**종합설계 프로젝트 보고서**

**제목: 스마트 도서관리 시스템**

**작성자 : 유종민**

**지도교수 : 구임수**

**차 례**

I. 프로젝트 개요

II. 시스템 구성

III. 각 모듈별 동작 원리

IV. 모듈별 설계

V. 전체 시스템 설계

VI. 제작내용 (회로도, 소스코드 등 첨부)

VII. 결과물 설명 (결과물 관련 이미지(사진) 첨부)

VIII. 프로젝트 수행 결과 분석

1. 재학중 취득한 기초지식의 활용 내용

2. 재학중 취득한 실험지식의 활용 내용

3. 본 프로젝트 수행과정에서의 설계 능력 향상 내용

4. 본 프로젝트 수행과정에서의 문제 해결 내용

5. 본 프로젝트 수행과정에서의 실무 능력 향상 내용

6. 본 프로젝트 수행과정에서의 팀원간 협동 내용

7. 개발된 결과물에 대한 전시 방법 계획

< 참고 문헌 >

< 종합설계 프로젝트 수행 후기 >

**I. 프로젝트 개요**

본 프로젝트는 SLAM(Simultaneous Localization and Mapping) 기반의 자율주행 로봇 스캔 시스템을 구현하는 것을 목표로 하였다. 이 시스템은 지도 생성을 위한 LIDAR(RPLIDAR A1M8) 센서, 이동 거리를 추정하는 Encoder 등을 활용하여 로봇이 스스로 환경을 인식하고 주행할 수 있도록 설계되었다.

특히 ROS Noetic 환경에서 Hector SLAM 알고리즘을 이용하여 지도 작성 및 위치 추정을 수행하고, AMCL 기반의 자율 주행 기능까지 통합하여 실시간 주행 및 위치 보정을 가능하게 하였다. 본 프로젝트는 제한된 예산, 팀 협력 , 실무적 기술 요소 등을 고려하여 학부 수준에서 산업적 구조와 유사한 실습 환경을 구현하였다.

도서 스캔 시스템은 도서관 내 책장을 대상으로 자동으로 이동 및 바코드 스캔을 수행하는 바코드 스캐너와 라즈베리파이,주행 제어를 위한 아두이노 우노, 이를 모니터링하는 웹 기반 인터페이스로 구성된다. 사용자는 웹에서 스캔시작 버튼 클릭만으로 책장 스캔을 지시할 수 있으며, 결과는 실시간 웹서버로 확인 가능하다.

**II. 시스템 구성**

**▶ 환경 (Platform)**

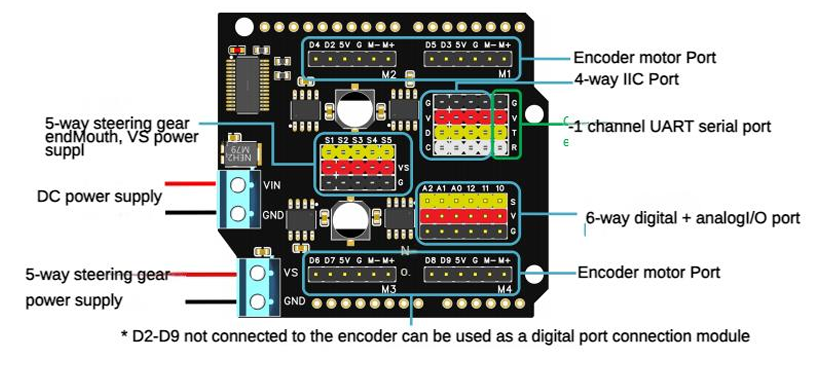
* 운영체제: Ubuntu 20.04 LTS (Raspberry Pi 4 기반)
* 프레임워크: ROS Noetic

**▶ 하드웨어 (Hardware)**

* Raspberry Pi 4: ROS Master, LIDAR 및 고급 연산 처리, 바코드 스캔 관련 명령 수신 및 처리, 로봇 하드웨어 제어
* Arduino UNO: 모터 제어 및 엔코더 데이터 수집
* RPLIDAR A1M8: 360° 2D 거리 측정 LIDAR 센서



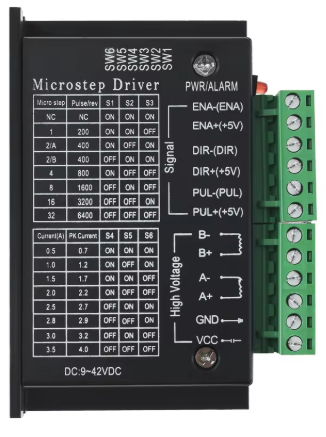
* YFROBOT IIC-Quad Motor Driver: I2C 기반 모터 제어



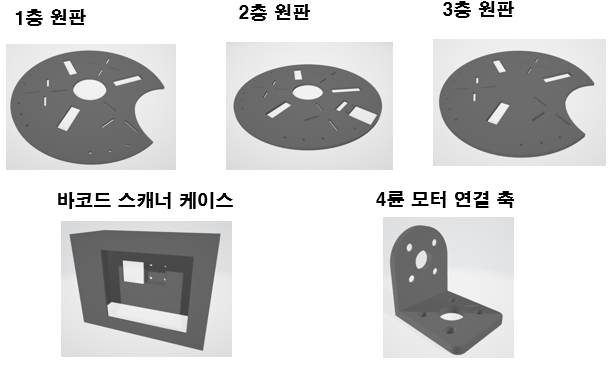
* DC 엔코더 모터 4개 (1:75 기어비, 360 CPR)



* 바코드 스캐너(TDS-9200): 도서의 바코드 스캔
* 
* 옴니휠: 바퀴
* 
* 리튬 이온 배터리 및 배터리홀더: 전원공급
* 리니어 슬라이더: 층별 스캔
* 
* TB6600 슬라이더 모터드라이버: 슬라이더 모터 전력 증대



* 자동차 차체: 3D프린터로 각 층별 제작

****

**▶ 소프트웨어 (라이다)**

* ROS 패키지: rplidar\_ros, hector\_slam, amcl, map\_server, move\_base
* 사용자 정의 노드: odom\_from\_serial.py (Python 기반 엔코더 연동)
* 통신: USB-UART 직렬통신, I2C 제어 버스

**▶ 소프트웨어 (서버)**

* Flask 웹 서버: 사용자 인터페이스 제공, 명령 수신 및 전송, 실시간 상태 업데이트
* Socket.IO: 실시간 양방향 통신 구현 (사용자 ↔ 서버 ↔ 라즈베리파이)
* 도서 정보 API: 도서 정보 획득

**III. 각 모듈별 동작 원리**

▶ LIDAR (RPLIDAR A1M8)

* 초당 5.5Hz ~ 10Hz 속도로 회전하며 거리 측정 데이터를 획득
* /scan 토픽으로 ROS에 퍼블리시
* Hector SLAM에 입력되어 실시간 2D 지도 작성에 활용됨

▶ Motor + Encoder + Driver

* 각 바퀴 회전에 따라 Encoder가 A 채널 펄스를 발생시킴 (360 CPR)
* 아두이노에서 시리얼을 통해 ROS로 속도/거리 정보 전달
* I2C YFROBOT 모터드라이버는 I2C를 통해 PWM 제어 신호로 엔코더모터 회전 조절
* 더 정확한 주행을 위해 PID제어 주행 알고리즘을 구현 Kp=1 Ki=0.5 Kd=0.1 직접 모터 성능을 확인하고 튜닝 계수값을 정함.

▶ 바코드 스캔

* 사용자 명령 입력
  + Flask 웹 서버에서 스캔시작 버튼 클릭 시 Socket.IO를 통해 명령이 라즈베리파이로 전송됨
* 로봇 제어
  + 라즈베리파이는 수신된 명령을 기반으로 아두이노에 주행 명령 전달
  + 아두이노는 지정된 위치로 로봇을 주행 시킴
* 바코드 인식 및 처리
  + 위치 도달 후 바코드 스캐너가 활성화되며, 스캔된 책바코드는 라즈베리파이에 입력됨
* 결과 전송 및 표시
  + 추출된 도서 정보를 Flask 서버로 전송하고, 서버는 Socket.IO를 통해 웹페이지에 실시간 반영함
  + 도서관에서 정한 책의 순서 위치를 바탕으로 책장상태를 컬러 시각화하여 도서관리자는 현재 책장에 있는 책의 상태를 확인할 수 있음

**IV. 모듈별 설계**

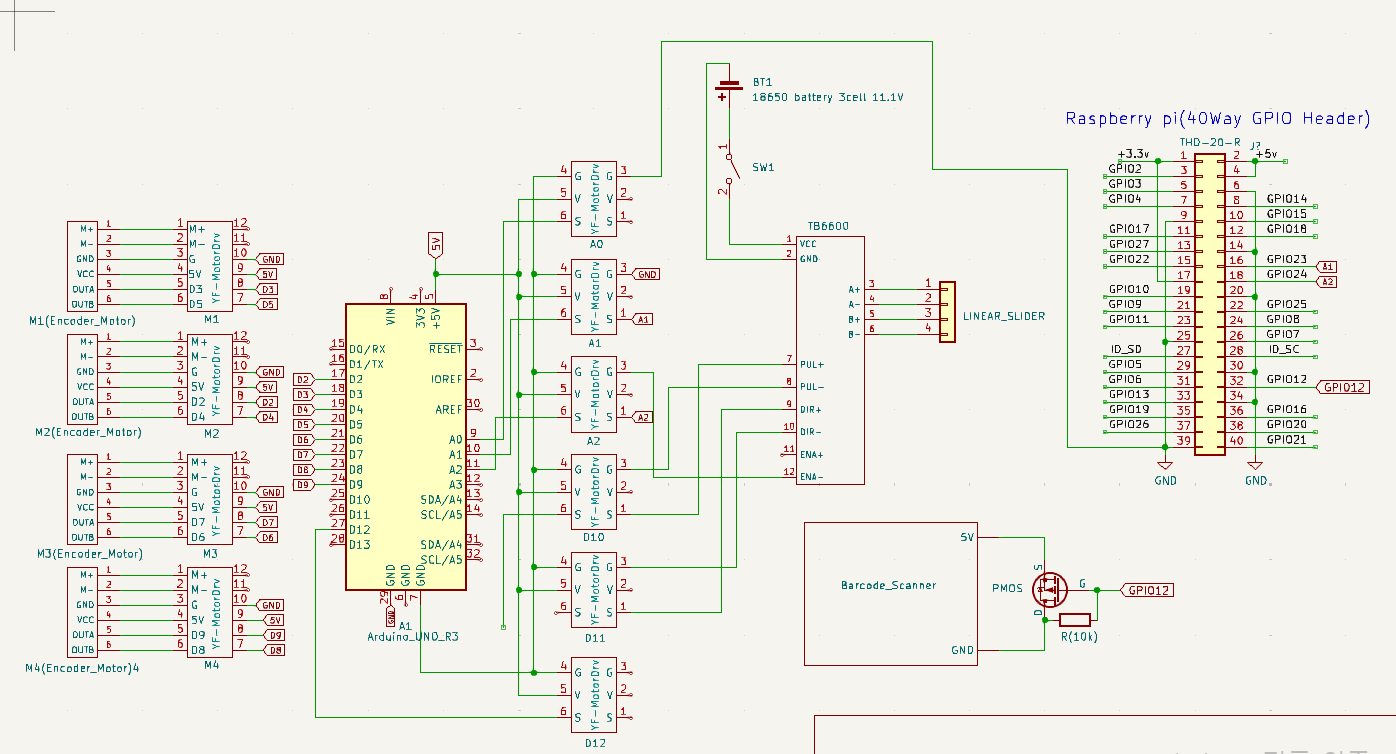
* LIDAR 설치: 장애물 감지가 잘 되는 전면 상단 중앙에 고정
* Encoder 배선: D2, D3, D7, D9 핀에 각각 A채널 연결, 인터럽트 기반 감지
* 모터 드라이버 연결: I2C 통신으로 Raspberry Pi와 연결, PWM 주파수 100Hz 설정
* 바코드 스캐너: 고속 스캔을 위해 트렌지스터를 추가하여 라즈베리파이에서 속도를 조절함

**V. 전체 시스템 설계**

* TF Tree 구성: map → odom → base\_link → laser
* 통합 launch 파일: 각 노드가 순차적으로 실행될 수 있도록 통합 구성 (hector\_slam.launch, move\_base.launch, odom\_from\_serial.py 등)
* RViz 시각화: 센서 및 로봇 상태를 실시간으로 모니터링 가능, 맵 생성과 위치 정보 시각적 확인
* 서버 구성: [사용자 버튼 클릭] -> [Flask 서버: Socket.IO emit] -> [라즈베리파이: 명령 수신 및 이동 명령 전송] -> [아두이노: 로봇 이동 → 도착 신호] -> [라즈베리파이: 바코드 스캐너 활성화] -> [도서 바코드 인식] -> [외부 API/DB로 책 정보 요청] -> [책 정보 → Flask로 전송] -> [Flask 서버: 실시간 웹소켓 emit] -> [웹 페이지: 책 정보 표시]

**VI. 제작 내용 (회로도, 소스코드 등 첨부)**

**▶** 회로도



* Raspberry Pi ↔ Arduino: UART 직렬통신 (TX/RX)
* Arduino ↔ Motor Driver Shiled: I2C 제어
* Arduino ↔ Encoder: 디지털 인터럽트 핀 (D2, D3, D7, D9)
* 각 모듈별 전원은 분리된 배터리 회로에서 공급 (18650 11.1V 3S 2200mAh)

▶ 소스코드 요약

1. 아두이노 PID 주행 및 슬라이더 제어 코드 소스코드

// 필요한 라이브러리 포함

#include <Wire.h> // I2C 통신용

#include <MotorDriver.h> // YFROBOT IIC RZ 모터 드라이버용 라이브러리

#include <PID\_v1.h> // PID 제어 라이브러리

#include <EnableInterrupt.h> // 외부 인터럽트용 라이브러리

// 모터 드라이버 객체 생성

#define MOTOR\_TYPE YF\_IIC\_RZ

MotorDriver motorDriver(MOTOR\_TYPE);

// 모터 ID 정의

#define M1 1

#define M2 2

#define M3 3

#define M4 4

// PID 제어용 기본 속도와 보정 주기

#define BASE\_SPEED 600

#define CORRECTION\_INTERVAL 100 // PID 보정 주기(ms)

// 엔코더 A 채널 핀

#define ENCODER\_M1\_A 3

#define ENCODER\_M2\_A 2

#define ENCODER\_M3\_A 7

#define ENCODER\_M4\_A 9

// 슬라이더용 스텝모터 핀

#define STEP\_PIN 10

#define DIR\_PIN 11

#define SLIDER\_TRIGGER\_PIN 12 // 토글 트리거 입력 (버튼 또는 신호)

// 슬라이더 이동 거리 정의

#define STEPS\_PER\_MM 400L // mm당 스텝 수

#define HEIGHT\_2F\_MM 300 // 2층 높이(mm)

#define HEIGHT\_2F\_STEPS (HEIGHT\_2F\_MM \* STEPS\_PER\_MM)

// 라즈베리파이에서 슬라이더 제어를 위한 GPIO 입력 핀

#define PI\_SIGNAL\_UP\_PIN A2 // 2층 이동 명령

#define PI\_SIGNAL\_DOWN\_PIN A1 // 1층 이동 명령

// 엔코더 값 저장용 변수

volatile long encM1 = 0, encM2 = 0, encM3 = 0, encM4 = 0;

// 슬라이더 위치 상태 및 동작 플래그

long current\_steps = 0; // 현재 위치 (스텝 수 기준)

bool isMoving = false; // 슬라이더 동작 중 여부

bool lastTriggerState = LOW; // 트리거 핀의 이전 상태 저장

// PID 제어 변수 (M1-M2, M3-M4 쌍으로 제어)

double diff12\_input = 0, output12 = 0, setpoint12 = 0;

double diff34\_input = 0, output34 = 0, setpoint34 = 0;

// PID 컨트롤러 객체 생성

PID pid12(&diff12\_input, &output12, &setpoint12, 1.0, 0.5, 0.1, DIRECT);

PID pid34(&diff34\_input, &output34, &setpoint34, 1.0, 0.5, 0.1, DIRECT);

// 엔코더 인터럽트 핸들러 (RISING 에지에서 카운터 증가)

void encoderM1() { encM1++; }

void encoderM2() { encM2++; }

void encoderM3() { encM3++; }

void encoderM4() { encM4++; }

void setup() {

Serial.begin(115200);

Wire.begin(); // I2C 초기화

motorDriver.begin(); // 모터 드라이버 초기화

motorDriver.setPWMFreq(100); // PWM 주파수 설정

// 엔코더 핀 입력 모드 설정 및 인터럽트 등록

pinMode(ENCODER\_M1\_A, INPUT);

pinMode(ENCODER\_M2\_A, INPUT);

pinMode(ENCODER\_M3\_A, INPUT);

pinMode(ENCODER\_M4\_A, INPUT);

enableInterrupt(ENCODER\_M1\_A, encoderM1, RISING);

enableInterrupt(ENCODER\_M2\_A, encoderM2, RISING);

enableInterrupt(ENCODER\_M3\_A, encoderM3, RISING);

enableInterrupt(ENCODER\_M4\_A, encoderM4, RISING);

// 슬라이더 핀 설정

pinMode(STEP\_PIN, OUTPUT);

pinMode(DIR\_PIN, OUTPUT);

pinMode(SLIDER\_TRIGGER\_PIN, INPUT);

// 라즈베리파이 GPIO 입력 설정

pinMode(PI\_SIGNAL\_UP\_PIN, INPUT);

pinMode(PI\_SIGNAL\_DOWN\_PIN, INPUT);

// PID 제어기 설정

pid12.SetMode(AUTOMATIC);

pid34.SetMode(AUTOMATIC);

pid12.SetOutputLimits(-100, 100);

pid34.SetOutputLimits(-100, 100);

delay(1000);

Serial.println("🔍 시스템 준비 완료");

}

void loop() {

// 2층 이동 명령 수신

if (!isMoving && digitalRead(PI\_SIGNAL\_UP\_PIN) == HIGH) {

Serial.println("📡 라즈베리파이 → 2층 이동 명령 수신");

stopRobot(); // 주행 중지

moveToWithLog("1층", "2층", HEIGHT\_2F\_STEPS);

return;

}

// 1층 이동 명령 수신

if (!isMoving && digitalRead(PI\_SIGNAL\_DOWN\_PIN) == HIGH) {

Serial.println("📡 라즈베리파이 → 1층 이동 명령 수신");

stopRobot(); // 주행 중지

moveToWithLog("2층", "1층", 0);

return;

}

// 슬라이더 트리거 버튼 처리 (HIGH 에지 감지)

bool triggerState = digitalRead(SLIDER\_TRIGGER\_PIN);

if (!isMoving && triggerState == HIGH && lastTriggerState == LOW) {

Serial.println("📥 슬라이더 트리거 감지됨 → 현재 방식 유지");

stopRobot(); // 주행 중지

delay(500); // 디바운싱

}

lastTriggerState = triggerState;

// 슬라이더 동작 중이 아니면 PID 주행

if (!isMoving) {

moveForwardWithPID(CORRECTION\_INTERVAL);

} else {

stopRobot();

}

}

// 슬라이더 이동 함수 + 로그 출력

void moveToWithLog(String from, String to, long target\_steps) {

isMoving = true;

long steps\_to\_move = target\_steps - current\_steps;

int dir;

Serial.println("====================================");

Serial.print("📍 From: "); Serial.println(from);

Serial.print("🎯 To: "); Serial.println(to);

Serial.print("📏 current\_steps: "); Serial.println(current\_steps);

Serial.print("📌 target\_steps: "); Serial.println(target\_steps);

Serial.print("➡ 이동 스텝 수: "); Serial.println(steps\_to\_move);

if (steps\_to\_move > 0) {

dir = HIGH; // 위로 이동

Serial.println("↑ 위로 이동");

} else {

dir = LOW; // 아래로 이동

Serial.println("↓ 아래로 이동");

steps\_to\_move = -steps\_to\_move;

