Colorimetro su Basys MX3 (PIC32MX370) con sensore TCS34725

1. Obiettivo

Realizzare un colorimetro che acquisisce i valori RGBC dal sensore **TCS34725**, li normalizza in RGB (0–255), classifica il colore e **conteggia gli eventi di "rosso"**. Il conteggio viene salvato in **memoria Flash SPI** alla pressione del pulsante **BTNC**. L'utente interagisce tramite **menu su UART** e visualizza i valori anche su **LCD**.

2. Hardware e collegamenti

Componenti

- Board: Basys MX3 con MCU PIC32MX370
- Sensore colore: TCS34725 (RGBC, I2C)
- LCD onboard (interfaccia PMP)
- Speaker/buzzer onboard (PWM via Output Compare)
- Memoria Flash SPI onboard
- LED RGB onboard, pulsante BTNC

Collegamenti chiave (verificati in hardware)

- **TCS34725** su header I2C dedicato: SDA, SCL, GND, 3V3
- **LED del sensore** collegato fisso a 3V3 (sempre acceso durante le misure)
- **UART4**: TX su RF12, RX su RF13 (PPS)
- **Beep**: OC1 su RB14 (PPS), Timer2

3. Clock e configurazione (config bits)

La board usa l'oscillatore esterno da **8 MHz** con **PLL attivo** per ottenere **SYSCLK = 80 MHz** e **PBCLK = 40 MHz**. Questa scelta garantisce una temporizzazione stabile per periferiche (UART/I2C/PMP) e delay basati su core timer.

4. Architettura software (modulare)

Il firmware è organizzato in moduli indipendenti: inizializzazione board, driver periferiche, driver sensore, logica applicativa a stati.

```
Struttura logica
- config_bits.h : configurazione clock / fuse
- utils.* : delay/millis con core timer
- board.* : init MCU (es. JTAG off, GPIO base)
- uart4.* : driver UART4 (PPS RF12/RF13, OERR, I/O non-blocking)
- i2c.* : I2C master (transazioni base)
- tcs34725.* : init + lettura RGBC + conversione RAW→RGB8
- lcd_pmp.* : LCD via PMP write-only con delay fissi
- flash_spi.* : salvataggio/lettura/erase conteggio in flash
- app.* / main.c: menu UART + macchina a stati (scan, stop, visualizza, reset)
```

5. Acquisizione e classificazione colore

Lettura sensore

Il TCS34725 fornisce valori **RAW a 16 bit** per i canali **R, G, B** e **Clear (C)**. I valori RAW possono superare 255 (non sono RGB "standard").

La lettura avviene in stato di "scan" e viene stampata in debug su UART e mostrata su LCD.

Normalizzazione RAW → RGB (0–255)

Per ottenere valori RGB comparabili, i canali R/G/B vengono **scalati rispetto al canale Clear (C)**: **RGB8 = clamp(RAW * 255 / C)**, con soglia minima su C per evitare rumore e divisioni instabili.

Output finale: tre interi 8-bit (0–255) pronti per classificazione e visualizzazione.

Regola di riconoscimento "ROSSO" (permissiva)

Dopo la conversione in RGB8, il colore rosso viene classificato con soglie robuste (tolleranti a luce e stampa):

```
ROSSO se:
- C >= C_MIN (soglia anti-rumore)
- R >= 200
- G <= 50
- B <= 50
```

6. Interazione utente: UART, LCD, beep, Flash

UART (menu + debug)

- UART4 con PPS su RF12 (TX) / RF13 (RX)
- Stampa continua dei valori e messaggi di stato
- Menu testuale per selezione funzioni
- Gestione errore OERR e routine non-blocking

LCD (PMP write-only)

- Scrittura senza busy-flag (solo delay fissi)
- Layout: riga 0 = R, riga 1 = G/B alternati
- Reset stato LCD a inizio/fine scansione

Beep inizio scansione (PWM)

- OC1 su RB14 (PPS), Timer2
- 10 kHz, duty 50% (feedback immediato di avvio)

