debug_buttons.dart

소스 코드

전체 코드

```dart
import 'package:buyoil/common/app_commands.dart';
import 'package:buyoil/viewmodel/vm_serial_port.dart';
import 'package:flutter/material.dart';
import 'package:flutter/src/widgets/framework.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';

class DebugButtons extends ConsumerStatefulWidget {
  @override
  ConsumerState<ConsumerStatefulWidget> createState() => DebugBtnState();

}

class DebugBtnState extends ConsumerState<DebugButtons> {
  @override
  Widget build(BuildContext context) {
    return Container(
      width: double.maxFinite,
      // height: 120,
      decoration: BoxDecoration(color: Colors.blue, borderRadius:
BorderRadius.circular(10)),
      child: Wrap(
        alignment: WrapAlignment.start,
          children: [
            _btn(PORT_RESPONSES.ok),
            _btnDivided("[ANS"),
            _btnDivided("]OK#"),
            _btn(PORT_RESPONSES.open),
            _btn(PORT_RESPONSES.fail),
            _btn(PORT_RESPONSES.notAuth),
            _btn(PORT_RESPONSES.reject),
            _btn(PORT_RESPONSES.ok),
            _btn(PORT_RESPONSES.full),
            _btnString("[ANS]O5.55W6.56E#"),
            _btn(PORT_RESPONSES.driverTrue),
            _btn(PORT_RESPONSES.driverFalse),
            _btnString("[CMD]SLEEP#", toMCU:true),
            _btn(PORT_RESPONSES.stmSleep),
            _btnPhone(),
          ],
      ),
    );
  }

  _btn(PORT_RESPONSES res) {
    return TextButton(onPressed: () {
```

```
45        ref.watch(serialPortVMProvider.notifier).listenDebug("${res.response}#");
46
47      },child: Container(
48          padding: EdgeInsets.all(10),
49          color: Colors.white,
50          child: Text(res.response+"#"))
51      );
52    }
53
54    _btnDivided(String text) {
55      return TextButton(onPressed: () {
56        ref.watch(serialPortVMProvider.notifier).listenDebug(text);
57
58      },child: Container(
59          padding: EdgeInsets.all(10),
60          color: Colors.white,
61          child: Text(text))
62      );
63    }
64    _btnString(String msg, {bool toMCU = false}) {
65      return TextButton(onPressed: () {
66        ref.watch(serialPortVMProvider.notifier).writeDebug(msg);
67      },
68        child: Container(
69        padding: EdgeInsets.all(10),
70        color: toMCU ? Colors.yellow: Colors.white,
71        child: Text(msg),
72      ),
73      );
74    }
75
76    _btnPhone() {
77      return TextButton(onPressed: () {
78        ref.watch(serialPortVMProvider.notifier).sendPhoneNumber("000000");
79      }, child: Container(
80        padding: EdgeInsets.all(10),
81        color: Colors.yellowAccent,
82        child: Text("[V]   000000 phone V click"),
83      ));
84    }
85
86  }
```

1 파일 개요

파일명: `lib/view/widget/step1/debug_button.dart`
역할: MCU/시리얼 포트와 관련된 **디버그용 버튼 위젯** 제공
주요 목적:

- 다양한 테스트용/디버그용 명령을 MCU나 시리얼 포트로 전송
- 버튼 클릭 시 **시리얼 통신 ViewModel (`vm_serial_port`)**과 연동
- 개발 중 통신 테스트, 상태 확인, 명령 전송 등에 사용

사용 맥락 예시:

- STM32, ESP32 등 MCU와 통신 테스트
- 시리얼 포트 응답 확인, 명령 전송
- 기능별 시나리오 테스트

2️⃣ 주요 기능

1. **다양한 버튼 제공**
    - `PORT_RESPONSES` enum/클래스 기반 버튼 (`ok`, `open`, `fail`, `notAuth` 등)
    - 문자열 직접 전송 버튼 (`_btnString`)
    - 전화번호 전송 버튼 (`_btnPhone`)
2. **버튼 클릭 시 동작**
    - `_btn`, `_btnDivided` → `listenDebug()` 호출
    - `_btnString` → `writeDebug()` 호출
    - `_btnPhone` → `sendPhoneNumber()` 호출
3. **버튼 UI**
    - `TextButton` + `Container` 구조
    - 버튼 내부 패딩: 10
    - 버튼 배경색: 일반 흰색, MCU 전송 버튼은 노란색
4. **레이아웃**
    - `Container`로 감싸고 전체 너비 최대(`double.maxFinite`)
    - `Wrap` 위젯 사용 → 버튼 여러 개를 자동 줄 바꿈 처리
    - `borderRadius: 10` → 라운드 테두리 적용
5. **상태 관리**
    - `ConsumerStatefulWidget` + `Riverpod` 사용
    - `ref.watch(serialPortVMProvider.notifier)` 통해 ViewModel 접근
    - 각 버튼 클릭 시 ViewModel 메서드 호출

