VesiScan-Basic 코드 리뷰 보고서

> **작성일**: 2026-03-15 **대상**: VesiScan BASIC 펌웨어 전체 (nRF52840 + S140 SoftDevice) **기준 회로**: 20305 VB0HW0100_Circuit.sch **현재 빌드**: DEBUG_MINIMAL_BOOT=1, BLE_DEV_MODE=1

목차

---

1. 삭제 대상: 미사용 하드웨어 드라이버

1.1 회로 기반 분석

VB0HW0100 회로에 실장된 주요 IC: - **nRF52840** (U26) — MCU - **ICM-42670-P** (U10) — 6축 IMU ✅ 사용 중 - **TMP235-Q1** (U22) — 온도 센서 ✅ 사용 중 - **ADC121S051** (U37) — 12-bit ADC ✅ 사용 중 (dr_adc121s051) - **ADL5513** (U23) — 로그 앰프 (아날로그, 드라이버 불필요) - **OPA2836** (U2) — 정밀 OP-AMP (아날로그, 드라이버 불필요) - **W25Q32** (U16) — SPI Flash (현재 코드에서 미사용) - **LT3463** (U32) — DC-DC 부스트 (GPIO 제어만, 별도 드라이버 불필요) - **LM27762** (U38) — 음전압 생성 (EN 핀 제어만) - **MCP1804T** (U33) — LDO (패시브) - **MCP73838** (U3) — 배터리 충전 (패시브)

1.2 회로에 없는 IC의 드라이버 (삭제 대상)

파일	IC/기능	회로 존재	판정	비고
`ad5272_i2c.c/h`	AD5272 디지털 포텐시오미터	❌	**삭제**	회로에 미실장. Full boot에서만 사용
`ada2200_spi.c/h`	ADA2200 동기 복조기	❌	**삭제**	회로에 미실장. Full boot에서만 사용
`mcp4725_i2c.c/h`	MCP4725 DAC	❌	**삭제**	회로에 미실장. LED 바이어스 전압 제어용이었음
`mcp4725_adc.c/h`	MCP4725 ADC 피드백	❌	**삭제**	MCP4725 의존
`ir_i2c.c/h`	IR LED 센서 (0x50~0x57)	❌	**삭제**	회로에 해당 I2C 디바이스 미실장
`LED_Parse.c/h`	LED ROM 파싱	❌	**삭제**	ir_i2c 의존. LED ROM 하드웨어 없음

1.3 기능적으로 미사용인 드라이버 (삭제 대상)

파일	기능	판정	비고
`measurements.c/h`	레거시 측정 루프	**삭제**	meas_pd_48 기반으로 대체됨. 현재 MINIMAL에서 미호출
`measurements_20.c/h`	20-샘플 측정 변형	**삭제**	measurements.c 변형. 미사용
`meas_pd_imm.c/h`	즉시 PD 측정	**검토 필요**	cmd에서 imm 명령 참조 확인 필요
`meas_pd_buff.c/h`	버퍼드 PD 측정	**검토 필요**	cmd에서 buff 명령 참조 확인 필요
`meas_pd_voltage_simple.c/h`	단순 전압 측정	**검토 필요**	simple 모드 cmd 참조 확인 필요
`meas_pd_voltage_half.c/h`	반파 전압 측정	**검토 필요**	half 모드 cmd 참조 확인 필요

파일	기능	판정	비고
`meas_pd_voltage_full.c/h`	전파 전압 측정	**검토 필요**	full 모드 cmd 참조 확인 필요
`meas_pd_voltage_custom.c/h`	커스텀 전압 측정	**삭제**	cmd_parse.h에서 이미 주석 처리됨
`full_agc.c/h`	자동 이득 제어	**삭제**	AD5272 + MCP4725 + ir_i2c 의존. 해당 HW 없음
`debuf_print.h`	오타 파일	**삭제**	debug_print.h가 올바른 파일

1.4 삭제 시 의존성 영향

`cmd_parse.h`가 대부분의 드라이버 헤더를 include하므로, 드라이버 삭제 시 반드시 함께 수정:

```
cmd_parse.h 에서 제거할 include:
- #include "ir_i2c.h"          (line 37)
- #include "ad5272_i2c.h"      (line 46)
- #include "ada2200_spi.h"     (line 47)
- #include "mcp4725_i2c.h"     (line 50)
- #include "measurements.h"    (line 51)
- #include "meas_pd_voltage_simple.h" (line 55)
- #include "meas_pd_voltage_half.h"   (line 56)
- #include "meas_pd_voltage_full.h"   (line 57)
- #include "full_agc.h"        (line 58)
- #include "meas_pd_imm.h"    (line 60)
```

`main.c` 에서 제거할 include (현재 #if !DEBUG_MINIMAL_BOOT 내부):

```
- #include "ada2200_spi.h"     (line 68 영역)
- #include "mcp4725_i2c.h"     (line 71 영역)
- 기타 삭제 대상 드라이버 헤더
```

