Versione N° 1.0

LaurTec

Orologio con display 7 segmenti

4 Digits Clock

Autore : *Ivo Colleoni*

ID: UP0012-IT

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.

AVVERTENZE

I progetti presentati non hanno la marcatura CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.

L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.

Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Indice

Introduzione

Il progetto nasce dalla richiesta di mio padre di avere una sveglia compatta e con visualizzazione del solo orario. La scelta progettuale ricade su un semplice circuito formato da un microcontrollore PIC18F2550 il quale pilota direttamente un display 7 segmenti a 4 cifre (HDSP-B09G) tramite tecnica multiplexing.

Nota dell'Autore

A mio padre dedico questo semplice progetto e lo ringrazio per tutto quello che mi ha insegnato e tramandato, compresa la grande passione per l'elettronica.

Ciao papà

Ivo Colleoni

Specifiche Tecniche

Alimentazione: 9VDC
Assorbimento: 50mA (valore tipico)
Dimensioni: 7x5cm

Orologio 24h
Secondi scanditi dai puntini lampeggianti
2 pulsanti per l'impostazione dell'ora
Display 7 segmenti a 4 cifre

Applicazioni

Oltre all'utilità di un orologio da tavolo/parete compatto e ben visibile a tutti gli orari del giorno e della notte, il progetto viene presentato come tutorial didattico. Tramite questo progetto vengono utilizzate le tecniche base di programmazione facendo uso del timer, interrupt, decodifica di valori tramite array, pulsanti.
Mantenendo lo stesso hardware e le tecniche base descritte in questo articolo si potrebbe creare anche un timer per bromografi o da cucina, oppure un contatore tramite pressione dei tasti o di passaggio tramite sensori di movimento.

La tecnica del multiplexing

La tecnica del multiplexing consiste nell'utilizzare una stessa linea dati per comandare contemporaneamente più dispositivi ad intervalli di tempo regolari.
In particolare nel progetto in questione si tratta di utilizzare più display a 7 segmenti (o un display unico a più cifre) condividendo le linee che comandano gli anodi dei LED dei display. I 4 catodi comuni, uno per ogni cifra, al posto di essere collegati direttamente a massa vengono collegati ognuno ad un collettore di un transistor pilotato da un'uscita del microcontrollore.
In questo modo per scrivere un valore su uno dei 4 display bisognerà inviare il dato sulla linea dati e mandare il transistor corrispondente al catodo del display desiderato in conduzione chiudendo così il circuito a massa e accendendo i segmenti selezionati. Effettuando questa operazione ciclicamente sui 4 display ed in modo rapido e continuo, grazie al fenomeno di persistenza dell'immagine sulla retina, i 4 display sembreranno tutti accesi contemporaneamente. Ovviamente il tutto può funzionare anche con display ad anodo comune invertendo la logica di funzionamento delle uscite del PIC che comandano i segmenti.

Analisi della tecnica multiplexing nel progetto

Dallo schema elettrico di Figura 1, si può notare il collegamento del display con gli anodi dei segmenti su RA0-RA6 e i 4 catodi comuni su RB4-RB7 (tramite un transistor).

All'interno del codice vanno prima di tutto definiti i collegamenti:

```
#define DISPLAY             LATA
#define CATHODE_1           LATBbits.LATB4
#define CATHODE_2           LATBbits.LATB5
#define CATHODE_3           LATBbits.LATB6
#define CATHODE_4           LATBbits.LATB7
```

Va poi creata una tabella di decodifica da decimale a segmenti da accendere per visualizzare il numero decimale. Per far questo basta creare un array contente i vari valori. Richiamando poi l'array con il numero decimale desiderato si selezionerà il byte di decodifica necessario per accendere i segmenti corretti.

```
unsigned char decode_table[10] = {
                                    0b00111111,    // 0
                                    0b00000110,    // 1
                                    0b01011011,    // 2
                                    0b01001111,    // 3
                                    0b01100110,    // 4
                                    0b01101101,    // 5
                                    0b01111101,    // 6
                                    0b00000111,    // 7
                                    0b01111111,    // 8
                                    0b01101111     // 9
};
```

Per far visualizzare quindi il numero desiderato sul display basterà mandare sulla PORTA il valore contenuto nell'array relativo al numero desiderato e abilitare il catodo del display desiderato.

```
DISPLAY = decode_table[3];
CATHODE_1 = ON;
```

In questo esempio si vedrà il numero 3 sulla primo display.

