2023년도 전자공학부 캡스톤디자인2 보고서

팀명	스트레스 멈춰		
과제명	도서 검색 로봇		
구성원	학과(전공)	학번	이름
	전자공학부	2017006262	김용현
	전자공학부	2018038740	김도현
	전자공학부	2020010764	유창선
과제유형	HW ( ) SW ( ) HW/SW (●)	지도교수	고현석 교수님 (인)
과제개요	도서관, 물류창고 등 다양한 종류의 보관 항목이 존재하는 공간에서 사용자가 원하는 물품을 찾아주는 과정을 대체해주는 로봇을 제작한다. 해당 로봇은 물품을 찾아 해당 위치 정보만을 전달해주던 기존의 방식에서 해당 물품을 선반에서 뽑아 사용자에게 직접 전달하는 것을 동작 목표로 한다.		

1. 과제의 목표 및 개요

1) 주제 선정 배경

도서관에서 책을 찾을 때, 도서를 검색한 후 도서의 일렬번호를 확인, 책장에서 해당 번호를 찾아 도서를 획득하게 된다. 그러나 도서가 많아질수록 일렬번호는 길어지고 책장의 수도 증가하게 된다. 이에 도서를 찾는 데 드는 시간과 노력을 줄이고자 기존의 책장에 프레임을 부착하여 작동하는 도서 검색 로봇을 고안하게 되었다.

캡스톤디자인1에서는 도서관에서의 활용을 목적으로 도서 검색을 위주로 주제를 선정하였으나, 물품의 품목이 다양하고, 해당 물품을 찾는 노력을 단축하고자 하는 요구가 있다는 점에서 물류창고와 같은 장소에도 적용하여 물류 자동화를 노릴 수 있다는 점에 착안하여 기존의 물품이 어디에 있는지 찾아내고, 물품의 위치 정보만을 사용자에게 제공, 사용자가 직접 해당 물품을 가지러 이동하는 기존의 방식에서 발전한 로봇이 직접 책장 또는 선반으로부터 물품을 뽑아 사용자에게 전달할 수 있는 시스템을 구축할 수 있도록 제작했다.

2) 작품 개요

1. 사용자는 앱을 통해 찾고자 하는 도서 또는 물품을 검색한다.
2. 앱은 서버에 찾고자 하는 물품의 정보를 전달하고, 장치는 서버로부터 찾고자 하는 정보를 받아온다
3. 로봇 장치가 동작하여 책장의 후방에서 이동하며 물품을 탐색한다.
4. 찾고자 하는 도서 또는 물품을 찾으면, 해당 물품을 밀어 전면의 책받침 위로 이동시킨다.
5. 장치를 사용자가 수령할 수 있는 위치로 이동하여 사용자에게 책을 전달한다.

2. 과제의 필요성 및 독창성

1) 과제의 필요성

-시간과 노력의 절약

도서 검색 로봇은 사용자가 원하는 물건을 찾기 위해 소비되는 시간과 노력을 줄일 수 있다. 사용자가 어플을 통해 원하는 물품을 입력하기만 하면 책장에서 해당 물품을 찾아내고 사용자에게 전달하게 된다. 이를 통해 책장 탐색하는 과정을 거치지 않고도 물품을 손쉽게 찾을 수 있다.

-효율적인 공간 활용

책장은 일반적으로 사용자의 접근이 가능한 높이까지 물품을 보관하도록 설계되어 있다. 그러나 도서 검색 로봇을 사용하면 책장의 높은 위치나 물건의 위치가 정해지지 않은 상황에서도 물품을 찾을 수 있다. 이는 공간을 효율적으로 활용할 수 있도록 한다.

-데이터 분석

앱을 통해 입력된 정보를 기반으로 작동하며, 책장의 내부를 스캔하여 원하는 물품을 빠르게 찾아내고, 정확한 위치로 이동하여 해당 물품을 사용자에게 전달한다. 이를 통해 재고 관리나 물품 검색 작업을 보다 효율적이고 정확하게 처리할 수 있게 된다.

-접근성 향상

노약자나 장애인 등 책장을 이용하는 데 불편을 겪을 수 있는 사람들에게 도움을 줄 수 있다. 물품을 찾아주고 무거운 물건을 옮기는 작업을 대신해줄 수 있다. 이를 통해 삶의 질의 향상과 편리함을 제공할 수 있다.

-시간 제약

시간에 제한 없는 운영이 가능하다. 사용자가 인터넷에 접속할 수 있는 환경이라면 언제든지 조작할 수 있다. 이는 서비스 제공 시간을 확장하고 사용자의 편의성을 높여준다.