`main_timer.c` 에서 제거할 include:

```
- #include "mcp4725_i2c.h"     (line 34)
- #include "ad5272_i2c.h"      (line 35)
- #include "meas_pd_voltage_full.h" (line 33)
```

`power_control.c` 에서:

```
- #include "cat_interface.h"   (line 17) → EEPROM 함수만 필요, 헤더 분리 검토
```

1.5 cat_interface.c (EEPROM 드라이버) 삭제 분석

1.5.1 결론: 삭제 가능하나 TWI 인스턴스 분리 필요

VB0HW0100 회로에 **EEPROM IC가 없음**. cat_interface.c는 I2C 주소 0x50의 EEPROM 드라이버이며, EEPROM이 없으면 모든 eeprom_xxx() 함수는 I2C NACK 에러를 반환하게 됨.

1.5.2 cat_interface.c가 제공하는 것

기능	호출자	EEPROM 없을 때	판정
`m_eeprom` TWI 인스턴스	i2c_manager.c, IMU 드라이버	**필수** (IMU I2C 공유)	**유지 (분리)**
`eeprom_initialize()` / `uninitialize()`	i2c_manager.c	TWI HW init — IMU에 필요	**유지 (리네이밍)**

기능	호출자	EEPROM 없을 때	판정
`eeprom_read/write_byte/word/page/bytes`	cmd_parse.c (20+ 곳)	I2C NACK → 에러 반환	**삭제**
`eeprom_read/write_uint16_array`	cmd_parse.c, full_agc.c	I2C NACK → 에러 반환	**삭제**
`eeprom_read/write_uint32`	cmd_parse.c	I2C NACK → 에러 반환	**삭제**
`eeprom_write_encrypted()` / `read_decrypted()`	cmd_parse.c, main.c	AES + I2C NACK → 실패	**삭제**
`encrypt_data()` / `decrypt_data()`	위 함수 내부	AES 자체는 동작하나 호출자 삭제 시 불필요	**삭제**
`aes_key[16]`, `aes_iv[16]`	위 함수 내부	⚠️ 하드코딩 키 보안위험	**삭제**

1.5.3 권장 작업

1. **cat_interface.c → twi_driver.c 로 리네이밍** (또는 기존 i2c_manager.c에 통합)
2. **유지할 것**: `m_eeprom` TWI 인스턴스, `eeprom_initialize()`, `eeprom_uninitialize()` (함수명을 `twi_master_init()` / `twi_master_uninit()` 로 변경)
3. **삭제할 것**: EEPROM read/write 전체, AES encrypt/decrypt 전체, `aes_key`, `aes_iv`
4. **cmd_parse.c EEPROM 호출 → FDS 전환** (아래 1.5.4절 참고)
5. **main.c:1417-1425** (버튼 롱프레스 리셋): EEPROM 쓰기를 FDS config_save()로 교체
6. **power_control.c**: `eeprom_control(OFF)` — EEP_WP(P0.24) GPIO 제어, EEPROM 없으면 불필요 → 삭제

1.5.4 cmd_parse.c EEPROM → FDS 전환 목록

현재 cmd_parse.c에서 EEPROM을 사용하는 모든 경로 (MINIMAL 부팅에서도 실행됨):

EEPROM 호출	위치	FDS 대체 방법
`eeprom_init_values_read()`	cmd_parse.c:371-480	**삭제** — FDS config_load()가 이미 동일 역할 수행
`eeprom_write_byte(0x0070, m_pd_adc_cnt)`	cmd_parse.c:1421	`m_config.pd_adc_cnt = val; config_save();`
`eeprom_write_word(0x0080, m_pd_delay_us)`	cmd_parse.c:1477	`m_config.pd_delay_us = val; config_save();`
`eeprom_write_byte(0x0060, bond_data_delete)`	cmd_parse.c:1601	`m_config.bond_data_delete = val; config_save();`
`eeprom_write_byte(0x0065, m_reset_status)`	cmd_parse.c:1605,1628	`m_config.reset_status = val; config_save();`
`eeprom_write_encrypted(0x0020, passkey, 6)`	cmd_parse.c:298,1708	`memcpy(m_config.static_passkey, val, 6); config_save();`
`eeprom_read_decrypted(0x0020, passkey, 6)`	cmd_parse.c:1720,1755	`memcpy(buf, m_config.static_passkey, 6);`
`eeprom_write_uint32(0x0090, m_life_cycle)`	cmd_parse.c:2025	FDS에 life_cycle 필드 추가 필요
`eeprom_read_uint32(0x0090, &m_life_cycle)`	cmd_parse.c:2054	FDS에서 읽기
`eeprom_write/read_uint16_array(0x0480, ...)`	cmd_parse.c:756,1886,1918,1952	AGC gain 배열 → FDS 별도 레코드 또는 삭제 (HW 없음)
`eeprom_read_uint16_array(0x0480, led_pd_dac_v, 48)`	cmd_parse.c:398	AGC gain — HW 없으므로 삭제

> **참고**: `m_life_cycle`은 현재 `config_data_t`에 없으므로 구조체에 `uint32_t life_cycle` 필드 추가 필요. AGC gain 배열은 AD5272+MCP4725 HW가 없으므로 관련 기능 전체 삭제 가능.

