Simulacro de Examen – Temporizador (min/seg) con Interrupciones + EEPROM

Objetivo

Implementar un temporizador configurable **minutos/segundos** usando **4 botones**, **2 LEDs**, **EEPROM**, **Serial** y **al menos dos interrupciones** (Arduino UNO, proyecto modular).

Estructura

- `main.ino` → `setup()` y `loop()` mínimos (delegan en funciones).
- `funciones.h` → pines, prototipos, y defines.
- `funciones.cpp` → lógica: FSM simple, EEPROM, Serial.

Pines sugeridos

- `BTN_UP` → : incrementa el **parámetro activo** (con interrupción).
- `BTN_DOWN` → : decrementa el **parámetro activo** (con interrupción).
- `BTN_MODE` → : alterna parámetro activo **MIN** ↔ **SEG** .
- `BTN_SAVE` → : guarda MIN y SEG en EEPROM (por polling).
- `LED_MIN`,`LED_SEG` → :(indican parámetro activo).

Requerimientos

1. **Parámetros**
    - `MIN` (0–59) y `SEG` (0–59).
    - `BTN_MODE` alterna el parámetro activo; `LED_MIN/LED_SEG` lo indican.
2. **Botones o configuracion sugerida**
    - `BTN_UP` : **+1** sobre el parámetro activo .
    - `BTN_DOWN` : **–1** sobre el parámetro activo .
    - `BTN_MODE` : alterna **MIN/SEG** .
    - `BTN_SAVE` : guarda valores en EEPROM y confirma por Serial/LEDs.
3. **Interrupciones**
    - Usar **dos ISRs** .
    - **funciones de interrupcion cortas**: sólo setean **banderas**.
4. **Antirrebote (OPCIONAL)**
    - Con `millis()` en el procesamiento de banderas.
    - No usar `delay()` dentro de ISRs(Funciones de interrupción).

5. **EEPROM**
    - Al iniciar: leer `MIN`, `SEG` .
    - Si inválido → iniciar `MIN=0`, `SEG=0`.
    - Al guardar: escribir `MIN`, `SEG`.