}

digitalWrite(DIR\_PIN, dir);

Serial.println("🚀 슬라이더 이동 시작");

// 슬라이더모터 구동 루프

for (long i = 0; i < steps\_to\_move; i++) {

digitalWrite(STEP\_PIN, HIGH);

delayMicroseconds(100);

digitalWrite(STEP\_PIN, LOW);

delayMicroseconds(100);

}

current\_steps = target\_steps;

isMoving = false;

Serial.print("✅ 이동 완료 → current\_steps = ");

Serial.println(current\_steps);

}

// PID 기반 직진 주행

void moveForwardWithPID(int interval) {

// 엔코더 값 초기화

encM1 = encM2 = encM3 = encM4 = 0;

// 초기 속도 설정

int s1 = -BASE\_SPEED;

int s2 = -BASE\_SPEED;

int s3 = BASE\_SPEED + 110;

int s4 = BASE\_SPEED + 110;

// PID 입력값 계산

diff12\_input = encM1 - encM2;

diff34\_input = encM3 - encM4;

// PID 보정 수행

pid12.Compute();

pid34.Compute();

// 보정값 적용

s1 += output12;

s2 -= output12;

s3 += output34;

s4 -= output34;

// 모터 속도 설정

motorDriver.setSingleMotor(M1, s1);

motorDriver.setSingleMotor(M2, s2);

motorDriver.setSingleMotor(M3, s3);

motorDriver.setSingleMotor(M4, s4);

// 디버깅 출력

Serial.print("▶ PID 직진 | Enc M1="); Serial.print(encM1);

Serial.print(" M2="); Serial.print(encM2);

Serial.print(" M3="); Serial.print(encM3);

Serial.print(" M4="); Serial.print(encM4);

Serial.print(" | PID12="); Serial.print(output12);

Serial.print(" | PID34="); Serial.println(output34);

delay(interval);

}

// 모든 모터 정지

void stopRobot() {

motorDriver.setSingleMotor(M1, 0);

motorDriver.setSingleMotor(M2, 0);

motorDriver.setSingleMotor(M3, 0);

motorDriver.setSingleMotor(M4, 0);

Serial.println("⏹ 정지 완료");

}

1. 라이다 소스코드

**\*\*MOVE\_BASE**

<launch>

<!-- 1. 맵 서버: 저장된 맵 파일을 로드해서 /map 토픽으로 퍼블리시 -->

<node pkg="map\_server" type="map\_server" name="map\_server"

args="$(find my\_robot)/maps/my\_new3\_map.yaml" />

<!-- 2. 정적 TF 설정: base\_link → laser 간의 고정 위치를 알려줌 -->

<node pkg="tf" type="static\_transform\_publisher" name="base\_to\_laser"

args="0 0 0 0 0 0 base\_link laser 100" />

<!-- 여기서 (0 0 0 0 0 0)은 위치(x,y,z) + 회전(RPY), 100은 전송 주기(Hz) -->

<!-- 3. AMCL 노드: 라이다와 odom을 이용해 로봇 위치 추정 (Localization) -->

<node pkg="amcl" type="amcl" name="amcl" output="screen">

<param name="odom\_frame\_id" value="odom"/> <!-- 오도메트리 기준 프레임 -->

<param name="base\_frame\_id" value="base\_link"/> <!-- 로봇의 본체 중심 프레임 -->

<param name="global\_frame\_id" value="map"/> <!-- 최종 위치 추정은 map 기준 -->

<param name="scan\_topic" value="/scan"/> <!-- 라이다 스캔 데이터 토픽 -->

<param name="transform\_tolerance" value="0.5" /> <!-- TF 동기화 허용 오차 (초) -->

</node>

<!-- 4. move\_base: 전역+지역 경로 계획 및 로봇 속도 명령 생성 -->

<node pkg="move\_base" type="move\_base" name="move\_base" output="screen">

<!-- 글로벌 플래너 지정 (A\* 기반) -->

<param name="base\_global\_planner" value="navfn/NavfnROS" />

<!-- 컨트롤러 반복 주기: 1초에 몇 번 속도 제어할지 (5Hz) -->

<param name="controller\_frequency" value="5.0" />

<!-- 공통 설정 로딩: 센서 범위, 장애물 처리, 인플레이션 등 -->

<rosparam file="$(find my\_robot)/config/costmap\_common\_params.yaml" command="load" />

<!-- 로컬 코스트맵 설정: 로봇 주변 단기 경로 계획 -->

<rosparam file="$(find my\_robot)/config/local\_costmap\_params.yaml" command="load" />

<!-- 글로벌 코스트맵 설정: 맵 전체 기반의 장기 경로 계획 -->

<rosparam file="$(find my\_robot)/config/global\_costmap\_params.yaml" command="load" />

<!-- 로컬 플래너 설정: 속도 제한, 장애물 회피 전략 등 -->

<rosparam file="$(find my\_robot)/config/base\_local\_planner\_params.yaml" command="load" />

</node>

<!-- 5. RViz 실행: 맵, 라이다, 위치 등 시각적으로 보여줌 -->

<node pkg="rviz" type="rviz" name="rviz"

args="-d $(find my\_robot)/rviz/move\_base\_config.rviz" />

</launch>

**\*\*ODOM\_FROM\_SERIAL**

#!/usr/bin/env python3

import rospy # ROS 노드용

from nav\_msgs.msg import Odometry # 오도메트리 메시지 형식

from geometry\_msgs.msg import Quaternion # 방향 표현용 쿼터니언

import tf # TF 브로드캐스터

import serial # 시리얼 통신용

import math # 수학 계산용

# === 설정 ===

PORT = "/dev/ttyACM0" # 아두이노와 연결된 포트

BAUD = 115200 # 시리얼 통신 속도 (아두이노와 맞춰야 함)

WHEEL\_RADIUS = 0.04 # 바퀴 반지름 (단위: m)

TICKS\_PER\_REV = 360 # 엔코더 한 바퀴당 틱 수 (CPR)

DIST\_PER\_TICK = 2 \* math.pi \* WHEEL\_RADIUS / TICKS\_PER\_REV # 한 틱당 거리 (m)

# 시리얼 포트 열기

ser = serial.Serial(PORT, BAUD, timeout=1)

# ROS 노드 초기화

rospy.init\_node("odom\_from\_serial")

rospy.loginfo(" /odom 퍼블리셔 시작")

# 퍼블리셔 및 TF 브로드캐스터 초기화

pub = rospy.Publisher("/odom", Odometry, queue\_size=50)

br = tf.TransformBroadcaster()

# 위치 및 상태 변수 초기화

x = 0.0 # X좌표

y = 0.0 # Y좌표

theta = 0.0 # 방향 각도 (라디안)

last\_time = rospy.Time.now() # 마지막 시간 기록

prev\_enc = [0, 0, 0] # 이전 엔코더 값 저장용

# 시리얼 데이터 파싱 함수

def parse\_encoder(line):

if not line.startswith("ENC,"):

return None # 형식이 다르면 무시

try:

parts = line.strip().split(",")

return [int(parts[1]), int(parts[2]), int(parts[3])] # 정수로 변환

except:

return None # 오류 발생 시 무시

# 루프 주기 설정 (30Hz = 0.033초마다 반복)

rate = rospy.Rate(30)

# === 메인 루프 시작 ===

while not rospy.is\_shutdown():

now = rospy.Time.now()

stable\_time = now - rospy.Duration(0.05) # TF 동기화 지연 보정용

# 아두이노로부터 한 줄 읽기

line = ser.readline().decode("utf-8", errors="ignore").strip()

enc = parse\_encoder(line) # ENC,d1,d2,d3 → 리스트로 파싱

if enc is None:

continue # 형식이 아니면 skip

# 시간 간격 계산

dt = (now - last\_time).to\_sec()

last\_time = now

# 엔코더 차이 계산 → 거리로 환산

d1 = (enc[0] - prev\_enc[0]) \* DIST\_PER\_TICK

d2 = (enc[1] - prev\_enc[1]) \* DIST\_PER\_TICK

d3 = (enc[2] - prev\_enc[2]) \* DIST\_PER\_TICK

# 평균 직진 거리

dx = (d1 + d2 + d3) / 3.0

vx = dx / dt if dt > 0 else 0.0 # 속도 = 거리 / 시간

vth = 0.0 # 각속도는 아직 고려 안 함 (직선만)

# 위치 업데이트 (현재 각도 고려한 dx 벡터 적용)

x += dx \* math.cos(theta)

y += dx \* math.sin(theta)

prev\_enc = enc # 현재 엔코더 → 다음 반복용

# 현재 방향을 쿼터니언으로 변환

quat = tf.transformations.quaternion\_from\_euler(0, 0, theta)

# === TF 전송 ===

br.sendTransform((x, y, 0), quat, stable\_time, "base\_link", "odom")

# === Odometry 메시지 작성 및 전송 ===

odom = Odometry()

odom.header.stamp = stable\_time # 시간

odom.header.frame\_id = "odom" # 전역 프레임

odom.child\_frame\_id = "base\_link" # 로봇 본체

# 위치 및 방향 설정

odom.pose.pose.position.x = x

odom.pose.pose.position.y = y

odom.pose.pose.orientation = Quaternion(\*quat)

# 선속도, 각속도 설정

odom.twist.twist.linear.x = vx

odom.twist.twist.angular.z = vth

# 퍼블리시!

pub.publish(odom)

# 디버그 출력 (x, y, 속도, 시간)

rospy.loginfo(f"[ODOM] x={x:.2f}, y={y:.2f}, vx={vx:.2f}, t={stable\_time.to\_sec():.2f}")

# 다음 루프까지 대기

rate.sleep()

\*\*CMD\_VEL\_TO\_SERIAL  
#!/usr/bin/env python3

import rospy # ROS 노드 기능

from geometry\_msgs.msg import Twist # /cmd\_vel 메시지 타입

import serial # 시리얼 통신용

import time # 시간 계산용

# 아두이노가 리셋되며 부팅될 시간을 주기 위해 잠깐 대기

time.sleep(1.0)

# === 시리얼 포트 설정 ===

try:

ser = serial.Serial('/dev/ttyACM0', 115200, timeout=1) # 포트 이름 & 속도

time.sleep(2.0) # 아두이노 재부팅 시간 확보

rospy.loginfo(" Serial port /dev/ttyACM0 opened")

except serial.SerialException as e:

rospy.logerr(f" Failed to open serial port: {e}")

exit(1) # 포트 연결 실패 시 종료

# 전송 타이밍 제어용 변수

last\_send\_time = 0

send\_interval = 0.1 # 0.1초 간격 = 10Hz

# === Twist 메시지를 수신하면 실행되는 콜백 함수 ===

def callback(msg):

global last\_send\_time

now = time.time()

# 전송 간격 제한 (10Hz)

if now - last\_send\_time >= send\_interval:

# Twist 메시지에서 속도 추출

vx = msg.linear.x # 전진/후진 속도

vy = msg.linear.y # 좌우 이동 속도 (옴니휠에서만 사용)

omega = msg.angular.z # 회전 속도 (z축)

# 문자열 포맷팅 (소수점 3자리) + 개행 문자 포함

data = f"{vx:.3f},{vy:.3f},{omega:.3f}\n"

try:

# 아두이노로 전송

ser.write(data.encode('utf-8')) # UTF-8로 인코딩

ser.flush() # 버퍼 비우기 (즉시 전송)

rospy.loginfo(f"[ROS→Arduino] Sent: {data.strip()}")

except serial.SerialException as e:

rospy.logerr(f"[ERROR] Serial write failed: {e}")

# 마지막 전송 시간 업데이트

last\_send\_time = now

# === 메인 함수: ROS 노드 실행 ===

def main():

rospy.init\_node('cmd\_vel\_to\_serial') # 노드 이름 설정

rospy.Subscriber("/cmd\_vel", Twist, callback) # /cmd\_vel 메시지 구독

rospy.loginfo(" cmd\_vel\_to\_serial node started")

rospy.spin() # 콜백 무한 대기 루프

# 실행 시작점

if \_\_name\_\_ == "\_\_main\_\_":

main()

* odom\_from\_serial.py: 엔코더 데이터를 수신하여 /odom, /tf 퍼블리시

1. 웹서버 소스 코드
   * 서버 소스코드

from flask import Flask, request, jsonify, render\_template

import json, os, subprocess, paramiko

from flask\_socketio import SocketIO, emit

app = Flask(\_\_name\_\_)

socketio = SocketIO(app, cors\_allowed\_origins="\*")

BOOKS\_FILE = "books.json"

EXPECTED\_BARCODES\_FILE = "expected\_barcodes.json"

RASPBERRY\_IP = "192.168.137.82"

RASPBERRY\_USER = "pi"

RASPBERRY\_PASS = "raspberry"

C\_CODE = "/home/pi/barcode\_scanner.c"

COMPILED = "/home/pi/scanner"

# JSON 파일에서 책 정보 불러오기

robot\_status = "normal"

def load\_books():

return json.load(open(BOOKS\_FILE, "r", encoding="utf-8")) if os.path.exists(BOOKS\_FILE) else []

books = load\_books()

# JSON 파일에 책 정보 저장하기

def save\_books():

json.dump(books, open(BOOKS\_FILE, "w", encoding="utf-8"), ensure\_ascii=False, indent=4)

# 서버 부팅 시 expected\_barcodes.json 생성

expected\_barcodes = {}

def generate\_expected\_barcodes():

global expected\_barcodes

location\_dict = {}

for book in books:

location = book["location"]

barcode = book["barcode"]

position = book["position"]

if location not in location\_dict:

location\_dict[location] = []

location\_dict[location].append((barcode, position))

# 정렬 후 저장 (position 기준)

expected\_barcodes = {

location: [barcode for barcode, \_ in sorted(barcodes, key=lambda x: x[1])]

for location, barcodes in location\_dict.items()

}

# JSON 파일로 저장

with open(EXPECTED\_BARCODES\_FILE, "w", encoding="utf-8") as file:

json.dump(expected\_barcodes, file, ensure\_ascii=False, indent=4)

print("expected\_barcodes.json 업데이트 완료!")

# 서버 부팅 시 expected\_barcodes.json 생성

generate\_expected\_barcodes()

@app.route('/robot\_status', methods=['GET'])

def get\_robot\_status():

return jsonify({"status": robot\_status})

@app.route('/set\_robot\_status', methods=['POST'])

def set\_robot\_status():

global robot\_status

data = request.get\_json()

status = data.get("status")

if status in ["normal", "scanning", "complete"]:

robot\_status = status

return jsonify({"success": True, "status": robot\_status})

else:

return jsonify({"success": False, "error": "Invalid status"}), 400

@app.route('/')

def home():

return render\_template('index.html', books=books)

@app.route('/books')

def books\_page():

return render\_template('books.html', books=books)

@app.route('/book\_shelf')

def book\_shelf():

grouped\_shelves = {} # location별로 그룹화된 책 저장

# 책 상태와 그룹화를 처리

for book in books:

if book['location'] in ['1F-1-A-1-a', '1F-1-A-2-a', '2F-1-A-1-a', '2F-1-A-2-a']:

status\_label = ""

color\_class = ""

if book.get("available", False):

status\_label = "책 있음"

color\_class = "available"

elif not book.get("available", False) and not book.get("misplaced", False) and not book.get("wrong-location", False):

status\_label = "책이 없음"

color\_class = "not-available"

elif book.get("misplaced", False):

status\_label = "순서 잘못됨"

color\_class = "misplaced"

elif book.get("wrong-location", False):

status\_label = "잘못 배치됨"

color\_class = "wrong-location"

# 책을 location별로 그룹화

if book['location'] not in grouped\_shelves:

grouped\_shelves[book['location']] = []

grouped\_shelves[book['location']].append({

'title': book['title'],

'barcode': book['barcode'],

'image\_url': book.get('image\_url', 'default.jpg'),

'status\_label': status\_label,

'color\_class': color\_class

})

return render\_template('book\_shelf.html', grouped\_shelves=grouped\_shelves)

# 1. C 코드 컴파일 및 실행 (터미널에서)

@app.route('/scan', methods=['POST'])

def scan\_book():

global robot\_status

robot\_status = "scanning"

ssh = paramiko.SSHClient()

ssh.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

try:

ssh.connect(RASPBERRY\_IP, username=RASPBERRY\_USER, password=RASPBERRY\_PASS)

# 터미널에서 scanner 실행 (종료는 따로)

command = (

f"DISPLAY=:0 lxterminal --command='bash -c \""

f"gcc {C\_CODE} -o {COMPILED} -ljansson -lcurl && "

f"{COMPILED}; "

f"exec bash\"'"

)

ssh.exec\_command(command)

return jsonify({"success": True, "message": "C 코드 실행 시작 (lxterminal에서)"})

except Exception as e:

return jsonify({"success": False, "message": "실행 오류", "error": str(e)})

# 2. scanner 종료 (kill)

@app.route('/scan\_exit', methods=['POST'])

def stop\_scanner():

global robot\_status

robot\_status = "complete"

ssh = paramiko.SSHClient()

ssh.set\_missing\_host\_key\_policy(paramiko.AutoAddPolicy())

try:

ssh.connect(RASPBERRY\_IP, username=RASPBERRY\_USER, password=RASPBERRY\_PASS)

# scanner 종료 (터미널 내에서 실행되었기 때문에 -f로 종료)

stop\_command = "pkill -9 -f './scanner'"

ssh.exec\_command(stop\_command)

# lxterminal도 함께 닫으려면 아래 추가

kill\_terminal = "pkill -9 -f lxterminal"

ssh.exec\_command(kill\_terminal)

return jsonify({"success": True, "message": "scanner 및 터미널 종료 완료"})

except Exception as e:

return jsonify({"success": False, "message": "종료 오류", "error": str(e)})

# 책 상태 업데이트 API

@app.route('/update\_book\_status', methods=['POST'])

def update\_book\_status():

global books, expected\_barcodes

data = request.get\_json()

scanned = data # 클라이언트에서 전송한 스캔 데이터 (예: {"3F-A-1-F": ["123", "234", "345", "456", "567"]})

# 스캔된 칸바코드가 있는지 확인

if not scanned:

return jsonify({"error": "잘못된 데이터입니다."}), 400

location = list(scanned.keys())[0] # 예: "3F-A-1-F"

scanned\_barcodes = scanned[location] # 예: ["123", "234", "345", "456", "567"]

# expected\_barcodes.json에서 예상 바코드 목록 가져오기

if location not in expected\_barcodes:

return jsonify({"error": "잘못된 책장 위치입니다."}), 400

expected\_barcodes\_list = expected\_barcodes[location] # 예: ["123", "234", "345", "456", "567"]

expected\_set = set(expected\_barcodes\_list) # 빠른 검색을 위한 집합 변환

scanned\_set = set(scanned\_barcodes) # 스캔된 바코드 집합 변환

# 1. expected에는 있지만 scanned에는 없는 바코드 찾기 (책이 없는 경우)

missing\_barcodes = expected\_set - scanned\_set

# 2. expected에는 없지만 scanned에 있는 바코드 찾기 (잘못된 위치)

wrong\_location\_barcodes = scanned\_set - expected\_set

# wrong-location에 해당하는 바코드는 제거

scanned\_barcodes = [b for b in scanned\_barcodes if b not in wrong\_location\_barcodes]

# 3. 버퍼에 분배

buffers = [] # 여러 개의 버퍼 (각각의 버퍼는 순서대로 스캔된 바코드들)

# 첫 번째 바코드는 첫 번째 버퍼에 넣기

current\_buffer = [scanned\_barcodes[0]]

# 나머지 바코드는 이전 바코드와의 순서를 비교하여 버퍼에 나누어 넣기

for i in range(1, len(scanned\_barcodes)):

prev\_barcode = scanned\_barcodes[i - 1]

curr\_barcode = scanned\_barcodes[i]

# prev\_barcode가 current\_buffer의 마지막 바코드보다 앞선 순서일 경우

if expected\_barcodes\_list.index(curr\_barcode) > expected\_barcodes\_list.index(prev\_barcode):

current\_buffer.append(curr\_barcode)

else:

# 새로운 버퍼를 시작

buffers.append(current\_buffer)

current\_buffer = [curr\_barcode]