1.5.5 B5, B10 재판정

항목	기존 판정	수정 판정
B5: EEPROM 고정값 48 vs LED_NUM 24	수정 필요	**삭제** (EEPROM 함수 전체 삭제)

항목	기존 판정	수정 판정
B10: eeprom_read_page() 디버그 과다	수정 필요	**삭제** (EEPROM 함수 전체 삭제)

---

2. 삭제 대상: 미사용 코드

2.1 AES 암호화 (cat_interface.c)

판정 변경: 전체 삭제

EEPROM이 없으므로 AES 암호화/복호화가 불필요. S/N과 Passkey는 FDS에 평문 저장. (FDS는 내장 Flash 영역이므로 외부 접근 불가, AES 불필요)

항목	위치	판정
`aes_key[16]` 하드코딩	cat_interface.c:38-41	**삭제**
`aes_iv[16]` (고정 IV = 0)	cat_interface.c:43	**삭제**
`encrypt_data()`	cat_interface.c:54-73	**삭제**
`decrypt_data()`	cat_interface.c:76-93	**삭제**
`eeprom_write_encrypted()`	cat_interface.c:95-103	**삭제**
`eeprom_read_decrypted()`	cat_interface.c:106-114	**삭제**

2.2 Serial Number / Passkey / HW Version 중복 선언 분석

현재 이 값들이 **3중 저장**되고 있음:

데이터	전역 변수 (런타임)	FDS (내장 Flash)	EEPROM (외부, 미실장)
Serial Number	`SERIAL_NO[12]` (cmd_parse.c:38)	`m_config.serial_no[12]`	addr 0x0030 (AES 암호화)
Passkey	`m_static_passkey[6]` (cmd_parse.c:40)	`m_config.static_passkey[6]`	addr 0x0020 (AES 암호화)
HW Version	`HW_NO[12]` (cmd_parse.c:48)	`m_config.hw_no[12]`	addr 0x0010 (미구현)

문제: - 전역 변수 ↔ FDS는 필요한 구조 (BLE 스택이 런타임 포인터 필요) - EEPROM 경로는 **완전 삭제 가능** (HW 없음) - `HW_NO[12]` 는 `m_config.hw_no` 의 로컬 복사본. BLE 명령 응답 시 `m_config.hw_no` 를 직접 사용하면 제거 가능

권장 작업: 1. EEPROM 경로 삭제 → `eeprom_init_values_read()` 함수 전체 삭제 2. `HW_NO[12]` 삭제 → `m_config.hw_no` 직접 사용 (cmd_parse.c:1975-1999) 3. `SERIAL_NO` , `m_static_passkey` 는 BLE 스택 인터페이스에 필요하므로 유지 4. 데이터 흐름 단순화: `FDS → 전역변수 → BLE 스택` (2단계만)

2.3 미사용 전역 변수

변수	위치	판정	비고
`roles_str[]`	main.c:206	**삭제**	suppress_unused_warnings()에서만 참조
`m_encrypted_text[16]`	main.c:211	**삭제**	어디서도 사용되지 않음
`m_encrypted_text2[16]`	main.c:212	**삭제**	어디서도 사용되지 않음
`m_decrypted_text[16]`	main.c:213	**삭제**	어디서도 사용되지 않음
`DEVICE_NAME`	cmd_parse.c:18	**삭제**	미사용 define. SERIAL_NO가 BLE 이름으로 사용
`resetCount`	cmd_parse.c:46	**삭제 검토**	meas_pd_voltage_simple.c에서 참조하지만 해당 모듈이 삭제 대상이면 함께 삭제
`HW_NO[12]`	cmd_parse.c:48	**삭제**	m_config.hw_no로 직접 대체 (2.2절 참고)
`AES_KEY_SIZE`	main.c:168	**삭제**	m_encrypted_text와 함께 삭제

변수	위치	판정	비고
`AES_BLOCK_SIZE`	main.c:169	**삭제**	main.c에서 미사용 (cat_interface.c에 별도 정의 있음)
`suppress_unused_warnings()`	main.c:268-273	**삭제**	roles_str 삭제 후 불필요
`c_addr[6]`	main.c:204	**삭제**	PM_EVT_PEER_DATA_UPDATE에서 쓰지만 이후 읽는 곳 없음
`led_pd_dac_v[LED_NUM]`	참조 다수	**삭제**	AGC gain 배열 — AD5272/MCP4725 HW 없음

2.4 주석 처리된 코드

위치	내용	판정
main_timer.c:72-96	FEATURE_DETAIL_VALUE_FULL 주석 블록	**삭제**
cmd_parse.c:324-359	serial_to_passkey_hash, test_eeprom_page_rw	**삭제**
cmd_parse.c:87-88	pd_adc_half/full_a_start extern	**삭제**
battery_saadc.c:22	//#include "fstorage.h"	**삭제**

---

3. DEBUG_MINIMAL_BOOT 조건부 컴파일 정리

현재 `DEBUG_MINIMAL_BOOT=1` 로 MINIMAL 모드만 사용. Full boot 경로는 실행되지 않음.