2) 과제의 독창성

현재 도서관에서 책을 검색, 정해진 위치에 놓이지 않은 책을 감지하는 로봇은 많이 있다. 주로 모바일 로봇의 형태를 가지고 있으며, 간혹 드론을 이용하기도 한다. 하지만 이들은 책을 직접적으로 다루지 않으며 해당 물품이 있는 장소를 관리자에게 안내, 관리자는 해당 자리의 물품을 꺼내주거나 오류를 수정하는 방식으로 운영되고 있다.

따라서 사용자는 로봇을 따라가거나, 로봇이 책을 찾으면 책장에서 책을 뽑으러 직접 이동해야 한다는 점에서는 기존의 방식과 차이가 없다. 로봇을 통해 검색 과정만을 자율화한 결과라고 볼 수 있다.

그림 1 LG 클로이 가이드 로봇

그림 2 RFID를 통해 선반을 탐색하는 과정

이는 물류 시스템에서도 마찬가지이다. 대표적인 예로 물류 자동화 선두 주자에 속하는 아마존의 물류 센터에서는 모바일 로봇을 이용한 시스템을 사용하고 있다. 로봇이 실어 나르고자 하는 물품이 있는 선반 앞에서 대기하면, 직원이 선반에서 해당 물건을 꺼내주는 방식, 또는 모바일 로봇이 선반 전체를 가져오면, 지정된 자리에 있는 로봇팔이 해당 물품을 꺼내서 다른 모바일 로봇에게 넘겨주는 방식이다.

그림 4 모바일 로봇을 이용한 물류 시스템

그림 3 로봇팔을 이용한 분류 과정

이렇듯, 아직 사람의 최종 역할이 필요한 부분이 존재하며, 이러한 부분까지 자동화하기 위하여 책장 또는 서랍에서 물건을 꺼내는 역할을 책장 스스로 할 수 있도록 하고자 하였다. 또한, 기존의 선반에서도 사용이 가능할 수 있도록 선반에 부착하여 설치할 수 있는 형태로 개발하였다.

3. 캡스톤 디자인1에서의 변경점

-도서 중심에서 도서, 물류 중심으로 변경

기존에는 도서관에서의 사용을 목표로 도서관 책장에 중점을 두었지만, 캡스톤을 진행할수록 책과 물품이 담긴 상자가 별 차이가 없음을 인지하고, 도서관도 하나의 물류 시스템이라는 결론에 도달, 주제를 도서 중심에서 물류 중심으로 변경, 해당 기술이 물류창고 등 다양한 환경에서도 사용이 가능하다는 점을 어필하고자 하였음.

-전면부 컨베이어벨트 제거

초기 구상에는 책을 밀어주는 후방에는 로봇 상자를, 전방에는 책을 받쳐주고 책을 사용자에게 전달할 컨베이어벨트까지 구상이 되어 있으나, 컨베이어벨트의 크기와 무게가 문제가 되었음. 컨베이어벨트를 전면부에 장착할 시, 무게중심을 맞추기 위해선 뒷부분의 무게 증가가 동시에 진행되어야 함. 그러나 하중이 증가할 시, 층 이동에 필요한 스텝 모터의 요구사양 또한 증가하여야 하며 이미 구현된 층 이동 장치를 전면 재구성할 필요가 있었음. 당시 전면부 제외 대부분 완성된 상태에서 재구성은 불가하다는 판단에 해당 컨베이어벨트 부분을 제거하자는 결론에 도달하였음.

-가드 구조체 제거

전면부 컨베이어벨트를 제거하면서 컨베이어벨트에 연동되어 동작하는 가드 구조체의 필요성을 따져본 결과, 가드 구조체의 목적인 책의 넘어짐 방지와 바코드 스캔 기능을 대체할 수 있어야 가드 구조체를 제거할 수 있다는 결론을 내리게 되었음. 이에 책이 일정한 방향으로 넘어질 수 있도록 하는 가드 역할의 경우 링크 구조체의 축과 링크 사이에 유격을 주면 링크 구조가 관절처럼 휘는 현상을 이용, 책이 무게중심에 따라 한 방향으로 넘어질 수 있도록 하여 대체하였다. 바코드 스캐닝 기능의 경우 바코드 리더기를 후방의 로봇 상자에 부착하여 로봇 상자와 함께 이동하면서 바코드를 읽어오도록 설계를 수정하였다. 가드 구조체의 역할을 후면부로 이동시키는 과정을 통해 전면부의 가드 구조체를 제거할 수 있었다.

