오늘 BLE 테스트 앱 개발 회고 요약

목표:

- Flutter를 사용하여 두 기기 간 직접적인 BLE(Bluetooth Low Energy) 통신 가능성을 검증하고, 통신 상태 및 근접성(RSSI 기반)을 시각적으로 확인하는 테스트 앱 개발.

주요 구현 내용:

- **역할 분리:** 사용자가 '거리측정자(Central)' 또는 '일꾼(Peripheral)' 역할을 선택하는 기능 구현.
- **상태 관리:** Riverpod (`StateNotifier`)와 `Equatable` (또는 Freezed)을 이용한 상태 관리 패턴 적용.
- **Central (거리측정자) 기능:**
  - 특정 서비스 UUID를 가진 주변 BLE 기기 스캔 및 목록 표시 (RSSI 포함).
  - 선택한 '일꾼' 기기에 연결/해제 기능.
  - 연결 상태(연결됨, 연결 중, 끊김)를 명확한 시각적 표시기(색상, 텍스트)로 제공.
  - 연결된 '일꾼'의 서비스 및 특성(Characteristic) 검색.
  - 특정 Characteristic에 대한 데이터 읽기/쓰기 기능.
  - Characteristic 값 변화 알림 구독/해제 기능 (Notify/Indicate).
  - 연결된 '일꾼'의 RSSI 값을 읽어 근접성 추정치 표시 기능 추가.
- **Peripheral (일꾼) 기능:**
  - 특정 서비스 UUID를 포함하여 BLE Advertising 시작/중지 기능.
  - Advertising 상태 표시.
  - '거리측정자'의 연결 상태(연결됨/끊김) 표시 기능 추가.
  - Write 요청 처리 로직 구현 (Central로부터 데이터 수신).

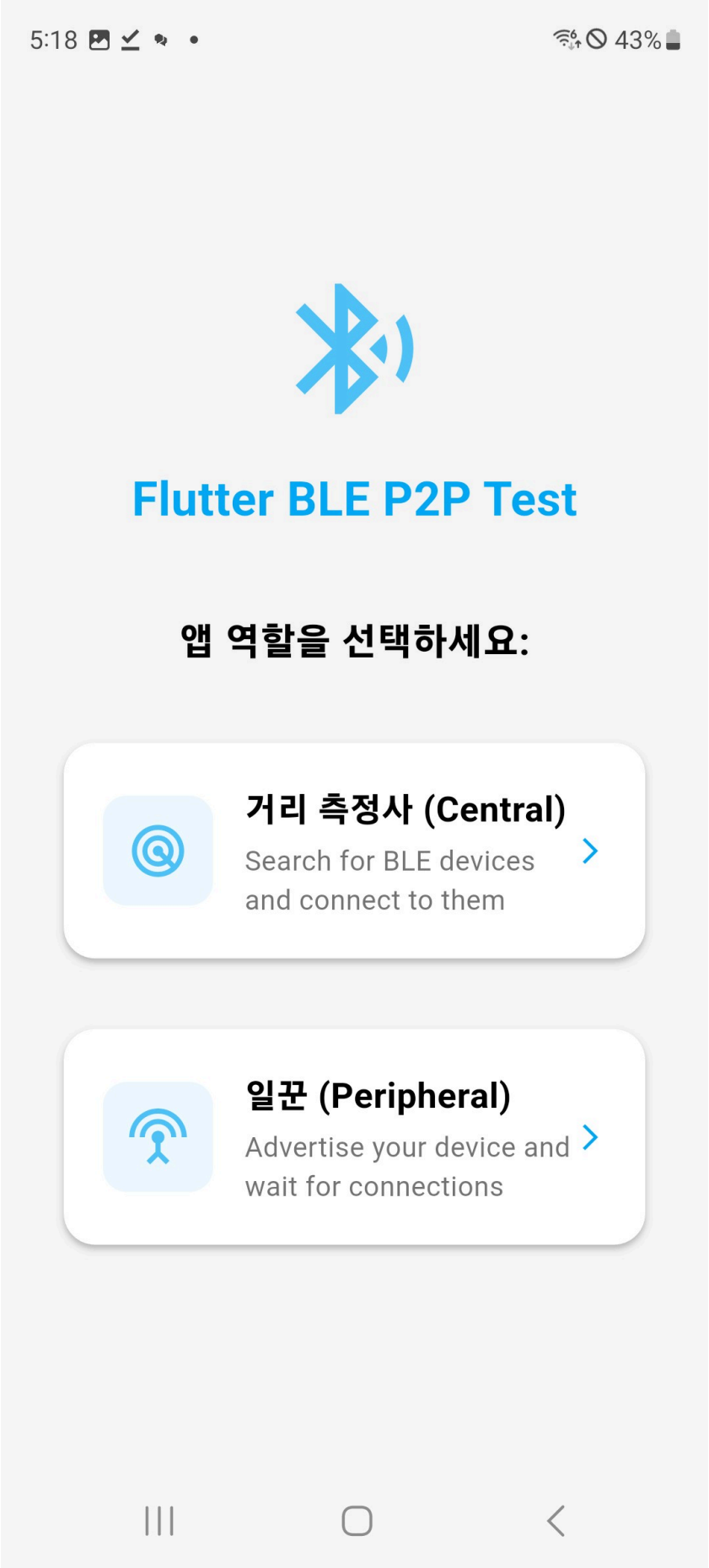
결론:

