신하여 데이터양의 획기적으로 줄이고 프린터에 맞는 프로토콜을 이용하여 해결하였다.
- 반대로 Mbed는 시리얼 통신으로 데이터 수신할 때 오류가 발생하였다. 다른 마이크로컨트롤러는 RX buffer의 값을 읽는 수신 인터럽트가 종료된 직후 RX에 데이터가 바로 수신된다. 하지만 Mbed는 연속적인 수신 data에 대해 수신 인터럽트가 연속적으로 발생하지 못하고 main문을 진행한다. 이를 해결하기 위해 수신된 Byte를 매번 확인하여 프로토콜에 적합하게 수신되었는지 비교하도록 하였다.

○ PC 소프트웨어 시행착오

- GUI 툴로 선정한 PyQt5 모듈이 thread safety하지 않은 코드가 많아 event 처리 방식이 아닌 subthread 방식으로 mutex lock을 하더라도 프로그램이 셧다운 되는 경우가 잦았다. 따라서 blocking code가 들어가는 음성재생 또는 polling 방식의 통신을 사용하는 경우를 제외하고 최대한 threading 사용을 배제하여 오류 가능성을 줄였다.
- TTS는 API를 이용하여 실행하기 때문에 음성 데이터를 PC에 저장한 후 재생하는 방식을 이용했었다. 그러나 API와 file stream간 충돌 때문에 두 번째 실행부터는 저장되지 않는 문제가 발생하였다. 따라서 10개의 history 기록을 만들고 새로 TTS를 실행할 때 가장 오래된 파일을 지우고 새로운 파일을 저장하도록 하여 오류를 해결하였다.

○ 회로 문제

- 솔레노이드 액추에이터를 사용하여 점자를 생성하는 방식을 선택하였다. 이는 솔레노이드 액추에이터가 빠른 속도로 상하 운동을 할 수 있기 때문이다. 하지만 솔레노이드가 빠르게 운동함에 따라 하강 시 회로 내부에 역방향으로 기전력이 발생하여 역전류가 발생해 회로에 손상을 야기하였다. 또한 액추에이터를 제어할 때 사용하는 릴레이 내 코일에도 사용하여 역전류가 발생하여 마이크로컨트롤러에서 스텝모터로 보내는 펄스에 노이즈를 발생시켰다. 액추에이터에는 프리휠링 다이오드를 연결하여 역기전력을 제거하였고, 릴레이에서 발생하는 노이즈는 세라믹 커패시터를 사용하여 제거하였다.

□ 개발결과물의 차별성

○ OCR 점역 독서대

기존 개발결과물의 문제점

- 기존 제작된 독서대의 점자 디스플레이의 크기는 실제 점자에 비해 매우 크다. 많은 핀을 동시에 솔레노이드로 제어할 때 많은 공간을 차지하기 때문이다. 즉, 실제 시각장애인들이 점자를 읽을 수 있는 크기로 소형화하는데 한계가 존재한다.
- 또한 정해진 규격의 책을 독서대에 올려놓았을 때에만 텍스트를 번역할 수 있다는 한계를 가진다. 동시에 점자를 읽을 수 없는 사용자는 위 개발결과물을 사용할 수 없어 사용범위가 제한적이다.

우리 개발결과물의 장점

- 우리팀의 개발결과물은 하나의 솔레노이드로 실제 점자 규격의 점자를 출력할 수 있다.
- 또한 점자를 읽을 수 없는 사람을 위해 음성 데이터를 생성하여 저장할 수 있는 기능을 갖추고 있기에 점자를 모르는 사람도 출력 결과물을 사용할 수 있다.

○ 점화 번역 웨어러블 기기(팀 : 점심)와 다른점

기존 개발결과물의 문제점

- 웨어러블 기기의 가장 큰 단점은 기기를 항상 소지하고 다녀야한다는 점이다.

우리 개발결과물의 장점

- 우리가 개발한 결과물은 출력 결과물이 음성이나 점자 텍스트이다.
- 따라서 누구나 항상 휴대하는 휴대폰에 음성데이터를 저장할 수 있고, 유인물 형식으로 보관하거나 휴대할 수 있다.