Per fare in modo che su ogni display sia visualizzato un numero diverso dando l'impressione che tutti i display siano accesi contemporaneamente va creata una routine di scrittura continua su tutti i display.
A livello Firmware questa routine è stata implementata utilizzando un interrupt che ad intervalli regolari e personalizzabili scrive ciclicamente i dati sui 4 display. Di seguito un esempio semplificato estratto dal codice sorgente. Come si può vedere dal codice sorgente in realtà lo stesso interrupt è usato per 2 funzioni e la scrittura di ore e minuti viene separata.

```
// interrupt causato da timer 2 (1ms)

if(PIR1bits.TMR2IF){
 static unsigned char _timer_multiplex = 0;
 unsigned char _value = 0;

   // incremento contatore ogni 1ms
   _timer_multiplex++;

   // ogni 10 ms scrivo tutte le cifre
   if(_timer_multiplex==10) {

      // decodifico il valore
      DISPLAY = decode_table[1];
      // abilito catodo primo display
      CATHODE_1 = ON;
      CATHODE_2 = OFF;
      CATHODE_3 = OFF;
      CATHODE_4 = OFF;

      // ritardo minimo di mantenimento
      Delay_us(2000);

      // decodifico il valore
      DISPLAY = decode_table[2];
      // abilito catodo secondo display
      CATHODE_1 = OFF;
      CATHODE_2 = ON;
      CATHODE_3 = OFF;
      CATHODE_4 = OFF;

      // ritardo minimo di mantenimento
      Delay_us(2000);

      // decodifico il valore
      DISPLAY = decode_table[3];
      CATHODE_1 = ON;
      // abilito catodo terzo display
      CATHODE_2 = OFF;
      CATHODE_3 = OFF;
      CATHODE_4 = OFF;

      // ritardo minimo di mantenimento
```

```
        Delay_us(2000);

        // decodifico il valore
        DISPLAY = decode_table[4];

        // abilito catodo quarto display
        CATHODE_1 = OFF;
        CATHODE_2 = OFF;
        CATHODE_3 = OFF;
        CATHODE_4 = ON;

        // ritardo minimo di mantenimento
        Delay_us(2000);

        // resetto contatore timer
        _timer_multiplex = 0;
}

        // resetto preload timer ed interrupt
        TMR2 = 131;
        PIR1bits.TMR2IF = 0;
}
```

Come si può notare dal codice il Timer 2 causa un interrupt ogni 1ms, ad ogni interrupt viene incrementata la variabile _timer_multiplex, solo quando questa raggiunge il valore stabilito vengono scritti tutti i valori sui display.
In questo esempio ogni 10ms verranno scritte tutte le cifre sui display e si avrà la percezione che tutti i display siano accesi mostrando "1234".
Variando il delay di mantenimento si può dare l'impressione di variare la luminosità delle cifre. All'interno del Firmware il tempo di refresh e di mantenimento sono definiti da costanti in modo da renderne semplice la modifica.

Il conteggio dei secondi tramite clock esterno ed interrupt

Il conteggio dei secondi avviene ad ogni interrupt causato dal Timer 1 del microcontrollore. Tale timer è configurato in modo da funzionare tramite clock esterno e causare un interrupt ogni secondo. Per generare il clock esterno si è fatto uso di un quarzo da 32,768KHz. Tale valore consente di avere un conteggio dei secondi abbastanza preciso anche se una certa tolleranza è sempre presente.

Per calcolare il preload da impostare nel timer per causare un interrupt ogni secondo basta utilizzare la formula:

$$Preload = 2^{Nbit} - \frac{T * Fosc}{Prescaler} = 2^{16} - \frac{1 * 32768}{1} = 32768$$

Dove T è il tempo richiesto per causare l'interrupt, Fosc è la frequenza del quarzo, Prescaler è il valore assegnato al prescaler e Nbit è il numero di bit del Timer.
Tuttavia utilizzando questo valore teorico si introduce un ritardo di 4s ogni 24h. A seguito di prove fatte il miglior valore di preload risulta essere 32770 (la teoria sui timer dice che una volta calcolato il preload del timer vanno aggiunti ulteriori 2 unità per eliminare il tempo delle 2 istruzioni di scrittura del preload all'interno del registro ed infatti la teoria è

confermata), con questo valore si introduce un anticipo di circa 900ms ogni 24h. Per correggere questo errore è stato inserito un delay di 10uS ogni secondo, tale valore è comunque personalizzabile in fase di compilazione tramite modifica della costante FINE_TUNE. In questo modo si ottiene un anticipo di 13 secondi nell'arco di 1 anno.

