

Elettronica In

Elettronica **I**nnovativa

€ 5,00

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

126

**Tastierino PS/2
programmabile
per PC**



**Cartellino orario
con fingerprint**



**Piattaforma
di sviluppo
per PIC**



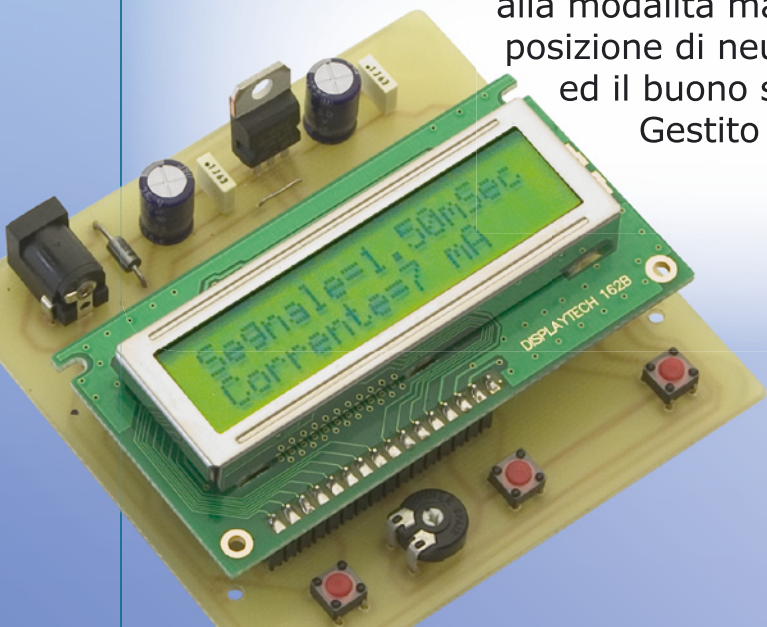
- Tester digitale per servocontrolli
- Datalogger USB firmware e software
- Progettare lo stadio di alimentazione

+++ ESCLUSIVO +++
CORSO EM1000
Programmazione
modulo Tlbbio

Tester digitale per servocontrolli

dell'Ing. Mirco Segatello

Permette di verificare l'efficienza dei servocontrolli analogici per modellismo. Grazie ai due programmi di test automatico e alla modalità manuale, consente di identificare con facilità la posizione di neutro nonché controllare la velocità di risposta ed il buono stato del gruppo ingranaggi del servo in test. Gestito tramite tre pulsanti, visualizza i parametri di funzionamento su un display alfanumerico.



Quello del modellismo radiocomandato è un hobby entusiasmante che coinvolge persone di tutte le età e di tutti i livelli di capacità: i giovanissimi, il più delle volte, all'inizio si accontentano di modelli più semplici sia da realizzare che da pilotare (tipicamente automobiline da corsa o fuori strada), così da potersi impratichire nella guida in tempo breve.

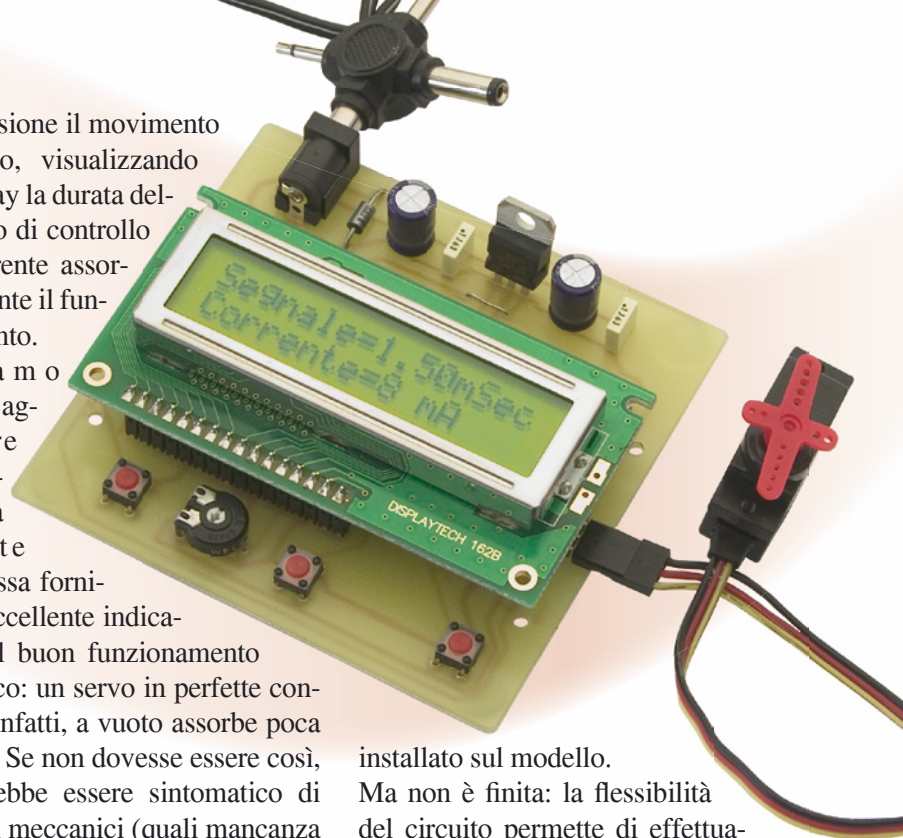
I professionisti e gli "anziani" del settore, invece, spesso dedicano dieci e più anni della propria vita per realizzare modelli radiocomandati più complessi (tipicamente navi, aerei, sommergibili ed elicotteri), talvolta così precisi e