# 마지막 버퍼도 추가

buffers.append(current\_buffer)

# 가장 큰 버퍼를 available로 간주

largest\_buffer = max(buffers, key=len)

available\_barcodes = []

misplaced\_barcodes = []

for buffer in buffers:

if buffer == largest\_buffer:

available\_barcodes.extend(buffer) # 가장 큰 버퍼의 바코드는 available

else:

misplaced\_barcodes.extend(buffer) # 나머지 버퍼의 바코드는 misplaced

# 책 상태 업데이트

for book in books:

if book["barcode"] in available\_barcodes:

book["available"] = True

book["misplaced"] = False

book["wrong-location"] = False

elif book["barcode"] in misplaced\_barcodes:

book["available"] = False

book["misplaced"] = True

book["wrong-location"] = False

elif book["barcode"] in wrong\_location\_barcodes:

book["available"] = False

book["misplaced"] = False

book["wrong-location"] = True

elif book["barcode"] in missing\_barcodes:

book["available"] = False

book["misplaced"] = False

book["wrong-location"] = False

save\_books() # 변경된 책 상태 저장

socketio.emit('refresh', {'message': 'update\_detected'})

return jsonify({

"available": available\_barcodes,

"misplaced": misplaced\_barcodes,

"wrong-location": list(wrong\_location\_barcodes),

"not-available": list(missing\_barcodes)

}), 200

@app.route('/get\_books', methods=['GET'])

def get\_books():

return jsonify(books)

@app.route('/search')

def search\_books():

query = request.args.get('query', '').lower()

books = load\_books()

filtered\_books = [book for book in books if query in book['title'].lower() or query in book['author'].lower()]

return jsonify(filtered\_books)

if \_\_name\_\_ == '\_\_main\_\_':

socketio.run(app, debug=True, host='0.0.0.0', port=5000)

* + 책장 별 바코드 소스코드

{

"1F-1-A-1-a": [

"0662345",

"0630407",

"0367900",

"0271270",

"0271268",

"W079266"

],

"1F-1-A-2-a": [

"0327791",

"0797050",

"0358409",

"0793104",

"0729806",

"W024258"

],

"2F-1-A-1-a": [

"0771933",

"0346961",

"0692505",

"0668313",

"0271262",

"W038498",

"W112143"

],

"2F-1-A-2-a": [

"0772556",

"0271263",

"0370230",

"0347286",

"0696520",

"0380227",

"0612227",

"0612228"

]

}

* 책 정보 소스코드

[

{

"barcode": "0662345",

"title": "한국인이 꼭 틀리는 영어 천 가지지",

"author": "김현후",

"location": "1F-1-A-1-a",

"position": 1,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate212/MidCate001/21105454.jpg",

"available": false,

"misplaced": false,

"wrong-location": false,

"status": "not-available"

},

{

"barcode": "0630407",

"title": "한자능력검정시험 2급",

"author": "한국어문교육연구회",

"location": "1F-1-A-1-a",

"position": 2,

"call\_number": "",

"image\_url": "https://image.yes24.com/goods/124095565/XL",

"available": false,

"misplaced": false,

"wrong-location": false,

"status": "not-available"

},

{

"barcode": "0367900",

"title": "한자능력검정시험 1급",

"author": "한국어문교육연구회",

"location": "1F-1-A-1-a",

"position": 3,

"call\_number": "",

"image\_url": "https://image.yes24.com/goods/129391866/XL",

"available": false,

"misplaced": false,

"wrong-location": false,

"status": "not-available"

},

{

"barcode": "0271270",

"title": "고석편 3-2",

"author": "최남규",

"location": "1F-1-A-1-a",

"position": 4,

"call\_number": "",

"image\_url": "https://contents.kyobobook.co.kr/sih/fit-in/400x0/pdt/9791167425799.jpg",

"available": false,

"misplaced": false,

"wrong-location": false,

"status": "not-available"

},

{

"barcode": "0271268",

"title": "고석편 2",

"author": "최남규",

"location": "1F-1-A-1-a",

"position": 5,

"call\_number": "",

"image\_url": "https://contents.kyobobook.co.kr/sih/fit-in/400x0/pdt/9791167425799.jpg",

"available": false,

"misplaced": false,

"wrong-location": false,

"status": "not-available"

},

{

"barcode": "W079266",

"title": "THE CAMBRIDGE GRAMMAR OF THE ENGLISH LANGUAGE",

"author": "HUDDLESTON AND PULLUM",

"location": "1F-1-A-1-a",

"position": 6,

"call\_number": "",

"image\_url": "https://m.media-amazon.com/images/I/81LdafQFT1L.\_SL1472\_.jpg",

"available": false,

"misplaced": false,

"wrong-location": false,

"status": "not-available"

},

{

"barcode": "0327791",

"title": "피아노 소나타 1987",

"author": "강유일일",

"location": "1F-1-A-2-a",

"position": 1,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate47/MidCate03/4625449.jpg",

"available": false,

"misplaced": false,

"wrong-location": false,

"status": "not-available"

},

{

"barcode": "0797050",

"title": "귀우",

"author": "이다모모",

"location": "1F-1-A-2-a",

"position": 2,

"call\_number": "",

"image\_url": "https://image.yes24.com/goods/130197852/XL",

"available": false,

"misplaced": false,

"wrong-location": false,

"status": "not-available"

},

{

"barcode": "0358409",

"title": "붉은 화살",

"author": "정병기",

"location": "1F-1-A-2-a",

"position": 3,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate60/MidCate07/5968080.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0793104",

"title": "구름문",

"author": "이다하",

"location": "1F-1-A-2-a",

"position": 4,

"call\_number": "",

"image\_url": "https://image.yes24.com/goods/124328263/XL",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0729806",

"title": "문학고전강의",

"author": "강유원",

"location": "1F-1-A-2-a",

"position": 5,

"call\_number": "",

"image\_url": "https://image.yes24.com/goods/40190523/XL",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "W024258",

"title": "LITERATURE",

"author": "KIRSZNER AND MANDELL",

"location": "1F-1-A-2-a",

"position": 6,

"call\_number": "",

"image\_url": "https://m.media-amazon.com/images/I/312Xv-nN17L.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0771933",

"title": "AI시대에 꼭 필요한 미래저널과 미래리딩",

"author": "박병기",

"location": "2F-1-A-1-a",

"position": 1,

"call\_number": "",

"image\_url": "https://image.yes24.com/goods/96645524/XL",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0346961",

"title": "유리심장2",

"author": "조례진",

"location": "2F-1-A-1-a",

"position": 2,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate0001/kepub/L\_67445.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0692505",

"title": "남부군",

"author": "이태",

"location": "2F-1-A-1-a",

"position": 3,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate407/MidCate008/40603553.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0668313",

"title": "정비석 문학 선집 3",

"author": "김현주",

"location": "2F-1-A-1-a",

"position": 4,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate236/MidCate001/23501146.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0271262",

"title": "서주금문의 어휘 허사편",

"author": "최남규",

"location": "2F-1-A-1-a",

"position": 5,

"call\_number": "",

"image\_url": "https://image.yes24.com/goods/120499261/XL",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "W038498",

"title": "Grammar for English Language Teachers",

"author": "parrott",

"location": "2F-1-A-1-a",

"position": 6,

"call\_number": "https://images-na.ssl-images-amazon.com/images/S/compressed.photo.goodreads.com/books/1348771462i/1380938.jpg",

"image\_url": "",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "W112143",

"title": "The Teacher's Grammar of English",

"author": "Ron cowan",

"location": "2F-1-A-1-a",

"position": 7,

"call\_number": "",

"image\_url": "https://image.yes24.com/goods/1575003/XL",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0772556",

"title": "한국고소설강의",

"author": "한국고소설학회",

"location": "2F-1-A-2-a",

"position": 1,

"call\_number": "",

"image\_url": "https://image.yes24.com/goods/68724092/XL",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0271263",

"title": "서주금문의 어휘 실사편편",

"author": "최남규",

"location": "2F-1-A-2-a",

"position": 2,

"call\_number": "",

"image\_url": "https://image.yes24.com/goods/120499261/XL",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0370230",

"title": "백범",

"author": "김별아",

"location": "2F-1-A-2-a",

"position": 3,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate224/MidCate003/19632491.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0347286",

"title": "유리심장1",

"author": "조례진",

"location": "2F-1-A-2-a",

"position": 4,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate0001/kepub/L\_67444.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0696520",

"title": "강철무지개",

"author": "최인석",

"location": "2F-1-A-2-a",

"position": 5,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate427/MidCate008/42672081.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0380227",

"title": "작전",

"author": "정철진",

"location": "2F-1-A-2-a",

"position": 6,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate67/MidCate09/6686113.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0612227",

"title": "화홍 연정만리",

"author": "이지환",

"location": "2F-1-A-2-a",

"position": 7,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate91/MidCate02/9017531.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

},

{

"barcode": "0612228",

"title": "화홍 월하정인",

"author": "이지환",

"location": "2F-1-A-2-a",

"position": 8,

"call\_number": "",

"image\_url": "https://image.yes24.com/momo/TopCate91/MidCate02/9017527.jpg",

"available": false,

"misplaced": false,

"wrong-location": false

}

]

* 메인 웹페이지 기능 소스코드

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8"> <!-- 문자 인코딩 UTF-8 설정 -->

<meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- 모바일 화면 대응 -->

<title>도서관 관리 시스템</title> <!-- 페이지 제목 -->

<link rel="stylesheet" href="{{ url\_for('static', filename='styles/styles.css') }}"> <!-- 외부 CSS 연결 -->

</head>

<body>

<!-- 상단 헤더 섹션 -->

<header>

<h1 onclick="refreshPage()">도서관 관리 시스템</h1> <!-- 제목 클릭 시 페이지 새로고침 실행 -->

</header>

<main>

<!-- 도서관 관련 이미지 3개를 보여주는 컨테이너 -->

<div class="image-container">

<img id="image1" src="{{ url\_for('static', filename='images/image1.png') }}" alt="도서관 이미지 1">

<img id="image2" src="{{ url\_for('static', filename='images/image2.png') }}" alt="도서관 이미지 2">

<img id="image3" src="{{ url\_for('static', filename='images/image3.png') }}" alt="도서관 이미지 3">

</div>

<!-- 검색창과 검색 버튼 -->

<div class="search-container">

<input type="text" id="searchInput" placeholder="책 이름 또는 저자를 입력하세요"> <!-- 검색어 입력 필드 -->

<button onclick="searchBook()">검색</button> <!-- 검색 버튼 -->

</div>

<!-- 검색 결과를 표시할 영역 -->

<div id="bookInfo"></div>

<!-- 여러 기능 버튼들이 모여 있는 영역 -->

<div class="button-container">

<button onclick="location.href='/books'">도서 목록 보기</button> <!-- 도서 목록 페이지 이동 -->

<button onclick="location.href='/book\_shelf'">책장 상태 보기</button> <!-- 책장 상태 페이지 이동 -->

<button id="startScanButton" onclick="scanBook()">📖 책 스캔 시작</button> <!-- 책 스캔 시작 -->

<button id="exitScanButton" onclick="exitscan()">스캔 종료</button> <!-- 스캔 종료 -->

</div>

<!-- 스캔 상태 메시지 표시 영역 -->

<div id="scanStatus"></div>

</main>

<script>

// 책 검색 함수

function searchBook() {

const query = document.getElementById('searchInput').value.trim(); // 입력값 가져오기

if (!query) { // 입력값 없으면 알림

alert('검색어를 입력하세요!');

return;

}

// 서버에 검색 요청 (GET /search?query=검색어)

fetch(`/search?query=${encodeURIComponent(query)}`)

.then(response => response.json()) // JSON 응답 받음

.then(data => {

let bookInfo = document.getElementById("bookInfo");

bookInfo.innerHTML = ""; // 기존 내용 초기화

if (data.length) { // 결과가 있으면 반복 출력

data.forEach(book => {

bookInfo.innerHTML += `

<div class="book-card">

<h3>${book.title}</h3>

<p>저자: ${book.author}</p>

<p>위치: ${book.location}</p>

<p>청구기호: ${book.call\_number ? book.call\_number : '없음'}</p>

<img src="${book.image\_url}" alt="${book.title}" class="book-image">

</div>

<hr>

`;

});

} else {

// 결과 없을 때 메시지 표시

bookInfo.innerHTML = "<p class='no-results'>책을 찾을 수 없습니다.</p>";

}

})

.catch(error => {

// 오류 처리

console.error('검색 오류:', error);

alert('검색 중 오류가 발생했습니다. 다시 시도해 주세요.');

});

}

// 책 스캔 시작 함수

function scanBook() {

let scanStatusElement = document.getElementById('scanStatus');

scanStatusElement.innerText = '책 스캔 중...'; // 상태 메시지 표시

scanStatusElement.style.display = 'block';

// 서버에 POST /scan 요청

fetch('/scan', { method: 'POST' })

.then(response => response.json())

.then(data => {

if (data.success) {

scanStatusElement.innerText = '책 스캔 완료!';

} else {

scanStatusElement.innerText = '책 스캔 실패!';

}

})

.catch(error => {

console.error('스캔 오류:', error);

scanStatusElement.innerText = '오류 발생!';

});

// 5초 후 상태 메시지 숨기기

setTimeout(() => {

scanStatusElement.innerText = '';

scanStatusElement.style.display = 'none';

}, 5000);

}

// 스캔 종료 함수

function exitscan() {

let scanStatusElement = document.getElementById('scanStatus');

scanStatusElement.innerText = '종료중...'; // 상태 메시지 표시

scanStatusElement.style.display = 'block';

// 서버에 POST /scan\_exit 요청

fetch('/scan\_exit', { method: 'POST' })

.then(response => response.json())

.then(data => {

if (data.success) {

scanStatusElement.innerText = '종료됨';

} else {

scanStatusElement.innerText = '종료 실패';

}

})

.catch(error => {

console.error('스캔 오류:', error);

scanStatusElement.innerText = '오류 발생!';

});

// 5초 후 상태 메시지 숨기기

setTimeout(() => {

scanStatusElement.innerText = '';

scanStatusElement.style.display = 'none';

}, 5000);

}

// 페이지 새로 고침 함수

function refreshPage() {

location.reload(); // 페이지를 새로고침함

}

</script>

</body>

</html>

* 책정보 웹페이지 기능 소스코드

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8"> <!-- 문자 인코딩 UTF-8 설정 -->

<meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- 모바일 화면 대응 -->

<title>도서 목록</title> <!-- 페이지 제목 -->

<link rel="stylesheet" href="{{ url\_for('static', filename='styles/styles3.css') }}"> <!-- 외부 CSS 연결 -->

<style>

/\* 테이블 기본 스타일 \*/

table {

width: 95%; /\* 테이블 너비 95% \*/

border-collapse: collapse; /\* 테두리 겹치지 않게 \*/

margin: 20px 0; /\* 위아래 마진 \*/

}

th, td {

padding: 10px; /\* 안쪽 여백 \*/

border: 1px solid #b4b3b3; /\* 연한 회색 테두리 \*/

text-align: left; /\* 텍스트 왼쪽 정렬 \*/

}

th {

background-color: #000000; /\* 헤더 배경 검정색 \*/

color: white; /\* 글자 흰색 (추가하면 좋음) \*/

}

/\* 상태별 배경색 및 글자색 지정 \*/

.available {

background-color: green !important; /\* 책 있음: 초록색 배경 \*/

color: white !important; /\* 흰 글자 \*/

}

.not-available {

background-color: red !important; /\* 책 없음: 빨간색 배경 \*/

color: white !important; /\* 흰 글자 \*/

}

.misplaced {

background-color: yellow !important; /\* 자리 변경됨: 노란색 배경 \*/

color: black !important; /\* 검정 글자 \*/

}

.wrong-location {

background-color: purple !important; /\* 잘못된 위치: 보라색 배경 \*/

color: white !important; /\* 흰 글자 \*/

}

</style>

</head>

<body>

<h1>도서 목록</h1> <!-- 페이지 제목 -->

<table>

<thead>

<tr>

<th>책 제목</th>

<th>저자</th>

<th>위치</th>

<th>청구기호</th>

<th>바코드</th>

<th>책 유무</th>

<th>책 이미지</th>

</tr>

</thead>

<tbody>

<!-- books 리스트를 반복하여 한 권씩 행으로 출력 -->

{% for book in books %}

<tr>

<td>{{ book.title }}</td> <!-- 책 제목 -->

<td>{{ book.author }}</td> <!-- 저자 -->

<td>{{ book.location }}</td> <!-- 위치 -->

<td>{{ book.call\_number }}</td> <!-- 청구기호 -->

<td>{{ book.barcode }}</td> <!-- 바코드 -->

<td>

<!-- 책 상태에 따라 다른 클래스와 텍스트 표시 -->

{% if book.available %}

<span class="available">책 있음</span>

{% elif not book.available and not book.misplaced and not book.wrong\_location %}

<span class="not-available">책 없음</span>

{% elif book.wrong\_location %}

<span class="wrong-location">위치 잘못됨</span>

{% elif book.misplaced %}

<span class="misplaced">순서 바뀜</span>

{% endif %}

</td>

<td>

<!-- 책 이미지 출력, 너비 100px로 고정 -->

<img src="{{ book.image\_url }}" alt="{{ book.title }}" width="100">

</td>

</tr>

{% endfor %}

</tbody>

</table>

<!-- 메인 화면으로 돌아가는 버튼 -->

<button class="back-button" onclick="location.href='/'">메인화면으로 돌아가기</button>

</body>

</html>

* 책장 상태 시각화 페이지 소스 코드

<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8"> <!-- 문자 인코딩 UTF-8 설정 -->

<meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- 모바일에서도 화면 크기에 맞게 표시 -->

<title>책장 상태</title> <!-- 문서 제목 -->

<script src="https://cdn.socket.io/4.5.4/socket.io.min.js"></script> <!-- Socket.IO 라이브러리 불러오기 -->

<link rel="stylesheet" href="{{ url\_for('static', filename='styles/styles2.css') }}"> <!-- style2.css 외부 CSS 파일 연결 -->

<style>

/\* 책장 그룹을 그리드 형식으로 배치 \*/

.shelf-group {

display: grid;

grid-template-columns: repeat(auto-fill, minmax(18px, 1fr)); /\* 자동으로 칸 나누기, 최소 크기 18px \*/

gap: 10px; /\* 칸 사이 간격 \*/

margin-top: 30px; /\* 위쪽 여백 \*/

margin-left: 20px; /\* 왼쪽 여백 \*/

}

/\* 개별 책장 아이템 스타일 \*/

.shelf-item {

width: 15%; /\* 부모 크기에 대한 비율 너비 \*/

height: 130px; /\* 높이 고정 \*/

display: flex;

flex-direction: column; /\* 세로 정렬 \*/

justify-content: space-between; /\* 위아래 공간 고르게 배치 \*/

align-items: center; /\* 수평 가운데 정렬 \*/

border: 1px solid #ddd; /\* 연한 테두리 \*/

padding: 10px; /\* 내부 여백 \*/

text-align: center; /\* 텍스트 가운데 정렬 \*/

font-size: 10px; /\* 텍스트 크기 \*/

overflow: hidden; /\* 넘치는 내용 숨김 \*/

}

/\* 책 이미지 크기 \*/

.shelf-item img {

width: 40px; /\* 너비 40px \*/

height: auto; /\* 비율에 맞게 높이 자동 조정 \*/

}

/\* 책 있음 상태 \*/

.available {

background-color: green !important;

color: white !important;

}

/\* 책 없음 상태 \*/

.not-available {

background-color: red !important;

color: white !important;

}

/\* 순서 잘못됨 상태 \*/

.misplaced {

background-color: yellow !important;

color: black !important;

}

/\* 잘못 배치됨 상태 \*/

.wrong-location {

background-color: purple !important;

color: white !important;

}

/\* 책장 그룹 제목 스타일 \*/

.shelf-title {

font-size: 12px;

margin-top: 60px;

margin-bottom: 8px;

text-align: center;

font-weight: bold;

