Progetto Sistemi Elettronici – Tesina 20

Descrizione Hardware

- Microcontrollore: PIC16F887
- Ambiente di sviluppo: Microchip MPLAB X IDE
- Linguaggio: Assembly
- Gestione eventi: microcontrollore in modalità sleep (se possibile) in assenza di eventi da processare.

Descrizione Software

Si realizzi un firmware che legga ogni minuto la temperatura dal sensore presente sulla scheda e suoni un allarme sonoro quando questa supera una certa soglia. Ad ogni lettura del sensore si faccia lampeggiare brevemente un LED.

- Eventi gestiti tramite interrupt;
- Overflow di un minuto gestito tramite TIMER1
- Blink di un led prima della misurazione
- Conversione del valore di temperatura da tensione a gradi centigradi;
- Suono di un buzzer quando la temperatura rilevata è superiore alla soglia

Descrizione Programma

Il programma viene gestito tramite l'utilizzo di interrupt.

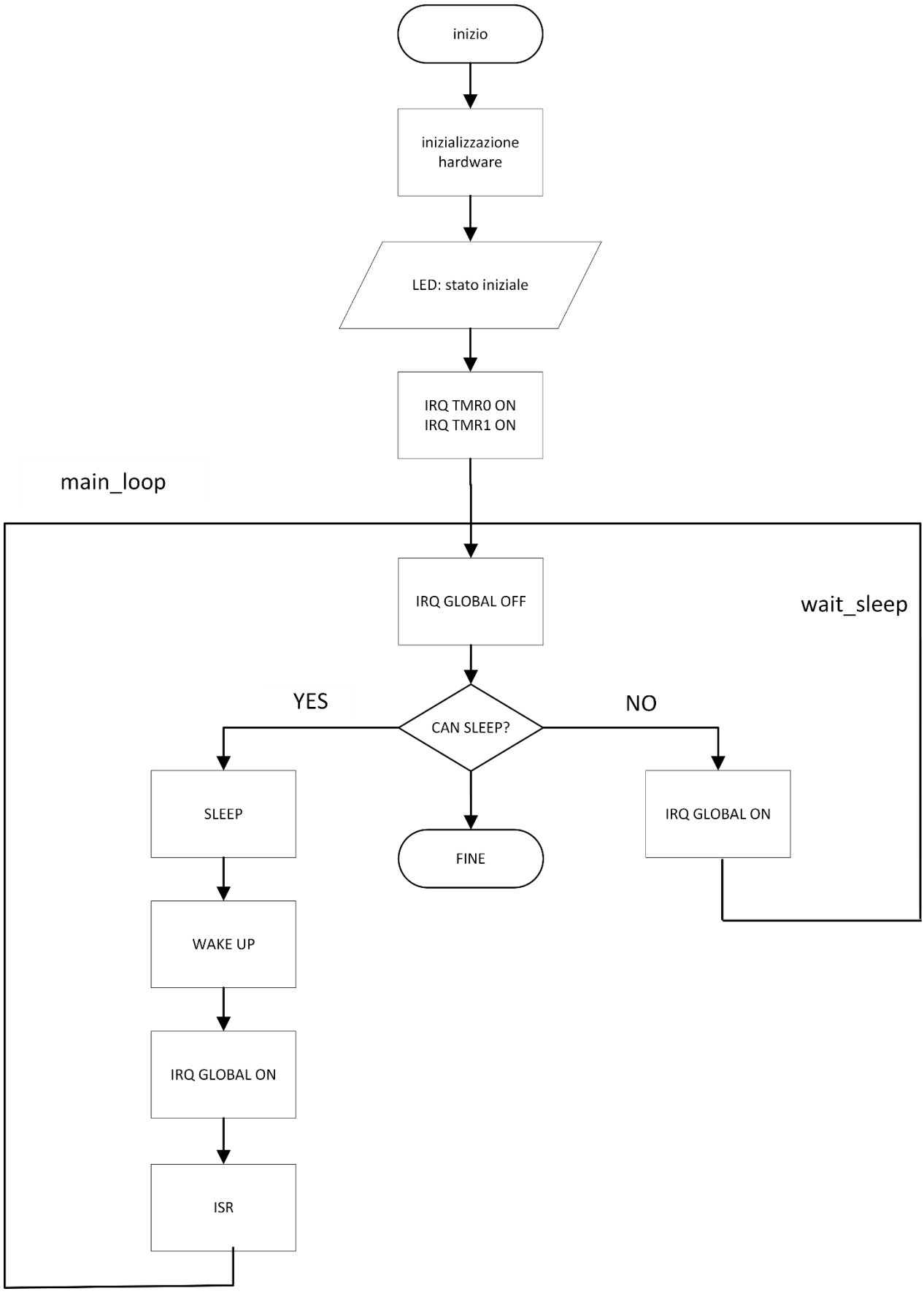
Nella fase iniziale del programma verrà caricato timer1 e un counter per gestire l'overflow del minuto. Nell'interrupt verranno gestite la maggior parte delle operazioni principali, per una questione di comodità e semplicità.