---

3 구조 분석

```
DebugButtons (ConsumerStatefulWidget)
 └── DebugBtnState (ConsumerState)
       ├─ build(): Container + Wrap
       |    ├─ _btn(PORT_RESPONSES)
       |    ├─ _btnDivided(String)
       |    ├─ _btnString(String, {bool toMCU})
       |    └─ _btnPhone()
       ├─ _btn(PORT_RESPONSES res) → listenDebug("${res.response}#")
       ├─ _btnDivided(String text) → listenDebug(text)
       ├─ _btnString(String msg, {bool toMCU}) → writeDebug(msg)
       └─ _btnPhone() → sendPhoneNumber("000000")
```

- **ConsumerStatefulWidget** 사용 → Riverpod과 연동
- **Wrap** 사용 → 버튼 자동 줄바꿈, 화면 너비에 따라 배치
- 버튼 메서드(`_btn`, `_btnDivided`, `_btnString`, `_btnPhone`) 별로 기능 분리

---

4 동작 흐름

1. 화면에 `DebugButtons` 위젯 렌더링
2. `Wrap` 위에 여러 버튼 표시
3. 버튼 클릭 시:
    - `ref.watch(serialPortVMProvider.notifier)`로 ViewModel 접근
    - `listenDebug`, `writeDebug`, `sendPhoneNumber` 메서드 호출
4. 시리얼 포트 통신 이벤트 발생 → MCU와 데이터 송수신 가능

> 참고: `PORT_RESPONSES`는 아마 MCU 응답 코드/상태를 정의한 enum 또는 클래스

---

5 장점

- **빠른 디버그 가능**: 여러 테스트 명령 버튼 제공
- **UI 간단**: 버튼 텍스트 기반으로 기능 확인
- **상태 관리와 통신 연동**: Riverpod ViewModel과 직접 연결
- **자동 줄바꿈 레이아웃**: Wrap 사용으로 여러 버튼 화면 대응

---

6 단점 / 개선점

1. **하드코딩된 버튼 목록**
    - `_btn`, `_btnDivided`, `_btnString` 버튼 배열이 `build` 내부에 직접 작성
    - 개선: 버튼 정의를 리스트로 분리하고 반복 생성 → 유지보수 용이
2. **UI 일관성**

- 버튼 색상과 스타일이 일부 하드코딩(`Colors.white`, `Colors.yellow`)
- 개선: Theme 또는 상수로 관리

3. **재사용성**
   - `DebugButtons` 위젯이 MCU/시리얼에 강하게 종속
   - 개선: 일반 커맨드 버튼/콜백 위젯으로 추상화 가능
4. **상태 관리**
   - 클릭 후 버튼 상태(비활성화/활성화, 로딩 등) 처리 없음
   - 개선: 명령 전송 중 로딩 표시, 완료 후 UI 업데이트
5. **매직 스트링**
   - 전화번호 `"000000"` 등 하드코딩
   - 개선: 매개변수화하여 동적 테스트 가능
6. **접근성 / UX**
   - 버튼 크기, 패딩, 텍스트 크기 고정 → 작은 화면에서 불편
   - 개선: 반응형, 스크롤 가능, 버튼 크기 조정

7️⃣ 개선된 구조 예시

```
final debugCommands = [
  PORT_RESPONSES.ok,
  PORT_RESPONSES.open,
  "[ANS]O5.55W6.56E#",
  "[CMD]SLEEP#"
];

Wrap(
  children: debugCommands.map((cmd) {
    return DebugCommandButton(
      command: cmd,
      onSend: (msg) => ref.read(serialPortVMProvider.notifier).writeDebug(msg),
    );
  }).toList(),
)
```

- 버튼 정의를 리스트로 관리 → 버튼 추가/삭제 용이
- 콜백 함수 전달로 ViewModel 종속 최소화
- 스타일을 Theme 또는 상수로 통일