Dal momento che non tutti i quarzi sono uguali il valore da utilizzare per effettuare la correzione può variare da caso a caso. Il quarzo ha inoltre una variazione della frequenza al variare della temperatura che in in questo caso non è stata corretta, vista la semplicità dell'hardware utilizzato.

Il fatto di poter collegare il quarzo direttamente al microcontrollore deriva dal fatto che il Timer 1 del microcontrollore PIC18F2550 incorpora un circuito di oscillazione pensato proprio per essere utilizzato con i quarzi a 32KHz. A livello hardware va solo aggiunto il quarzo sui pin T1OSO e T1OSI e i relativi condensatori di carico collegati tra il quarzo e massa.

All'interno del codice bisogna abilitare il circuito di oscillazione e specificare che la sorgente del clock di questo timer è esterna ed indipendente dal clock principale. Di seguito il codice per queste impostazioni.

```
T1CONbits.T1CKPS0 =0;           // prescaler 1:1
T1CONbits.T1CKPS1 =0;

T1CONbits.RD16 = 1;             // lettura scrittura 16bit

T1CONbits.T1OSCEN = 1;          // oscillatore abilitato
T1CONbits.T1SYNC = 1;           // asincrono
T1CONbits.TMR1CS = 1;           // clock esterno

TMR1 = TMR1_PRELOAD;            // preload timer 1 (dichiarato in costante)

T1CONbits.TMR1ON = 1;           // timer attivo
```

Analisi del progetto

Il progetto, il cui schema elettrico è riportato in Figura 1, come già anticipato è relativamente semplice. Si possono notare il regolatore di tensione LM7805, il PIC18F2550 e il display a 4 cifre. Come si può vedere le linee di controllo per il display sono 12, 8 per gli anodi dei LED che formano le cifre più il punto e 4 che comandano 4 transistor BC547 che chiudono a massa i catodi. Completano l'hardware il quarzo a 32.768KHz, 2 pulsanti e la porta di programmazione. Sui 2 pulsanti non è necessario installare le resistenze di pull-up in quanto vengono utilizzate le resistenze di pull-up interne al PIC18F2550.
Come si vede dallo schema è presente solo il quarzo a 32.768KHz necessario al calcolo dei secondi. Per minimizzare i componenti esterni, come clock di sistema si è fatto uso dell'oscillatore interno al PIC18F2550.