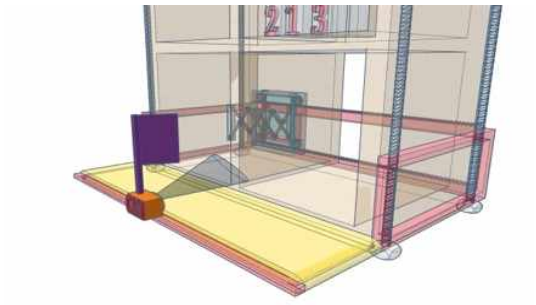


그림 5 기존 전면부 구상안

그림 6 최종 전면부 모습

-링크 구조 변화

기존 계획하였던 3절 링크 구조의 경우, 축에 연결되는 링크가 6개로 링크 하나의 두께를 1cm로 잡으면 링크 구조의 총 두께가 6cm까지 증가하게 되었다. 이렇게 되면 6cm 이하의 얇은 책이나 물품은 링크 구조를 통해 밀어줄 수 없게 되기 때문에, 2절 링크 구조로 변경하게 되었다. 2절 링크를 이용하면 4cm 두께 이상의 물품을 밀어줄 수 있어 기존의 3절 링크보다 더 다양한 물품을 밀어줄 수 있게 되었다.

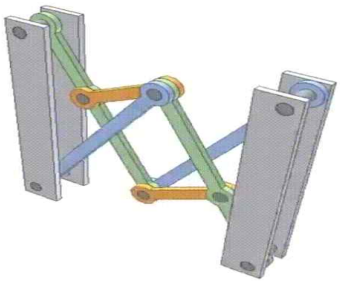


그림 7 기존 구상 3절 링크 구조

그림 8 변경 후 2절 링크 구조

4. 진행 과정

1) 주제 선정 및 아이디어 구상

주제 선정 과정을 통해 '도서 검색 로봇'을 주제로 선정하였다. 해당 주제를 선정하게 된 배경, 주제를 구현할 때 얻을 수 있는 효과 등에 대하여 논하였으며, 이후 해당 주제를 구현하기 위한 구체적인 논의를 진행했다. 도서 검색 로봇의 모델링을 제작하여 예상되는 문제점 등을 수정해가며 아이디어를 보완하였다.

2) 재료와 부품 선정

주제로 선정한 도서 검색 로봇의 구현에 필요한 부품을 파악하였다.

- 모터 제어 : 스텝 모터를 이용하여 프레임의 상하 이동과 로봇 상자의 좌우 이동을 구현한다. 좌우 이동의 경우 회전력만으로 제어가 가능하나, 상하 이동의 경우 정지 상태에서도 무게를 버티기 위해 모터에 지속적인 토크가 가해지며 모터에 부하가 발생한다. 이는 기기 고장의 원인이 될 수 있다. 이에 모터가 회전하지 않으면 스크류가 고정되어 정지 상태에서도 부하가 적다는 장점이 있는 볼 스크류 방식의 모터 제어 방식 채택.

- 볼스크류(Linear Motion), 모터 선정

지름 16인 볼스크류를 사용, 예상 하중 5kg, 예상 부하 0.2, 목표 이동속도 0.5cm/s일 때 모터 선정

g: $9.81 m/s^2$

예상 하중(m): $5kg$

예상 부하: $5kgf = 5 \times 9.81 = 49.05N$

예상 마찰계수: $\mu = 0.2$

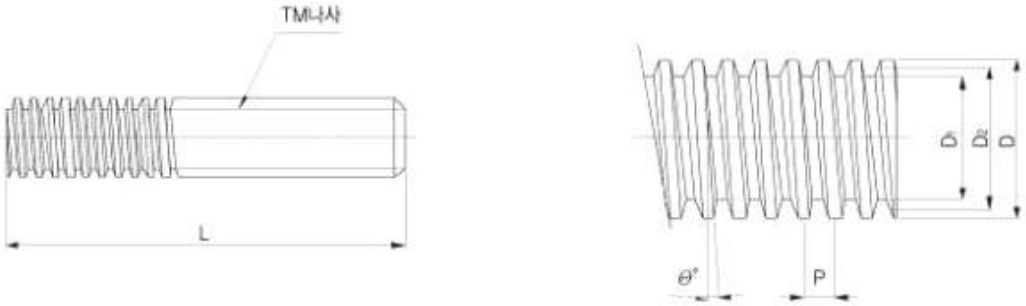
예상 이동속도: $S = 0.5cm/s = 0.5 \times 10^{-2} m/(\frac{1}{60}\min) = 0.3\, m/\min$