raffinati da meritare la medaglia d'oro nei campionati nazionali ed internazionali. In mezzo a questi due estremi troviamo tutto il resto degli appassionati di modellismo RC, più o meno bravi o impazienti; nonostante le grandi differenze in termini di modelli, dettagli del modello e capacità del modellista, esiste, però, una particolare situazione che accomuna tutti gli appassionati. Sia il giovanissimo, che ha fatto sacrifici per racimolare (risparmiando la paghetta settimanale) il necessario per acquistare la prima macchinina RC, sia colui che ha impiegato lunghi anni per realizzare un modello fuori dal comune, sono >

terrorizzati dall'idea di perdere o danneggiare il proprio modello per colpa di un'anomalia ad uno dei dispositivi di controllo: stiamo parlando dei servomotori, delle riceventi e delle trasmettenti, tutti dispositivi che, sfortunatamente, talvolta si guastano; il guaio è che si guastano solamente se sono accesi, ossia mentre si sta facendo muovere il modello... In questi casi, quanto possa accadere, soprattutto con aerei, elicotteri, motoscafi o automobili in movimento e fuori controllo, beh, lo potete immaginare.

Tra i vari dispositivi, certamente, il servocontrollo è tra i più delicati in quanto comprende sia parti elettroniche che parti meccaniche, queste ultime tanto più delicate e soggette ad usura quanto più piccolo è il servo: bene, in queste pagine presentiamo un circuito molto utile, con il quale è possibile testare un gran numero di servocontrolli analogici da modellismo. In questa versione, il test completo è affidato ad un microcontrollore, gestibile tramite un display LCD e tre pulsanti: si tratta di un circuito più completo di quello realizzato con un semplice NE555, presentato sul fascicolo 116 nell'articolo dedicato al braccio robot; questa versione, infatti, permette di controllare con estre-

ma precisione il movimento del servo, visualizzando sul display la durata dell'impulso di controllo e la corrente assorbita durante il funzionamento. A b b i a m o voluto aggiungere la misura della corrente perché essa fornisce un'eccellente indicazione sul buon funzionamento meccanico: un servo in perfette condizioni, infatti, a vuoto assorbe poca corrente. Se non dovesse essere così, ciò potrebbe essere sintomatico di problemi meccanici (quali mancanza di lubrificazione, grippaggi da polvere, ingranaggi con denti rotti, eccetera) che obbligano il motore a rimanere sempre "in tiro"; il servo, pertanto, andrebbe sostituito. Relativamente al nostro dispositivo, vedremo come con tre tasti ed alcune semplici impostazioni sarà possibile effettuare il controllo sia manuale che automatico del movimento del servo: quest'ultima funzione è particolarmente utile per verificare il corretto funzionamento anche dopo che il servo è stato



installato sul modello.

Ma non è finita: la flessibilità del circuito permette di effettuare la verifica anche dei regolatori di velocità (Electronic Speed Control) usati in modellismo, senza la necessità della connessione alla ricevente radio. Alcune prove "al banco", poi, vengono fatte sui motori elettrici da gara, durante il rodaggio e il test di trazione: in questi casi un circuito che comanda con precisione il regolatore di velocità è utilissimo. Seguiteci nelle prossime pagine per scoprire come è stato realizzato questo versatile tester per servocontrolli.