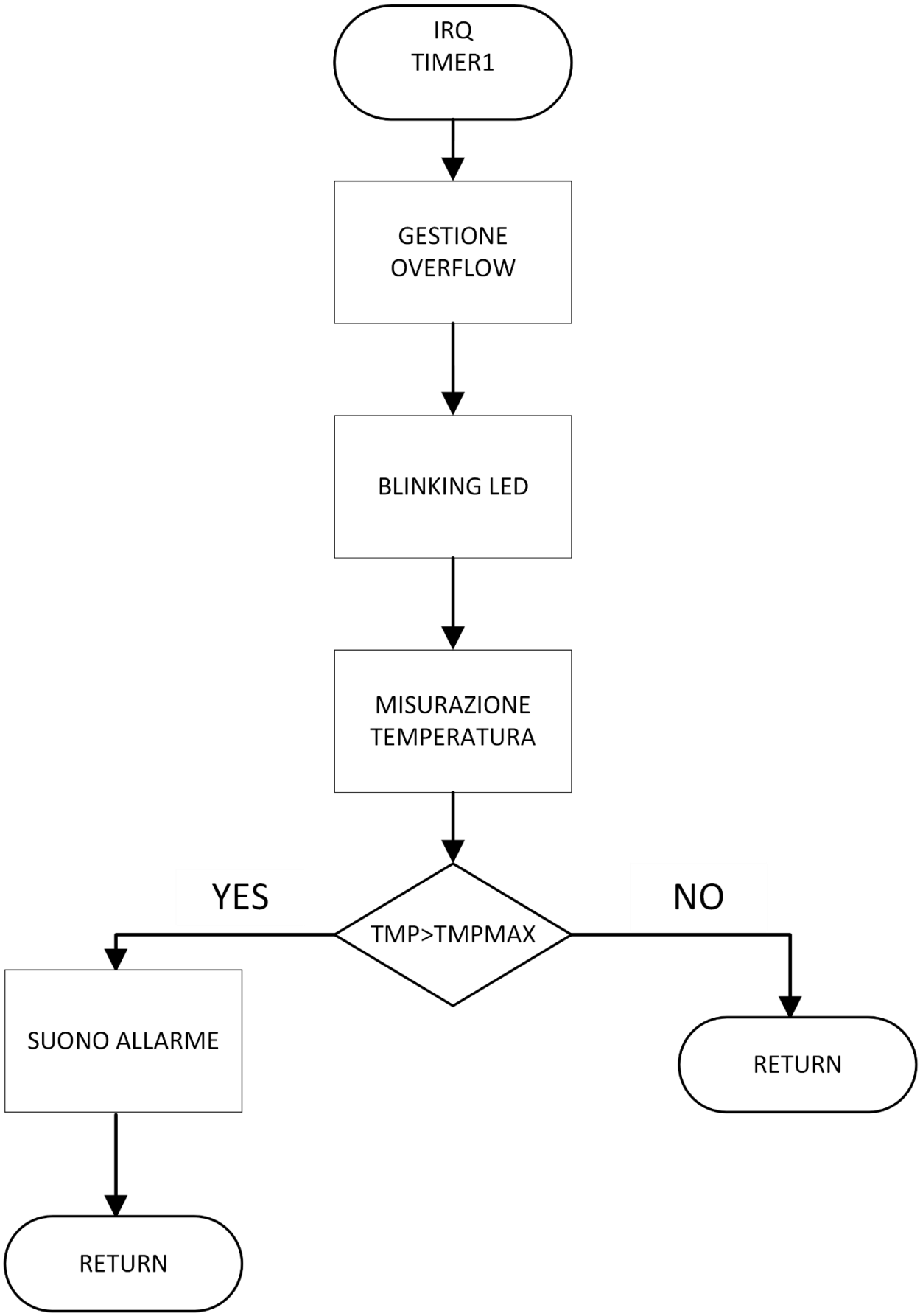
Come visto sopra, una volta eseguito l'overflow di timer1, verrà fatto lampeggiare il led sulla board per qualche secondo, in questo modo capiremo che il programma sta per avviare la misurazione.