그림 9 볼스크류(TMR16)

조건

1) 안정계수: $f_s \geq 4$ (선반에 도서를 적재하는 과정 등에서 충격진동이 예상됨)
2) 내구성(너트 재질 BC6 기준): $pV < 3.6$ ($pV < 1.2$: 권장수치)

1) TM SCREW: TMR16-1000L(오른나사)



품번	PITCH	리드 각	외경	곡경	유효경	길이
	P	$\theta°$	D	D_1	D_2	L
TMR 16	3	3.77	16	12.5	14.5	1000, 1500, 2000, 3000

2) TM NUT: TTM16

단위 : mm

품번	PITCH	곡경	내경	유효경	D_3	D_4	L	L_1	PCD	B	L_2	a	동적허용추력kgf
	P	D	D_1	D_2									F0
TTM 16	3	16.5	13.5	14.5	28	51	35	6	38	6.6	14.5	1.5	640

1) 접촉면압(p):

$$p = \frac{P_F}{F_0} \times 9.8 = \frac{5}{640} \times 9.8 \approx 0.0766 N/mm^2$$

2) 미끄럼속도(V):

$$V = \frac{\pi D_2 n}{\cos(\theta^\circ) \times 10^3} = \frac{\pi \times 14.5 \times n}{\cos(3.77^\circ) \times 10^3} \approx 0.045652 n\,[m/\min],\ (n: \text{분당회전수}\,[\min^{-1}])$$

3) 이동속도(S):

$$n = \frac{S}{R \times 10^{-3}} = \frac{S}{P \times 10^{-3}} = \frac{S}{3 \times 10^{-3}} = \frac{1000}{3} S[m/\min]$$

4) 효율(η):

$$\eta = \frac{1 - \mu tan(\theta^\circ)}{1 + \mu/\tan(\theta^\circ)} = \frac{1 - 0.2 \times tan(3.77^\circ)}{1 + 0.2/\tan(3.77^\circ)} \approx 0.22$$

5) 발생 추력(F_a):

$$F_a = \frac{2\pi\eta T}{R \times 10^{-3}} = \frac{2\pi\eta T}{P \times 10^{-3}} = \frac{2\pi \times 0.22 \times T}{3 \times 10^{-3}} \approx 460.77\, T[N] \quad (T\text{의 단위: } [N \cdot m])$$

조건1 검증) $f_s = \frac{f_T F_0}{P_F} = \frac{1.0 \times 640}{5} = 128 \gg 4$ (f_T: 온도계수, 상온의 경우 1.0)

▶ 만족

조건2 검증) 계산3 & 예상 이동속도 고려

$n = \frac{1000}{3} \times 0.3 = 100$ (n : 분당회전수 $[\min^{-1}]$)

$V = 0.045652n = 0.045652 \times 100 = 4.5652$

$pV = 0.0766 \times 4.5652 \simeq 0.3497 < 1.2$

▶ 만족

모터 선정) 예상 부하의 30% 여유를 기준으로 선정

[조건1] 정격 속도

▶ n의 값을 고려하여 100RPM ~ 130RPM 정도로 선정

[조건2] 정격 토크

①. 부하(선반)에 의한 토크

$F_a = 460.77\,T > 49.05 \times 1.3 = 63.765[N]$

$\Leftrightarrow T > \frac{63.765}{460.77} = 0.138[N \cdot m]$

②. TM SCREW 가감속에 의한 토크

$I_{TM} = \frac{1}{2}MD^2 = \frac{1}{2}\sigma(\frac{\pi D^2 h}{4})D^2 = \frac{1}{8}\sigma\pi D^4 h = \frac{1}{8}(7810\frac{kg}{m^3})\pi(0.016)^4 = 2.009 \times 10^{-4}$

($M \simeq 1.57029kg$, SM20C 재질의 비중(7.81)은 ㈜한국특강의 MSDS(Material Safety Data Sheet)를 참고하였음)

I_{TM}이 $\Delta t = 0.1s$동안 100RPM을 도달하기 위한 토크는 $0.021[N \cdot m]$

→ 부하(선반)에 의한 토크에 비해 크기가 충분히 작으므로 무시

▶ 정격 토크가 $0.138[N \cdot m]$ 이상인 경우를 선정

[조건3] 모터의 출력

$P = T\omega = 0.138 \times \frac{100 \times 2\pi}{60s} = 1.445\,W$

▶ 정격 출력이 1.445W급 이상

위 과정을 통해 kh56km2 유니폴라 스텝 모터를 선정하게 되었다.