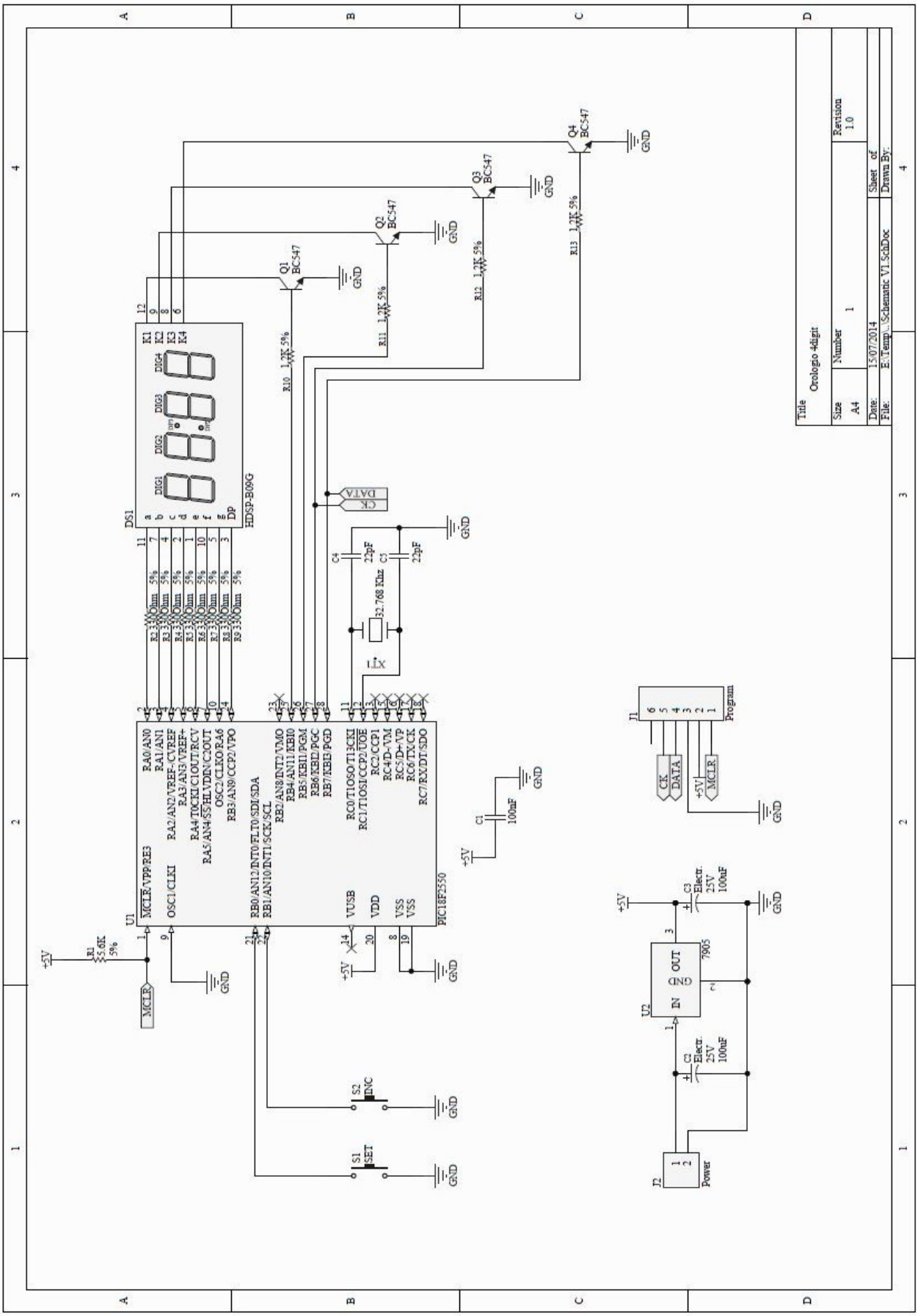


Figura 1: *Schema elettrico.*

Lista Componenti

Resistori

R1 = 5.6KΩ %5 SMD 805
R2 = 330Ω %5 SMD 805
R3 = 330Ω %5 SMD 805
R4 = 330Ω %5 SMD 805
R5 = 330Ω %5 SMD 805
R6 = 330Ω %5 SMD 805
R7 = 330Ω %5 SMD 805
R8 = 330Ω %5 SMD 805
R9 = 330Ω %5 SMD 805
R10 = 1,2KΩ %5 SMD 805
R11 = 1,2KΩ %5 SMD 805
R12 = 1,2KΩ %5 SMD 805
R13 = 1,2KΩ %5 SMD 805

Condensatori

C1 = 100nF ceramico 50V SMD 805
C2 = 100uF Elettrolitico 25V
C3 = 100uF Elettrolitico 25V
C4 = 22pF ceramico 50V SMD 805
C5 = 22pF ceramico 50V SMD 805

Transistor

Q1 = BC547 TO92
Q2 = BC547 TO92
Q3 = BC547 TO92
Q4 = BC547 TO92

Circuiti Integrati

U1 = PIC18F2550 SOIC28
U2 = LM7905 TO220

Quarzi

XT2 = 32,768KHz

Pulsanti

S1 = Strip line 2 pin per collegamento pulsante
S2 = Strip line 2 pin per collegamento pulsante