Avvenuta la misurazione, che è l'unico evento non gestito nell'interrupt, sia andrà a verificare se la temperatura rilevata è superiore quella di soglia, suonando eventualmente, l'allarme.

Di seguito viene riportato il flowChart e una spiegazione più dettagliata del progetto nelle sue componenti.

Flowchart:


inizio
inizializzazione hardware
LED: stato iniziale
IRQ TMR0 ON
IRQ TMR1 ON
main_loop
IRQ GLOBAL OFF
wait_sleep
YES
CAN SLEEP?
NO
SLEEP
FINE
IRQ GLOBAL ON
WAKE UP
IRQ GLOBAL ON
ISR


IRQ
TIMER1
GESTIONE
OVERFLOW
BLINKING LED
MISURAZIONE
TEMPERATURA
YES
TMP>TMPMAX
NO
SUONO ALLARME
RETURN
RETURN

Timer0

La scelta di utilizzare Timer0 è giustificata dall'utilizzo di una subroutine chiamata DELAY, che ci permette di avere un piccolo intervallo temporale per la durata di led e suono del buzzer.
Per far questo bisogna effettuare un conteggio di circa 50ms e quindi caricare nel registro TMR0 una costante specifica definita tramite la direttiva EQU.
Dalla teoria abbiamo studiato come è possibile, tramite una proporzione, scegliere il numero di tick da caricare per far conteggiare al timer un intervallo temporale a piacimento:

Ttick:1 tick=Tvoluto:x

Dove **Ttick = Ps/(Fosc/4) = 256/1MHz = 1μs**

Di conseguenza il periodo di un ciclo completo è pari a 2^8* Ttick = 256 μs

Dunque, la costante che deve essere caricata in TMR0 per permettergli di contare un intervallo di tempo pari a 50ms ed andare in overflow solo dopo aver contato 50 ms, ottenuto dalla proporzione:

256us:1=50ms:x

Avremo come valore di costante:

tmr_50ms EQU (.256-.195)

TIMER1

La scelta di Timer1 è giustificata per le tempistiche. A differenza di timer0 e timer2, timer1 ha 16 bit e dunque è in grado di contare fino 65536. Dovendo raggiungere 60 secondi di intervallo si è scelto di utilizzare timer1 di 1 secondo. Ragionando in proporzione dovremmo scrivere quindi:

Ttick:1 tick=Tvoluto:x

Avendo scelto come sorgente di Clock il quarzo esterno di frequenza 32.768 kHz e un prescaler impostato a 1:1 avremo che

Ttick = 30.512 ms

Quindi **Tciclo = 2^16*Ttick = 2s**

Quindi per ottenere il periodo da noi desiderato si usa la proporzione:

30.512:tick = 1s:x

Avremo come valore di costante:

tmr_1s EQU (.65536 - .32774)

Gestione Loop minuto

Il minuto viene gestito tramite un loop. Il loop consiste in un ciclo compiuto 60 volte (si utilizza un counter definito come variabile e caricata a 60) gestito all'interno dell'Interrupt. Il ragionamento dietro è molto semplice: ad ogni ciclo si va a chiamare Reload-timer, che permetterà di contare 1 secondo e si va a decrementare il contatore. Questo finché il counter non arriva a zero; in questo

modo solo al termine del minuto potranno essere compiute le operazioni necessarie al controllo della temperatura.

