HeartBeat

Programmazione di Microcontrollori (ProMC)

Sommario

Questo rapporto descrive, in maniera sintetica, un sistema di monitoraggio del battito cardiaco (*Heart Beat*) realizzato sulla board BasysMX3 (con microcontrollore `PIC32MX370F512L`) e utilizzante il sensore `KY-039`. Verranno illustrate le funzioni principali del programma, corredate da alcuni snippet di codice per dare un'idea della struttura generale del progetto.

1. Introduzione

Il sistema Heart Beat misura il battito cardiaco posizionando un dito sul sensore `KY-039` (basato su LED IR e fototransistor). I BPM (battiti per minuto) calcolati vengono mostrati:

- Sull'LCD integrato (via *PMP*);
- Sul Serial Plotter collegato tramite `UART`.

Inoltre sarà possibile salvare il valore massimo di bpm in una memoria Flash On-Board, e ripristinarlo in un secondo momento, oppure eliminarlo (*reset*).

2. Funzionalità Principali

All'avvio, il LED RGB si illumina di colore verde, e un menù sul LCD propone tre opzioni:

1. **HeartBeat**: avvio del monitoraggio;
2. **Max bpm**: visualizzazione del bpm massimo memorizzato;
3. **Reset**: cancellazione dati in memoria.

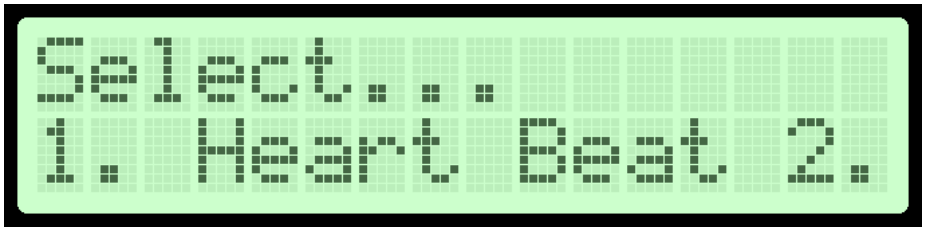


Figure 1: Menu

3. Struttura del Codice

Inizializzazione

Il microcontrollore viene inizializzato configurando le varie periferiche (`Timer`, `UART`, `PMP`, `SPI`, `ADC`, ...). Le funzioni e procedure di inizializzazione di ogni periferica sono contentute nelle librerie specifiche ad ognuna di esse, e sono perlopiù simili o identiche a quelle sviluppate durante le ore di laboratorio.

Fanno eccezione solo alcune (come `Timer`, `GPIO`, `PMP` e `KY-039`) che sono state personalizzate per il progetto.

```c
int main(void) {
    // GPIO Initialization
    rgb_pins_init();
    btn_pins_init();
    btn_interrupt_init();
    speaker_pins_init();
    // Timer Initialization
    timer_init();
    // UART Initialization
    uart_init_pins();
    uart_init(9600);
    // PMP Initialization
    lcd_init();
    // KY-039 Initialization
    ky39_init();
    // ADC Initialization
    adc_init();
    // Output Compare Initialization
    oc_init();
    // SPI Initialization
    init_spi1();
    rgb_set_color(0, 1, 0);
    stop_lcd = 0;
    // Initialize the sensor readings buffer
    for (int i = 0; i < READ_BUFSIZE; i++) reading_buf[i] = 0.0;
    while (1) {
        // ...
    }
    return 0;
}
```

Main Loop (*Gestione Menu*)

Nel *loop principale* del programma, questo mostra il menù in *Figure 1* e attende la ricezione di una scelta, da parte dell'utente, tramite la console seriale (`UART`).

La selezione invoca la funzione corrispondente:

```
// ...
while (1) {
    char *hb = "1";
    char *mb = "2";
    char *rs = "3";
    start_menu();
    if (flag_rx) {
        char *str = get_strg();
        if (strcmp(str, hb) == 0) {
            heart_beat();
        } else if (strcmp(str, mb) == 0) {
            max_bpm();
        } else if (strcmp(str, rs) == 0) {
            reset_max_bpm();
        } else {
            invalid();
        }
    }
    flag_rx = 0;
    sleep(20);
}
```

Funzione `heart_beat()`

La funzione `heart_beat()` si occupa di monitorare il battito cardiaco, calcolando i BPM e visualizzandoli sul display.