3.1 제거 대상 #if 블록

파일	라인	블록	판정
main.c:309-326	`#if !DEBUG_MINIMAL_BOOT` full_gpio_init()	**삭제** (함수 전체)	
main.c:399-433	`#if !DEBUG_MINIMAL_BOOT` load_eeprom_config()	**삭제** (함수 전체)	
main.c:1438-1477	Full boot 경로 (ada2200_init, mcp4725_init 등)	**삭제**	
main.c:1521-1525	GPIO init 분기	**MINIMAL 코드만 유지**	
power_control.c:84-111	Full boot power sequence	**삭제** (minimal 경로만 유지)	

3.2 정리 후 main.c 부팅 흐름 (MINIMAL만)

```
main()
  → power_hold_init()
  → uart_init() + log_init()
  → minimal_gpio_init()
  → timers_init()
  → load_default_config()
  → buttons_leds_init()
  → BLE stack init (sdh → gap → gatt → svc → adv → conn)
  → FDS init → config_load() → load_flash_config()
  → peer_manager_init()
  → meas_config_load()
  → timers_start() → main loop (idle_state_handle)
```

3.3 DEBUG_MINIMAL_BOOT 제거 시 함께 정리

- `#define DEBUG_MINIMAL_BOOT 1` 삭제 (main.c:110)
- 모든 `#if DEBUG_MINIMAL_BOOT` / `#if !DEBUG_MINIMAL_BOOT` 조건 제거
- MINIMAL 코드 경로만 유지
- `BLE_DEV_MODE` define은 유지 (DEV/PROD 전환에 필요)

4. Keil 프로젝트 파일 정리

4.1 삭제할 소스 참조 (uvprojx)

드라이버 삭제 시 `ble_app_bladder_patch_s140.uvprojx` 에서 제거할 `<File>` 항목:

```
ad5272_i2c.c
ada2200_spi.c
mcp4725_i2c.c
mcp4725_adc.c
ir_i2c.c
LED_Parse.c
measurements.c
measurements_20.c
full_agc.c
meas_pd_voltage_custom.c  (이미 주석 처리 상태 확인 필요)
```

4.2 추가 확인 필요

meas_pd_imm, meas_pd_buff, meas_pd_voltage_simple/half/full은 BLE 커맨드에서 참조 여부 확인 후 결정.

---

5. 버그 및 오류 수정

5.1 심각도 높음 (HIGH)

B1: binary_tx_handler() APP_ERROR_CHECK 크래시 위험

파일: main.c:1287-1296 **문제**: `do { ... APP_ERROR_CHECK(err_code); } while (err_code == NRF_ERROR_RESOURCES);` APP_ERROR_CHECK는 에러 시 assert(크래시)하므로, NRF_ERROR_RESOURCES 외의 에러가 발생하면 디바이스가 리셋됨. `dr_binary_tx_safe()` 는 이를 올바르게 처리하지만, `binary_tx_handler()` 는 여전히 위험.

수정: binary_tx_handler()를 dr_binary_tx_safe()로 교체하거나, APP_ERROR_CHECK 제거.

B2: nrf_delay_ms(10) 블로킹 대기 (BLE TX)

파일: main.c:1293 **문제**: `nrf_delay_ms(10)` 은 에러가 NRF_ERROR_RESOURCES가 **아닌** 경우에만 실행되는데, 코드 의도와 반대로 작동. 조건문 로직 오류.

```
// 현재 코드 (버그):
if ((err_code != NRF_ERROR_INVALID_STATE) &&
    (err_code != NRF_ERROR_RESOURCES) &&
    (err_code != NRF_ERROR_NOT_FOUND))
{
    nrf_delay_ms(10);  // ← RESOURCES일 때 실행 안 됨!
}
APP_ERROR_CHECK(err_code);  // ← RESOURCES일 때도 CHECK됨!
```

수정: 이 함수를 dr_binary_tx_safe() 패턴으로 전면 교체.

B3: extern + 초기화 동시 사용

파일: cmd_parse.c:79 **문제**: `extern int8_t pd_adc_counts[4] = {8, 16, 24, 32};` `extern` 과 초기화를 동시에 사용하면 컴파일러 경고 발생. 실질적으로 정의(definition)가 됨.