바이폴라 스텝 모터가 아닌 유니폴라 스텝 모터를 선정하게 된 이유는 로봇이 이동하는 속도는 빠를수록 좋다고 판단하여 고속에서 높은 토크가 나오는 유니폴라 방식의 스텝 모터를 선정하게 되었다.

그림 15 kh56km2 스텝 모터

-링크 구조

2절 링크 구조를 통해 책장 후면의 로봇 상자에서 책을 밀어 전면부의 책받침 부분까지 이동해야 만큼, 2절 링크를 이용하여 충분한 거리를 밀어줄 수 있도록 설계하였다. 링크 구조에는 물품을 밀어줄 만큼의 힘이 필요하여 스텝 모터보다는 DC 모터가 적합하기에 감속비 1/100의 4kg/cm의 DC모터를 선정하였다. DC 모터를 정확한 위치를 지정하여 동작시키기 위해 PD 제어기를 이용했다. 이를 통해 링크 구조가 책장의 끝단까지 이동 후 다시 돌아오도록 제작할 수 있었다.

그림 16 링크 구조

-바코드

시중의 바코드 모듈의 경우 USB를 이용하는 방식을 사용하고 있었으나, 본 작품에서는 아두이노를 사용할 예정이었기에 사용에 어려움이 있었다. 이에 시리얼 통신을 지원하는 GM-65 바코드 모듈을 선정하게 되었다. 통신에 있어 TX, RX 핀을 이용하는 모듈이었으나, esp 8266 칩셋 아두이노 보드에서는 와이파이 이용 시 TX, RX 핀을 사용할 때 와이파이를 이용하는데 제약이 있어 esp 32 아두이노 보드로 교체하여 작업을 진행했다.

그림 17 GM-65 바코드 모듈

3) 하드웨어 제작

선정한 부품을 이용하여 작품을 제작하였다. 제작 과정에서 완제품이 없어 구매에 어려움이 없는 경우 3D 프린터를 이용하여 제작했다. 초기 구상에서 프레임의 전,후방 길이와 높이 등은 무게중심을 맞추는 과정에서 조정되었다.

3D 프린터를 이용하여 프레임과 볼스크류를 연결하기 위한 브라켓, 스텝 모터를 고정하기 위한 모터 상자, 볼스크류의 떨림을 잡기 위한 서포트 유닛 등을 제작했다.

4) 소프트웨어 제작

해당 작품이 무선으로 작동되기 위해 인터넷 서버 통신을 이용하여 동작할 수 있도록 제작하였다. 소프트웨어 부분은 앱, 앱과 아두이노의 통신을 중개할 firebase, 아두이노 3곳으로 진행되었다.

-앱

사용자가 도서관에서 책을 검색하면 해당 책에 대한 정보를 표시하며, 대출 버튼을 누르면 firebase에 찾고자 하는 책의 정보를 제공한다.

-firebase

앱으로부터 전달받은 정보를 arduino의 각 함수가 사용하기 위한 데이터로 분리, 아두이노로부터 변경된 데이터값을 앱에게 전달하는 등의 상호작용을 위한 매개채로 활용했다.

-아두이노

아두이노는 firebase로부터 입력받은 값을 기반으로 동작하며, 해당 동작을 완료할 시 firebase에 저장되어 있는 값을 초기화시키는 동작을 하도록 제작되었다. 층을 이동하는 스텝 모터 4개를 작동시키는 아두이노와 로봇 상자를 동작시키는 아두이노, 총 2개의 아두이노를 이용하여 제작되었다.