Ciò viene effettuato dapprima avviando l'acquisizione di dati dal sensore `KY-039`, notificando l'avvio tramite un *beep* dallo speaker.

Una volta operativa, il `LED RGB` cambia colore, da verde a blu, ed il display LCD viene popolato con il valore di BPM corrente, che viene aggiornato circa ogni 10 secondi.

Nota:

> il valore acquisito dal sensore non è immediatamente traducibile in battiti al minuto, e quindi si devono prevedere almeno $20 - 30$ secondi di acquisizione prima di avere una lettura realistica; inoltre il sensore è molto suscettibile alla luce naturale ed artificiale presente nell'ambiente di utilizzo e quindi è consigliabile utilizzarlo in condizioni di scarsa luminosità o di coprirlo durante la lettura.

```c
void heart_beat(void) {
    rgb_set_color(0, 0, 1);
    // ...
    busy_measure = 1;
    beep(10);
    t3_start();
    while(busy_measure) {
        int read = ky39_read();
        // The signal received from the sensor is passed through a Low-Pass
        // filter which trims off high frequencies
        float tmp = (0.1 * read) + ((1.0 - 0.1) * previous_lowpass);
        previous_lowpass = tmp;
        // The signal is then stored in a circular buffer of size READ_BUFSIZE
        buf_sum -= reading_buf[read_buf_index];
        reading_buf[read_buf_index] = tmp;
        buf_sum += reading_buf[read_buf_index];
        read_buf_index = (read_buf_index + 1) % READ_BUFSIZE;
        filtered_sig = buf_sum / READ_BUFSIZE;
        // Every 1000 reading (10 seconds) are saved in a vector, which
        // is then used to compute the BPM through an approximated peak count.
        if (cur_reading_idx < READINGS_LEN)
            readings[cur_reading_idx++] = filtered_sig;
        else {
            bpm = current_bpm();
            cur_reading_idx = 0;
            if (max_bpm_m <= bpm) max_bpm_m = bpm;
        }
        // To show a graph on the serial plotter be sure to transmit a
        // "carriage return" after every data entry.
        sprintf(str_bpm, "%f\r\n", filtered_sig);
        uart_puts_4(str_bpm);
    }
    write_flash(MAX_BPM_FADDR, max_bpm_m);
    t3_stop();
    stop_lcd = 0;
    rgb_set_color(0, 1, 0);
}
```

Il segnale letto viene inoltre passato in un *low-pass filter* che riesce a rimuovere le alte frequenze che disturbano la lettura, e viene memorizzato in un buffer circolare di dimensione `READ_BUFSIZE`.

Una volta acquisiti 1000 valori (circa 10 secondi), questi vengono utilizzati per calcolare i BPM attraverso un conteggio approssimato dei picchi. Questo valore viene infine memorizzato in una variabile globale `max_bpm_m` e scritto in memoria Flash.

Per arrestare la misurazione, l'utente può premere il pulsante `BTNC`.

Funzione `max_bpm()`

La funzione `max_bpm()` si occupa di visualizzare il valore massimo di BPM memorizzato in memoria Flash.

Se la memoria Flash si trova a stato di `RESET` (valore `255`), viene visualizzato `--` al posto del valore in BPM.

```
void max_bpm(void) {
    int max_bpm = read_flash(MAX_BPM_FADDR);
    char str_bpm[16];
    if (max_bpm == 255) sprintf(str_bpm, "Max BPM:      --");
    else sprintf(str_bpm, "Max BPM: %7d", max_bpm);
    busy_measure = 1;
    while (busy_measure) {
        clr_lcd();
        puts_lcd(str_bpm);
        nl_lcd();
        puts_lcd("################");
        sleep(1000);
    }
}
```

Per uscire dalla visualizzazione della massima frequenza cardiaca, l'utente può premere il pulsante `BTNC`.