}

/\* 책 제목 텍스트 줄임 표시 \*/

.shelf-item .title {

white-space: nowrap; /\* 한 줄로 유지 \*/

overflow: hidden; /\* 넘치면 숨김 \*/

text-overflow: ellipsis; /\* 넘치면 ... 표시 \*/

font-size: 10px;

}

/\* 상태별 책 개수 요약 박스 스타일 \*/

.status-summary {

position: absolute; /\* 화면 위에 고정 \*/

top: 320px; /\* 위에서 320px 위치 \*/

right: 20px; /\* 오른쪽에서 20px 위치 \*/

font-size: 40px;

font-weight: bold;

background-color: rgba(12, 12, 12, 0.5); /\* 반투명 검정 배경 \*/

color: rgb(255, 255, 255); /\* 흰색 텍스트 \*/

padding: 30px;

border-radius: 10px; /\* 둥근 모서리 \*/

}

/\* 필터 버튼들 가로 배치 \*/

.filter-buttons {

display: flex;

gap: 10px; /\* 버튼 사이 간격 \*/

justify-content: center; /\* 가운데 정렬 \*/

margin-bottom: 20px; /\* 아래 여백 \*/

}

/\* 필터 버튼 공통 스타일 \*/

.filter-buttons button {

padding: 10px 20px; /\* 버튼 내부 여백 \*/

border: none; /\* 테두리 없음 \*/

border-radius: 5px; /\* 둥근 모서리 \*/

font-size: 16px;

font-weight: bold;

cursor: pointer; /\* 클릭 시 커서 변경 \*/

}

/\* 각 필터 버튼 배경색 및 글자색 \*/

.filter-available {

background-color: green;

color: white;

}

.filter-not-available {

background-color: red;

color: white;

}

.filter-misplaced {

background-color: yellow;

color: black;

}

.filter-wrong-location {

background-color: purple;

color: white;

}

</style>

</head>

<body>

<header class="main-header">

<h1>책장 상태</h1> <!-- 페이지 제목 -->

</header>

<!-- 상태별 책 개수 표시 -->

<div class="status-summary">

<p>책 있음: <span id="available-count">0</span></p>

<p>책 없음: <span id="not-available-count">0</span></p>

<p>순서 잘못됨: <span id="misplaced-count">0</span></p>

<p>잘못 배치됨: <span id="wrong-location-count">0</span></p>

</div>

<!-- 도서관 이미지 -->

<div class="image-container">

<img id="image4" src="{{ url\_for('static', filename='images/image4.png') }}" alt="도서관 이미지 4">

</div>

<!-- 상태별 필터 버튼 -->

<div class="filter-buttons">

<button class="filter-available" onclick="filterStatus('available')">책 있음</button>

<button class="filter-not-available" onclick="filterStatus('not-available')">책 없음</button>

<button class="filter-misplaced" onclick="filterStatus('misplaced')">순서 잘못됨</button>

<button class="filter-wrong-location" onclick="filterStatus('wrong-location')">잘못 배치됨</button>

<button class="filter-all" onclick="showAll()">전체 보기</button>

</div>

<!-- 서버에서 넘어온 grouped\_shelves 데이터를 반복하여 책장 그룹과 책장 아이템 출력 -->

{% for location, shelves\_group in grouped\_shelves.items() %}

<div class="shelf-group">

<div class="shelf-title">{{ location }}</div> <!-- 책장 위치 이름 -->

{% for shelf in shelves\_group %}

<div class="shelf-item {{ shelf.color\_class }}"> <!-- 색깔 클래스 포함 -->

<div class="title">{{ shelf.title }}</div> <!-- 책 제목 -->

<div class="barcode">{{ shelf.barcode }}</div> <!-- 책 바코드 -->

<img src="{{ shelf.image\_url }}" alt="책 이미지"> <!-- 책 이미지 -->

<div>{{ shelf.status\_label }}</div> <!-- 상태 텍스트 -->

</div>

{% endfor %}

</div>

{% endfor %}

<!-- 메인화면으로 돌아가는 버튼 -->

<button class="back-button" onclick="location.href='/'">메인화면으로 돌아가기</button>

<script>

let currentFilter = null; // 현재 필터 상태 저장 변수

// 상태별 책 개수 카운트 및 화면 표시 업데이트 함수

function updateStatusSummary() {

let availableCount = 0;

let notAvailableCount = 0;

let misplacedCount = 0;

let wrongLocationCount = 0;

// 모든 책장 아이템을 순회하면서 상태 라벨로 개수 세기

document.querySelectorAll('.shelf-item').forEach(item => {

const statusLabel = item.querySelector('div:last-child').textContent.trim();

if (statusLabel === '책 있음') {

availableCount++;

} else if (statusLabel === '책이 없음') {

notAvailableCount++;

} else if (statusLabel === '순서 잘못됨') {

misplacedCount++;

} else if (statusLabel === '잘못 배치됨') {

wrongLocationCount++;

}

});

// HTML 요소에 카운트 업데이트

document.getElementById('available-count').textContent = availableCount;

document.getElementById('not-available-count').textContent = notAvailableCount;

document.getElementById('misplaced-count').textContent = misplacedCount;

document.getElementById('wrong-location-count').textContent = wrongLocationCount;

}

// 페이지 로드 시 상태 요약 업데이트

window.onload = updateStatusSummary;

// 필터 버튼 클릭 시 해당 상태만 표시하고 나머지는 숨기기

function filterStatus(statusClass) {

currentFilter = statusClass;

document.querySelectorAll('.shelf-item').forEach(item => {

item.style.display = item.classList.contains(statusClass) ? 'flex' : 'none';

});

}

// 전체 보기 버튼 클릭 시 모든 책장 아이템 표시

function showAll() {

currentFilter = null;

document.querySelectorAll('.shelf-item').forEach(item => {

item.style.display = 'flex';

});

}

</script>

</body>

</html>

* 메인 페이지 디자인 소스코드

/\* 전체 페이지의 기본 스타일 설정 \*/

body {

font-family: 'Arial', sans-serif; /\* 폰트 설정 \*/

background-color: #f5f5f5; /\* 배경색 설정 \*/

margin: 0; /\* 기본 여백 제거 \*/

padding: 20px; /\* 페이지 여백 설정 \*/

color: #333; /\* 기본 텍스트 색상 \*/

text-align: center; /\* 텍스트를 중앙 정렬 \*/;

}

header {

background-color: #3498db; /\* 파란색 배경 \*/

padding: 20px 20px; /\* 상하 20px, 좌우 20px 패딩 \*/

color: white; /\* 텍스트 흰색 \*/

font-size: 2rem; /\* 헤더 텍스트 크기 \*/

text-align: center; /\* 중앙 정렬 \*/

cursor: pointer;

}

/\* 제목(h1)의 스타일 설정 \*/

h1 {

color: #2c3e50; /\* 글자 색상 설정 \*/

font-size: 50px; /\* 글자 크기 확대 \*/

font-weight: bold; /\* 굵은 글씨 \*/

font-family: 'Poppins', sans-serif; /\* 고급스러운 폰트 \*/

text-transform: uppercase; /\* 모든 글자를 대문자로 \*/

letter-spacing: 2px; /\* 글자 사이 간격 추가 \*/

margin-bottom: 30px; /\* 제목 하단 여백 \*/

margin-top: 30px; /\* 제목 상단 여백 (아래로 이동) \*/

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.1); /\* 제목에 그림자 효과 추가 \*/

}

/\* 버튼 스타일 설정 \*/

button {

background-color: #3498db; /\* 버튼 배경 색상 \*/

color: white; /\* 버튼 글자 색상 \*/

border: none; /\* 버튼 테두리 제거 \*/

padding: 12px 24px; /\* 버튼 내 여백 설정 \*/

margin: 40px; /\* 버튼 외부 여백 \*/

font-size: 40px; /\* 버튼 글자 크기 \*/

cursor: pointer; /\* 마우스 커서가 버튼 위에 있을 때 손 모양으로 변경 \*/

border-radius: 5px; /\* 버튼 테두리 둥글게 설정 \*/

transition: all 0.2s ease-in-out; /\* 애니메이션 효과 설정 \*/

margin-top: 50px; /\* 버튼을 화면 아래로 이동 \*/

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2); /\* 버튼에 그림자 효과 추가 \*/

}

/\* 버튼 호버(마우스를 올렸을 때) 스타일 \*/

button:hover {

background-color: #0410bb; /\* 버튼 호버 시 배경 색상 변경 \*/

transform: scale(1.05); /\* 버튼 크기 살짝 확대 \*/

border-color: #3498db; /\* 버튼 테두리 색상 변경 \*/

box-shadow: 0 6px 15px rgba(0, 0, 0, 0.3); /\* 호버 시 그림자 강조 \*/

}

/\* 버튼 클릭 시(액티브 상태) 스타일 \*/

button:active {

transform: scale(0.95); /\* 클릭 시 버튼 크기 살짝 줄어드는 효과 \*/

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2); /\* 클릭 시 그림자 복원 \*/

}

/\* image1 스타일 \*/

#image1 {

width: 400px; /\* 이미지 크기 설정 \*/

margin-top: 40px; /\* 제목과 이미지 사이에 여백을 추가 \*/

margin-bottom: 10px; /\* 이미지 아래에 여백 추가 \*/

display: block; /\* 이미지가 중앙에 오도록 블록 요소로 설정 \*/

margin-left: auto; /\* 좌측 중앙 정렬 \*/

margin-right: auto; /\* 우측 중앙 정렬 \*/

}

/\* image2 스타일 \*/

#image2 {

width: 300px; /\* 이미지 크기 설정 \*/

position: absolute; /\* 절대 위치 지정 \*/

right: calc(100% - 480px); /\* 왼쪽에서 450px 떨어지도록 설정 \*/

top: 150px; /\* 상단에서 120px만큼 떨어지게 설정 \*/

}

/\* image3 스타일 \*/

#image3 {

width: 300px; /\* 이미지 크기 설정 \*/

position: absolute; /\* 절대 위치 지정 \*/

left: calc(100% - 480px); /\* 오른쪽에서 480px 떨어지도록 설정 \*/

top: 170px; /\* 상단에서 170px만큼 떨어지게 설정 \*/

}

/\* 검색창 스타일 \*/

.search-container {

display: flex; /\* Flexbox 레이아웃 사용 \*/

justify-content: center; /\* 수평 중앙 정렬 \*/

align-items: center; /\* 수직 중앙 정렬 \*/

margin-top: 10px; /\* 검색창을 화면 아래로 이동 \*/

}

/\* 텍스트 입력창 스타일 \*/

input[type="text"] {

width: 500px; /\* 입력창 너비 설정 \*/

padding: 12px 16px; /\* 입력창 내 여백 설정 \*/

font-size: 30px; /\* 입력창 글자 크기 \*/

border: 2px solid #3498db; /\* 파란색 테두리 \*/

border-radius: 25px; /\* 입력창 테두리 둥글게 설정 \*/

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1); /\* 입력창에 부드러운 그림자 효과 추가 \*/

outline: none; /\* 입력창 포커스 시 외곽선 제거 \*/

transition: all 0.3s ease-in-out; /\* 포커스 시 애니메이션 효과 설정 \*/

}

/\* 입력창 포커스 시 스타일 \*/

input[type="text"]:focus {

border-color: #3498db; /\* 포커스 시 테두리 색상 변경 \*/

box-shadow: 0 2px 6px rgba(52, 152, 219, 0.5); /\* 포커스 시 그림자 강조 \*/

}

/\* 입력창 플레이스홀더 스타일 \*/

input[type="text"]::placeholder {

color: #aaa; /\* 플레이스홀더 글자 색상 \*/

font-style: italic; /\* 플레이스홀더 글자 이탤릭체 \*/

}

/\* 도서 정보 div 스타일 \*/

#bookInfo div {

background: linear-gradient(to right, #f3f5f5, #97c5f0);

padding: 70px; /\* 내부 여백 설정 \*/

margin: 10px 0; /\* 상하 여백 설정 \*/

border-radius: 15px; /\* 부드러운 둥글게 설정 \*/

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1); /\* 부드러운 그림자 효과 \*/

font-family: 'Lora', serif; /\* 고급스러운 폰트 사용 \*/

display: block; /\* 항목들이 세로로 정렬되도록 설정 \*/

text-align: left; /\* 텍스트 왼쪽 정렬 \*/

transition: transform 0.3s ease-in-out, box-shadow 0.3s ease-in-out; /\* 호버 효과 추가 \*/