Connettori

J1 = Strip line 6 pin

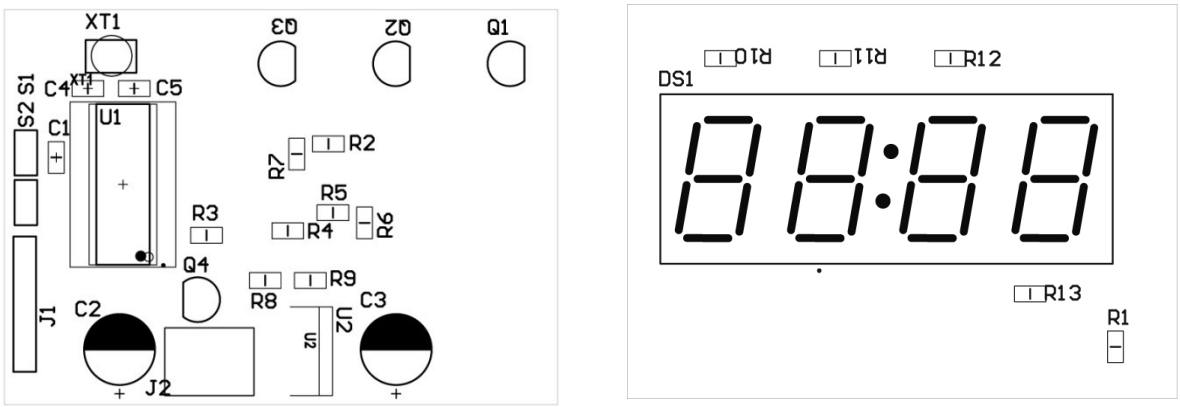
Nella Tabella 1 è riportata l'associazione tra i vari pin del microcontrollore e le varie funzioni del sistema.

Pin	Nome Pin	Direzione	Nome Linea	Funzione
1	MCLR	Input	RESET	Reset della scheda.
2	RA0	Output		Anodo segmento a dei display
3	RA1	Output		Anodo segmento b dei display
4	RA2	Output		Anodo segmento c dei display
5	RA3	Output		Anodo segmento d dei display
6	RA4	Output		Anodo segmento e dei display
7	RA5	Output		Anodo segmento f dei display
8	VSS		GND	GND
9	OSCI		GND	GND
10	RA6	Output		Anodo segmento g dei display
11	RC0	Input		Cristallo
12	RC1	Input		Cristallo
13	RC2	Input		Non Collegato
14	VUSB	NC		Non Collegato
15	RC4	Input		Non Collegato
16	RC5	Input		Non Collegato
17	RC6	Input		Non Collegato
18	RC7	Input		Non Collegato
19	VSS		GND	GND
20	VDD		+5V	+5V
21	RB0	Input		Switch Set
22	RB1	Input		Switch Incremento
23	RB2	Input		Non Collegato
24	RB3	Output		Anodo punto dei display
25	RB4	Output		Catodo display prima cifra
26	RB5	Output		Catodo display seconda cifra
27	RB6	Output		Catodo display terza cifra
28	RB7	Output		Catodo display quarta cifra

Tabella 1: *Tabella riassuntiva delle connessioni tra il* PIC18F2550 *e le periferiche di sistema. Per una visione completa delle funzionalità di ogni pin si rimanda al datasheet del PIC18F2550.*

Realizzazione del circuito

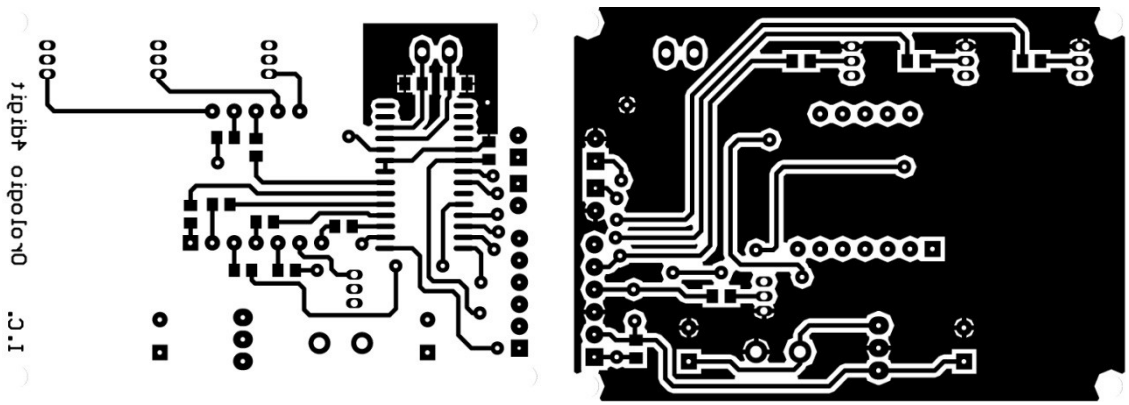
Il circuito è stato realizzato su una piastra doppia faccia con il metodo della fotoincisione e sono stati utilizzati principalmente componenti SMD per avere un circuito compatto e di semplice realizzazione. Lo schema di montaggio è riportato in Figura 2.



