TEMA D'ESAME

Esercizio 1 (max 30 punti)

Sviluppare le seguenti funzionalità utilizzando la scheda LANDTIGER e il system-on-chip LPC1768.

1) Si vuole usare il TIMER 1 dell'LPC1768 come generatore di numeri. Per questo, si programmi il timer in modo ciclico senza interruzioni con un periodo di 2,43 minuti.
2) Si devono acquisire **N** numeri interi da inserire in un vettore **VETT**. I numeri da memorizzare sono senza segno e minori di 256 (unsigned char).
   - La pressione del pulsante KEY1 permette di acquisire un valore leggendo il registro TC dal timer predisposto
   - Dal valore letto sono estratti 8 bits (da indice 23 a indice 16) per essere scritti in una variabile unsigned char chamata VAL.
3) Durante l'acquisizione dei numeri, i LED da 11 (MSB) fino a 4 (LSB) mostrano la codifica binaria dell'indice del valore da acquisire attualmente. Ad esempio, inizialmente i LED saranno tutti spenti, una volta acquisito il primo numero sarà mostrato 1, e così via fino a N-1.
4) Ogni volta che si acquisisce un valore, si deve lanciare la seguente funzione scritta in linguaggio ASSEMBLER che ordina il dato nuovo nel vettore VETT; i valori maggiori devono occupare le posizioni meno significative del vettore, alla posizione 0 risulta essere memorizzato il massimo attuale.

```
unsigned char get_and_sort(unsigned char* VETT, unsigned char VAL, int n);
/* dove VAL è il nuovo numero acquisito */
 la funzione restituisce
      o Il valore VAL
```

5) Una volta acquisiti gli **N** valori in VETT, questi devono essere mostrati in sequenza sui LED con una frequenza di ½ Hz, seguita da LED tutti spenti.
6) Il processo ricomincia da 2).

Sicuramente servono le librerie per timer, LED e pulsanti (le altre possono essere rimosse).

Il timer (TC) inizia a contare e, raggiunto un certo valore massimo, si resetta: premendo KEY1 vogliamo prelevare il valore del timer in quel momento, ma senza fermarlo o resettarlo.

Il timer è da 2.43 minuti, quindi $T = 2.43\text{min} = 2.43 \cdot 60\text{s} \rightarrow \text{count} = T \cdot 2.5 \cdot 10^6 = 0\text{xD9424940}$: i tre bit relativi a Stop/Reset/Interrupt devono essere settati in modo tale che non interrompa (I = 0), resetti al fondo (R = 1) e non stoppi (S = 0), quindi vale 010 = 2.

Per isolare i bit dal 16 al 23, si può:

- usare una maschera 0x00FF0000 e poi shiftare a destra di 16 bit (migliore per byte);
- shiftare a sinistra di 8 bit e poi a destra di 24 bit (migliore per non multipli del byte).

N.B: al punto 3 vuole i LED accesi con posizioni inverse da quelle fornite dal normale LED_out, quindi bisogna modificare la funzione (al posto di `LED_on(i)` fare `LED_on(7 – i)`).

sample.c

```
int main(void){
   …
   //init_timer(timer, prescaler, MR, SRI)
   init_timer(0, 0, 0, 2, 0x); //T = 2s -> count = 2 * 2.5 * 10^6 -> count_16 = …
   init_timer(1, 0, 1, 2, 0xD9424940); //MR1
   enable_timer(1);
}
```

IRQ_RIT.c

```c
#define N 5
unsigned char vett[N];
int val_TC;
char val_char;
int n = 0;

void RIT_IRQHandler(void){
   down++;
   if((LPC_GPIO2->PIOPIN & (1 << 11)) == 0){
      reset_RIT();
      switch(down){
         case 1:
            //disabilitare e riabilitare rende preciso il campionamento
            disable_timer(1);
            val_TC = LPC_TIM1->TC; //lettura di TC
            enable_timer(1);

            val_char = (val_int & 0x00FF0000) >> 16; //estraggo il byte
            vect[n] = val_char;
            n++;
            get_and_sort(vect, val_char, n); //funzione che ordina il vettore

            if(n < N){
               LED_Out(n);
            }
            else{
               LED_Out(vect[0]);
               enable_timer(0);
               i++;
            }
            break;

         else:
            down = 0;
            disable_RIT();
            reset_RIT();
            NVIC_EnableIRQ(EINT1_IRQn);
            LPC_PINCON->PINSEL4 |= 1 << 22;
            break;

         default:
            break;
      }
}
```