Blink LED

Una richiesta del programma è che venga implementato un lampeggio di un led prima della misurazione della temperatura.

Per far questo si va a caricare un counter che ci permetterà di avere una temporizzazione soddisfacente.

Si andrà a chiamare la funzione loop_blink che andrà a creare un ciclo, della durata del contatore, che andrà ad accendere e spegnere il led utilizzando la funzione delay, e quindi timer0, per rendere visibile ad occhio umano l'operazione.

Loop Alarm

Per quanto riguarda il suono dell'allarme anche qui si ha l'utilizzo di loop, gestito in egual modo all'operazione di blink led.

Si andrà quindi a caricare un counter e a chiamare la funzione loop_alarm:

Ogni ciclo il counter verrà decrementato e chiamata la funzione play_alarm che farà partire il suono. Una volta che il counter sarà arrivato a zero, verrà chiamata la funzione stop_alarm che andrà a spegnere il suono.

Il suono del buzzer viene gestito tramite PWM e timer2 dove si andrà a creare un'onda sinusoidale caratterizzata da un duty-cycle del 50% che ci dirà che il ton sarà pari alla metà del periodo da caricare in PR2.

Reload_counter

Un'operazione importante che deve essere effettuata prima della chiamata a irq_end è ricaricare il counter del minuto e il timer.

MODULO ADC

Nel mondo reale la maggior parte delle variabili sono di tipo analogico. Il microcontrollore, lavorando con segnali di tipo discreto, necessita di un sistema che gli permetta di acquisire e poter lavorare con i tipi di segnali del mondo esterno.

Il microcontrollore è dotato di un intermediario, che si occupa dell'acquisizione del segnale di tipo analogico e la sua conversione in segnale di tipo discreto.

Il modulo ADC infatti, nel nostro caso, viene sfruttato per l'acquisizione e la conversione del segnale relativo alla temperatura, il quale sarà poi trasformato da segnale di tensione a valore in gradi.

Andando ad utilizzare le formule presenti nel datasheet del sensore di temperatura, si nota come per poter passare da un segnale di tensione a un valore in gradi centigradi sarà necessario implementare la seguente formula:

$$T = (N_{adc} - 31) * \frac{2}{3}$$

Questa operazione, presente nella sub routine computeTemp è giustificata dalla necessità di confrontare il valore massimo di temperatura da noi impostato all'inizio del programma con quello acquisito dal sensore.

Funzionamento

Nel mio caso d'esame, viene tutto gestito, tranne le operazioni di misura della temperatura, nella ISR.

L'overflow del minuto viene gestito tramite TIMER1 caricato di 1 secondo, andato a effettuare un ciclo di 60 volte, chiedendo al termine di ogni ciclo se la variabile counter si è scaricata.

Dopo il minuto viene effettuata la chiamata a loop_blink

Infine, verrà eseguita la misurazione e conversione del valore della temperatura tramite il sensore posto sulla PIC
Tale valore verrà confrontato con una soglia massima scelta ad inizio programma come variabile.

Si potranno verificare due situazioni:

1) La temperatura attuale supera la temperatura massima scelta;
2) La temperatura attuale è minore della temperatura massima scelta.

Nel primo caso ci sarà la chiamata a loop_alarm per gestire il suono dell'allarme, mentre nel secondo non verrà effettuata alcuna azione.
Al termine delle due possibili situazioni, vi sarà la chiamata a reload_counter, ricaricando il counter minuto per riniziare da capo il ciclo e con la successiva uscita dall'interrupt con il relativo ripristino di contesto.