}

/\* 카드 호버 시 스타일 \*/

#bookInfo div:hover {

transform: translateY(-5px); /\* 위로 살짝 떠오르는 효과 \*/

box-shadow: 0 6px 15px rgba(0, 0, 0, 0.15); /\* 그림자 강조 \*/

}

/\* 책 제목 스타일 \*/

#bookInfo h3 {

font-size: 32px; /\* 글자 크기 설정 (책 제목을 강조) \*/

color: #2c3e50; /\* 짙은 색상으로 제목 강조 \*/

font-weight: bold; /\* 굵은 글씨로 제목 강조 \*/

margin: 0; /\* 마진 제거 (상하 여백을 없애고 싶을 때) \*/

margin-bottom: 30px; /\* 제목 하단에 10px 여백 추가 (책 제목과 그 다음 항목 사이의 여백) \*/

}

/\* 책 정보 텍스트 스타일 \*/

#bookInfo p {

font-size: 24px; /\* 본문 텍스트의 크기를 24px로 설정하여 가독성을 높임 \*/

color: #000505; /\* 중간 검정색 \*/

margin: 5px 0; /\* 위아래 마진을 각각 5px로 설정하여 문단 간 간격을 균일하게 유지 \*/

}

/\* 책 이미지 스타일 \*/

.book-image {

width: 250px; /\* 책 이미지 크기 설정 \*/

height: auto; /\* 이미지 비율 유지 \*/

border-radius: 10px; /\* 이미지 둥글게 처리 \*/

margin-right: 20px; /\* 텍스트와 이미지 간의 여백 \*/

}

/\* 구분선(hr) 스타일 \*/

hr {

border: 0; /\* 기본 테두리 제거 \*/

border-top: 1px solid #ccc; /\* 상단에만 테두리 추가 \*/

}

/\* 책을 찾을 수 없습니다 문구 스타일 \*/

.no-results {

color: #ee0808 !important; /\* 빨간색 텍스트 \*/

font-size: 24px; /\* 글자 크기 설정 \*/

font-weight: bold; /\* 글씨 굵게 \*/

margin-top: 20px; /\* 상단 여백 추가 \*/

margin-bottom: 20px; /\* 하단 여백 추가 \*/

text-align: center; /\* 중앙 정렬 \*/

text-transform: uppercase; /\* 대문자 변환 \*/

animation: shake 0.5s ease-in-out; /\* 흔들리는 애니메이션 효과 \*/

}

/\* 검색 중 오류 문구 스타일 \*/

.error-message {

color: #e74c3c !important; /\* 빨간색 텍스트 \*/

font-size: 24px; /\* 글자 크기 설정 \*/

font-weight: bold; /\* 글씨 굵게 \*/

margin-top: 20px; /\* 상단 여백 추가 \*/

margin-bottom: 20px; /\* 하단 여백 추가 \*/

text-align: center; /\* 중앙 정렬 \*/

text-transform: uppercase; /\* 대문자 변환 \*/

animation: shake 0.5s ease-in-out; /\* 흔들리는 애니메이션 효과 \*/

}

/\* 흔들리는 애니메이션 \*/

@keyframes shake {

0% {

transform: translateX(-10px); /\* 왼쪽으로 10px 이동 \*/

}

25% {

transform: translateX(10px); /\* 오른쪽으로 10px 이동 \*/

}

50% {

transform: translateX(-10px); /\* 다시 왼쪽으로 10px 이동 \*/

}

75% {

transform: translateX(10px); /\* 다시 오른쪽으로 10px 이동 \*/

}

100% {

transform: translateX(0); /\* 원래 위치로 복귀 \*/

}

}

/\* 페이드 인 애니메이션 정의 \*/

@keyframes fadeIn {

from {

opacity: 0; /\* 시작 시 투명하게 \*/

transform: translateY(20px); /\* 살짝 아래에서 올라오는 효과 \*/

}

to {

opacity: 1; /\* 끝나는 시점에 보이게 \*/

transform: translateY(0); /\* 원래 위치로 돌아옴 \*/

}

}

/\* 책 정보 텍스트 스타일 \*/

#bookInfo p {

font-size: 24px; /\* 본문 텍스트의 크기를 24px로 설정하여 가독성을 높임 \*/

color: #000505; /\* 중간 검정색 \*/

margin: 5px 0; /\* 위아래 마진을 각각 5px로 설정하여 문단 간 간격을 균일하게 유지 \*/

}

/\* 책 이미지 스타일 \*/

.book-image {

width: 250px; /\* 책 이미지 크기 설정 \*/

height: auto; /\* 이미지 비율 유지 \*/

border-radius: 10px; /\* 이미지 둥글게 처리 \*/

margin-right: 20px; /\* 텍스트와 이미지 간의 여백 \*/

}

/\* 구분선(hr) 스타일 \*/

hr {

border: 0; /\* 기본 테두리 제거 \*/

border-top: 1px solid #ccc; /\* 상단에만 테두리 추가 \*/

}

/\* 책을 찾을 수 없습니다 문구 스타일 \*/

.no-results {

color: #ee0808 !important; /\* 빨간색 텍스트 \*/

font-size: 24px; /\* 글자 크기 설정 \*/

font-weight: bold; /\* 글씨 굵게 \*/

margin-top: 20px; /\* 상단 여백 추가 \*/

margin-bottom: 20px; /\* 하단 여백 추가 \*/

text-align: center; /\* 중앙 정렬 \*/

text-transform: uppercase; /\* 대문자 변환 \*/

animation: shake 0.5s ease-in-out; /\* 흔들리는 애니메이션 효과 \*/

}

/\* 검색 중 오류 문구 스타일 \*/

.error-message {

color: #e74c3c !important; /\* 빨간색 텍스트 \*/

font-size: 24px; /\* 글자 크기 설정 \*/

font-weight: bold; /\* 글씨 굵게 \*/

margin-top: 20px; /\* 상단 여백 추가 \*/

margin-bottom: 20px; /\* 하단 여백 추가 \*/

text-align: center; /\* 중앙 정렬 \*/

text-transform: uppercase; /\* 대문자 변환 \*/

animation: shake 0.5s ease-in-out; /\* 흔들리는 애니메이션 효과 \*/

}

/\* 흔들리는 애니메이션 \*/

@keyframes shake {

0% {

transform: translateX(-10px); /\* 왼쪽으로 10px 이동 \*/

}

25% {

transform: translateX(10px); /\* 오른쪽으로 10px 이동 \*/

}

50% {

transform: translateX(-10px); /\* 다시 왼쪽으로 10px 이동 \*/

}

75% {

transform: translateX(10px); /\* 다시 오른쪽으로 10px 이동 \*/

}

100% {

transform: translateX(0); /\* 원래 위치로 복귀 \*/

}

}

/\* 페이드 인 애니메이션 정의 \*/

@keyframes fadeIn {

from {

opacity: 0; /\* 시작 시 투명하게 \*/

transform: translateY(20px); /\* 살짝 아래에서 올라오는 효과 \*/

}

to {

opacity: 1; /\* 끝나는 시점에 보이게 \*/

transform: translateY(0); /\* 원래 위치로 돌아옴 \*/

}

}

/\* 책 스캔 중 문구 스타일 \*/

#scanStatus {

font-size: 40px; /\* 글자 크기 증가 \*/

font-weight: bold; /\* 글씨 굵게 \*/

color: #3498db; /\* 파란색 텍스트 \*/

font-family: 'Arial', sans-serif; /\* 폰트 설정 \*/

text-transform: uppercase; /\* 대문자 변환 \*/

margin-top: 20px; /\* 상단 여백 추가 \*/

margin-bottom: 20px; /\* 하단 여백 추가 \*/

animation: blink 1s step-end infinite; /\* 점 깜빡임 애니메이션 \*/

display: none; /\* 처음에는 숨겨두기 \*/

}

/\* 미디어 쿼리: 화면이 768px 이하일 때 \*/

@media screen and (max-width: 768px) {

/\* 전체 페이지 기본 스타일 유지 \*/

body {

padding: 10px; /\* 여백을 조금 더 줄여서 화면을 꽉 채움 \*/

font-size: 16px; /\* 작은 화면에 맞게 글자 크기 조정 \*/

}

/\* 헤더 \*/

header {

font-size: 1.5rem; /\* 모바일에서 글자 크기 줄이기 \*/

padding: 10px 15px; /\* 패딩 조정 \*/

}

/\* 제목(h1) \*/

h1 {

font-size: 35px; /\* 모바일에서 제목 글자 크기 축소 \*/

margin-top: 20px; /\* 제목 상단 여백 \*/

margin-bottom: 20px; /\* 제목 하단 여백 \*/

}

/\* 버튼 스타일 \*/

button {

font-size: 15px; /\* 모바일에서 버튼 글자 크기 축소 \*/

padding: 10px 20px; /\* 버튼 크기 줄이기 \*/

margin: 25px 10px; /\* 여백을 조정하여 화면에 맞게 배치 \*/

}

.button-container {

flex-wrap: nowrap; /\* 모바일에서도 줄 바꿈 없이 가로로 나열 \*/

}

.button-container button {

font-size: 15px; /\* 버튼 글자 크기 축소 \*/

padding: 10px 10px; /\* 버튼 여백 조정 \*/

}

/\* 이미지1 스타일 \*/

#image1 {

width: 40%; /\* 이미지를 화면 크기에 맞게 조정 \*/

margin-top: 20px; /\* 제목과 이미지 간의 간격 조정 \*/

margin-bottom: 15px; /\* 이미지 아래 여백 \*/

}

/\* 이미지2, 이미지3 스타일 - 모바일에서 절대 위치를 사용하지 않도록 수정 \*/

#image2 {

width: 40%; /\* 이미지 크기 조정 \*/

margin: 10px; /\* 이미지 간 간격 \*/

display: inline-block; /\* 세로로 정렬된 이미지를 좌우로 나열 \*/

vertical-align: top; /\* 수직 정렬을 위쪽으로 설정 \*/

top: 65px; /\* 상단에서 120px만큼 떨어지게 설정 \*/

right: calc(100% - 430px); /\* 왼쪽에서 450px 떨어지도록 설정 \*/

}

#image3{

width: 40%; /\* 이미지 크기 조정 \*/

margin: 10px; /\* 이미지 간 간격 \*/

display: inline-block; /\* 세로로 정렬된 이미지를 좌우로 나열 \*/

vertical-align: top; /\* 수직 정렬을 위쪽으로 설정 \*/

top: 80px; /\* 상단에서 120px만큼 떨어지게 설정 \*/

left: calc(100% - 440px); /\* 왼쪽에서 450px 떨어지도록 설정 \*/

}

/\* 이미지들이 같은 줄에 오도록 flexbox 사용 \*/

.image-wrapper {

display: flex; /\* 플렉스를 이용해 이미지 좌우 정렬 \*/

justify-content: space-between; /\* 좌우 이미지 간격을 일정하게 \*/

align-items: flex-start; /\* 상단 정렬 \*/

flex-wrap: wrap; /\* 화면에 맞게 이미지가 줄 바꿈하도록 \*/

margin-top: 20px; /\* 상단 여백 \*/

}

/\* 검색창 스타일 \*/

.search-container {

width: 100%; /\* 화면 전체를 차지하도록 설정 \*/

display: flex; /\* Flexbox 레이아웃 \*/

justify-content: center; /\* 중앙 정렬 \*/

margin-top: 20px; /\* 위 여백 \*/

}

/\* 텍스트 입력창 스타일 \*/

input[type="text"] {

width: 50%; /\* 입력창 너비를 100%로 설정하여 화면을 꽉 채움 \*/

font-size: 13px; /\* 폰트 크기 줄이기 \*/

padding: 12px; /\* 여백 조정 \*/

border-radius: 20px; /\* 둥근 테두리 \*/

}

/\* 책 정보 카드 스타일 \*/

#bookInfo div {

padding: 20px; /\* 내부 여백을 줄여서 화면에 맞게 \*/

margin: 10px 0; /\* 상하 여백 설정 \*/

border-radius: 10px; /\* 둥글게 처리 \*/

}

/\* 책 제목 스타일 \*/

#bookInfo h3 {

font-size: 24px; /\* 책 제목 폰트 크기 줄이기 \*/

margin-bottom: 15px; /\* 하단 여백 \*/

}

/\* 책 이미지 스타일 \*/

.book-image {

width: 150px; /\* 책 이미지 크기 조정 \*/

}

/\* 책 정보 텍스트 \*/

#bookInfo p {

font-size: 16px; /\* 텍스트 크기 줄이기 \*/

}

/\* 책을 찾을 수 없습니다 문구 스타일 \*/

.no-results {

font-size: 20px; /\* 글자 크기 축소 \*/

margin-top: 20px;

text-transform: none; /\* 대문자 변환 제거 \*/

}

/\* 검색 중 오류 문구 스타일 \*/

.error-message {

font-size: 20px; /\* 글자 크기 축소 \*/

}

/\* 스캔 중 문구 스타일 \*/

#scanStatus {

font-size: 30px; /\* 글자 크기 축소 \*/

}

}

* 책장상태 웹페이지 디자인 소스코드

/\* 전체 페이지의 기본 스타일 설정 \*/

body {

font-family: 'Arial', sans-serif; /\* 폰트 설정 \*/

background-color: #f5f5f5; /\* 배경색 설정 \*/

margin: 0; /\* 기본 여백 제거 \*/

padding: 20px; /\* 페이지 여백 설정 \*/

color: #333; /\* 기본 텍스트 색상 \*/

text-align: center; /\* 텍스트를 중앙 정렬 \*/;

}

header {

background-color: #3498db; /\* 파란색 배경 \*/

padding: 20px 20px; /\* 상하 20px, 좌우 20px 패딩 \*/

color: white; /\* 텍스트 흰색 \*/

font-size: 2rem; /\* 헤더 텍스트 크기 \*/

text-align: center; /\* 중앙 정렬 \*/

cursor: pointer;

}

/\* 제목(h1)의 스타일 설정 \*/

h1 {

color: #2c3e50; /\* 글자 색상 설정 \*/

font-size: 50px; /\* 글자 크기 확대 \*/

font-weight: bold; /\* 굵은 글씨 \*/

font-family: 'Poppins', sans-serif; /\* 고급스러운 폰트 \*/

text-transform: uppercase; /\* 모든 글자를 대문자로 \*/

letter-spacing: 2px; /\* 글자 사이 간격 추가 \*/

margin-bottom: 30px; /\* 제목 하단 여백 \*/

margin-top: 30px; /\* 제목 상단 여백 (아래로 이동) \*/

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.1); /\* 제목에 그림자 효과 추가 \*/

}

/\* 버튼 스타일 설정 \*/

button {

background-color: #3498db; /\* 버튼 배경 색상 \*/

color: white; /\* 버튼 글자 색상 \*/

border: none; /\* 버튼 테두리 제거 \*/

padding: 12px 24px; /\* 버튼 내 여백 설정 \*/

margin: 40px; /\* 버튼 외부 여백 \*/

font-size: 40px; /\* 버튼 글자 크기 \*/

cursor: pointer; /\* 마우스 커서가 버튼 위에 있을 때 손 모양으로 변경 \*/

border-radius: 5px; /\* 버튼 테두리 둥글게 설정 \*/

transition: all 0.2s ease-in-out; /\* 애니메이션 효과 설정 \*/

margin-top: 50px; /\* 버튼을 화면 아래로 이동 \*/

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2); /\* 버튼에 그림자 효과 추가 \*/

}

/\* 버튼 호버(마우스를 올렸을 때) 스타일 \*/

button:hover {

background-color: #0410bb; /\* 버튼 호버 시 배경 색상 변경 \*/

transform: scale(1.05); /\* 버튼 크기 살짝 확대 \*/

border-color: #3498db; /\* 버튼 테두리 색상 변경 \*/

box-shadow: 0 6px 15px rgba(0, 0, 0, 0.3); /\* 호버 시 그림자 강조 \*/

}

/\* 버튼 클릭 시(액티브 상태) 스타일 \*/

button:active {

transform: scale(0.95); /\* 클릭 시 버튼 크기 살짝 줄어드는 효과 \*/

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2); /\* 클릭 시 그림자 복원 \*/

}

/\* image1 스타일 \*/

#image1 {

width: 400px; /\* 이미지 크기 설정 \*/

margin-top: 40px; /\* 제목과 이미지 사이에 여백을 추가 \*/

margin-bottom: 10px; /\* 이미지 아래에 여백 추가 \*/

display: block; /\* 이미지가 중앙에 오도록 블록 요소로 설정 \*/

margin-left: auto; /\* 좌측 중앙 정렬 \*/

margin-right: auto; /\* 우측 중앙 정렬 \*/

}

/\* image2 스타일 \*/

#image2 {

width: 300px; /\* 이미지 크기 설정 \*/

position: absolute; /\* 절대 위치 지정 \*/

right: calc(100% - 480px); /\* 왼쪽에서 450px 떨어지도록 설정 \*/

top: 150px; /\* 상단에서 120px만큼 떨어지게 설정 \*/

}

/\* image3 스타일 \*/

#image3 {

width: 300px; /\* 이미지 크기 설정 \*/

position: absolute; /\* 절대 위치 지정 \*/

left: calc(100% - 480px); /\* 오른쪽에서 480px 떨어지도록 설정 \*/

top: 170px; /\* 상단에서 170px만큼 떨어지게 설정 \*/

}

/\* 검색창 스타일 \*/

.search-container {

display: flex; /\* Flexbox 레이아웃 사용 \*/

justify-content: center; /\* 수평 중앙 정렬 \*/

align-items: center; /\* 수직 중앙 정렬 \*/

margin-top: 10px; /\* 검색창을 화면 아래로 이동 \*/

}

/\* 텍스트 입력창 스타일 \*/

input[type="text"] {

width: 500px; /\* 입력창 너비 설정 \*/

padding: 12px 16px; /\* 입력창 내 여백 설정 \*/

font-size: 30px; /\* 입력창 글자 크기 \*/

border: 2px solid #3498db; /\* 파란색 테두리 \*/

border-radius: 25px; /\* 입력창 테두리 둥글게 설정 \*/

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1); /\* 입력창에 부드러운 그림자 효과 추가 \*/

outline: none; /\* 입력창 포커스 시 외곽선 제거 \*/

transition: all 0.3s ease-in-out; /\* 포커스 시 애니메이션 효과 설정 \*/

}

/\* 입력창 포커스 시 스타일 \*/

input[type="text"]:focus {

border-color: #3498db; /\* 포커스 시 테두리 색상 변경 \*/

box-shadow: 0 2px 6px rgba(52, 152, 219, 0.5); /\* 포커스 시 그림자 강조 \*/

}

/\* 입력창 플레이스홀더 스타일 \*/

input[type="text"]::placeholder {

color: #aaa; /\* 플레이스홀더 글자 색상 \*/

font-style: italic; /\* 플레이스홀더 글자 이탤릭체 \*/

}

/\* 도서 정보 div 스타일 \*/

#bookInfo div {

background: linear-gradient(to right, #f3f5f5, #97c5f0);

padding: 70px; /\* 내부 여백 설정 \*/

margin: 10px 0; /\* 상하 여백 설정 \*/

border-radius: 15px; /\* 부드러운 둥글게 설정 \*/

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1); /\* 부드러운 그림자 효과 \*/

font-family: 'Lora', serif; /\* 고급스러운 폰트 사용 \*/

display: block; /\* 항목들이 세로로 정렬되도록 설정 \*/

text-align: left; /\* 텍스트 왼쪽 정렬 \*/

transition: transform 0.3s ease-in-out, box-shadow 0.3s ease-in-out; /\* 호버 효과 추가 \*/