수정: `extern` 제거 → `int8_t pd_adc_counts[4] = {8, 16, 24, 32};`

B4: memset 버퍼 크기 불일치

파일: main.c:333, 336, 375, 379 **문제**: `SERIAL_NO[12]` 인데 `memset(SERIAL_NO, 0, 16)` — 4바이트 오버런. `m_static_passkey[6]` 인데 `memset(m_static_passkey, 0, 16)` — 10바이트 오버런.

수정:

```
memset(SERIAL_NO, 0, sizeof(SERIAL_NO));       // 12
memset(m_static_passkey, 0, sizeof(m_static_passkey)); // 6
```

> ⚠️ **이것은 스택/힙 오버플로우로 인한 메모리 손상 가능성이 있는 심각한 버그입니다.**

~~B5: EEPROM 고정값 48 vs LED_NUM 24 불일치~~ → 삭제 대상

cat_interface.c EEPROM 함수 전체 삭제로 해소됨 (1.5절 참고).

5.2 심각도 중간 (MEDIUM)

B6: battery_event_handler에서 floating-point 연산

파일: battery_saadc.c:169 **문제**: `batt_lvl_in_milli_volt_1 = (batt_lvl_in_milli_volt_0) *1.42;` nRF52840에 FPU가 있어 동작하지만, 정수 연산으로 변경하면 더 효율적: `batt_lvl_in_milli_volt_1 = (batt_lvl_in_milli_volt_0 * 142) / 100;`

B7: main_timer 단발 모드 문제

파일: main_timer.c:214 **문제**: `APP_TIMER_MODE_SINGLE_SHOT` 으로 생성되어, main_loop 내에서 매번 `main_timer_start()` 를 수동 호출해야 함. 이는 의도적 설계일 수 있으나, 타이밍 갭이 발생할 수 있음.

B8: processing_start_tick 미사용

파일: cmd_parse.c:61 **문제**: `static uint32_t processing_start_tick = 0;` 선언 후 사용되지 않음.

B9: config_load() goto 기반 흐름

파일: fstorage.c:157-261 **문제**: `goto cfg_load_start` 패턴이 무한 루프 가능성. FDS 초기화 실패 시 retry 10회 후에도 실패하면 default를 쓰고 다시 goto로 재진입하는데, 쓰기도 실패하면 무한 루프에 빠질 수 있음.

수정: goto를 for 루프로 변환하고, 최대 재시도 횟수 제한.

~~B10: eeprom_read_page() 디버그 출력 과다~~ → 삭제 대상

cat_interface.c EEPROM 함수 전체 삭제로 해소됨 (1.5절 참고).

5.3 심각도 낮음 (LOW)

B11: `debuf_print.h` 파일명 오타

판정: 이 파일이 어디선가 include되는지 확인 후 삭제.

B12: 함수 중복 선언

파일: main.h:48-51 **문제**: `static` 함수를 헤더에 선언하면 모든 include 단위에서 독립 복사본이 생성됨. `t_power_off_timeout_handler`, `power_control_handler`, `main_s`, `PM_s` 는 main.c에서만 사용하므로 main.h에서 제거해야 함.

B13: LED_ALLOFF, PD_ALLOFF 매크로 정의 위치

파일: main.c:906-907 (BLE 연결 해제 시 호출) **문제**: MINIMAL 모드에서 이 매크로가 정의되어 있는지 확인 필요. `boards.h` 에 정의되어 있으면 OK, 없으면 빌드 에러.

---

6. 저전력 최적화

6.1 현재 전력 구조 분석

구간	상태	예상 소비전류
BLE 연결 대기 (Advertising)	SoftDevice idle	~10-15μA
BLE 연결 상태 (Idle)	SoftDevice + app_timer	~20-30μA
IMU 측정 (1초 중 54ms)	I2C + IMU 650μA	평균 ~41μA
배터리 측정 (5초마다)	SAADC 샘플링	~200μA (순간)
UART 활성	1Mbps	~1mA (상시)

6.2 개선 항목

P1: UART 비활성화 (가장 큰 절약 — 예상 ~1mA 절감)

파일: main.c:1103-1128 **문제**: UART가 1Mbps로 항상 초기화됨. 프로덕션에서 UART는 불필요. **수정**: `#if BLE_DEV_MODE` 조건으로 UART 초기화 감싸기. 프로덕션에서는 UART 비활성화.

```
#if BLE_DEV_MODE
    uart_init();
#endif
```

P2: SEGGER RTT 비활성화 (프로덕션)

파일: debug_print.h **문제**: `ENABLE_PRINTF=1`이면 SEGGER_RTT가 항상 활성. RTT는 J-Link 없이도 CPU 사이클 소비. **수정**: 프로덕션 빌드에서 `ENABLE_PRINTF=0` 설정. 또는 BLE_DEV_MODE에 연동:

```
#if BLE_DEV_MODE
    #define ENABLE_PRINTF 1
#else
    #define ENABLE_PRINTF 0
#endif
```