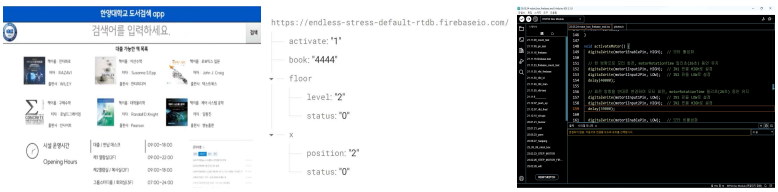


그림 18
앱, firebase, 아두이노
개발 환경

[firebase로부터 floor/level 값을 받아와 층을 이동시키는 아두이노 코드]

```
#include <Arduino.h>
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>

//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

#include <AccelStepper.h>

#define WIFI_SSID "stress" //wifi id
#define WIFI_PASWORD "stopstop" //wifi pw

// Insert Firebase project API Key
#define API_KEY "AIzaSyBHircR-AiLzAB9NqoWrZKe6fOHUbIHlN4"

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL "https://endless-stress-default-rtdb.firebaseio.com/"

// Insert RTDB AUTH the RTDB URL */
#define DATABASE_AUTH "zryLInlDYv9qeJ6FPFR047XUYyyUh7Xq4qemV168"

//Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;
String floorlevel;
String status;
bool signupOK = false;

AccelStepper stepper(1, 12, 13);

void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASWORD);
  Serial.println("wifi connecting");

  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
```

```
    delay(500);
  }
  Serial.println();
  Serial.println("WIFI connected : ");
  Serial.println(WiFi.localIP());

  /* Assign the api key (required) */
  config.api_key = API_KEY;

  /* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL;

  /* Sign up */
  if (Firebase.signUp(&config, &auth, "", "")){
    Serial.println("ok");
    signupOK = true;
  }
  else{
    Serial.printf("%s₩n", config.signer.signupError.message.c_str());
  }

  /* Assign the callback function for the long running token generation task */
  config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  stepper.setMaxSpeed(30720);
  stepper.setAcceleration(30720);
}

void loop() {
  int position;

  if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 2000 || sendDataPrevMillis == 0)) {
    sendDataPrevMillis = millis();
    if (Firebase.RTDB.getString(&fbdo, "/floor/level")) {
      if (fbdo.dataType() == "string") {
        floorlevel = fbdo.stringData();
        Serial.print("ordered floor level : ");
        Serial.println(floorlevel);
        if(floorlevel == "1"){
          //firebase 1층 입력 시
          position = 0;
          if (stepper.currentPosition() == position){
            Firebase.RTDB.setString(&fbdo, "floor/status", "0");
          }
          if (stepper.currentPosition() != position){
```

```
            Firebase.RTDB.setString(&fbdo, "floor/status", "1");
            stepper.moveTo(position);
            stepper.runToPosition();
            Serial.println("Move To Floor 1");
          }
        }
        if(floorlevel == "2"){
          //firebase 2층 입력 시
          position = 300000*3.4;
          if (stepper.currentPosition() == position){
            Firebase.RTDB.setString(&fbdo, "floor/status", "0");
          }
          if (stepper.currentPosition() != position){
            Firebase.RTDB.setString(&fbdo, "floor/status", "1");
            stepper.moveTo(position);
            stepper.runToPosition();
            Serial.println("Move To Floor 2");
          }
        }
        if(floorlevel == "3"){
          //firebase 3층 입력 시
          position = 300000*3.4*2;
          if (stepper.currentPosition() == position){
            Firebase.RTDB.setString(&fbdo, "floor/status", "0");
          }
          if (stepper.currentPosition() != position){
            Firebase.RTDB.setString(&fbdo, "floor/status", "1");
            stepper.moveTo(position);
            stepper.runToPosition();
            Serial.println("Move To Floor 3");
          }
        }
      }
    }
    else {
      Serial.println(fbdo.errorReason());
    }
  }
}
```