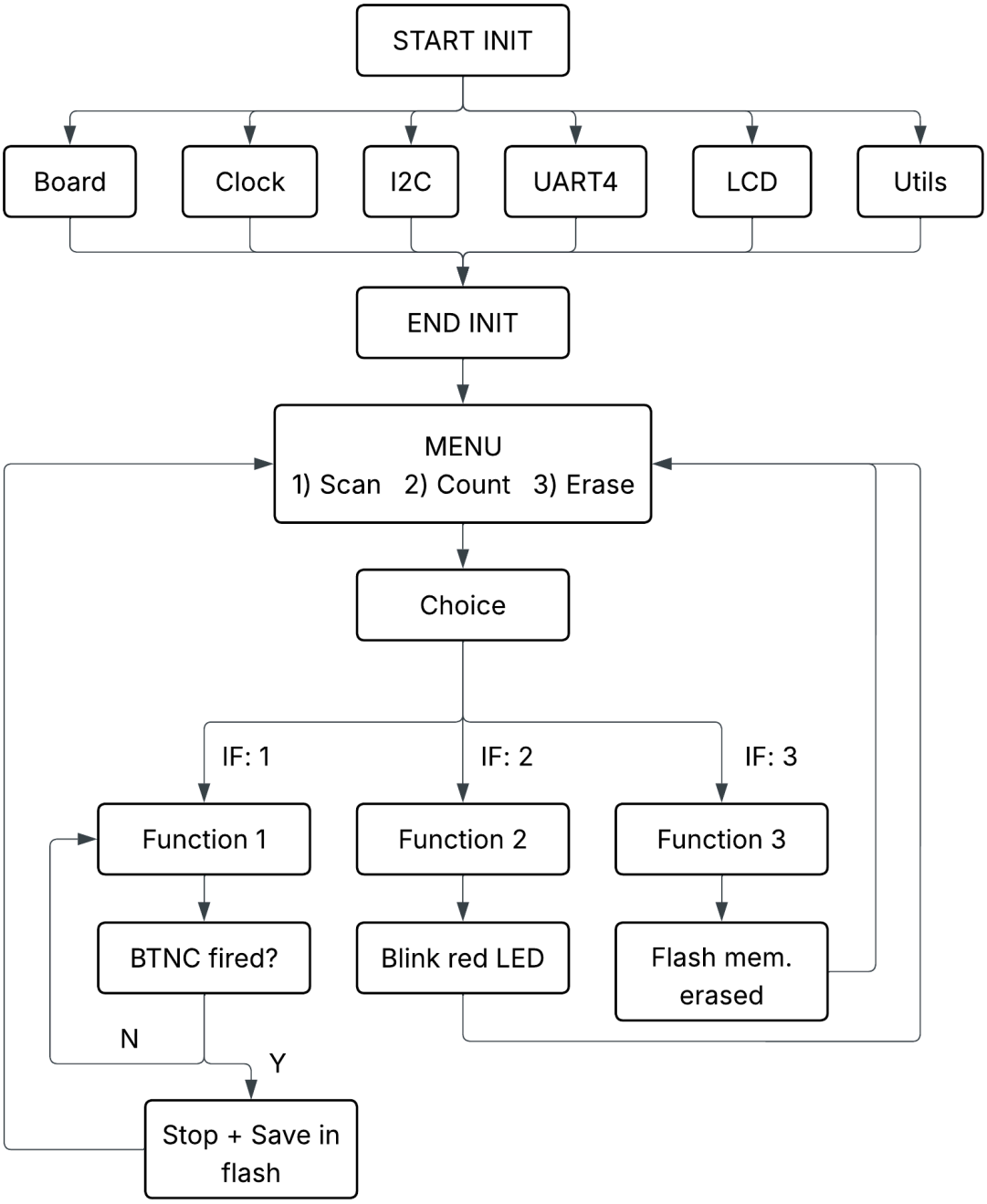
BTNC + Flash SPI

- BTNC su interrupt esterno (INT4): stop scansione
- Salvataggio del conteggio "rosso" in Flash SPI
- Funzione "erase" per azzeramento memoria

7. Funzioni richieste (menu UART)

- **Funzione 1 – Scan**: beep, lettura continua RGBC → RGB8, classificazione, conteggio rosso, output UART + LCD. BTNC interrompe e salva in Flash.
- **Funzione 2 – Mostra conteggio**: legge il valore in Flash, lo stampa su UART e fa lampeggiare LED RGB rosso (0.5 s × N).
- **Funzione 3 – Reset**: erase della Flash (conteggio azzerato) e conferma su UART.

8. Flowchart del programma



9. Note di compilazione/esecuzione (essenziali)

Compilazione e flash da MPLAB X (XC32). Per il test: aprire un terminale seriale (PuTTY) sulla porta COM della Basys MX3 e selezionare il baudrate impostato nel progetto; usare il menu UART per avviare scan/lettura/reset.