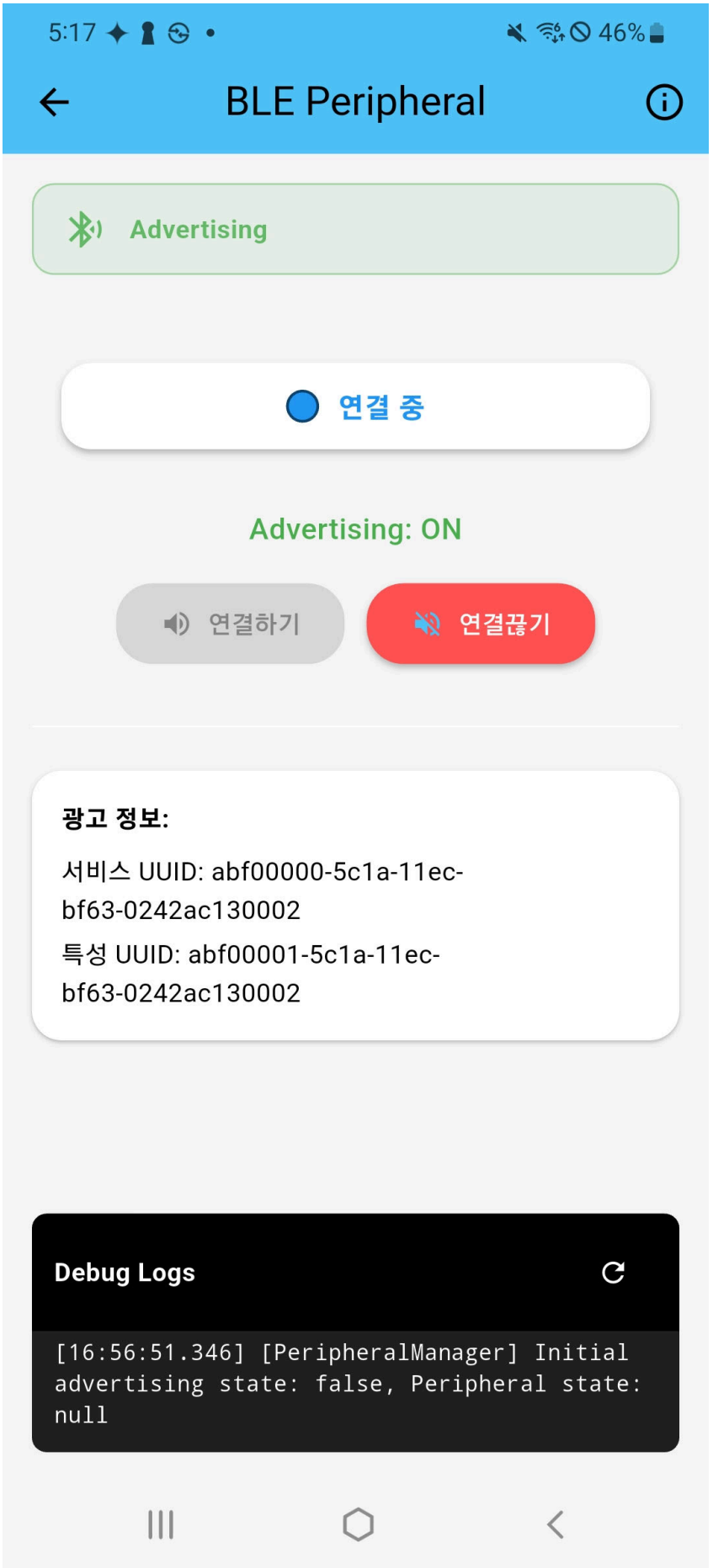
- Flutter 환경에서 `flutter_blue_plus`와 `flutter_ble_peripheral` 라이브러리를 활용하여 BLE Central 및 Peripheral 역할의 기본적인 통신 흐름 구현 가능성을 성공적으로 확인했습니다.

- 상태 관리 라이브러리(Riverpod)를 통해 복잡한 비동기 BLE 상태를 효과적으로 관리하고 UI에 반영하는 방법을 연습했습니다.
- RSSI 값과 메시지 송수신 기능을 통해 기본적인 P2P 상호작용의 기반을 마련했습니다.

배운 점/개선점:

- BLE 통신의 비동기적 특성과 상태 관리의 중요성을 확인함.
- Central과 Peripheral 역할 구현 시 사용하는 라이브러리가 다르고, 각 라이브러리의 API 및 동작 방식에 대한 이해가 필요함.
- 오류 처리 및 예외 상황 대응을 더 강화할 필요가 있음.
- (향후) 메시지 기능 고도화, 데이터 구조화, 로컬 저장소 연동 등을 통해 실제 사용 가능한 수준으로 발전시킬 수 있음.


5:18
43%
Flutter BLE P2P Test
앱 역할을 선택하세요:
거리 측정사 (Central)
Search for BLE devices and connect to them
일꾼 (Peripheral)
Advertise your device and wait for connections


5:17
46%
BLE Peripheral
Advertising
연결 중
Advertising: ON
연결하기
연결끊기
광고 정보:
서비스 UUID: abf00000-5c1a-11ec-bf63-0242ac130002
특성 UUID: abf00001-5c1a-11ec-bf63-0242ac130002
Debug Logs
[16:56:51.346] [PeripheralManager] Initial advertising state: false, Peripheral state: null