[서버에서 받은 책 데이터를 찾아주고 firebase 값을 초기화시키는 아두이노 코드]

```
#include <Arduino.h>
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>

//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

#include <AccelStepper.h>

#define WIFI_SSID "stress" //wifi id
#define WIFI_PASWORD "stopstop" //wifi pw

// Insert Firebase project API Key
#define API_KEY "AIzaSyBHircR-AiLzAB9NqoWrZKe6fOHUbIHlN4"

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL "https://endless-stress-default-rtdb.firebaseio.com/"

// Insert RTDB AUTH the RTDB URL */
#define DATABASE_AUTH "zryLInlDYv9qeJ6FPFR047XUYyyUh7Xq4qemV168"

//Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;
String x;
String status;
String readbarcode;
String activate;
String level;
String barcodeDatabase[1];

bool signupOK = false;

AccelStepper stepper(8, 23, 13, 5, 12);

//바코드 핀
#define RXD2 25
```

```
#define TXD2 26

//L298N DC 모터 핀
const int motor1EnablePin = 14;  //Enable pin (ENA)
const int motor1Input1Pin = 27;  //IN1
const int motor1Input2Pin = 16;  //IN2

bool barcodeScannerEnabled = true; // 바코드 스캐너 활성화 상태를 추적하는 변수

void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFI_SSID, WIFI_PASWORD);
  Serial.println("wifi connecting");

  while (WiFi.status() != WL_CONNECTED){
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.println("WIFI connected : ");
  Serial.println(WiFi.localIP());

  /* Assign the api key (required) */
  config.api_key = API_KEY;

  /* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL;

  /* Sign up */
  if (Firebase.signUp(&config, &auth, "", "")){
    Serial.println("ok");
    signupOK = true;
  }
  else{
    Serial.printf("%s₩n", config.signer.signupError.message.c_str());
  }

  /* Assign the callback function for the long running token generation task */
  config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  stepper.setMaxSpeed(2000);
  stepper.setAcceleration(2000);

  Serial2.begin(9600, SERIAL_8N1, RXD2, TXD2);
  Serial.println("Serial Txd is on pin: " + String(TX));
```

```
  Serial.println("Serial Rxd is on pin: " + String(RX));

  pinMode(motor1EnablePin, OUTPUT);
  pinMode(motor1Input1Pin, OUTPUT);
  pinMode(motor1Input2Pin, OUTPUT);

  digitalWrite(motor1EnablePin, LOW);
}

void fin() {
  while (Serial2.available()) {
    String barcode_Data = Serial2.readStringUntil('₩n');
    Serial.print("read: ");
    Serial.println(barcode_Data);

    String processedBarcode = removeSpaces(barcode_Data);

    if (barcodeScannerEnabled && checkBarcode(processedBarcode)) {
      Serial.println("바코드 일치!");
      activateMotor();
      reset();
    }
  }
  delay(20);
}
bool checkBarcode(String barcode) {
  if (Firebase.RTDB.getString(&fbdo, "book")) {
      if (fbdo.dataType() == "string") {
        readbarcode = fbdo.stringData();
        barcodeDatabase[0]=readbarcode;
      }
  }
  for (const auto& dbBarcode : barcodeDatabase) {
    if (barcode.equals(dbBarcode)) {
      return true;
    }
  }
  return false;
}
String removeSpaces(String barcode) {
  barcode.trim(); // 공백 제거
  barcode.replace(" ", ""); // 중간 공백 제거
  return barcode;
}

void activateMotor() {
  digitalWrite(motor1EnablePin, HIGH);  // 모터 활성화
  // 한 방향으로 모터 회전, motorRotationTime 밀리초(26초) 동안 유지
```

```
  digitalWrite(motor1Input1Pin, HIGH);  // IN1 핀을 HIGH로 설정
  digitalWrite(motor1Input2Pin, LOW);   // IN2 핀을 LOW로 설정
  delay(40000);
  // 회전 방향을 반대로 변경하여 모터 회전, motorRotationTime 밀리초(26초) 동안 유지
  digitalWrite(motor1Input1Pin, LOW);   // IN1 핀을 LOW로 설정
  digitalWrite(motor1Input2Pin, HIGH);  // IN2 핀을 HIGH로 설정
  delay(39000);
  digitalWrite(motor1EnablePin, LOW);   // 모터 비활성화
}

void reset(){
  Firebase.RTDB.setString(&fbdo, "activate", "0");
  Firebase.RTDB.setString(&fbdo, "floor/level", "1");
  Firebase.RTDB.setString(&fbdo, "x/position", "1");
  Firebase.RTDB.setString(&fbdo, "book", "0");
}

void loop() {
  int position;
  int i=1;
  if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 2000 || sendDataPrevMillis == 0)) {
    sendDataPrevMillis = millis();
    if (Firebase.RTDB.getString(&fbdo, "activate")) {
      if (fbdo.dataType() == "string") {
        activate = fbdo.stringData();
      }
    }
    if (Firebase.RTDB.getString(&fbdo, "floor/status")) {
      if (fbdo.dataType() == "string") {
        status = fbdo.stringData();
      }
    }
    if (Firebase.RTDB.getString(&fbdo, "/x/position")) {
      if (fbdo.dataType() == "string") {
        x = fbdo.stringData();
        if(x == "1"){
          //firebase 중앙으로 이동 시
          position = 0;
          if (stepper.currentPosition() != position){
            stepper.moveTo(position);
            stepper.runToPosition();
            Serial.println("Move To Mid");
          }
        }
        if(x == "0" && status == "0"){
          //firebase 왼쪽으로 이동 시
          Serial.println("x=0, move");
          while (activate != "0") {
```

```
            if(Firebase.RTDB.getString(&fbdo, "activate")) {
              if (fbdo.dataType() == "string") {
                activate = fbdo.stringData();
              }
            }
            position = -256*i;
            i++;
            if (stepper.currentPosition() != position){
              stepper.moveTo(position);
              stepper.runToPosition();
              Serial.println("Move To left");
              }
              fin();
          }
        }
        if(x == "2" && status == "0"){
          //firebase 오른쪽으로 이동 시
          Serial.println("x=0, move");
          while (activate != "0") {
            if(Firebase.RTDB.getString(&fbdo, "activate")) {
              if (fbdo.dataType() == "string") {
                activate = fbdo.stringData();
                Serial.println("activate");
                Serial.println(activate);
              }
            }
            position = 256*i;
            i++;
            if (stepper.currentPosition() != position){
              stepper.moveTo(position);
              stepper.runToPosition();
              Serial.println("Move To Right");
            }
            fin();
          }
        }
      }
    }
    else {
      Serial.println(fbdo.errorReason());
    }
  }
}
```