○ 추가적인 차별점

현 시장의 문제점

- 최근 개발중인 시각장애인 용품은 전부 사용자를 시각장애인에 초점을 두고 있다.
- 또한 실제 개발된 프린터들은 모두 대형 출판업을 기반으로 제작되어 구매 및 유지가격이 매우 비싸다.
- 실제 시각장애인은 한 권 분량의 책을 장기간에 걸쳐 번역하여 받는 것보다 적은 양의 다양한 텍스트를 빠른 시간 내에 받을 수 있는 방법을 필요로 한다.

개발결과물의 우수성

- 우리팀의 개발결과물은 시각장애인의 주변인에게 도움을 줄 수 있다는 점에서 유용하다.
- 점역 또는 음성 번역된 데이터를 시각장애인에게 상시 제공할 수 있어야하는 기관에서 사용할 수 있다. 즉, 시각장애인에게 텍스트로 된 문서를 점자 또는 음성자료로 저장하여 건네줄 수 있다. (주민센터의 서류, 민원에 대한 응답기록물 등)
- 또한 저렴한 가격으로 일반 가정에서 점자도서관을 거치지 않고 시각장애인에게 텍스트를 점역 또는 음성 번역하여 제공할 수 있다는 점에서 짧은 기간 내에 번역물을 제공받을 수 있고, 텍스트의 내용을 제 3자에게 밝히지 않아도 된다는 장점을 가진다.

□ 개발 일정

내용	2020年																			
	6月				7月				8月				9月				10月			
프로젝트 아이디어 선정																				
	H/W 및 프린터 프로그래밍 제작팀 (김상민, 김상일, 윤인재)																			
H/W 설계																				
H/W 제작																				
H/W 기능 테스트																				
H/W 기능 구현																				
	PC 프로그래밍 담당 (최호승, 조현우)																			
S/W 기능 선정																				
S/W 기능 테스트																				
S/W 기능 구현																				
	최종 팀원 작업 과정																			
시스템 통합 및 테스트																				
시스템 수정 및 보완																				

※ 아이디어 선정 이후, 팀을 H/W 및 임베디드 프로그래밍, PC 프로그래밍으로 나누어 프로젝트를 진행하였다.

※ 중간 과정에서 부품의 손상 및 배송 기간으로 인해 기능 테스트 및 구현 기간이 길어졌다.

※ COVID-19로 인해 직접 모여서 작업할 수 있는 시간과 공간이 제한적이었다. 대부분 회의는 가급적 원격 화상 회의로 진행하였다.

□ 팀 업무 분장

No	구분	성명	참여인원의 업무 분장
1	팀장	김상민	<p>전체 업무 총괄 및 업무 분배 / 프린터 제작 및 MBed 프로그래밍</p> <ul style="list-style-type: none"> - MBed 프로그래밍 병합 / MBed 시리얼 통신 프로그래밍 - 점자 출력 알고리즘 및 함수 제작 - 프린터 도면 제작 및 형상 수정
2	팀원	최호승	<p>PC 프로그래밍 담당 (정) - Python</p> <ul style="list-style-type: none"> - 파이썬 기반 PC 시리얼 통신 프로그래밍 - PyQt를 이용한 GUI 프로그래밍 - 파이썬 프로그래밍 병합
3	팀원	김상일	<p>프린터 및 회로 설계, 제작</p> <ul style="list-style-type: none"> - 프린터 부품 3D 프린팅 - 프린터 형상 조립 및 가공 - 액추에이터 회로 설계 및 실험
4	팀원	윤인재	<p>회로 및 MBed 프로그래밍 / 프린터 제작</p> <ul style="list-style-type: none"> - 스텝모터 회로 설계 및 실험 - MBed 통신 프로그래밍 - 프린터 조립 및 가공
5	팀원	조현우	<p>PC 프로그래밍 (부) - Python</p> <ul style="list-style-type: none"> - 점자 번역 프로그램 모듈 생성 - 번역된 점자 분류 알고리즘 제작 - 통신용 Data List 생성 알고리즘 제작