}

/\* 카드 호버 시 스타일 \*/

#bookInfo div:hover {

transform: translateY(-5px); /\* 위로 살짝 떠오르는 효과 \*/

box-shadow: 0 6px 15px rgba(0, 0, 0, 0.15); /\* 그림자 강조 \*/

}

/\* 책 제목 스타일 \*/

#bookInfo h3 {

font-size: 32px; /\* 글자 크기 설정 (책 제목을 강조) \*/

color: #2c3e50; /\* 짙은 색상으로 제목 강조 \*/

font-weight: bold; /\* 굵은 글씨로 제목 강조 \*/

margin: 0; /\* 마진 제거 (상하 여백을 없애고 싶을 때) \*/

margin-bottom: 30px; /\* 제목 하단에 10px 여백 추가 (책 제목과 그 다음 항목 사이의 여백) \*/

}

/\* 책 정보 텍스트 스타일 \*/

#bookInfo p {

font-size: 24px; /\* 본문 텍스트의 크기를 24px로 설정하여 가독성을 높임 \*/

color: #000505; /\* 중간 검정색 \*/

margin: 5px 0; /\* 위아래 마진을 각각 5px로 설정하여 문단 간 간격을 균일하게 유지 \*/

}

/\* 책 이미지 스타일 \*/

.book-image {

width: 250px; /\* 책 이미지 크기 설정 \*/

height: auto; /\* 이미지 비율 유지 \*/

border-radius: 10px; /\* 이미지 둥글게 처리 \*/

margin-right: 20px; /\* 텍스트와 이미지 간의 여백 \*/

}

/\* 구분선(hr) 스타일 \*/

hr {

border: 0; /\* 기본 테두리 제거 \*/

border-top: 1px solid #ccc; /\* 상단에만 테두리 추가 \*/

}

/\* 책을 찾을 수 없습니다 문구 스타일 \*/

.no-results {

color: #ee0808 !important; /\* 빨간색 텍스트 \*/

font-size: 24px; /\* 글자 크기 설정 \*/

font-weight: bold; /\* 글씨 굵게 \*/

margin-top: 20px; /\* 상단 여백 추가 \*/

margin-bottom: 20px; /\* 하단 여백 추가 \*/

text-align: center; /\* 중앙 정렬 \*/

text-transform: uppercase; /\* 대문자 변환 \*/

animation: shake 0.5s ease-in-out; /\* 흔들리는 애니메이션 효과 \*/

}

/\* 검색 중 오류 문구 스타일 \*/

.error-message {

color: #e74c3c !important; /\* 빨간색 텍스트 \*/

font-size: 24px; /\* 글자 크기 설정 \*/

font-weight: bold; /\* 글씨 굵게 \*/

margin-top: 20px; /\* 상단 여백 추가 \*/

margin-bottom: 20px; /\* 하단 여백 추가 \*/

text-align: center; /\* 중앙 정렬 \*/

text-transform: uppercase; /\* 대문자 변환 \*/

animation: shake 0.5s ease-in-out; /\* 흔들리는 애니메이션 효과 \*/

}

/\* 흔들리는 애니메이션 \*/

@keyframes shake {

0% {

transform: translateX(-10px); /\* 왼쪽으로 10px 이동 \*/

}

25% {

transform: translateX(10px); /\* 오른쪽으로 10px 이동 \*/

}

50% {

transform: translateX(-10px); /\* 다시 왼쪽으로 10px 이동 \*/

}

75% {

transform: translateX(10px); /\* 다시 오른쪽으로 10px 이동 \*/

}

100% {

transform: translateX(0); /\* 원래 위치로 복귀 \*/

}

}

/\* 페이드 인 애니메이션 정의 \*/

@keyframes fadeIn {

from {

opacity: 0; /\* 시작 시 투명하게 \*/

transform: translateY(20px); /\* 살짝 아래에서 올라오는 효과 \*/

}

to {

opacity: 1; /\* 끝나는 시점에 보이게 \*/

transform: translateY(0); /\* 원래 위치로 돌아옴 \*/

}

}

/\* 책 정보 텍스트 스타일 \*/

#bookInfo p {

font-size: 24px; /\* 본문 텍스트의 크기를 24px로 설정하여 가독성을 높임 \*/

color: #000505; /\* 중간 검정색 \*/

margin: 5px 0; /\* 위아래 마진을 각각 5px로 설정하여 문단 간 간격을 균일하게 유지 \*/

}

/\* 책 이미지 스타일 \*/

.book-image {

width: 250px; /\* 책 이미지 크기 설정 \*/

height: auto; /\* 이미지 비율 유지 \*/

border-radius: 10px; /\* 이미지 둥글게 처리 \*/

margin-right: 20px; /\* 텍스트와 이미지 간의 여백 \*/

}

/\* 구분선(hr) 스타일 \*/

hr {

border: 0; /\* 기본 테두리 제거 \*/

border-top: 1px solid #ccc; /\* 상단에만 테두리 추가 \*/

}

/\* 책을 찾을 수 없습니다 문구 스타일 \*/

.no-results {

color: #ee0808 !important; /\* 빨간색 텍스트 \*/

font-size: 24px; /\* 글자 크기 설정 \*/

font-weight: bold; /\* 글씨 굵게 \*/

margin-top: 20px; /\* 상단 여백 추가 \*/

margin-bottom: 20px; /\* 하단 여백 추가 \*/

text-align: center; /\* 중앙 정렬 \*/

text-transform: uppercase; /\* 대문자 변환 \*/

animation: shake 0.5s ease-in-out; /\* 흔들리는 애니메이션 효과 \*/

}

/\* 검색 중 오류 문구 스타일 \*/

.error-message {

color: #e74c3c !important; /\* 빨간색 텍스트 \*/

font-size: 24px; /\* 글자 크기 설정 \*/

font-weight: bold; /\* 글씨 굵게 \*/

margin-top: 20px; /\* 상단 여백 추가 \*/

margin-bottom: 20px; /\* 하단 여백 추가 \*/

text-align: center; /\* 중앙 정렬 \*/

text-transform: uppercase; /\* 대문자 변환 \*/

animation: shake 0.5s ease-in-out; /\* 흔들리는 애니메이션 효과 \*/

}

/\* 흔들리는 애니메이션 \*/

@keyframes shake {

0% {

transform: translateX(-10px); /\* 왼쪽으로 10px 이동 \*/

}

25% {

transform: translateX(10px); /\* 오른쪽으로 10px 이동 \*/

}

50% {

transform: translateX(-10px); /\* 다시 왼쪽으로 10px 이동 \*/

}

75% {

transform: translateX(10px); /\* 다시 오른쪽으로 10px 이동 \*/

}

100% {

transform: translateX(0); /\* 원래 위치로 복귀 \*/

}

}

/\* 페이드 인 애니메이션 정의 \*/

@keyframes fadeIn {

from {

opacity: 0; /\* 시작 시 투명하게 \*/

transform: translateY(20px); /\* 살짝 아래에서 올라오는 효과 \*/

}

to {

opacity: 1; /\* 끝나는 시점에 보이게 \*/

transform: translateY(0); /\* 원래 위치로 돌아옴 \*/

}

}

/\* 책 스캔 중 문구 스타일 \*/

#scanStatus {

font-size: 40px; /\* 글자 크기 증가 \*/

font-weight: bold; /\* 글씨 굵게 \*/

color: #3498db; /\* 파란색 텍스트 \*/

font-family: 'Arial', sans-serif; /\* 폰트 설정 \*/

text-transform: uppercase; /\* 대문자 변환 \*/

margin-top: 20px; /\* 상단 여백 추가 \*/

margin-bottom: 20px; /\* 하단 여백 추가 \*/

animation: blink 1s step-end infinite; /\* 점 깜빡임 애니메이션 \*/

display: none; /\* 처음에는 숨겨두기 \*/

}

/\* 미디어 쿼리: 화면이 768px 이하일 때 \*/

@media screen and (max-width: 768px) {

/\* 전체 페이지 기본 스타일 유지 \*/

body {

padding: 10px; /\* 여백을 조금 더 줄여서 화면을 꽉 채움 \*/

font-size: 16px; /\* 작은 화면에 맞게 글자 크기 조정 \*/

}

/\* 헤더 \*/

header {

font-size: 1.5rem; /\* 모바일에서 글자 크기 줄이기 \*/

padding: 10px 15px; /\* 패딩 조정 \*/

}

/\* 제목(h1) \*/

h1 {

font-size: 35px; /\* 모바일에서 제목 글자 크기 축소 \*/

margin-top: 20px; /\* 제목 상단 여백 \*/

margin-bottom: 20px; /\* 제목 하단 여백 \*/

}

/\* 버튼 스타일 \*/

button {

font-size: 15px; /\* 모바일에서 버튼 글자 크기 축소 \*/

padding: 10px 20px; /\* 버튼 크기 줄이기 \*/

margin: 25px 10px; /\* 여백을 조정하여 화면에 맞게 배치 \*/

}

.button-container {

flex-wrap: nowrap; /\* 모바일에서도 줄 바꿈 없이 가로로 나열 \*/

}

.button-container button {

font-size: 15px; /\* 버튼 글자 크기 축소 \*/

padding: 10px 10px; /\* 버튼 여백 조정 \*/

}

/\* 이미지1 스타일 \*/

#image1 {

width: 40%; /\* 이미지를 화면 크기에 맞게 조정 \*/

margin-top: 20px; /\* 제목과 이미지 간의 간격 조정 \*/

margin-bottom: 15px; /\* 이미지 아래 여백 \*/

}

/\* 이미지2, 이미지3 스타일 - 모바일에서 절대 위치를 사용하지 않도록 수정 \*/

#image2 {

width: 40%; /\* 이미지 크기 조정 \*/

margin: 10px; /\* 이미지 간 간격 \*/

display: inline-block; /\* 세로로 정렬된 이미지를 좌우로 나열 \*/

vertical-align: top; /\* 수직 정렬을 위쪽으로 설정 \*/

top: 65px; /\* 상단에서 120px만큼 떨어지게 설정 \*/

right: calc(100% - 430px); /\* 왼쪽에서 450px 떨어지도록 설정 \*/

}

#image3{

width: 40%; /\* 이미지 크기 조정 \*/

margin: 10px; /\* 이미지 간 간격 \*/

display: inline-block; /\* 세로로 정렬된 이미지를 좌우로 나열 \*/

vertical-align: top; /\* 수직 정렬을 위쪽으로 설정 \*/

top: 80px; /\* 상단에서 120px만큼 떨어지게 설정 \*/

left: calc(100% - 440px); /\* 왼쪽에서 450px 떨어지도록 설정 \*/

}

/\* 이미지들이 같은 줄에 오도록 flexbox 사용 \*/

.image-wrapper {

display: flex; /\* 플렉스를 이용해 이미지 좌우 정렬 \*/

justify-content: space-between; /\* 좌우 이미지 간격을 일정하게 \*/

align-items: flex-start; /\* 상단 정렬 \*/

flex-wrap: wrap; /\* 화면에 맞게 이미지가 줄 바꿈하도록 \*/

margin-top: 20px; /\* 상단 여백 \*/

}

/\* 검색창 스타일 \*/

.search-container {

width: 100%; /\* 화면 전체를 차지하도록 설정 \*/

display: flex; /\* Flexbox 레이아웃 \*/

justify-content: center; /\* 중앙 정렬 \*/

margin-top: 20px; /\* 위 여백 \*/

}

/\* 텍스트 입력창 스타일 \*/

input[type="text"] {

width: 50%; /\* 입력창 너비를 100%로 설정하여 화면을 꽉 채움 \*/

font-size: 13px; /\* 폰트 크기 줄이기 \*/

padding: 12px; /\* 여백 조정 \*/

border-radius: 20px; /\* 둥근 테두리 \*/

}

/\* 책 정보 카드 스타일 \*/

#bookInfo div {

padding: 20px; /\* 내부 여백을 줄여서 화면에 맞게 \*/

margin: 10px 0; /\* 상하 여백 설정 \*/

border-radius: 10px; /\* 둥글게 처리 \*/

}

/\* 책 제목 스타일 \*/

#bookInfo h3 {

font-size: 24px; /\* 책 제목 폰트 크기 줄이기 \*/

margin-bottom: 15px; /\* 하단 여백 \*/

}

/\* 책 이미지 스타일 \*/

.book-image {

width: 150px; /\* 책 이미지 크기 조정 \*/

}

/\* 책 정보 텍스트 \*/

#bookInfo p {

font-size: 16px; /\* 텍스트 크기 줄이기 \*/

}

/\* 책을 찾을 수 없습니다 문구 스타일 \*/

.no-results {

font-size: 20px; /\* 글자 크기 축소 \*/

margin-top: 20px;

text-transform: none; /\* 대문자 변환 제거 \*/

}

/\* 검색 중 오류 문구 스타일 \*/

.error-message {

font-size: 20px; /\* 글자 크기 축소 \*/

}

/\* 스캔 중 문구 스타일 \*/

#scanStatus {

font-size: 30px; /\* 글자 크기 축소 \*/

}

}

/\* 전체 페이지의 기본 스타일 설정 \*/

body {

font-family: 'Arial', sans-serif; /\* 폰트 설정 \*/

background-color: #f5f5f5; /\* 배경색 설정 \*/

margin: 0; /\* 기본 여백 제거 \*/

padding: 20px; /\* 페이지 여백 설정 \*/

color: #333; /\* 기본 텍스트 색상 \*/

text-align: center; /\* 텍스트를 중앙 정렬 \*/;

}

/\* 제목(h1)의 스타일 설정 \*/

h1 {

color: #2c3e50; /\* 글자 색상 설정 \*/

font-size: 50px; /\* 글자 크기 확대 \*/

font-weight: bold; /\* 굵은 글씨 \*/

font-family: 'Poppins', sans-serif; /\* 고급스러운 폰트 \*/

text-transform: uppercase; /\* 모든 글자를 대문자로 \*/

letter-spacing: 2px; /\* 글자 사이 간격 추가 \*/

margin-bottom: 30px; /\* 제목 하단 여백 \*/

margin-top: 30px; /\* 제목 상단 여백 (아래로 이동) \*/

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.1); /\* 제목에 그림자 효과 추가 \*/

}

header {

background-color: #3498db !important; /\* 파란색 배경 \*/

padding: 40px 20px; /\* 상하 40px, 좌우 20px 패딩 \*/

color: white; /\* 텍스트 흰색 \*/

font-size: 2rem; /\* 헤더 텍스트 크기 \*/

text-align: center; /\* 중앙 정렬 \*/

}

/\* 메인화면으로 돌아가기 버튼 스타일 \*/

.back-button {

padding: 12px 24px; /\* 충분한 여백을 주어 버튼 크기 키우기 \*/

font-size: 30px; /\* 글자 크기 조정 \*/

font-weight: bold; /\* 글자 두껍게 \*/

text-transform: uppercase; /\* 대문자로 변환 \*/

border: none; /\* 기본 버튼 테두리 제거 \*/

background-color: #0b63e7; /\* 파란색 배경 \*/

color: white; /\* 흰색 글자 \*/

border-radius: 25px; /\* 둥근 모서리 \*/

cursor: pointer; /\* 커서가 버튼 위에 올 때 포인터로 바뀌게 \*/

transition: background-color 0.3s, transform 0.3s; /\* 배경 색상 및 크기 변화 시 부드럽게 \*/

position: absolute; /\* 절대 위치 설정 \*/

top: 80px; /\* 화면 상단에서 80px 아래로 \*/

right: 80px; /\* 화면 오른쪽에서 80px 떨어지게 \*/

}

/\* 버튼에 마우스를 올렸을 때 효과 \*/

.back-button:hover {

background-color: #0410bb; /\* 어두운 파란색으로 변화 \*/

transform: scale(1.1); /\* 크기를 조금 키워주는 효과 \*/

}

/\* 버튼에 클릭했을 때 효과 \*/

.back-button:active {

background-color: #388e3c; /\* 클릭했을 때 색상 \*/

transform: scale(0.95); /\* 클릭 시 버튼 크기 작게 \*/

}

/\* 버튼의 포커스 상태 (키보드로 탭하여 이동 시) \*/

.back-button:focus {

outline: none; /\* 기본 포커스 테두리 제거 \*/

box-shadow: 0 0 5px rgba(0, 0, 0, 0.5); /\* 그림자 효과 추가 \*/

}

/\* 개별 책장의 스타일 \*/

.shelf-item {

display: flex;

flex-direction: column;

align-items: center;

justify-content: space-between;

width: 100px; /\* 기본 가로 크기 짧게 설정 \*/

height: 180px; /\* 기본 세로 크기 유지 \*/

padding: 10px;

border: 1px solid #ddd;

border-radius: 10px; /\* 둥근 테두리 \*/

background-color: #f5f5f5; /\* 배경색 \*/

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /\* 그림자 효과 \*/

transition: all 0.3s ease-in-out; /\* 모든 속성에 부드러운 애니메이션 효과 추가 \*/

animation: fadeIn 1s ease-in-out; /\* 로드 시 페이드인 \*/

}

/\* 서서히 나타나는(fade in) 애니메이션 정의 \*/

@keyframes fadeIn {

from {

opacity: 0; /\* 완전히 투명한 상태에서 시작 \*/

}

to {

opacity: 1; /\* 완전히 불투명한 상태로 끝남 \*/

}

}

.shelf-item:hover {

width: 220px; /\* 가로 크기 확대 \*/

height: 220px; /\* 세로 크기 확대 \*/

transform: scale(1.1); /\* 전체 크기 확대 \*/

box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2); /\* 그림자 확대 \*/

}

/\* 책장 항목 안의 이미지에 마우스를 올렸을 때 확대 효과 \*/

.shelf-item:hover img {

width: 100px; /\* 이미지 크기를 키움 \*/

height: auto; /\* 비율 유지 \*/

transition: all 0.3s ease-in-out; /\* 이미지 확대도 부드럽게 \*/

}

.status-summary {

position: absolute; /\* 부모 요소를 기준으로 절대 위치 지정 \*/

top: 250px; /\* 위쪽에서 250픽셀 떨어진 위치에 배치 \*/

right: 100px; /\* 오른쪽에서 100픽셀 떨어진 위치에 배치 \*/

font-size: 24px; /\* 큰 텍스트로 가독성 향상 \*/

font-weight: 600; /\* 중간 굵기의 텍스트 \*/

background-color: #faf3e0; /\* 따뜻한 크림 베이지 배경 \*/

color: #4b3a27; /\* 깊이 있는 브라운 색상 \*/

padding: 25px; /\* 충분한 여백으로 텍스트 공간 확보 \*/

border-radius: 20px; /\* 더 부드러운 둥근 모서리 \*/

box-shadow: 0 10px 20px rgba(0, 0, 0, 0.3); /\* 깊고 은은한 그림자 \*/

border: 3px solid #d1b591; /\* 고급스러운 황갈색 테두리 \*/

font-family: 'Palatino Linotype', 'Book Antiqua', serif; /\* 클래식 서체 \*/

text-align: left; /\* 텍스트 왼쪽 정렬 \*/

line-height: 1.6; /\* 가독성을 높이기 위한 줄 간격 \*/

background-image: linear-gradient(to bottom, #faf3e0, #f0e3c2); /\* 은은한 그라데이션 \*/

}

.status-summary::after {

content: ""; /\* 가상 요소 생성, 내용은 없음 \*/

position: absolute; /\* 부모 요소를 기준으로 절대 위치 지정 \*/

bottom: -30px; /\* 돼지꼬리가 박스에서 떨어지도록 설정 \*/

left: -25px; /\* 왼쪽 끝에 배치 \*/

width: 20px; /\* 요소의 가로 길이 20픽셀 \*/

height: 20px; /\* 요소의 세로 길이 20픽셀 \*/

border-radius: 50%; /\* 원형 모양으로 만들기 \*/

background-color: #faf3e0; /\* 배경색상과 동일 \*/

box-shadow: 0 0 0 5px #d1b591; /\* 테두리 효과 \*/

transform: rotate(45deg); /\* 회전 효과로 돼지꼬리 느낌 추가 \*/

}

.status-summary::before {

content: ""; /\* 가상 요소 생성, 내용은 없음 \*/

position: absolute; /\* 부모 요소를 기준으로 절대 위치 지정 \*/

bottom: -50px; /\* 두 번째 돼지꼬리 위치 (첫 번째 꼬리 아래) \*/

left: -40px; /\* 두 번째 꼬리 왼쪽 위치 \*/

width: 15px; /\* 크기 작게 설정 \*/

height: 15px; /\* 크기 작게 설정 \*/

border-radius: 50%; /\* 원형 모양으로 만들기 \*/

background-color: #faf3e0; /\* 배경색상과 동일 \*/

box-shadow: 0 0 0 4px #d1b591; /\* 테두리 크기도 줄임 \*/

transform: rotate(45deg); /\* 회전 효과로 돼지꼬리 느낌 추가 \*/

}

.status-summary p {

margin: 15px 0; /\* 텍스트 간격 \*/

font-size: 22px; /\* 텍스트 크기 \*/

font-weight: 500; /\* 가독성을 위한 중간 굵기 \*/

color: #5c4a33; /\* 약간 더 밝은 갈색으로 강조 \*/

}

.status-summary span {

font-weight: 700; /\* 강조된 텍스트 \*/

color: #9b7847; /\* 따뜻한 금빛 톤으로 포인트 \*/

text-decoration: underline; /\* 강조 효과를 위해 밑줄 추가 \*/

}

.filter-buttons {

display: flex;

gap: 15px; /\* 버튼 간 간격을 조금 더 넓게 \*/

justify-content: center; /\* 버튼들을 컨테이너 가로 중앙에 정렬 \*/

margin-bottom: 25px; /\* 버튼 아래 여백 \*/

}

.filter-buttons button {

padding: 12px 24px; /\* 버튼 크기 조정 \*/

border: none; /\* 버튼의 기본 테두리 제거 \*/

border-radius: 8px; /\* 둥근 모서리 \*/

font-size: 16px; /\* 버튼 글자 크기를 16px로 설정 \*/

font-weight: bold; /\* 글자를 굵게 표시 \*/

cursor: pointer; /\* 마우스 커서를 손가락 모양으로 변경해 클릭 가능함을 표시 \*/

transition: all 0.3s ease; /\* 부드러운 전환 효과 \*/

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /\* 약간의 그림자 효과 \*/

}

/\* 호버 시 버튼 밝아지는 효과 \*/

.filter-buttons button:hover {

transform: translateY(-3px); /\* 위로 살짝 이동 \*/

filter: brightness(1.1); /\* 버튼 색상을 더 밝게 \*/

}

/\* 눌렀을 때 살짝 눌리는 효과 \*/

.filter-buttons button:active {

transform: translateY(1px); /\* 아래로 살짝 이동 \*/

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2); /\* 그림자 줄이기 \*/

}

.shelf-item:hover {

border: 2px solid #000307; /\* 마우스 오버 시, 검정에 가까운 진한 회색 테두리 2px로 표시 \*/

background-color: #f8f9fa; /\* 마우스 오버 시, 밝은 회색 배경색으로 변경 \*/

}

.shelf-group {

border: 2px solid #ccc; /\* 네 면 모두 실선 \*/

width: 60%; /\* 가로 길이를 전체 폭의 60%로 설정 \*/

margin: 0 auto 10px auto; /\* 수직 간격과 가로 중앙 정렬 \*/

padding: 10px; /\* 내부 여백 \*/

box-sizing: border-box; /\* 테두리를 포함해 크기를 정확히 계산 \*/

}

/\* image4 스타일 \*/

#image4 {

width: 450px; /\* 이미지 크기 설정 \*/

position: absolute; /\* 절대 위치 지정 \*/

right: calc(100% - 1290px); /\* 왼쪽에서 470px 떨어지도록 설정 \*/

top: 400px; /\* 상단에서 50px만큼 떨어지게 설정 \*/

}

@media screen and (max-width: 768px) {

.back-button {

font-size: 18px; /\* 버튼 글자 크기 줄이기 \*/

padding: 10px 20px;

position: absolute; /\* 절대 위치 설정 \*/

top: 90px; /\* 화면 상단에서 80px 아래로 \*/

right: 10px; /\* 화면 오른쪽에서 10px 떨어지게 \*/

}

}

/\* 반응형 모바일 화면 (max-width: 768px) \*/

@media screen and (max-width: 768px) {

body {

padding: 10px; /\* 모바일에서는 여백을 줄여줍니다. \*/

text-align: left; /\* 모바일에서는 텍스트 왼쪽 정렬 \*/

}

h1 {

font-size: 30px; /\* 제목 크기를 줄여줍니다. \*/

margin-top: 20px;

}

header {

font-size: 1.5rem; /\* 헤더 텍스트 크기를 작게 \*/

padding: 18px; /\* 패딩 줄이기 \*/

}

.back-button {

font-size: 15px; /\* 버튼 글자 크기 줄이기 \*/

padding: 10px 20px;

top: 20px; /\* 화면 상단에서 위치 조정 \*/

right: 20px; /\* 오른쪽 위치 조정 \*/

position: absolute; /\* 절대 위치 설정 \*/

top: 90px; /\* 화면 상단에서 90px 아래로 \*/

right: 15px; /\* 화면 오른쪽에서 15px 떨어지게 \*/

}

.shelf-item {

width: 80px; /\* 모바일에서는 책장의 크기를 작게 \*/

height: 150px; /\* 세로 크기 줄이기 \*/

padding: 8px; /\* 여백 줄이기 \*/

}

.shelf-item:hover {

width: 160px; /\* 가로 크기 줄이기 \*/

height: 160px; /\* 세로 크기 줄이기 \*/

}

.status-summary {

position: static; /\* 위치를 일반 흐름에 따르게 설정 \*/

margin: 0 auto 20px auto;; /\* 위 아래 여백 + 중앙 정렬 \*/

transform: translateY(-100px); /\* 요소를 수직 방향으로 위쪽으로 100픽셀 이동시킴 \*/

top: auto; /\* 필요 없음 \*/

right: auto; /\* 필요 없음 \*/

font-size: 20px;

padding: 15px;

width: 60%; /\* 화면 폭에 맞게 넓게 \*/

max-width: 140px; /\* 최대 폭 제한 \*/

}

.filter-buttons {

gap: 10px; /\* 간격 줄이기 \*/

}

.shelf-group {

width: 90%; /\* 책장 그룹의 가로 크기를 늘림 \*/

margin: 10px auto; /\* 수직 여백 추가 \*/

transform: translateY(300px); /\* 책장을 20px 아래로 이동 \*/

}

/\* 이미지 스타일 \*/

#image4 {

width: 250px; /\* 이미지 크기를 모바일에 맞게 조정 \*/

right: 200px; /\* 오른쪽 위치를 조정 \*/

top: 280px; /\* 위치 조정 \*/

}

.status-summary::before {

content: ""; /\* 가상 요소 내용 비움 \*/

position: absolute; /\* 부모 기준 절대 위치 지정 \*/

bottom: -20px; /\* 두 번째 돼지꼬리 위치 (첫 번째 꼬리 아래) \*/

left: -50px; /\* 두 번째 꼬리 왼쪽 위치 \*/

width: 15px; /\* 크기 작게 설정 \*/

height: 15px; /\* 크기 작게 설정 \*/

border-radius: 50%; /\* 둥근 원 모양 \*/

background-color: #faf3e0;

box-shadow: 0 0 0 4px #d1b591; /\* 테두리 크기도 줄임 \*/

transform: rotate(45deg); /\* 45도 회전시켜 돼지꼬리 모양 연출 \*/

}

}

* 도서 정보 웹페이지 디자인 소스코드

/\* 전체 배경 및 폰트 스타일 \*/

body {

font-family: Arial, sans-serif; /\* 전체 폰트 스타일을 Arial로 설정 \*/

background-color: #3498db; /\* 페이지 배경색을 밝은 파란색으로 설정 \*/

margin: 20px; /\* 페이지 전체 여백을 20px로 설정 \*/

color: #000000; /\* 기본 텍스트 색상을 어두운 회색으로 설정 \*/

}

h1 {

text-align: center; /\* 제목을 중앙 정렬 \*/

color: #2c3e50; /\* 제목 색상을 어두운 파란색으로 설정 \*/

font-size: 3rem; /\* 제목 크기를 3배로 설정 \*/

margin-bottom: 20px; /\* 제목 아래 여백을 20px로 설정 \*/

}

/\* 테이블 스타일 \*/

table {

width: 100%; /\* 테이블의 너비를 부모 요소에 맞게 100%로 설정 \*/

border-collapse: collapse; /\* 테이블 테두리를 겹치도록 설정 \*/

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /\* 테이블에 그림자 효과 추가 \*/

background-color: rgb(255, 255, 255); /\* 테이블 배경색을 흰색으로 설정 \*/

}

th, td {

padding: 15px; /\* 테이블 셀 내부 여백을 15px로 설정 \*/

border: 1px solid #ffffff; /\* 테이블 셀 테두리 색상을 흰색으로 설정 \*/

text-align: left; /\* 텍스트를 왼쪽 정렬 \*/

}

th {

background-color: #007bff; /\* 테이블 헤더의 배경색을 파란색으로 설정 \*/

color: white; /\* 테이블 헤더 텍스트 색상을 흰색으로 설정 \*/

font-weight: bold; /\* 테이블 헤더 텍스트를 두껍게 설정 \*/

}

/\* 도서 상태 색상 \*/

.available {

background-color: green; /\* 책 있음 상태를 초록색으로 표시 \*/

color: white; /\* 텍스트 색상을 흰색으로 설정 \*/

padding: 5px 10px; /\* 상태 표시 내부 여백 설정 \*/

border-radius: 5px; /\* 상태 표시 모서리를 둥글게 처리 \*/

}

.not-available {

background-color: red; /\* 책 없음 상태를 빨간색으로 표시 \*/

color: white; /\* 텍스트 색상을 흰색으로 설정 \*/

padding: 5px 10px; /\* 상태 표시 내부 여백 설정 \*/

border-radius: 5px; /\* 상태 표시 모서리를 둥글게 처리 \*/

}

.misplaced {

background-color: yellow; /\* 책이 잘못 배치된 상태를 노란색으로 표시 \*/

color: black; /\* 텍스트 색상을 검정으로 설정 \*/

padding: 5px 10px; /\* 상태 표시 내부 여백 설정 \*/

border-radius: 5px; /\* 상태 표시 모서리를 둥글게 처리 \*/

}

.wrong-location {

background-color: purple; /\* 책이 잘못된 위치에 있는 상태를 보라색으로 표시 \*/

color: white; /\* 텍스트 색상을 흰색으로 설정 \*/

padding: 5px 10px; /\* 상태 표시 내부 여백 설정 \*/

border-radius: 5px; /\* 상태 표시 모서리를 둥글게 처리 \*/

}

/\* 이미지 스타일 \*/

td img {

border-radius: 8px; /\* 이미지의 모서리를 둥글게 처리 \*/

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1); /\* 이미지에 약간의 그림자 효과 추가 \*/

