**step\_06\_20250911\_v1에 대한 미세 개선 사항**

본 문서는 스케치 step\_06\_20250911\_v1의 안정성·가독성·조작감을 향상하기 위한 소규모(리스크 낮은) 개선 제안들을 정리한 것입니다. 각 항목은 독립적으로 적용 가능하며, 실사용 환경(Arduino Nano/Every + SAM9703 기반 GM/GS 모듈)에서의 체감 개선을 기준으로 작성했습니다.

## 1) 스플래시 지연 단축

초기 LCD 스플래시에서 1.2초 대기(delay(1200))는 체감 응답성을 떨어뜨립니다. 가독성을 해치지 않는 선에서 0.5~0.8초로 단축을 권장합니다.

/\* boot banner \*/  
print16("Nano Ardule vX");  
// 기존: delay(1200);  
delay(800); // 또는 delay(500);

## 2) 타이머 드리프트 보정(LED/LCD 등 주기 작업)

현재는 특정 시점에 도달하면 다음 시점을 interval만큼 더하는 방식입니다. 프레임 누락이 생기면 다음 틱을 한 번에 건너뛰어 오차가 누적될 수 있습니다. while 루프를 이용해 누락된 틱을 모두 보상하면 장시간 구동에서 깔끔합니다.

// 기존 예시  
if (millis() >= nextLcdAt) { nextLcdAt += LCD\_REFRESH\_MS; updateLCD(); }  
  
// 개선안(드리프트 보정)  
while ((long)(millis() - nextLcdAt) >= 0) {  
 nextLcdAt += LCD\_REFRESH\_MS;  
 updateLCD();  
}  
  
// 매크로/함수화 예시  
inline void tickCatchUp(unsigned long &next, unsigned long interval, void (\*fn)()){  
 while ((long)(millis() - next) >= 0){ next += interval; fn(); }  
}

## 3) 액티비티/파트 LED 가시성 개선

LED 블링크 시간이 5ms 등 너무 짧으면 육안으로 인지하기 어렵습니다. 활동 LED는 25~40ms, 파트/브라우저 블링크는 250~500ms 권장(사용자 취향에 맞춰 조절).

#define LED\_BLINK\_MS 30 // 활동 LED  
#define PART\_LED\_BLINK\_MS 350 // 파트 표시

## 4) LCD 패딩/문자열 처리 미세 최적화

빈칸 채우기 패딩은 while 루프 대신 한 번에 채우면 CPU 점유율을 줄일 수 있습니다. 현재도 배열 길이 상한이 있어 안전하나, 아래와 같은 헬퍼로 가독성과 성능을 함께 확보할 수 있습니다.

static inline void pad16(char\* dst, size\_t curLen){  
 while (curLen < 16){ dst[curLen++] = ' '; }  
 dst[16] = 0;  
}  
  
// 사용 예시  
char line1[17]; // 내용 채운 뒤  
pad16(line1, strlen(line1));  
lcd.print(line1);

## 5) PROGMEM 문자열 복사 루프 가드 점검

`pgm\_read\_byte(p++)`로 널 종료까지 복사하는 루프는 `buflen-1` 상한이 이미 있어 안전합니다. 다만, 여러 군데 사용할 경우 공용 헬퍼로 묶으면 재사용/테스트가 쉬워집니다.

static inline void pgmCopy(char\* buf, size\_t buflen, const char\* p){  
 size\_t i=0; char c;  
 while (i<buflen-1 && (c = pgm\_read\_byte(p++))) buf[i++] = c;  
 buf[i] = 0;  
}

## 6) 드럼 뱅크 전환 시 LSB(32) 동시 전송

일부 모듈(SAM9703 등)에서 드럼 전환 시 MSB=121만 보낼 경우 드물게 선택 상태가 불안정할 수 있습니다. LSB=0을 함께 전송하고, 중복 코드는 헬퍼로 통일합니다.

static inline void setDrumKit(uint8\_t pc){  
 sendRawCC(CH\_DRUM, 0, 121); // Bank MSB (GS Drum)  
 sendRawCC(CH\_DRUM, 32, 0); // Bank LSB  
 sendPC(CH\_DRUM, pc);  
}

## 7) 볼륨 감도 표준화(VOL\_STEP)

엔코더 하드웨어 차이(펄스×4/×2 등)와 사용자 취향을 흡수하기 위해 `VOL\_STEP` 매크로로 일괄 조절하고, `UI\_VOL` 분기에서는 오직 이 기준만 사용합니다.

#define VOL\_STEP 4  
  
static inline void adjustVolume(int delta){  
 int step = (delta>0 ? VOL\_STEP : -VOL\_STEP);  
 // MODE\_AB / MODE\_A / MODE\_B / MODE\_CH10 각각에 대해 0~127 범위 보정 + CC7 전송  
}

## 8) 브라우저 MSB/LSB 표기 혼동 방지(주석/리네임)

구현 상 `browser.lsb`가 실제로는 MSB 의미로 쓰이는 경우가 있습니다. 장기 유지보수를 위해 변수명을 정정하거나(권장), 최소한 파일 상단에 굵은 주석을 남깁니다.

// NOTE: 이 펌웨어에서 'browser.lsb'는 장치 사양상 MSB 용도로 사용됩니다.  
// 가능하면 변수명을 'browser.msb'로 변경하세요.

## 9) millis() 비교 관용구 통일

오버플로우(약 49.7일) 상황에서도 안전하도록 아래 관용구를 프로젝트 전반에서 통일합니다.

if ((long)(millis() - nextDeadline) >= 0) {  
 nextDeadline += interval;  
 // ...  
}

## 10) LCD 갱신 스로틀 값 점검

`LCD\_REFRESH\_MS`가 지나치게 짧으면 깜빡임이 보이고, 너무 길면 반응성이 떨어집니다. 보통 60~120ms 범위에서 테스트하여 최적점을 찾으세요. (현재 체감상 80~100ms 권장)

※ 모든 변경은 독립적으로 적용 가능하며, 기능 변경보다 사용자 체감 및 안정성 개선을 우선한 제안입니다.