- **Visualizzazione:** tramite display alfanumerico "2 righe x 16 caratteri".
- **Impostazione:** tramite i tre pulsanti "P+", "P-" e "Enter".
- **Modalità di test:** manuale e automatico.
- **Test automatici:** Slow e Fast.
- **Parametri modificabili:** valore minimo e massimo degli impulsi di controllo.
- **Verifiche possibili sul servocontrollo:**
 - verifica dell'estensione della rotazione;
 - identificazione della posizione di neutro;
 - misura della corrente assorbita a riposo (max. 1 A);
 - misura della corrente assorbita in movimento, sia a vuoto che a carico (max. 1 A);
 - verifica dello stato degli ingranaggi tramite il test "Slow";
 - verifica della velocità di risposta tramite il test "Fast".
- **Alimentazione:** 9 ÷ 12 Vdc, 100 mA minimo, 1 A massimo. La corrente massima è circa uguale a quella assorbita dal servocontrollo.

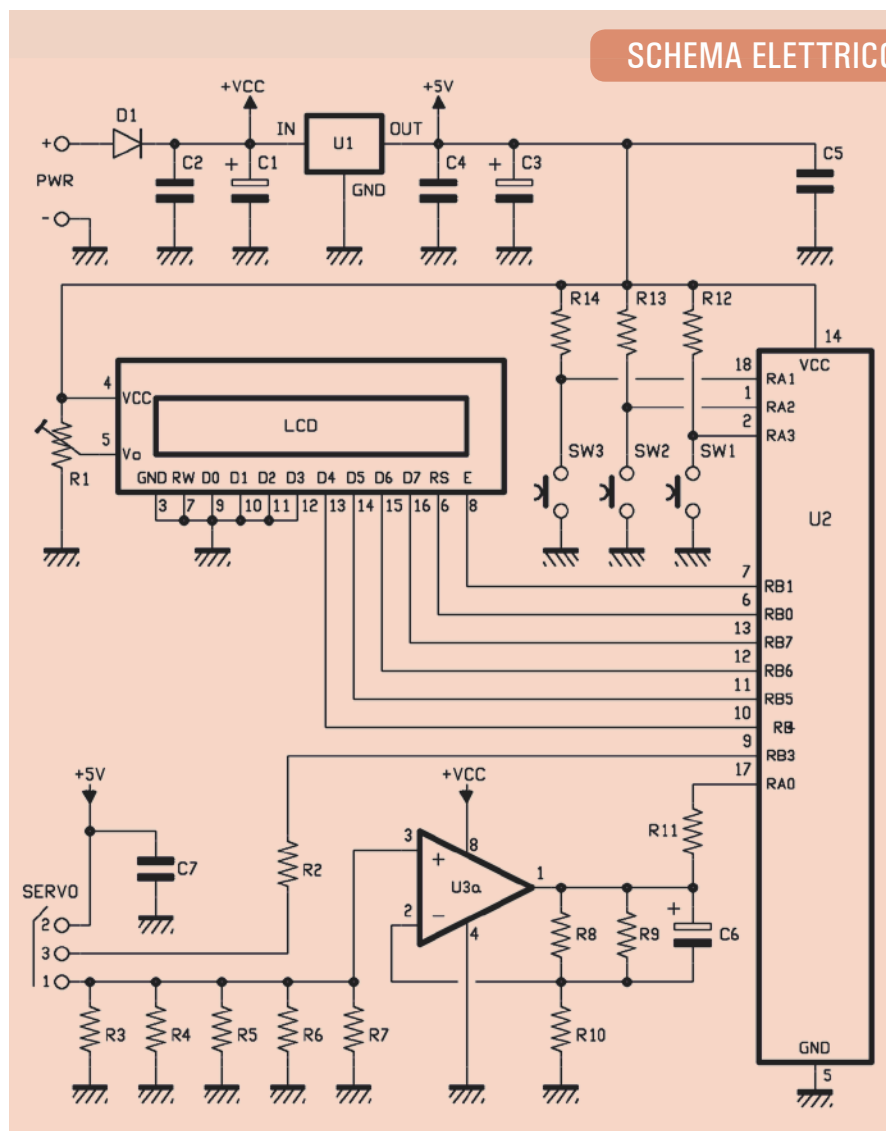
Schema elettrico

Come potete notare, il circuito è molto semplice ed è costituito essenzialmente da un microcontrollore della Microchip modello PIC16F818 (U2) e da un operazionale (U3a). Il PIC, dal costo modesto, è completo di stadio oscillatore interno e di svariati ingressi analogici, e queste sono esattamente le funzioni che servono a noi. Dispone anche di un'uscita PWM, ma, sfortunatamente, non può essere usata in quanto di risoluzione non adeguata. Il generatore PWM interno, infatti, ha una risoluzione massima di 10 bit, che permettono di avere un duty-cycle (ciclo utile) dell'onda quadra in uscita variabile tra 1023 va-

lori. Per gestire correttamente un servo, il segnale di controllo deve essere un impulso a onda quadra di durata compresa tra 1 e 2 ms, che si ripete ogni 20 msec: a seconda della durata dell'impulso, il servo assumerà una determinata posizione tra $\pm 60^\circ$ (nota: l'angolo di movimento dipende dal tipo di servocontrollo). Disponendo di 1023 step da $19,5 \mu s$ ($20 \text{ msec} / 1023 = 19,5 \mu s$ circa) per gestire gli impulsi da 1 a 2 msec utilizzeremo solamente 51 step ($51 \times 19,5 \mu s = 1 \text{ msec}$). Programmando, invece, il PIC per generare un impulso di durata variabile tramite la funzione PULSOUT del compilatore PicBasic, si ottiene una precisione doppia in quanto questa istruzione ha una risoluzione di $10 \mu s$, corrispondenti a cento step ogni millisecondo.