P3: 배터리 타이머 주기 최적화

파일: battery_saadc.c:49 **현재**: `BATTERY_LOOP_INTERVAL = 5000ms` (5초) **제안**: BLE 미연결 시 30초, 연결 시 10초로 동적 조절.

```
void battery_timer_start_dynamic(bool connected) {
    uint32_t interval = connected ? 10000 : 30000;
    app_timer_start(m_battery_loop_timer_id, APP_TIMER_TICKS(interval), NULL);
}
```

P4: IMU 타이머 조건부 시작

파일: main.c:1455, 1479 **현재**: 부팅 시 무조건 `imu_active_timer_start()` 호출. **문제**: BLE 미연결 시에도 1초마다 IMU를 읽음 → 전력 낭비. **수정**: BLE 연결 시에만 IMU 타이머 시작, 연결 해제 시 중지.

```
// BLE_GAP_EVT_CONNECTED 핸들러에:
imu_active_timer_start();

// BLE_GAP_EVT_DISCONNECTED 핸들러에:
imu_active_timer_stop();
```

P5: main_loop() 타이머 폴링 — 연속 측정 모드 효율화

파일: main_timer.c:41-214

현재 구조 분석: main_loop()는 `APP_TIMER_MODE_SINGLE_SHOT` (10ms) 타이머 콜백. BLE 명령이나 버튼 이벤트가 `main_timer_start()` 를 호출할 때만 동작.

```
[idle: main_loop 미동작, CPU sleep] ← 저전력 OK
    ↓ BLE 명령 수신
[cmd_parse.c] → 플래그 세팅 + main_timer_start()
    ↓ 10ms 후
[main_loop()] → 플래그 확인 → 작업 실행 → (연속이면 main_timer_start, 아니면 종료)
    ↓ 단발 작업이면
[idle: main_loop 미동작, CPU sleep] ← 저전력 OK
```

idle 상태에서는 main_loop가 호출되지 않으므로 저전력 문제 없음.

문제가 되는 경우: BLE 명령으로 연속 모드 진입 시

모드	트리거	동작	문제점
`motion_raw_data_enabled` (msp? 명령)	BLE 명령	10ms마다 main_loop 재호출 → IMU 읽기	100Hz 폴링, IMU FIFO 미사용
`adc_enabled`	BLE 명령	10ms마다 main_loop 재호출 → cnt_adc++	의미 없는 카운터만 증가
단발 측정 (scj, sej 등)	BLE 명령	1~2회 호출 후 종료	✅ 문제 없음
`go_device_power_off` 등	BLE/버튼	1회 호출 후 전원 OFF	✅ 문제 없음

개선 방안 (난이도 순):

방안 A: 단발 이벤트를 직접 호출로 전환 (쉬움)

`go_device_power_off`, `go_NVIC_SystemReset`, `go_batt` 같은 단발 이벤트는 10ms 타이머를 경유할 이유가 없음. 이벤트 소스에서 즉시 실행:

```
// 변경 전 (현재):
go_device_power_off = true;
main_timer_start();  // 10ms 후 main_loop → go_device_power_off 확인 → device_power_off()

// 변경 후 (직접 호출):
device_power_off();  // 즉시 실행
```

이렇게 하면 main_loop()에서 단발 플래그 처리 코드를 제거할 수 있고, main_loop()는 연속 측정 전용으로 축소됨.

방안 B: app_scheduler 이벤트 큐 (보통, 권장)

nRF5 SDK의 `app_scheduler`를 사용하여 타이머 폴링을 이벤트 큐로 교체:

```
#include "app_scheduler.h"
#define SCHED_MAX_EVENT_DATA_SIZE  8
#define SCHED_QUEUE_SIZE           16

// 초기화 (main)
APP_SCHED_INIT(SCHED_MAX_EVENT_DATA_SIZE, SCHED_QUEUE_SIZE);

// 이벤트 등록 (BLE 명령 수신 시):
static void battery_meas_handler(void *p_event_data, uint16_t event_size) {
    battery_level_meas();
}
app_sched_event_put(NULL, 0, battery_meas_handler);

// main loop:
for (;;) {
    app_sched_execute();   // 큐에 이벤트 있으면 즉시 실행
```

```
    idle_state_handle();   // 없으면 nrf_pwr_mgmt_run() → WFE (CPU sleep)
}
```

장점: - ISR 컨텍스트에서 main 컨텍스트로 안전한 전환 - 타이머 10ms 지연 제거 → 응답 속도 향상 - idle_state_handle()에서 `nrf_pwr_mgmt_run()` = **WFE** 호출로 확실한 sleep 보장 - main_timer 관련 플래그 전역변수 전부 제거 가능

방안 C: IMU FIFO + DRDY 인터럽트 (어려움, 최적)