5. 과제 결과

앱을 이용하여 원하는 책을 검색 후 대출 버튼을 누르면 앱이 서버로 찾고자 하는 책에 대한 정보가 전달된다. 아두이노가 서버로부터 정보를 받아오면 장치가 동작한다. 해당 층에 이동 후, 로봇 상자가 좌, 우로 이동하며 해당 도서를 검색한다. 해당하는 책이 감지되면 링크 구조를 이용하여 책을 밀어 책받침 위로 이동시킨다. 이후 서버의 값을 초기화하고 장치 또한 초기 위치로 되돌아간다.

6. 토의 및 개선 방향

캡스톤 디자인을 통해 구상했던 작품을 실체화하는 과정에서 발생하는 문제를 해결해나가는 과정에서 문제 해결 능력을 키울 수 있었으며, 작품을 직접 만드는 성취감을 얻을 수 있었다.

도서 검색 로봇은 도서관에서 활용될 목적으로 만들어졌으나, 작품을 진행할수록 이러한 장치는 물류 창고와 같은 장소에서 요구하는 공간의 효율성을 극대화, 시간 제약 없이 운영이 가능하다는 점에서 기존의 물류 자동화 능력을 한층 끌어올릴 수 있는 작품이라는 점을 체감하게 되었다. 그러나 해당 제품이 상용화 되기 전에 보완해야 할 점 또한 존재하였으며 해당 사항은 다음과 같다.

-속도 개선

해당 작품이 2층 좌측 칸의 중앙 부분의 도서를 찾는 동작이 완료되기까지 3분가량의 시간이 소요된다. 현재 상황에서도 속도를 높일 수 있으나 스텝 모터 동기 운전 이탈의 위험이 존재한다. 해당 문제 해결을 위해선 더 높은 사양의 모터가 필요하다.

-컨베이어벨트

컨베이어벨트를 이용하여 책장에서 뽑은 도서를 모바일 로봇이나 메인 컨베이어벨트에 전달하여 사용자에게 전달하는 초기 구상과 달리, 책장에서 도서를 뽑는 과정까지만 구현되었다. 앞서 말했던 스텝 모터의 업그레이드가 선행된 후 컨베이어벨트를 추가되었으면 하는 아쉬움이 있다.

-링크 구조

링크 구조의 경우 3D프린터로 출력하여 사용하다 보니 테스트 가동 중 축이 하중을 버티지 못하고 부러지는 등의 내구성에 문제가 있었다. 링크 구조를 PLA 소재가 아닌 알루미늄 소재의 금속으로 제작하였으면 더 강한 내구성을 갖추게 될 것으로 예상된다.

7. 참고문헌 및 부록

-도서관에서 로봇 활용에 대한 사례 연구, 김경철 (2020)
-사물인터넷(IoT) 기반의 대학 도서관 서비스에 관한 연구, 노동조, 손태익, (2016)
-베이징 긱플러스 테크놀러지 씨오. 로봇, 운반시스템 및 방법. 특허 출원번호 10-2020-0016303(출원일 2018.07.25, 등록일 2022.02.10)
-(주)오트. 로봇 도서관 시스템(System for robotic library). 특허 출원번호 10-2021-0061897(출원일 2021.05.13, 등록일 2022.11.22.)
-YTN (2023.01.22). 도서관에 웬 로봇?!...와우, 열 일 하네!
(https://www.ytn.co.kr/_ln/0105_202301220543400486)
-전자신문(2023.01.19). LG'로봇 사서' 책 운반 검색 척척(https://www.etnews.com/20230119000141)
-Digital Today(2018.06.08). 아마존, 로봇 10만대 준비하고도 작업자(사람) 대체 못했다...왜?
(https://www.digitaltoday.co.kr/news/articleView.html?idxno=200427)
-로봇신문(2022.11.11). 아마존, 물류창고용 로봇 팔 '스패로우' 공개
(http://m.irobotnews.com/news/articleView.html?idxno=30029)