Torniamo ora al circuito, progettato per essere alimentato con un economico adattatore di rete non stabilizzato da 9-12 volt da 10-20 W. Lo stadio di alimentazione è formato da un diodo di protezione contro l'inversione di polarità (D1) e da uno stabilizzatore di tensione 7805 (U1) per l'alimentazione del circuito e del servo a 5 volt. Come potete notare dallo schema, sono previste due alimentazioni denominate +5V, per il micro e il display, e +Vcc per l'alimentazione dell'operazionale: più avanti vi spiegheremo il perché di questa seconda tensione di alimentazione.

Il PIC gestisce direttamente sia il display LCD (il cui contrasto viene regolato tramite il trimmer R1) che i tre pulsanti (collegati alle linee RA1, RA2 e RA3) necessari per l'utilizzo mentre la lettura della corrente assorbita dal servo avviene tramite un amplificatore operazionale ed una resistenza di shunt realizzata col parallelo di cinque resistenze da 1 ohm. Con il termine "shunt" viene indicata una resistenza di piccolo valore posta in serie all'elemento del quale si vuole misurare la corrente. La caduta di tensione su questa resistenza, proporzionale alla corrente assorbi-



ta, deve essere di valore modesto rispetto alla tensione di alimentazione totale del circuito (ad esempio 1/20). Il perché siano state utilizzate cinque resistenze in parallelo è presto detto: occorre una resistenza di valore piccolo, in grado di sopportare una corrente massima di 1 ampere, di buona precisione così da rendere il circuito di misura sufficientemente preciso senza la necessità di regolazioni aggiuntive. Le resistenze di potenza, economiche e facilmente reperibili, hanno una tolleranza del 10%, non adeguata per la nostra applicazione. Utilizzando cinque resistenze da $1/4 \text{ W}$ al 5% poste in parallelo, invece, si ottengono una precisione e una potenza cinque volte superiori (teoricamente 1% e 1,25 W). La caduta

di tensione sullo shunt (poche centinaia di millivolt sempre positivi, riferiti a massa) è, però, troppo piccola per essere letta direttamente dal PIC, quindi va amplificata. Per fare ciò, utilizziamo l'amplificatore operazionale U3a, un LM358 in grado di funzionare a tensione singola riferita a massa e di amplificare segnali che partono da zero volt (sempre riferiti a massa). In questo modo, dimensionando opportunamente le resistenze di retroazione, è possibile prelevare dall'uscita di U3a una tensione compresa tra 0 e 5 V, idonea per essere applicata all'ingresso analogico del microcontrollore.