`motion_raw_data_enabled` 연속 모드의 10ms 폴링을 완전 제거:

```
현재: 10ms 타이머 → main_loop → hw_i2c_init → IMU 레지스터 읽기 → uninit
개선: IMU FIFO 버퍼링 → FIFO 워터마크 → INT1 핀 → GPIOTE ISR → 한 번에 읽기
```

ICM-42670-P FIFO 설정: - FIFO_CONFIG1 = FIFO 활성화 - FIFO_WM (워터마크) = 12바이트 × N샘플 - INT_SOURCE0 = FIFO_WM 인터럽트 활성화 - INT1 핀 → nRF52840 GPIO → GPIOTE 인터럽트

장점: 10ms 폴링 완전 제거, CPU wakeup 횟수 대폭 감소, IMU 전원 관리 단순화

P6: NRF_LOG 비활성화 고려

파일: sdk_config.h **현재**: NRF_LOG 모듈이 활성화 상태. **문제**: `NRF_LOG_PROCESS()` 가 idle_state_handle()에서 매 루프 호출됨. 로그가 없어도 오버헤드 존재. **수정**: 프로덕션에서 `NRF_LOG_ENABLED=0` 설정.

6.3 예상 절감 효과

최적화	예상 절감	난이도
P1: UART 비활성화	~1mA	쉬움
P2: RTT 비활성화	~0.1-0.5mA	쉬움
P4: IMU 타이머 조건부 시작	~41μA (미연결 시)	쉬움
P3: 배터리 타이머 동적	~5μA	보통
P5: main_loop 단발→직접호출 (방안 A)	응답 개선 (전력 변화 미미)	쉬움
P5: main_loop → app_scheduler (방안 B)	코드 품질 + 확장성	보통
P5: IMU FIFO+인터럽트 (방안 C)	연속측정 시 ~50μA 절감	어려움
P6: NRF_LOG 비활성화	~10μA	쉬움
합계 (A+B 적용 기준)	**~1.1-1.6mA**	—

---

7. 코드 품질 개선

7.1 구조 개선

Q1: main.h에서 static 함수 선언 제거

main.h에 `static` 함수 프로토타입이 선언되어 있음 (line 48-54). 이들은 main.c에서만 사용하므로 main.c 내부 forward declaration으로 이동.

Q2: cat_interface.c → TWI 드라이버 분리

현재 `cat_interface.c` 는 EEPROM 드라이버 + TWI 인스턴스. EEPROM 삭제 후 TWI init/uninit만 남김 → `i2c_manager.c` 에 통합 권장 (1.5절 참고).

Q3: extern 변수 과다 사용

cmd_parse.c에 약 30개 이상의 `extern` 변수가 있음. 이는 모듈 간 강한 결합을 나타냄. 장기적으로: - 관련 변수를 구조체로 묶기 - getter/setter 함수 제공 - 전역 상태 최소화

Q4: main_timer.c main_loop() 플래그 기반 폴링

`go_batt`, `go_temp`, `go_pdread`, `go_device_power_off`, `go_NVIC_SystemReset` 등 전역 bool 플래그로 이벤트를 전달하는 패턴. 이벤트 큐 또는 상태 머신으로 개선 가능.

Q5: 매직 넘버 제거

- `cnt_s < 150` → `#define BUTTON_SHORT_PRESS_TICKS 150` (main.c:1404)
- `cnt_s > 1000` → `#define BUTTON_LONG_PRESS_TICKS 1000` (main.c:1411)
- `m_reset_status == 5` → enum 또는 define 사용 (main.c:456)
- EEPROM 주소 (`0x0060`, `0x0065`, `0x0020` 등) → define 사용

7.2 코딩 스타일

항목	현재	권장
인덴트	탭/스페이스 혼용	통일 (탭 권장)
중괄호 위치	혼재 (K&R + Allman)	K&R로 통일
변수명	일부 헝가리안, 일부 snake_case	snake_case 통일
주석 언어	한국어/영어 혼재	코드 주석은 영어 권장

---

8. 작업 우선순위 요약

즉시 수정 (안전/안정성)

#	항목	심각도	파일
1	B4: memset 버퍼 오버런	CRITICAL	main.c
2	B1: binary_tx_handler 크래시	HIGH	main.c
3	B2: BLE TX 에러 핸들링 로직 반전	HIGH	main.c
4	B3: extern + 초기화 동시 사용	HIGH	cmd_parse.c

1차 정리 (미사용 코드/EEPROM 삭제)

#	항목	영향 파일 수
5	미사용 HW 드라이버 삭제 (1.2절)	12 파일
6	cat_interface.c EEPROM 기능 삭제 → TWI만 유지 (1.5절)	5 파일
7	cmd_parse.c EEPROM 호출 → FDS 전환 (1.5.4절)	1 파일
8	미사용 전역변수 + HW_NO 삭제 (2.2-2.3절)	2 파일
9	DEBUG_MINIMAL_BOOT 조건 제거 (3절)	2 파일
10	Keil uvprojx 소스 참조 제거 (4절)	1 파일
11	주석 처리된 코드 삭제 (2.4절)	3 파일