Funzione `reset_max_bpm()`

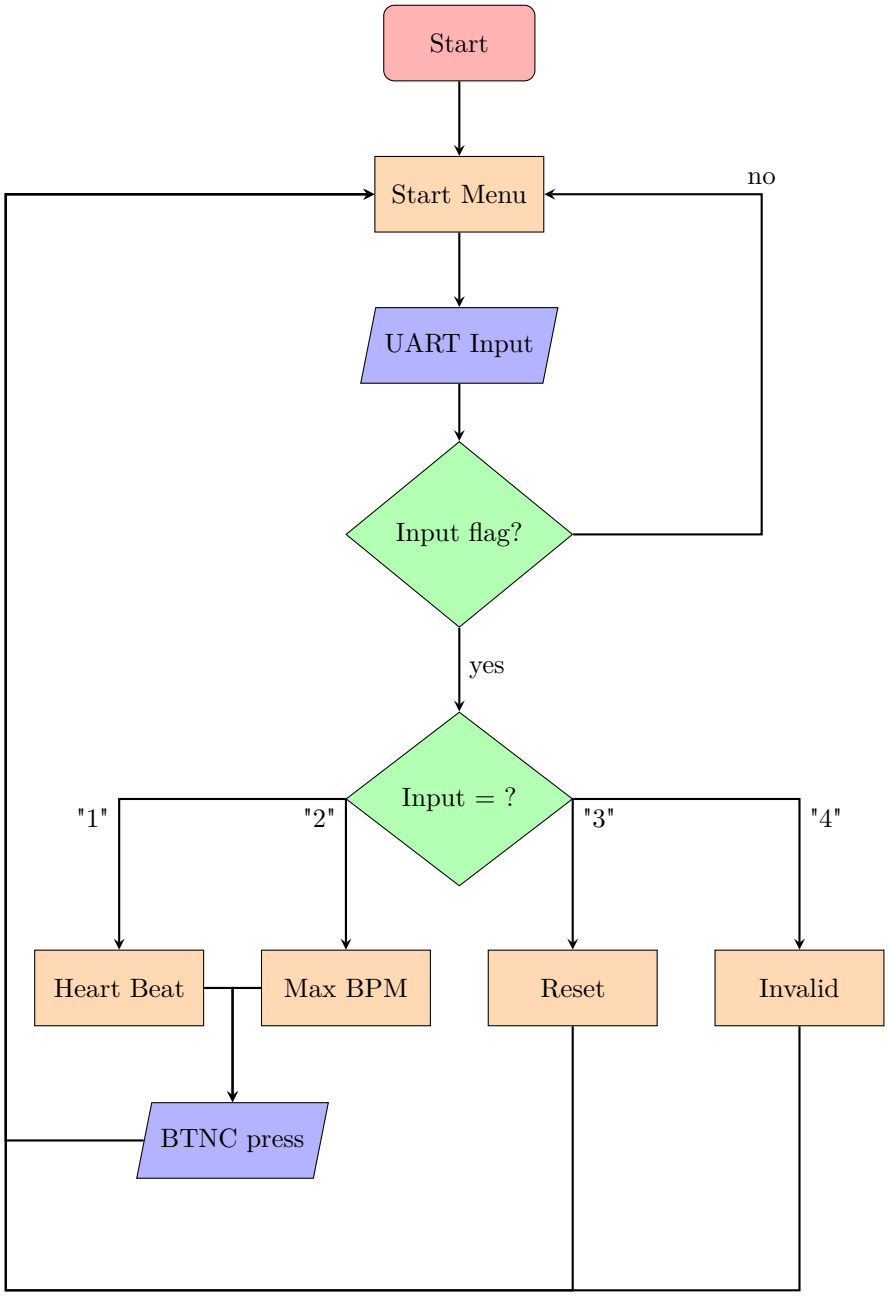
La funzione `reset_max_bpm()` si occupa di azzerare il valore massimo di BPM memorizzato in memoria Flash.