A questo punto, però, occorre tenere presente un particolare molto importante: la tensione minima prelevabile ➤

in uscita all'LM358 è di zero volt, ma quella massima è circa due volt inferiore a quella di alimentazione; ciò significa che, per garantire in uscita un'escursione di 5 V, l'operazionale deve essere alimentato con una tensione leggermente superiore (almeno $2 \div 3$ V). Ecco spiegato il perché della seconda alimentazione (+Vcc): essa serve per alimentare l'operazionale e viene prelevata prima del regolatore di tensione, dove sono presenti alcuni volt in più. Il condensatore C1 filtra il ripple residuo proveniente dall'adattatore di rete non stabilizzato, in modo da alimentare l'operazionale con una tensione più pulita. Data la tensione di alimentazione superiore a 5 V ($8 \div 12$ V), teoricamente l'uscita dell'operazionale potrebbe andare oltre i 5 V: in questo caso, cosa succederebbe all'ingresso del microcontrollore? Verrebbe bruciato? Per rispondere a queste domande, vi facciamo notare la presenza della resistenza R11 (da 1 kohm): in condizioni normali, la corrente di ingresso di un canale analogico del PIC è di circa 500 nA e quindi la caduta di tensione su R11 è irrilevante tanto da poter affermare che tutta la tensione in uscita dall'operazionale giunge all'ingresso del PIC. Nel caso l'operazionale dovesse fornire in uscita una tensione troppo alta, entrerebbero in conduzione i diodi di protezione interni al PIC e la resistenza R11 limiterebbe la corrente su questi diodi, garantendo una protezione efficace dell'ingresso. In

alternativa si sarebbe potuto usare un amplificatore operazionale con uscita Rail-to-Rail (dall'inglese da rotaia a rotaia) che garantisce una tensione in uscita compresa tra il potenziale di massa e la tensione di alimentazione (ad esempio, se alimentato a 5 volt, la tensione in uscita può variare tra 0 e 5 volt), ma questo tipo di operazioni è più difficile da reperire nei negozi. A questo punto non ci resta che calcolare il valore dell'amplificazione per l'operazionale e nel fare questo consideriamo alcuni elementi: la risoluzione del convertitore interno del PIC è di 10 bit, quindi la conversione fornisce un numero compreso tra 0 e 1023. Il PIC converte in digitale i segnali compresi tra 0 e 5 volt, ovvero a zero volt fornisce il valore digitale 0 e a 5 volt fornisce il valore digitale 1023. Per evitare di dover fare calcoli complicati all'interno del micro, usiamo una scala molto diretta, ovvero:

- 0 mA corrispondono al valore decimale 0,
- 1.023 mA corrispondono al valore decimale 1023.

A questo punto i calcoli sono semplici, dato che l'uscita dell'operazionale deve valere 5 volt quando scorrono 1,023 A nello shunt. Sapendo che la resistenza di shunt è di 0,2 ohm, la caduta di tensione massima ai suoi capi equivale a

$0,2 \text{ ohm} \times 1,023 \text{ A} = 0,2046 \text{ V}$
per cui l'amplificazione dell'operazionale deve essere

$$5 \text{ V} / 0,2046 \text{ V} = 24,45.$$

L'amplificatore viene usato in configurazione non invertente ed il suo guadagno è dato dalla formula

$$A_v = [(R8 // R9) / R10] + 1$$

e quindi il rapporto

$$(R8 // R9) / R10$$

deve valere 23,45.

Abbiamo dovuto utilizzare un parallelo di due resistenze (R8 e R9) perché in commercio non sono disponibili due resistenze il cui rapporto dia come risultato 23,45. Con i valori scelti, il parallelo di R8 (2,7 kohm) e R9 (18 kohm) dà come risultato 2.348 ohm che, in rapporto a R10 (100 ohm), determinano il valore di



Futaba 14MZ: con i suoi 14 canali programmabili, i comandi vocali, il display LCD a colori Touch Screen, i programmi Aereo - Elicottero - Aliante precaricati, il doppio processore e la trasmissione a 2,4 GHz, è uno dei più evoluti radiocomandi per elimodellismo e aeromodellismo oggi in vendita.