2차 정리 (저전력 최적화)

#	항목	예상 절감
12	P1: UART 비활성화	~1mA
13	P2: RTT 비활성화	~0.1-0.5mA

#	항목	예상 절감
14	P4: IMU 타이머 BLE 연결 시에만 동작	~41μA
15	P5-A: main_loop 단발 이벤트 직접 호출로 전환	응답 개선
16	P5-B: app_scheduler 도입 (선택)	코드 품질
17	P3: 배터리 타이머 동적 조절	~5μA

3차 정리 (코드 품질)

#	항목
18	Q1: main.h static 함수 제거
19	Q2: cat_interface.c TWI를 i2c_manager.c에 통합
20	Q5: 매직 넘버 define화
21	B9: config_load() goto 제거

부록: 삭제 대상 파일 목록

확정 삭제 (회로에 미실장된 HW 드라이버)

```
project/ble_peripheral/ble_app_bladder_patch/ad5272_i2c.c
project/ble_peripheral/ble_app_bladder_patch/ad5272_i2c.h
project/ble_peripheral/ble_app_bladder_patch/ada2200_spi.c
project/ble_peripheral/ble_app_bladder_patch/ada2200_spi.h
project/ble_peripheral/ble_app_bladder_patch/mcp4725_i2c.c
project/ble_peripheral/ble_app_bladder_patch/mcp4725_i2c.h
project/ble_peripheral/ble_app_bladder_patch/mcp4725_adc.c
project/ble_peripheral/ble_app_bladder_patch/mcp4725_adc.h
project/ble_peripheral/ble_app_bladder_patch/ir_i2c.c
project/ble_peripheral/ble_app_bladder_patch/ir_i2c.h
project/ble_peripheral/ble_app_bladder_patch/LED_Parse.c
project/ble_peripheral/ble_app_bladder_patch/LED_Parse.h
project/ble_peripheral/ble_app_bladder_patch/measurements.c
project/ble_peripheral/ble_app_bladder_patch/measurements.h
project/ble_peripheral/ble_app_bladder_patch/measurements_20.c
project/ble_peripheral/ble_app_bladder_patch/measurements_20.h
project/ble_peripheral/ble_app_bladder_patch/full_agc.c
project/ble_peripheral/ble_app_bladder_patch/full_agc.h
project/ble_peripheral/ble_app_bladder_patch/meas_pd_voltage_custom.c
project/ble_peripheral/ble_app_bladder_patch/meas_pd_voltage_custom.h
project/ble_peripheral/ble_app_bladder_patch/debuf_print.h
```

확정 삭제 (EEPROM 미실장 → 기능 삭제)

```
cat_interface.c 에서 삭제할 코드:
  - aes_key[], aes_iv[]
  - encrypt_data(), decrypt_data()
  - eeprom_write_encrypted(), eeprom_read_decrypted()
  - eeprom_write/read_byte/word/page/bytes/uint16_array/uint32()
  (TWI 인스턴스 m_eeprom, eeprom_initialize/uninitialize만 유지 → 리네이밍)

cat_interface.h 에서 삭제할 선언:
  - 위 함수들의 프로토타입 전체
  - EEPROM_I2C_ADDRESS, EEPROM_PAGE_SIZE, SERIAL_ADDRESS 매크로
```

대체 필요 (EEPROM → FDS 전환)

```
cmd_parse.c:
  - eeprom_init_values_read() 함수 전체 삭제 (FDS config_load가 대체)
  - 약 20개소의 eeprom_xxx() 호출 → m_config.xxx + config_save() 패턴으로 전환

main.c:
  - 1417-1425: 버튼 롱프레스 리셋 시 EEPROM → FDS로 전환
  - 1474-1476: Full boot 경로 — DEBUG_MINIMAL_BOOT 삭제와 함께 제거

i2c_manager.c:
  - sw_i2c_init_once()의 mcp4725_init() 호출 삭제 (MCP4725 HW 없음)
  - SW I2C 모드 자체가 MCP4725용이었으므로 SW 모드 전체 삭제 검토
```

삭제 검토 필요 (BLE 커맨드 참조 확인 후 결정)

```
project/ble_peripheral/ble_app_bladder_patch/meas_pd_imm.c/h
project/ble_peripheral/ble_app_bladder_patch/meas_pd_buff.c/h
project/ble_peripheral/ble_app_bladder_patch/meas_pd_voltage_simple.c/h
project/ble_peripheral/ble_app_bladder_patch/meas_pd_voltage_half.c/h
project/ble_peripheral/ble_app_bladder_patch/meas_pd_voltage_full.c/h
```