Viene cancellata tutta la memoria flash, per assicurarsi che nessun dato sia presente, e viene visualizzato un messaggio per 2 secondi. Fatto ciò, il sistema torna al menù principale.

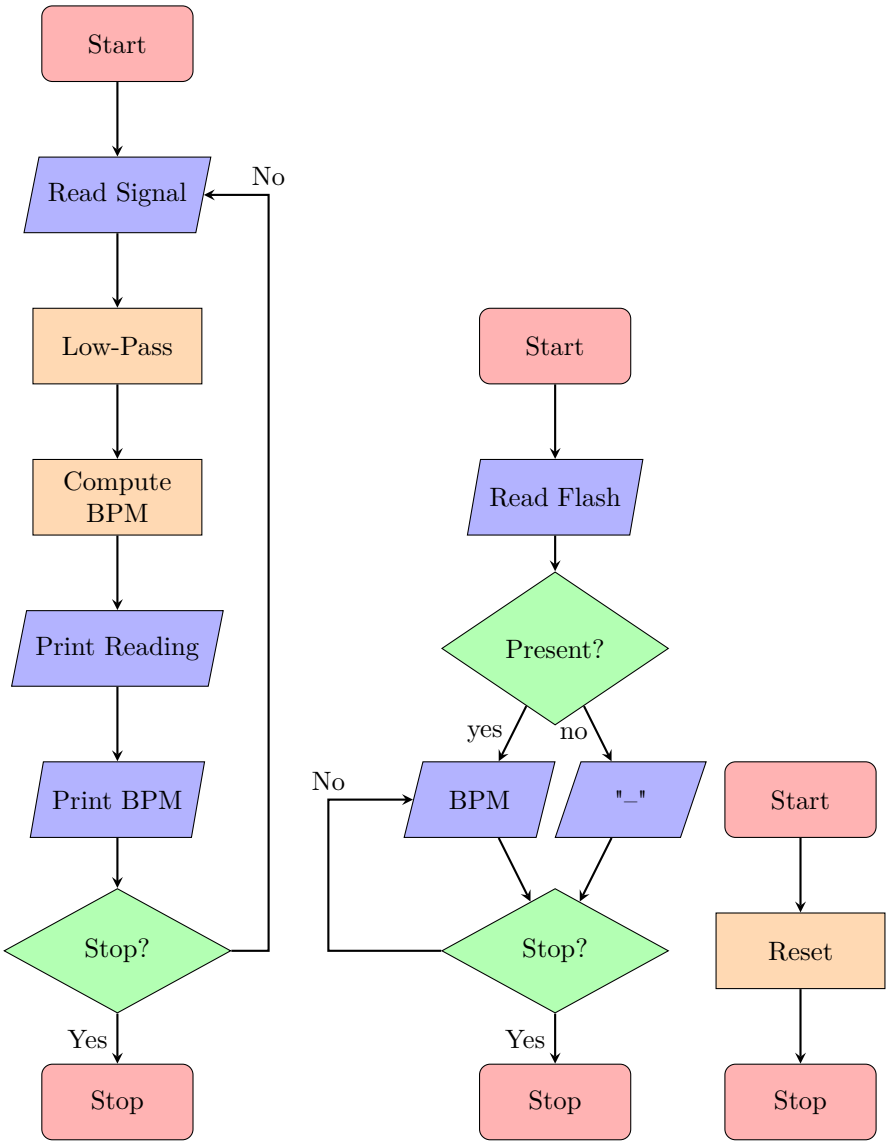
```
void reset_max_bpm(void) {
    // ...
    erase_flash();
    // ...
}
```

Flowchart

Main Loop



Heart Beat, Max BPM, Reset



Conclusioni

Il progetto *HeartBeat* è stato realizzato con successo, e tutte le funzionalità previste sono state implementate e testate.

Il sistema è in grado di misurare il battito cardiaco, visualizzarlo sul display LCD e salvarlo in memoria Flash, per essere recuperato in un secondo momento.