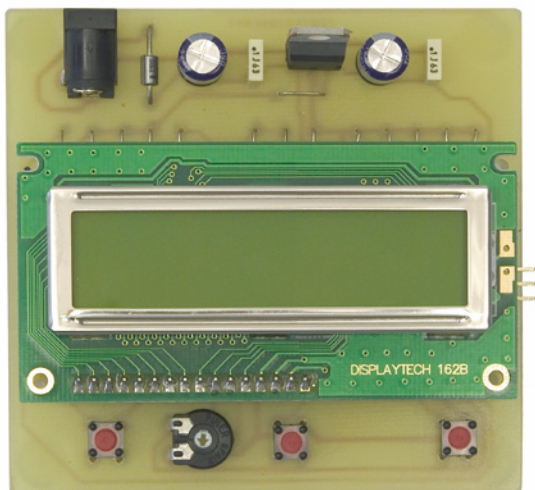
Splendido modello in scala 1:100 della Fregata ESPERO F576 (MMI). Realizzato da E. Effalli in otto anni e medaglia d'oro ai campionati Italiani, è navigante e dispone di ben 14 servocontrolli che movimentano, oltre a motori e timone, l'elicottero sul ponte di volo, le ancore, i cannoni e il radar.

23,48 ed il conseguente guadagno di 24,48, poco lontano da quello richiesto di 24,45.

Il condensatore C6, infine, crea un filtro passa basso utile per attenuare i segnali di disturbo generati da picchi di corrente del servo, sempre presenti durante il normale funzionamento.

Realizzazione pratica

Per prima cosa, occorre realizzare il circuito stampato monofaccia: a



Durante il montaggio, non dimenticate di inserire i quattro ponticelli a filo nelle vicinanze del microcontrollore, dell'operazionale e dello stabilizzatore di tensione. Il display può essere fissato meccanicamente con due torrette esagonali da 3MA, poste su due vertici diagonalmente opposti.

ELENCO COMPONENTI:

R1: Trimmer MO 10 kohm

R2, R10: 100 ohm

R3÷R7: 1 ohm

R8: 2,7 kohm

R9: 18 kohm

R11: 1 kohm

R12÷R14: 10 kohm

C1: 220 µF 25 VL elettrolitico

C2, C4, C5: 100 nF multistrato

C3: 100 µF 25 VL elettrolitico

C6: 10 µF 25 VL elettrolitico

C7: 100 nF multistrato

U1: 7805

U2: PIC16F818 (MF741)

U3: LM358

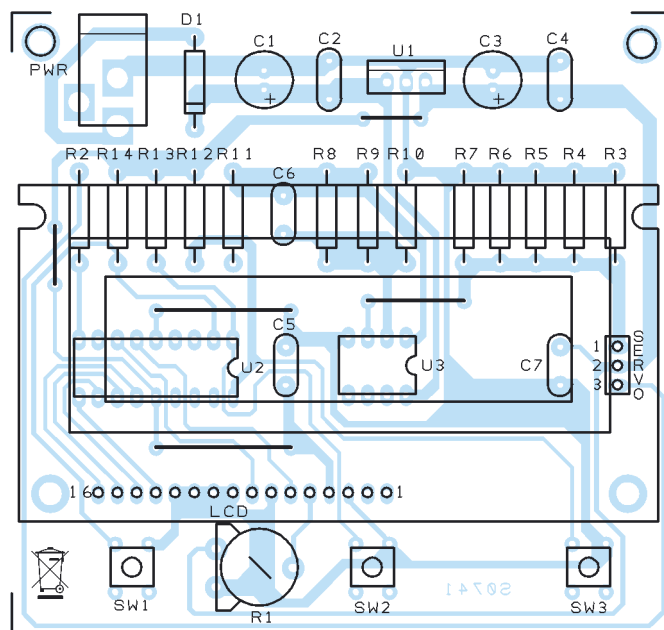
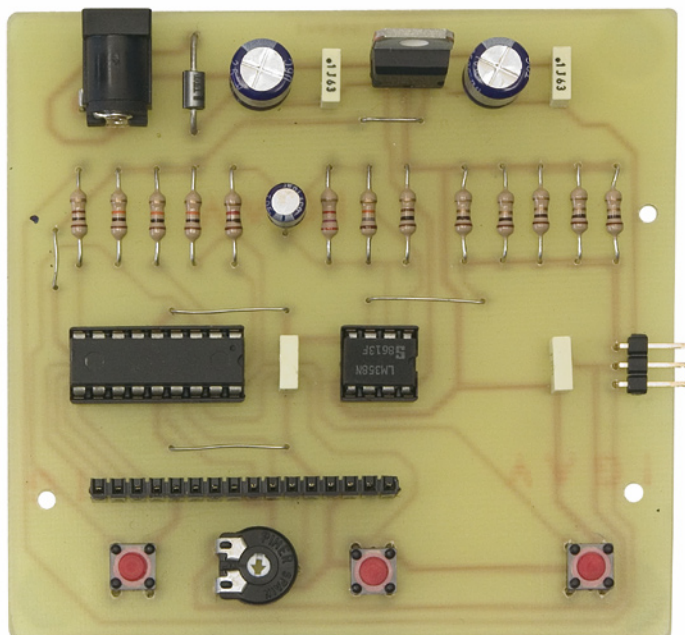
D1: 1N4007