}

/\* 메인화면으로 돌아가기 버튼 스타일 \*/

.back-button {

padding: 12px 24px; /\* 충분한 여백을 주어 버튼 크기 키우기 \*/

font-size: 30px; /\* 글자 크기 조정 \*/

font-weight: bold; /\* 글자 두껍게 \*/

text-transform: uppercase; /\* 대문자로 변환 \*/

border: none; /\* 기본 버튼 테두리 제거 \*/

background-color: #0b63e7; /\* 파란색 배경 \*/

color: white; /\* 흰색 글자 \*/

border-radius: 25px; /\* 둥근 모서리 \*/

cursor: pointer; /\* 커서가 버튼 위에 올 때 포인터로 바뀌게 \*/

transition: background-color 0.3s, transform 0.3s; /\* 배경 색상 및 크기 변화 시 부드럽게 \*/

position: absolute; /\* 절대 위치 설정 \*/

top: 25px; /\* 화면 상단에서 25px 아래로 \*/

right: 90px; /\* 화면 오른쪽에서 90px 떨어지게 \*/

}

/\* 버튼에 마우스를 올렸을 때 효과 \*/

.back-button:hover {

background-color: #0410bb; /\* 어두운 파란색으로 변화 \*/

transform: scale(1.1); /\* 크기를 조금 키워주는 효과 \*/

}

/\* 버튼에 클릭했을 때 효과 \*/

.back-button:active {

background-color: #388e3c; /\* 클릭했을 때 색상 \*/

transform: scale(0.95); /\* 클릭 시 버튼 크기 작게 \*/

}

/\* 버튼의 포커스 상태 (키보드로 탭하여 이동 시) \*/

.back-button:focus {

outline: none; /\* 기본 포커스 테두리 제거 \*/

box-shadow: 0 0 5px rgba(0, 0, 0, 0.5); /\* 그림자 효과 추가 \*/

}

tr:nth-child(even) {

background-color: #f2f2f2; /\* 짝수 번째 테이블 행에 연한 회색 배경색 적용 \*/

}

tr:hover {

background-color: #d9eaf7; /\* 마우스를 올린 행에 연한 파란색 배경 적용 \*/

transform: scale(1.02); /\* 살짝 확대 \*/

transition: transform 0.2s, background-color 0.2s; /\* 배경색과 확대 효과가 0.2초 동안 부드럽게 변하도록 설정 \*/

}

td img:hover {

transform: scale(1.5); /\* 이미지에 마우스를 올리면 1.5배 확대 \*/

transition: transform 0.3s; /\* 확대 효과가 0.3초 동안 부드럽게 적용 \*/

box-shadow: 0 8px 15px rgba(0, 0, 0, 0.2); /\* 이미지 확대 시 그림자도 강하게 \*/

}

/\* 미디어 쿼리: 화면이 768px 이하일 때 \*/

@media screen and (max-width: 768px) {

/\* 메인화면으로 돌아가기 버튼 스타일 \*/

.back-button {

padding: 12px 12px; /\* 충분한 여백을 주어 버튼 크기 키우기 \*/

}

}

* 파이와연결된 바코드스캐너를 동작시키는 소스코드

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <termios.h>

#include <jansson.h>

#include <curl/curl.h>

#include <ctype.h>

#include <signal.h>

#include <sys/types.h>

#include <sys/wait.h>

#define SERVER\_URL "http://223.194.155.129:5000/update\_book\_status"

#define MAX\_BARCODE\_LEN 128

#define MAX\_BARCODES 100

#define MAX\_COMPLETED\_SHELVES 100

pid\_t scanner\_pid; // Python 스크립트 프로세스 PID

char completed\_shelves[MAX\_COMPLETED\_SHELVES][MAX\_BARCODE\_LEN];

int completed\_shelf\_count = 0;

// 터미널을 비차단 모드로 설정

void set\_terminal\_mode(struct termios \*original) {

struct termios new\_settings;

tcgetattr(STDIN\_FILENO, original);

new\_settings = \*original;

new\_settings.c\_lflag &= ~(ICANON | ECHO);

tcsetattr(STDIN\_FILENO, TCSANOW, &new\_settings);

}

// 터미널을 원래 상태로 복원

void reset\_terminal\_mode(struct termios \*original) {

tcsetattr(STDIN\_FILENO, TCSANOW, original);

}

// 중복 바코드인지 확인

int is\_duplicate(char barcodes[][MAX\_BARCODE\_LEN], int count, const char \*barcode) {

for (int i = 0; i < count; i++) {

if (strcmp(barcodes[i], barcode) == 0) {

return 1;

}

}

return 0;

}

// 완료된 책장인지 확인

int is\_completed\_shelf(const char \*barcode) {

for (int i = 0; i < completed\_shelf\_count; i++) {

if (strcmp(completed\_shelves[i], barcode) == 0) {

return 1;

}

}

return 0;

}

// 완료된 책장 추가

void add\_completed\_shelf(const char \*barcode) {

if (completed\_shelf\_count < MAX\_COMPLETED\_SHELVES) {

strcpy(completed\_shelves[completed\_shelf\_count++], barcode);

} else {

fprintf(stderr, "완료된 책장 버퍼가 가득 찼습니다.\n");

}

}

// 책장 코드인지 판별 (대소문자 구분 없이 -f로 시작)

int is\_shelf\_code(const char \*buffer) {

return strchr(buffer, '-') && tolower(buffer[1]) == 'f';

}

// JSON 데이터를 생성하여 서버에 전송

void send\_to\_server(const char \*location, char barcodes[][MAX\_BARCODE\_LEN], int count) {

CURL \*curl;

CURLcode res;

struct curl\_slist \*headers = NULL;

json\_t \*root, \*barcode\_array;

char \*json\_data;

root = json\_object();

barcode\_array = json\_array();

for (int i = 0; i < count; i++) {

json\_t \*barcode\_str = json\_string(barcodes[i]);

if (barcode\_str) {

json\_array\_append\_new(barcode\_array, barcode\_str);

} else {

fprintf(stderr, "바코드 문자열 변환 실패: %s\n", barcodes[i]);

}

}

json\_object\_set\_new(root, location, barcode\_array);

json\_data = json\_dumps(root, 0);

json\_decref(root);

if (!json\_data) {

fprintf(stderr, "JSON 변환 실패. 데이터가 전송되지 않습니다.\n");

return;

}

curl\_global\_init(CURL\_GLOBAL\_ALL);

curl = curl\_easy\_init();

if (curl) {

headers = curl\_slist\_append(headers, "Content-Type: application/json");

curl\_easy\_setopt(curl, CURLOPT\_URL, SERVER\_URL);

curl\_easy\_setopt(curl, CURLOPT\_POSTFIELDS, json\_data);

curl\_easy\_setopt(curl, CURLOPT\_HTTPHEADER, headers);

res = curl\_easy\_perform(curl);

if (res != CURLE\_OK) {

fprintf(stderr, "curl\_easy\_perform() failed: %s\n", curl\_easy\_strerror(res));

} else {

printf("%s 책장의 데이터가 서버로 전송되었습니다.\n", location);

}

curl\_slist\_free\_all(headers);

curl\_easy\_cleanup(curl);

} else {

fprintf(stderr, "curl 초기화 실패\n");

}

curl\_global\_cleanup();

free(json\_data);

}

// scanner\_trigger.py 종료

void terminate\_scanner\_trigger() {

if (scanner\_pid > 0) {

printf("scanner\_trigger.py 종료 시도 중 (PID: %d)...\n", scanner\_pid);

kill(scanner\_pid, SIGTERM);

sleep(1);

if (kill(scanner\_pid, 0) == 0) {

kill(scanner\_pid, SIGKILL);

printf("scanner\_trigger.py가 강제 종료되었습니다.\n");

} else {

printf("scanner\_trigger.py 정상 종료됨.\n");

}

waitpid(scanner\_pid, NULL, 0); // 좀비 방지

}

}

int main() {

// 백그라운드로 scanner\_trigger.py 실행

scanner\_pid = fork();

if (scanner\_pid == 0) {

execlp("python3", "python3", "scanner\_trigger.py", NULL);

perror("execlp 실패");

exit(1);

} else if (scanner\_pid < 0) {

perror("fork 실패");

return 1;

}

struct termios original\_settings;

char buffer[MAX\_BARCODE\_LEN];

int index = 0;

char scanning\_location[MAX\_BARCODE\_LEN] = "";

char barcodes[MAX\_BARCODES][MAX\_BARCODE\_LEN];

int barcode\_count = 0;

int unique\_count = 0;

set\_terminal\_mode(&original\_settings);

printf("바코드를 스캔하세요 (종료하려면 'exit' 입력):\n");

while (1) {

char c = getchar();

if (c == '\n') {

buffer[index] = '\0';

if (strcmp(buffer, "exit") == 0) {

break;

}

if (is\_shelf\_code(buffer)) {

if (is\_completed\_shelf(buffer)) {

printf("이미 완료된 책장(%s)입니다. 무시합니다.\n", buffer);

} else if (strlen(scanning\_location) == 0) {

strcpy(scanning\_location, buffer);

barcode\_count = 0;

unique\_count = 0;

printf("%s 책장 스캔 시작!\n", scanning\_location);

} else if (strcmp(scanning\_location, buffer) == 0) {

if (unique\_count >= 2) {

printf("%s 책장 스캔 종료! 데이터 전송 중...\n", scanning\_location);

send\_to\_server(scanning\_location, barcodes, barcode\_count);

add\_completed\_shelf(scanning\_location);

if (strcasecmp(scanning\_location, "1F-1-A-2-a") == 0) {

printf("GPIO 제어 신호 전송 중 (linear\_gpio24.py 실행)...\n");

system("python3 linear\_gpio24.py"); //linear up

}

if (strcasecmp(scanning\_location, "2F-1-A-1-a") == 0) {

printf("GPIO 제어 신호 전송 중 (linear\_gpio23.py 실행)...\n");

system("python3 linear\_gpio23.py"); //linear down

}

scanning\_location[0] = '\0';

printf("바코드를 스캔하세요 (종료하려면 'exit' 입력):\n");

} else {

printf("책을 최소 2권 이상 스캔해야 책장 스캔을 종료할 수 있습니다.\n");

}

} else {

printf("다른 책장(%s)의 스캔이 진행 중입니다. 먼저 종료하세요.\n", scanning\_location);

}

} else if (strlen(scanning\_location) > 0) {

if (is\_duplicate(barcodes, barcode\_count, buffer)) {

printf("중복된 책 바코드(%s)입니다. 무시됩니다.\n", buffer);

} else if (barcode\_count < MAX\_BARCODES) {

strcpy(barcodes[barcode\_count++], buffer);

unique\_count++;

printf("책 %s 스캔됨. (버퍼에 저장됨)\n", buffer);

} else {

printf("버퍼가 가득 찼습니다. 책장 스캔을 종료하고 데이터를 전송하세요.\n");

}

} else {

printf("먼저 책장 바코드를 입력하여 스캔을 시작하세요.\n");

}

index = 0;

} else {

if (index < MAX\_BARCODE\_LEN - 1) {

buffer[index++] = c;

}

}

}

reset\_terminal\_mode(&original\_settings);

terminate\_scanner\_trigger(); // 종료 시 Python 프로세스도 종료

printf("프로그램을 종료합니다.\n");

return 0;

}

* 아두이노23, 24핀에 연결된 슬라이더를 제어하는 파이 핀

import lgpio

import time

import sys

h = lgpio.gpiochip\_open(0) # 기본 GPIO 칩 열기

MOVE\_PIN = 24

lgpio.gpio\_claim\_output(h, MOVE\_PIN)

lgpio.gpio\_write(h, MOVE\_PIN, 1) # 이동 LOW

print("(GPIO24 HIGH)")

time.sleep(1)

lgpio.gpio\_write(h, MOVE\_PIN, 0) # 정지 HIGH

print("(GPIO24 LOW)")

lgpio.gpiochip\_close(h)

import lgpio

import time

import sys

h = lgpio.gpiochip\_open(0) # 기본 GPIO 칩 열기

MOVE\_PIN = 23

lgpio.gpio\_claim\_output(h, MOVE\_PIN)

lgpio.gpio\_write(h, MOVE\_PIN, 1) # 이동 LOW

print("(GPIO23 HIGH)")

time.sleep(1)

lgpio.gpio\_write(h, MOVE\_PIN, 0) # 정지 HIGH

print("(GPIO23 LOW)")

lgpio.gpiochip\_close(h)

* 바코드스캐너의 트랜지스터 게이트를 제어하는 파이 소스코드

import lgpio

import time

import signal

import sys

scan\_trigger = 12

# GPIO 핸들러 열기

h = lgpio.gpiochip\_open(0)

running = True

def handle\_sigterm(signum, frame):

global running

running = False

signal.signal(signal.SIGTERM, handle\_sigterm)

try:

while True:

# ON

lgpio.gpio\_write(h, scan\_trigger, 1)

time.sleep(0.075)

# OFF

lgpio.gpio\_write(h, scan\_trigger, 0)

time.sleep(0.075)

except KeyboardInterrupt:

print("exit")

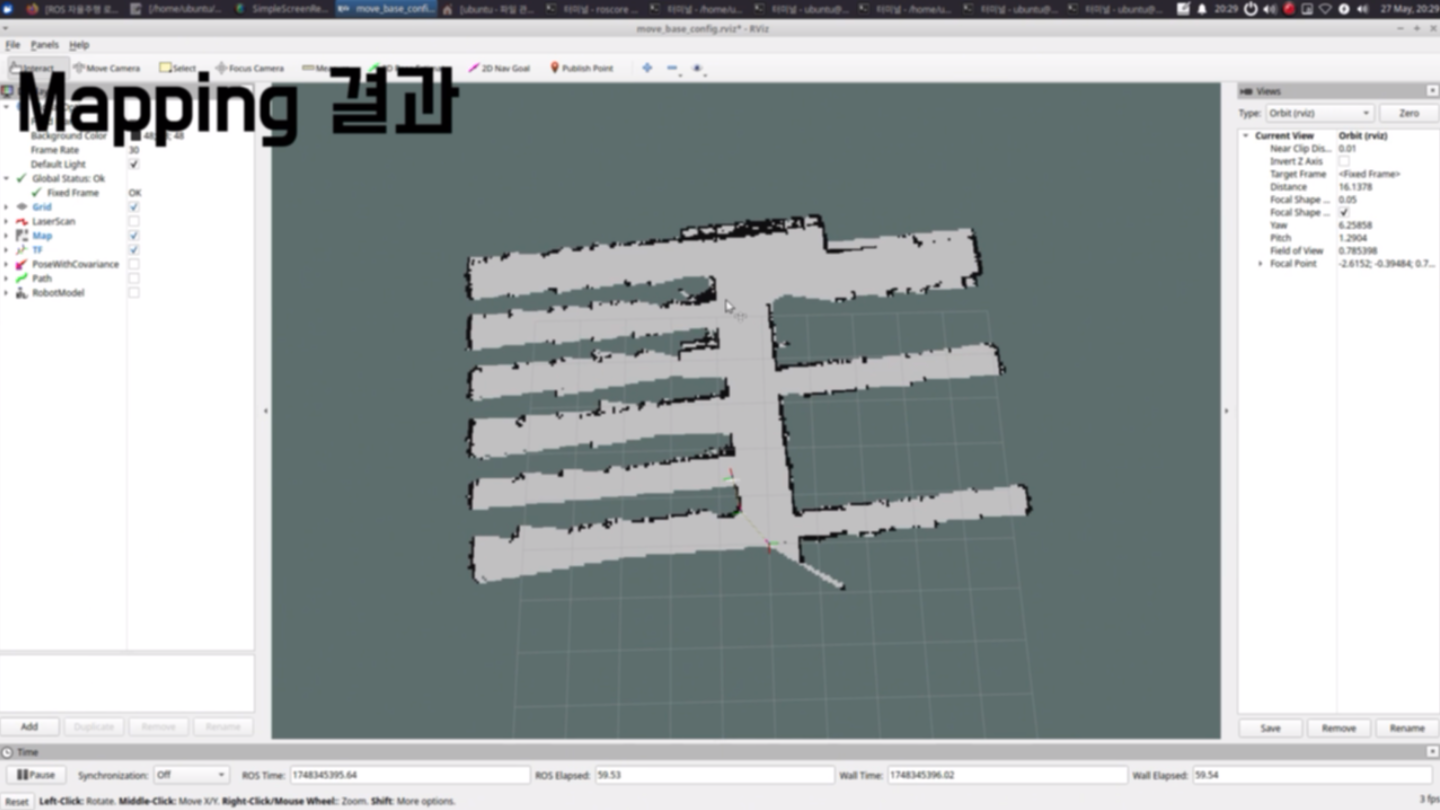
finally:

lgpio.gpiochip\_close(h)