a) *Top* *b)* *Bottom*

Figura 2: *Schema di montaggio del PCB.*

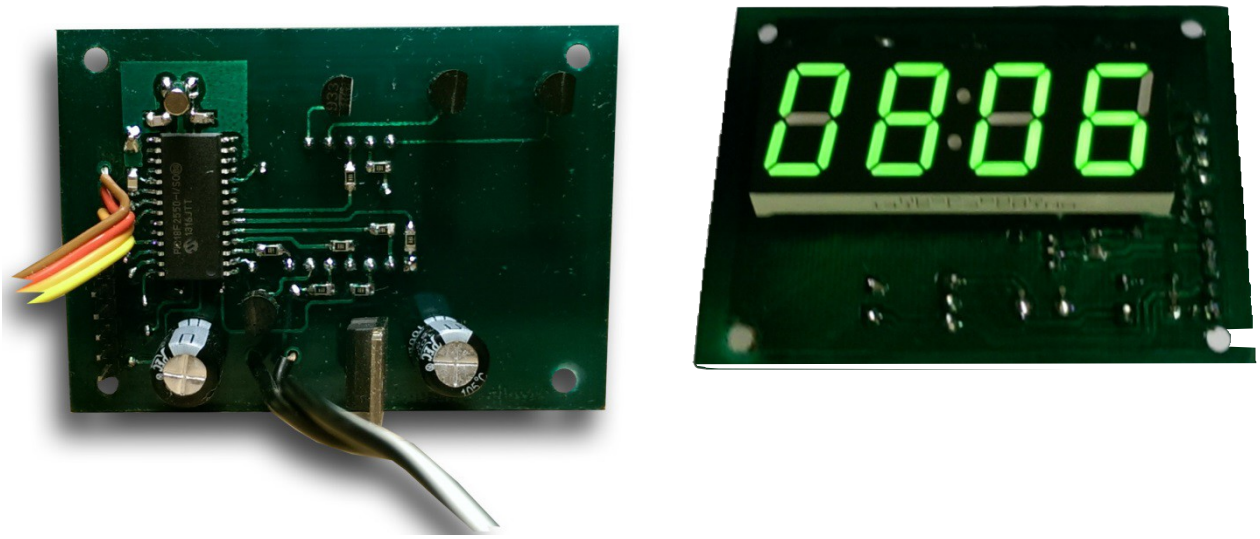
Il PCB, realizzato su doppia faccia è riportato in Figura 3.



a) *Top* *b)* *Bottom*

Figura 3: *PCB a doppia faccia del sistema.*

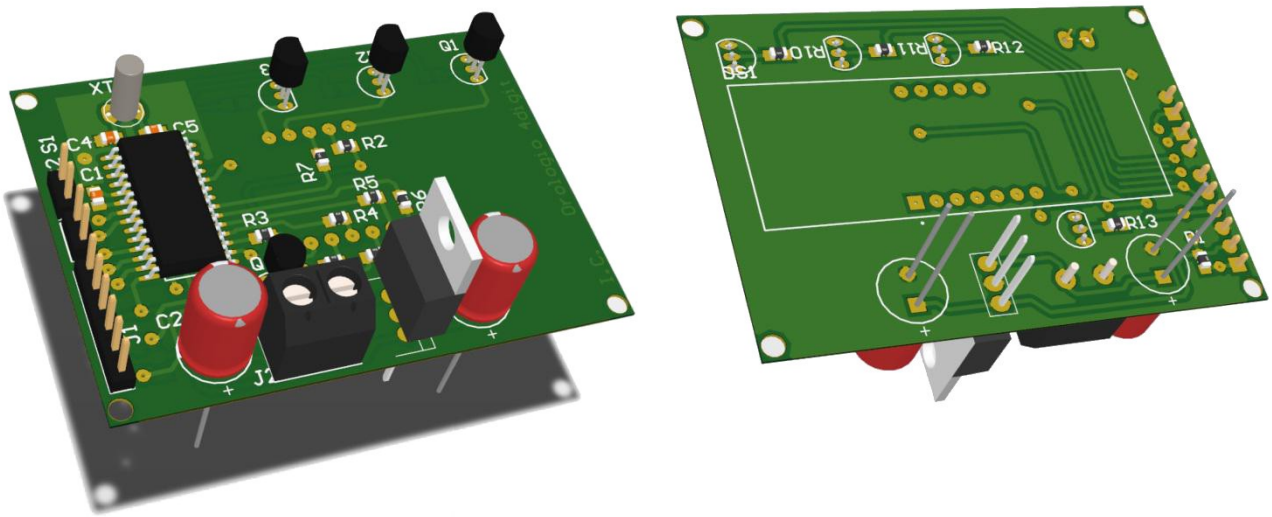
Il sistema assemblato è riportato in Figura 4.



