Progetto Sistemi Embedded

Fabio Casiraghi 807398 f.casiraghi@campus.unimib.it
Tommaso Carboni 808431 t.carboni@campus.unimib.it
Giacomo Elemi 806904 g.elemi@campus.unimib.it

17 luglio 2018

Livella con sensore di temperatura

Il progetto, come da indicazioni, rappresenta una livella con sensore di temperatura, il codice eseguito sul microcontrollore calcola gli angoli rispetto all'inclinazione dello stesso rispetto ai 3 assi e le mostra a display. Sotto queste misure viene visualizzata la temperatura percepita dal termometro. Il progetto è suddiviso su più file .c con rispettivi header. Abbiamo scelto questo tipo di implementazione per alleggerire il più possibile il codice (data la scarsa quantità di memoria disponibile sul microcontrollore) e rendere più chiaro comprendere il compito svolto da ogni segmento di codice.

File main.c

Il file main contiene la routine principale, che si occupa in primo luogo di settare i principali registri del microcontrollore, successivamente inizializza e attiva il timer 4 e per ultimo eseguo un ciclo infinito in cui vengono, ad ogni iterazione, eseguite le funzioni relative a ciascuna delle flag attive.

- **init()** Funzione che si occupa di settare i registri principali, in particolare quelli relativi al watchdog, velocità del ciclo di clock, abilita le interrupt globali.
- **init_t4()** Funzione che inzializza il timer 4, settando starting e reload value.

- **t4()** Eseguita ogni volta che viene generato il segnale d'interrupt 16, legato all'overflow del timer 4 (ogni 100ms). Utilizza dei contatori per tenere il conto del tempo passato e attivare le flag corrispondenti.
- **main()** Esegue tutte le inizializzazioni necessarie e avvia il loop infinito che controlla le varie flag ed esegue le funzioni relative se queste flag sono attive.

File smbus.c

Il file contiene l'interrupt service routine relativa al Smbus, necessarie per l'invio e ricezione di dati lungo il bus.

- **SM_Send()** Procedura per l'invio dei dati lungo l'Smbus. Prende in input il dispositivo a cui inviare i dati (chip_select), la sorgente dei dati da inviare (*src), la quantità di dati da inviare (len) e la modalità relativa alla trasmissione (mode).
- **SM_Receive()** Procedura per la ricezione dei dati dalla scheda lungo l'Smbus. Prende in input il dispositivo da cui ricevere i dati (chip_select), la destinazione in cui salvare i dati (*dest) e la quantità di dati da leggere (len).
- **SW_Routine()** ISR dell'Smbus, legata all'interrupt 7, che contiene la routine di switch relativa al registro di stato del bus. Ogni stato esegue le procedure standard consigliate nella tabella 1 del pdf Serial Communication with the SMBus della Silicon Labs.

File acc.c

File contenente le procedure relative alla lettura e calcolo della posizione, lungo i 3 assi, recepita dall'accelerometro.

- **read_angles()** Funzione che riceve le posizioni calcolate dall'accelerometro e le salva nei rispettivi buffer.
- **med_angles()** Calcola la media delle posizioni contenute in ciascun buffer.
- **compose_line()** Scrive le medie calcolate nelle posizioni corrette del buffer formante la prima linea visualizzata sul display.
- **accMain()** Main relativo all'accelerometro, esegue le tre funzioni descritte precedentemente in sequenza.

File pwm.c

Il file contiene le istruzioni necessario per l'inizializzazione e gestione del PWM relativo alla retroilluminazione dell'LCD.

- **timer0()** Inizializza il timer 0 settando valori iniziali e di reload
- **interrupt_timer0()** Funzione collegata all'interrupt 1 (overflow timer 0), gestisce l'onda del duty cicle.
- **timer2()** Inizializza il timer 2, prende come parametro il valore da cui far ripartire il timer dopo il reload.
- **interrupt_timer2()** Legata all'interrupt 5 (overflow timer 2), esegue ogni 10ms e controlla se il pulsante P3.7 viene tenuto premuto per più di un secondo, settando o meno la modalità di configurazione.
- **init_button()** Inizializza il bottone P3.7 e setta il valori base del Led.
- **click_button()** Legata all'interrupt 19, entra in esecuzione ogni qual volta il pulsante P3.7 viene premuto. Quando ciò avviene viene lanciato il timer 2 per controllare per quanto a lungo il pulsante è stato premuto e avviando la sequenza di modulazione nel caso venga schiacciato per più di un secondo.
- **pwm()** Setta i valori iniziali per la procedura di modulazione del segnale, come ad esempio luminosità, modalità, stato del Led e direzione della modulazione.
- **pwmMain()** Main relativo al procedimento PWM, esegue le prime due funzioni per le inizializzazioni necessarie e l'ultima per avviare il timer 0.

File temp.c

File contenente i segmenti di codice relativi alla ricezione e conversione della temperatura percepita.

- **ShiftTemp()** Questa funzione ha lo scopo principale di convertire e unire i due registri da 8 bit su cui è diviso il dato riguardante la temperatura. La funzione converte i due registri in valori interi, successivamente pulisce ciascun registro per evitare bit "sporchi" e infine unisce i due registri in un'unica variabile.

- **Divide()** La funzione divide il valore della temperatura in cifre (decine e unità) ed a ciascun valore somma l'esadecimale 0x30 per ottenere così il valore ASCII relativo alla cifra.

- **tempMain()** Main del file. Prima invia la richiesta di ricezione dei registri contenenti la temperatura successivamente esegue le due funzioni precedentemente descritte per pulire, unire e convertire i valori ottenuti.