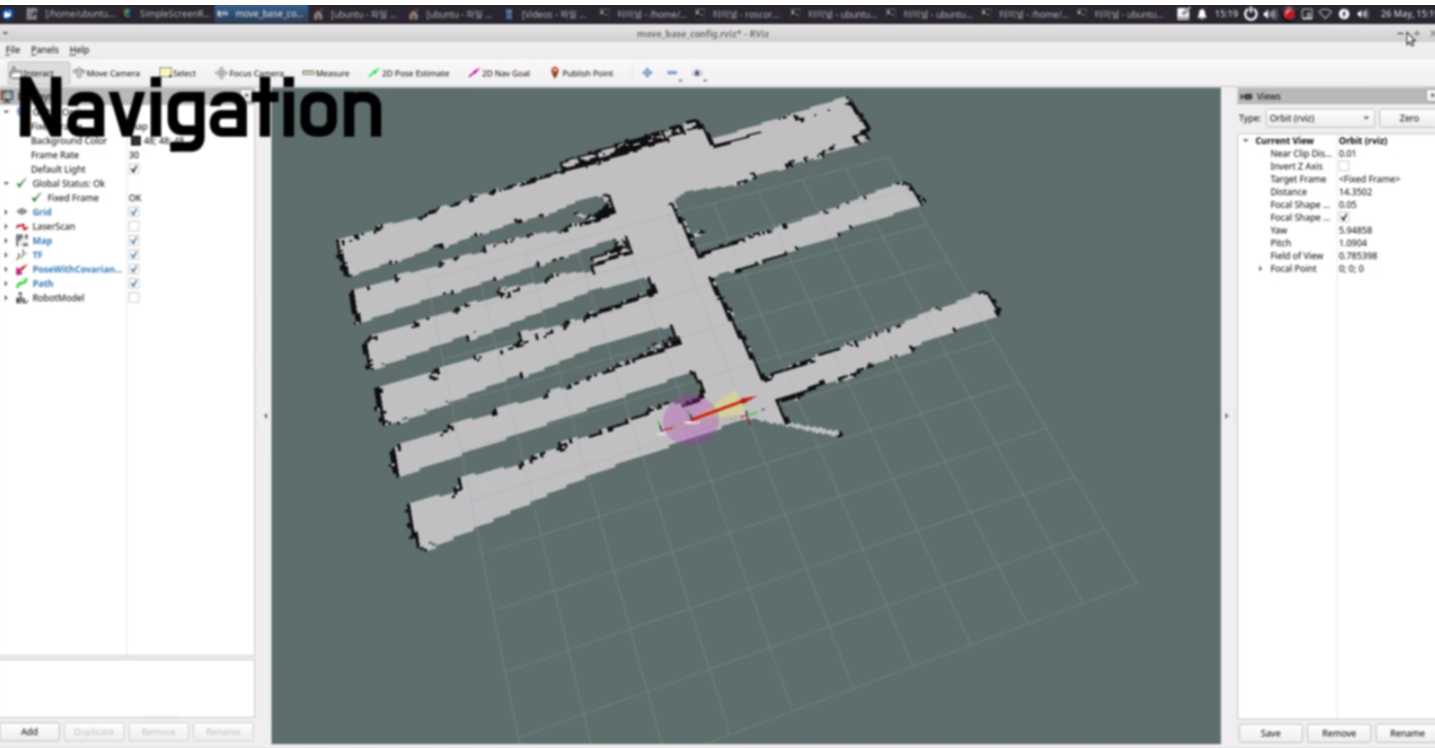
sys.exit(0)

**VII. 결과물 설명 (결과물 관련 이미지 첨부)**

* 지도 작성 (Mapping): Hector SLAM을 통해 벽, 장애물 등 환경의 구조를 정확히 반영한 2D 지도 작성 성공



* 오도메트리 결과: Encoder 기반의 실시간 위치 추정이 /odom 토픽으로 시각화됨 (RViz 캡처 첨부 가능)



* 웹에서 버튼 클릭 시, 로봇이 책장 스캔 동작 수행 및 데이터 전송하여 서버에서 처리하도록 함



* 웹서버에서 도서 목록 보기를 클릭하면 도서관의 도서 목록을 볼 수 있음



* 웹 서버에서는 실시간으로 현재 책장의 도서 상태를 확인 가능(도서 정보, 도서 유무, 도서 순서 등)

****

**VIII. 프로젝트 수행 결과 분석**

1. **재학 중 습득한 기초지식 활용**

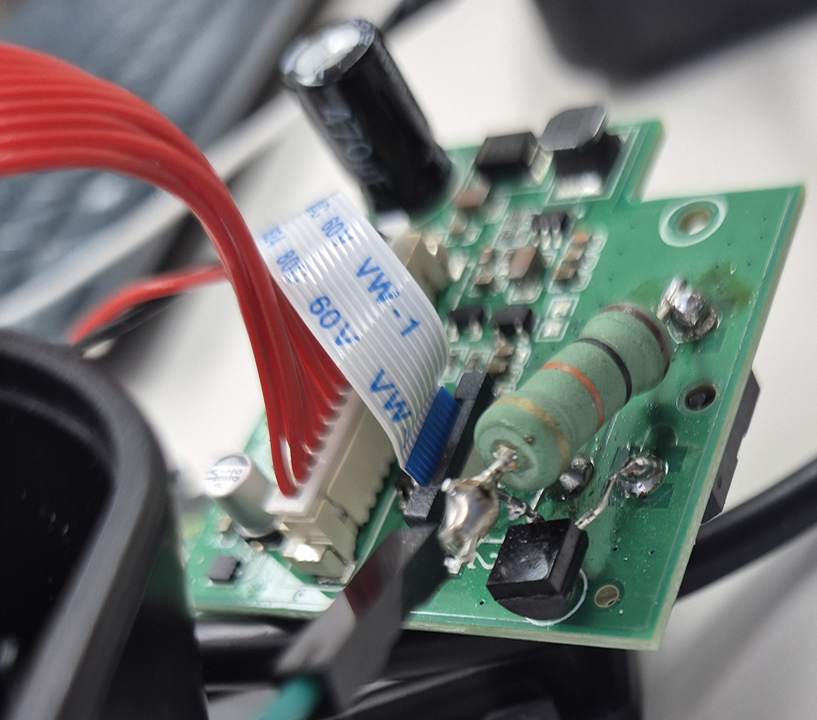
활용한 기초지식:

* 회로이론: 로봇의 전원 배선 구성, 모터와 센서 간 전류 흐름 이해, 슬라이더 모터 드라이버 및 주행용 모터 드라이버 회로 구성 시 전압·전류 계산
* 논리회로 및 제어공학: 엔코더의 A/B 위상을 분석하여 바퀴 회전 방향과 속도를 계산하는 알고리즘 구현, PWM 신호를 활용한 모터 속도 제어 개념 적용 (아두이노, 슬라이더 드라이버 제어), PID제어 알고리즘 구현
* 마이크로프로세서 실험: 아두이노 및 라즈베리파이를 이용한 I/O 핀 제어, 시리얼 통신 구현
* C 언어 프로그래밍: Arduino 및 모터 드라이버 제어, 센서 데이터 수집/처리 등 C언어 기반 기초 코드 작성

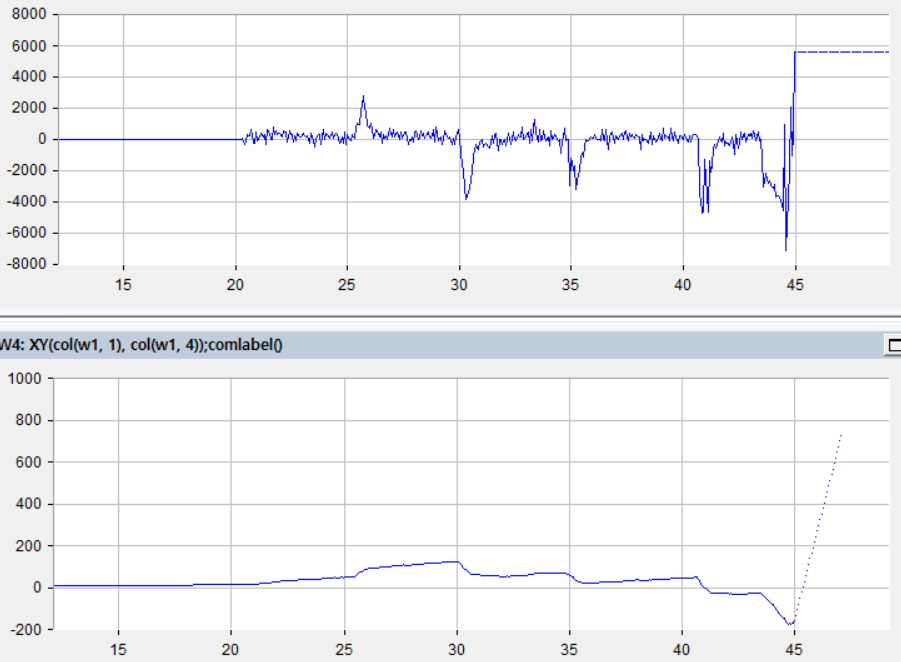
>> 회로이론, 논리회로,제어공학, 마이크로프로세서 실험 등의 과목에서 배운 내용을 바탕으로 모터 제어, 센서 연동, ROS 기반 데이터 처리 시스템을 구축하였다. 특히 아두이노와 드라이버, 슬라이더 등 부품 간의 전기적 연결과 PWM 제어 원리에 대한 이론이 실제 하드웨어 설계에 직접적으로 활용되었다.

1. **재학 중 습득한 실험지식 활용**

* 임베디드 시스템: 임베디드시스템 과목을 통해 익힌 아두이노와 외부 센서 간의 연동 경험은 바코드 리더기, 엔코더, IMU 등의 장치를 아두이노와 안정적으로 연결하고 데이터를 수집하는 데 활용되었다. 이를 통해 로봇의 위치 추정 및 바코드 인식 기능을 신뢰성 있게 구현할 수 있었다.
* 마이크로프로세서 실험: 마이크로프로세서 실험에서 학습한 UART, I2C, SPI 등의 통신 프로토콜에 대한 이해를 바탕으로 라즈베리파이와 아두이노 간의 시리얼 통신 및 I2C 기반 모터 드라이버 제어를 원활하게 수행할 수 있었다. 또한, 인터럽트를 활용하여 엔코더 신호를 실시간으로 처리하고, PWM 신호를 통해 모터 속도를 정밀하게 조절하는 방식도 실험 지식을 직접 적용한 사례였다.
* 전기전자회로 실험: 전기전자회로실험에서 저항, 트랜지스터 등의 소자를 활용한 실제 회로 구성 경험을 바탕으로 스캐너가 연속스캔모드 보다 기계적 스위치동작이 스캐닝이 더 빠르다는 것을 확인하여 스위치 부분에 트랜지스터를 부착하고 게이트와 라즈베리파이의 핀을 연결하여 고속 제어함

****

* 디지털 신호처리: 디지털신호처리에서 배운 샘플링, 펄스 신호, 노이즈 처리 개념은 BMI160 IMU센서에서 발생하는 노이즈,로봇 주행 중 발생하는 예측 불가능한 값을 디지털로 처리하는 로직을 작성하는 데에 도움을 주었다.



IMU센서에서 발생한 값을 50ms 마다 측정하여 데이터 분석을 하였다. 위쪽 그래프에서 로봇이 주행할 때 미세하게 떨리는 부분이 노이즈이고, 값이 커질 땐 yaw값의 +변위가 커질 때(좌회전), 값이 작아질 땐 yaw값의 -변위가 커질 때(우회전) 할 때 이다.

회전 임계값을 설정하여, 해당 변위가 커짐에 따라 가중치를 주어 로봇의 직진 주행을 보정하려 하였지만, 사용중인 옴니휠의 특성상 회전값을 보정할 때 고려해야할 사항이 너무 많아 보정하기에 한계가 있어 차용하지 않았다.

**3. 본 프로젝트 수행과정에서의 설계 능력 향상 내용**

* 본 프로젝트를 통해 센서 기반 자율주행 로봇 시스템의 전체적인 설계 능력을 크게 향상시킬 수 있었다. SLAM 기반 지도 생성, 바코드 스캔 자동화, 실시간 웹 인터페이스 구현까지 여러 기술 요소를 하나의 로봇 시스템에 통합하면서, 하드웨어와 소프트웨어 간의 연계 설계 역량을 체계적으로 개발하였다.
* 시스템 통합 설계 역량 향상:
* LIDAR(RPLIDAR A1M8) 센서와 엔코더를 기반으로 한 위치 추정 시스템을 ROS Noetic의 Hector SLAM과 AMCL로 구성하고, 자율주행 알고리즘과 실제 주행 하드웨어를 연동하였다. 이를 통해 센서 입력 → 위치 추정 → 목표 경로 생성 → 모터 제어까지 전 과정을 연결하는 시스템 전체 설계 능력을 기를 수 있었다.
* 모듈 간 통신 및 분산 제어 구조 설계 능력 향상:
* 라즈베리파이에서 SLAM 및 바코드 데이터 처리를 수행하고, 아두이노에서는 모터 및 센서 하드웨어 제어를 담당하는 분산 제어 구조를 직접 설계하였다. 이를 위해 UART 시리얼 통신을 활용하고 I2C 기반 모터드라이버를 제어하는 등 다양한 통신 방식을 실무적으로 연동하였다.
* 실용적인 구조와 사용자 편의성을 고려한 설계 경험:
* 웹 인터페이스를 통해 사용자가 ‘스캔 시작’ 버튼만 누르면 로봇이 책장을 주행하며 자동으로 책의 바코드를 인식하고 결과를 서버에 전송하도록 설계하였다. 이를 통해 실제 사용 시나리오를 고려한 시스템 설계와 사용자 인터페이스 설계 역량도 함양할 수 있었다.
* 하드웨어 설계의 현실적 문제 해결 능력 향상:
* 슬라이더와 바코드 스캐너의 높이 조정 메커니즘, 배터리 전원 구성, 센서 배치 최적화 등 실제 구현 과정에서 발생한 물리적 제약을 해결하며 제작 가능한 설계를 위한 반복적 수정과 개선을 경험하였다.
* 이러한 과정을 통해 단순한 부품 연결 수준을 넘어, 로봇 시스템 전체의 설계 흐름을 기획하고 구성하는 실전 중심의 설계 능력이 크게 향상되었다.

1. **문제 해결 내용**

* LIDAR 인식 실패 시 포트 재설정
* RViz에서 SLAM 맵이 겹치거나 깨지는 문제 해결 (TF 확인,)
* 바코드 스캐너의 스캔 속도 및 정확성 문제 해결
* 우리의 목표가 주행 중 스캔이다 보니 스캔율을 높일 방법을 찾아야만 하였다.스캐너가 연속스캔모드 보다 기계적 스위치동작이 스캐닝이 더 빠르다는 것을 확인하여 스위치 부분에 트랜지스터를 부착하고 게이트와 아두이노의 핀을 연결하여 고속제어하였음

1. **실무 능력 향상 내용**

* 본 프로젝트를 수행하며 하드웨어 제작, 시스템 통합, 센서 제어, 웹 인터페이스 구축 등 실제 산업 현장에서 요구되는 실무 역량을 종합적으로 향상시킬 수 있었다.

#ROS 기반 자율주행 시스템 설계 및 운용 능력

* 라이다를 활용한 Hector SLAM 지도 생성과 AMCL 기반 위치 보정을 통해 로봇이 실시간으로 경로를 인식하고 주행하도록 구성하였다. 이를 통해 산업용 로봇에서 사용되는 ROS 기반의 실무적 운용 방식을 익힐 수 있었다.

#하드웨어 통합 및 임베디드 제어 능력 향상

* 아두이노와 라즈베리파이를 활용하여 센서와 모터드라이버, 리니어 슬라이더, 바코드 스캐너 등 다양한 하드웨어 모듈을 직접 연결하고 제어하였다. I2C, UART, PWM, 인터럽트 처리 등의 다양한 제어 기법을 활용하여 임베디드 시스템 설계 실력을 강화하였다.
* 시스템 연동 및 실시간 데이터 처리 경험
* 센서로부터 획득한 데이터를 기반으로 로봇의 상태를 판단하고, ROS 노드 간 통신을 통해 동작을 제어하는 구조를 설계하며, 데이터 흐름과 타이밍 제어에 대한 실무적인 감각을 익혔다.

#웹 기반 인터페이스 연동 및 사용자 편의성 구현

* 로봇 제어 결과를 실시간으로 웹에 시각화하고, 버튼 클릭만으로 스캔을 시작할 수 있도록 구성함으로써, 사용자 중심의 시스템 설계와 클라이언트-서버 구조의 기본 개념을 익힐 수 있었다.

#문제 해결 및 협업 능력 향상

* 하드웨어 오작동, 센서 오류, 통신 불안정 등 다양한 문제를 직접 디버깅하고 해결해 나가는 과정을 통해 문제 분석력과 실무적 대응 능력, 팀원 간의 협업 능력을 함께 향상시킬 수 있었다.

1. **팀워크 향상 내용**

* 본 프로젝트는 하드웨어 제작, 소프트웨어 개발, 센서 통합, 웹 인터페이스 구축 등 다양한 작업이 동시에 진행되어야 하는 팀 기반 프로젝트로, 팀원 간의 협업과 원활한 소통이 핵심이었다.

#역할 분담을 통한 효율적인 작업 진행

* 팀 내에서 하드웨어 제어, ROS SLAM 설정, 바코드 인식 알고리즘 개발, 웹 서버 구축 등 각자의 전공 역량과 관심 분야에 맞춰 역할을 분담함으로써 작업의 전문성과 효율성을 높일 수 있었다.

#정기적인 회의와 의견 조율을 통한 문제 해결

* 시스템 통합 과정에서 발생한 충돌이나 예기치 못한 오류들을 해결하기 위해 정기적인 회의를 통해 문제를 공유하고, 서로의 의견을 조율하며 최적의 해결 방안을 도출하였다.

#문서화 및 코드 공유를 통한 협업 (https://github.com/YJM00/capston\_project)

* GitHub를 활용하여 소스코드 및 회로도, 설계 내용을 관리하고 공유함으로써 작업 중복을 방지하고 팀원 간의 연속적인 작업 흐름을 유지할 수 있었다.

#다양한 시도와 피드백을 통한 상호 학습 경험

* 각 팀원이 맡은 영역 외에도 서로의 작업을 이해하고 의견을 교환하며, 서로에게 기술적인 피드백을 주고받는 과정을 통해 팀 전체의 역량이 향상되었다.
* 이러한 협업 과정을 통해 기술적인 성과 뿐 아니라 현장 실무에서도 중요한 협동심, 의사소통 능력, 문제 해결 능력 등 실질적인 팀워크 역량을 크게 강화할 수 있었다.

1. **개발된 결과물에 대한 전시 방법 계획**

* 스캔시작 버튼을 통해 로봇이 주행하고 1층 책장을 스캐닝 하고

슬라이더가 이동해 2층 책장을 스캔하여 웹 서버에 실시간 도서 정보 연동되는 모습 전시

**참고 문헌**

* ROS Wiki: http://wiki.ros.org/
* Slamtec RPLIDAR A1M8 Datasheet
* I2C YFROBOT RZ7889 모터 드라이버 매뉴얼
* TB6600 Datasheet

**종합설계 프로젝트 수행 후기**

이번 종합설계 프로젝트는 단순한 수업 과제를 넘어서, 실제 산업 현장에서 요구되는 기술을 직접 설계하고 구현해보는 값진 경험이었다. 라이다 기반의 SLAM 지도 생성, 바코드 스캔 자동화, ROS 시스템 운용, 웹 인터페이스 연동 등 다양한 기술을 한 프로젝트에 통합하면서 전자공학 전공자로서 배운 이론과 실험 지식을 실제 동작하는 시스템에 적용할 수 있었다.

개별 부품의 단순한 제어를 넘어서, 전체 시스템이 하나의 로직으로 유기적으로 작동하도록 설계하고 조율하는 과정은 쉽지 않았지만, 그만큼 큰 성취감을 얻을 수 있었다. 특히 예상치 못한 문제를 팀원들과 함께 분석하고 해결해 나가면서 현장감 있는 문제 해결력과 팀워크의 중요성을 다시 한 번 체감할 수 있었다.

이번 프로젝트를 통해 하드웨어와 소프트웨어가 통합된 시스템의 설계 및 구현 전 과정을 직접 경험하였으며, 졸업 후 산업 현장에서도 활용 가능한 실무 능력과 자신감을 키울 수 있었다. 추후 더 복잡한 로봇 시스템이나 IoT 응용 분야에도 도전하고 싶은 동기를 얻게 된 뜻깊은 시간이었다.