a) Top *b) Bottom*

Figura 4: *Sistema a montaggio ultimato.*

Un Modello 3D del circuito è riportato in Figura 5.



a) Top *b) Bottom*

Figura 5: *Sistema a montaggio ultimato (modello 3D).*

Per realizzare i PCB è possibile scaricare i file per la stampa dal sito www.LaurTec.it alla pagina associata al progetto UP0012.

Collaudo e messa in funzione

Dal momento che il circuito fa uso di componenti a montaggio superficiale, prima di accendere il sistema è bene accertarsi che non sia presente alcun cortocircuito. Un semplice test potrebbe essere il controllo per mezzo di un tester, della presenza di cortocircuiti tra il terminale + e – sia prima che dopo il regolatore lineare.

Impostazione del timer

Se l'orologio dovesse essere poco preciso è possibile modificare il valore di preload del timer e una costante di regolazione di precisione a riga 33 e 34 del file Main.h

```
#define TMR1_PRELOAD  32770
#define FINE_TUNE     10
```

Dai test effettuati il miglior valore di preload risulta essere 32770, con questo valore si introduce un anticipo di circa 900ms ogni 24h. Per correggere questo errore è stato inserito un delay di 10uS ogni secondo, tale valore è comunque personalizzabile in fase di compilazione tramite la modifica della costante FINE_TUNE. In questo modo si ottiene un anticipo di 13 secondi nell'arco di 1 anno.

Utilizzo dell'orologio

Quando si alimenta l'orologio per la prima volta, i Display espongono 00:00 lampeggiante per richiamare l'attenzione ad impostare l'ora.

Impostazione dell'ora

Premere il pulsante SET. Il valore dell'ora lampeggia, premere il pulsante INC per incrementarne il valore oppure tenere premuto per incrementare velocemente il valore (velocità di incremento in 2 step: i primi 5 valori lentamente e poi più velocemente). Premere nuovamente il pulsante SET per passare ad impostare i minuti nella stessa modalità. Premere SET per chiudere il ciclo di impostazione.

Se fosse necessario reimpostare l'ora premere il pulsante SET per entrare nel ciclo di impostazione dell'orario.

Il Firmware

Il Firmware è stato scritto in C (compilatore XC8 1.34 versione free) ed il progetto allegato è stato creato con MPLAB X. Il progetto utilizza solamente 2 librerie esterne: una per la gestione dei pulsanti e una per la gestione dei delay, quest'ultima fa parte della suite di librerie di Mauro Laurenti (con qualche modifica per adattarla allo stile di scrittura del progetto).

All'avvio vengono inizializzati i moduli del microcontrollore per settare ingressi e uscite, i timer e gli interrupt necessari; viene poi richiamata la funzione di impostazione dell'orario. Una volta impostato l'orario per la prima volta, il ciclo principale non fa altro che controllare se viene premuto il pulsante set, per andare a reimpostare l'ora.
Il resto del programma è stato implementato nei cicli di interrupt.

