s_opening_door.dart

소스 코드

전체 코드

```dart
import 'package:buyoil/model/ui_state_step1.dart';
import 'package:buyoil/view/widget/circular_prograss.dart';
import 'package:buyoil/view/widget/w_step_nav.dart';
import 'package:buyoil/viewmodel/vm_step1.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:go_router/go_router.dart';

import '../../common/app_colors.dart';
import '../../common/app_strings.dart';
import '../../common/app_styles.dart';
import '../../router.dart';
import '../widget/w_header.dart';

class OpeningDoorScreen extends ConsumerStatefulWidget {
  const OpeningDoorScreen({Key? key}) : super(key: key);

  @override
  ConsumerState<ConsumerStatefulWidget> createState() => OpeningDoorScreenState();
}

class OpeningDoorScreenState extends ConsumerState<OpeningDoorScreen> {
  @override
  void initState() {
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        children: [
          HeaderWidget(),
          Expanded(
            child: _openDoorBody()
          )
        ],
      )
    );
  }

  _openDoorBody() {
    return Container(
      width: double.maxFinite,
```

```
      height: double.maxFinite,
      color: AppColors.EFFDF6,
      alignment: Alignment.center,
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Container(
            width: 110, height: 110,
            child: RotatingImageCircularProgressBar(
              progress: 50,
              imagePath: "${AppStrings.assetPath}img_circular.png",),
          ),
          SizedBox(height: 24 - 3,),
          Text(AppStrings.doorOpening.tr(), style: AppStyles.tsDoorOpeningText,),
          SizedBox(height: 3,),
        ],
      ),
    );
  }
}
```

OpeningDoorScreen 구조 분석

이 소스 파일은 사용자가 문을 여는 프로세스에서 **"문 열림 처리 중(Opening Door)"** 상태를 표시하기 위한 화면(View)이다.
MVVM 기반 구조에서 본 파일은 **View 계층**에 해당한다.

클래스 구성

`class OpeningDoorScreen extends ConsumerStatefulWidget`

- Riverpod 기반 상태 사용이 가능한 StatefulWidget이다.
- 화면 구성 및 Provider 접근을 위한 표준 형태이다.

`class OpeningDoorScreenState extends ConsumerState<OpeningDoorScreen>`

- 실제 화면을 렌더링하는 State 클래스.
- Provider의 상태를 모니터링할 수 있으나, 현 구현에서는 상태 리스닝 로직이 없다.
- 화면은 **로딩 애니메이션 + 문 열림 텍스트**를 표시하는 정적인 UI로 구성된다.

UI 계층 구조

전체 UI 구조는 다음과 같이 단순하다.

```
Scaffold
└ Column
   ├ HeaderWidget
   └ Expanded
      └ _openDoorBody()
         └ Container
            └ Centered Column
               ├ RotatingImageCircularProgressBar
               ├ SizedBox
               └ Text("doorOpening")
```

주요 구성 요소 분석

1. build() 메서드

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Column(
      children: [
        HeaderWidget(),
        Expanded(child: _openDoorBody())
      ],
    )
  );
}
```

역할:

- 기본 화면 구조 설정
- 상단 헤더 + 본문 영역(문 열림 애니메이션) 구성

---

2. `_openDoorBody()`

이 메서드는 화면의 핵심 UI를 구성한다.

```
Container
 width: infinity
 height: infinity
 color: AppColors.EFFDF6
 alignment: center
 └ Column
    mainAxisAlignment: center
    ├ RotatingImageCircularProgressBar
    ├ Gap
    └ Text(AppStrings.doorOpening)
```

포함된 위젯

RotatingImageCircularProgressBar

- 커스텀 로딩 애니메이션으로 추정된다.
- 내부적으로 회전 애니메이션 + 이미지 오버레이 구조일 가능성이 높다.
- `progress: 50`으로 설정되어 있으나, 실제 프로그레스 바는 고정된 애니메이션일 수도 있다.

텍스트 영역

```
Text(AppStrings.doorOpening.tr(), style: AppStyles.tsDoorOpeningText)
```

- 다국어 문자열 적용(상태 메시지)
- 문이 열리는 중임을 사용자에게 표시

ViewModel과 상태 연동 구조

현재 파일에서는 상태(ViewModel) 참조 없음

다음 import가 있으나, 실제 사용되지 않는다.

```
import 'package:buyoil/model/ui_state_step1.dart';
import 'package:buyoil/viewmodel/vm_step1.dart';
```

즉:

- 실제 문 열림 완료 여부를 체크하는 로직 없음
- 다음 화면 이동(`context.go(...)`)과 같은 동작 없음
- 단순 'Wait / Loading 화면' 역할만 수행

프로젝트 아키텍처 상에서는
외부(ViewModel or SerialPort VM)에서 문 열림 완료 이벤트가 발생하면 다른 화면으로 이동하도록 설계해야 한다.

현재 파일만 보면 이동 처리 기능은 다른 레이어에서 수행되고,
OpeningDoorScreen은 단순 디스플레이 역할만 담당하고 있다.

아키텍처상 기능 위치

본 프로젝트는 다음과 같은 화면 흐름 구조를 따르고 있다:

```
Step1 → OpeningDoorScreen → Step2 → Step3 → Step4
```

OpeningDoorScreen의 역할은 다음과 같다:

1. 문을 여는 명령 전송은 ViewModel에서 실행됨
   예: `vm_step1` 또는 `vm_driver`에서 SerialPort로 “문 여는 명령” 전송
2. 실제 문 열림 센서 응답 수신 (UIStateStep1 등)

3. 응답이 도착하면 ViewModel에서 Step2 화면으로 이동
   예:

```
if (doorOpened == true) { context.goNamed(RouteGroup.Step2.name); }
```

따라서 OpeningDoorScreen은 **"상태 변화가 발생할 때까지 사용자에게 대기 화면만 보여주는 UI"**이다.

요약

- OpeningDoorScreen은 UI 계층(View)이며 비즈니스 로직을 포함하지 않는다.
- ViewModel과 상태 구조는 import되어 있지만 본 화면에서는 직접 사용하지 않음.
- 단일 목적 화면: "문 열림 중" 상태를 나타내는 로딩 화면.
- GoRouter 이동 책임은 상위 흐름(ViewModel)에서 담당.
- MVVM 구조에서 "중간 표시 화면(Loading/Waiting UI)" 역할 수행.