- Il primo interrupt, impostato a livello di priorità alto, viene causato da un overflow del Timer 1, come già detto è impostato in modo da causare un interrupt ogni secondo. Nel ciclo di interrupt viene invertito lo stato dei due punti e vengono incrementate le variabili dell'ora codificate in Packed BCD.
- Il secondo interrupt, impostato a livello di priorità basso (cioè non può a sua volta interrompere un interrupt con livello di priorità alto) viene causato da un overflow del Timer 2; questo è impostato in modo da avere 1 interrupt ogni 1ms. Il ciclo di gestione di questo interrupt è diviso in 2 parti: la prima viene eseguita dopo N interrupt del Timer 2, dove N corrisponde al valore della costante DISPLAY_MUX, mentre la seconda viene eseguita dopo N interrupt del Timer 2, dove N corrisponde al valore della costante BLINK_DELAY. In questo modo i parametri sono facilmente personalizzabili modificando i valori delle costanti in fase di compilazione e viene utilizzato solo un timer per due scopi.

La prima parte dell'interrupt si occupa di gestire il multiplex dei display, in poche parole ogni 10ms (DISPLAY_MUX di default vale 10) vengono scritti i valori sui display. Come si nota dal codice si possono scrivere solamente le ore o solamente i minuti o entrambi i valori, questo perché in fase di impostazione le ore e i minuti lampeggiano separatamente.
Le seconda parte dell'interrupt inverte il flag di lampeggio dopo 300ms (BLINK_DELAY di default vale 300ms).

Per ulteriori dettagli sul codice si rimanda alla lettura dei sorgenti allegati.

Sorgenti

Main.c:	Corpo principale del programma
Main.h:	Configurazioni generali
Prototipi.h	Prototipi di funzioni

Librerie allegate

Delay.c:	Cicli di ritardo
Manage_input.c	Gestione pulsanti
Manage_input_settings.h	Configurazioni per debounce e ritardi

All'interno del file Main.h si possono personalizzare alcuni parametri:

BLINK_DELAY
Valore in ms rappresentante il tempo di lampeggio (ON e OFF).

PERSIST
Valore in us rappresentante il tempo in cui i dati vengono mantenuti su ogni display, agire su questa costante per modificare la luminosità dei display.

DISPLAY MUX
Valore in ms rappresentante il tempo di aggiornamento dei display.

TMR1_PRELOAD
Valore di preload del Timer1.

FINE_TUNE
Valore in us rappresentante il ritardo da aggiungere ad ogni secondo per una maggiore precisione.

Analisi finale

La scelta di componenti SMD ha permesso la realizzazione di un circuito molto compatto grazie anche al montaggio dei display sul lato inferiore del PCB.
Tutto sommato la sveglia è piuttosto precisa e si presta decisamente bene al lavoro che deve svolgere.

Come già anticipato è possibile modificare il Firmware trasformando la sveglia in un timer. Per far questo basta sottrarre un'unità ogni interrupt del timer 1 e ovviamente anche il ciclo di setup andrebbe rivisto per impostare minuti e secondi piuttosto che ore e minuti di partenza. Con una semplice modifica hardware sarebbe anche possibile aggiungere un relè comandato da un'uscita del microcontrollore, al fine di attivare o disattivare un circuito esterno allo scadere del timer (per esempio un circuito di lampade UV per un bromografo o un buzzer per un allarme).

Un'altra possibile modifica, eliminando la parte di calcolo dei secondi necessaria per una sveglia o un timer, potrebbe essere quella di creare un contatore da incrementare ad ogni pressione di uno dei tasti, oppure sostituendo i tasti con 2 sensori di movimento contare il numero di ingressi/uscite in un determinato ambiente a seconda della sequenza con la quale vengono attivati i sensori.

Allegati

Al fine di permettere la realizzazione del progetto da parte di ogni utente, è possibile scaricare dal sito www.LaurTec.it i file relativi alla realizzazione dei PCB e il relativo Firmware associati al progetto UP0012.

- PDF per permettere la fotoincisione delle schede.
- Progetto MPLAB X e relativi sorgenti.

Indice Alfabetico

Bibliografia

[1] www.LaurTec.it: sito ufficiale del progetto dove pter scaricare tutti i file per la realizzazione dello stesso e relativi aggiornamenti.

[2] www.microchip.com : sito dove scaricare i datasheet del PIC18F2550.

History

Data	Versione	Autore	Revisione	Descrizione Cambiamento
21.06.16	1.0	Ivo Colleoni	Mauro Laurenti	Versione Originale.