LCD: Display LCD 16x2

SW1÷SW3: Microswitch

Varie:

- Plug alimentazione
- Zoccolo 4+4
- Zoccolo 9+9
- Strip maschio 3 poli
- Strip maschio 16 poli
- Strip femmina 16 poli
- Circuito stampato S0741



tal proposito potete, come sempre, scaricare gratuitamente il master in formato 1:1 dal nostro sito (www.elettronica.in.it).

Iniziate il montaggio saldando i quattro ponticelli a filo (aiutatevi con le illustrazioni del piano di montaggio) quindi inserite tutti i componenti a basso profilo (resistenze, diodi, condensatori non polarizzati, il trimmer e i tre pulsanti). Saldare, successivamente, i condensatori elettrolitici ed il plug per l'alimentazione, facendo

attenzione al verso dei componenti polarizzati (diodi e condensatori elettrolitici). Terminate il montaggio inserendo i due zoccoli DIL, lo stabilizzatore di tensione U1 (montato verticalmente e con il lato metallico rivolto all'esterno del circuito) e le due pin strip (una da 16 pin femmina, per il display e una da 3 pin maschio, per il servocontrollo) a passo 2,54 mm nelle apposite sedi. Preparate, infine, il display saldando ad esso una pin strip maschio a passo

2,54 mm, sul lato opposto al vetrino alfanumerico.

Controllate attentamente che il montaggio non presenti errori, quindi, senza inserire ancora i chip e il display, alimentate il circuito fornendogli una tensione continua di 9÷12 V: verificate che sul pin 8 dell'operazionale siano presenti circa 9÷12 V (a meno della caduta di tensione del diodo D1) e che sui pin 14 del micro e 4 del display siano presenti 5 V. Spegnete il circuito, attendete qual-

Listato 1

Riportiamo di seguito il codice sorgente completo, commentato in ogni linea. Vi ricordiamo che il firmware permette di eseguire due test, uno lento e uno veloce, durante i quali il servocontrollo viene fatto muovere tra due posizioni (minima e massima, definibili dall'utente). In entrambi i casi, durante il test vengono mostrate sia la durata dell'impulso di controllo (in tempo reale) che la corrente media assorbita durante un ciclo di posizionamento. Quest'ultima viene mostrata ad ogni cambio di direzione del movimento.

```

*****
* Name       : TESTDIGSERVI.BAS
* Author     : Mirco Segatello
* Version    : PIC 16F818 V1.0
*****

@ DEVICE INTRC_OSC
@ DEVICE BOD_OFF
@ DEVICE LVP_OFF
@ DEVICE CPD_OFF
@ DEVICE WDT_OFF
@ DEVICE PWRT_OFF
@ DEVICE MCLR_OFF
*****

Definizioni
*****
DEFINE LCD_DREG PORTB ' I/O porta dati.
DEFINE LCD_DBIT 4      ' Bit iniziale.
DEFINE LCD_RSREG PORTB ' Linea RS PORTB
DEFINE LCD_RSBIT 0     ' Bit 0.
DEFINE LCD_EREG PORTB  ' Linea E PORTB
DEFINE LCD_EBIT 1      ' Bit 1.
DEFINE LCD_BITS 4      ' 4-bit data bus.
DEFINE LCD_LINES 2     ' LCD con linee.
DEFINE LCD_COMMANDS 2000 ' Ritardo comando.
DEFINE LCD_DATAUS 50   ' Ritardo dato.
DEFINE ADC_BITS 10     ' Risoluzione ADC 10 bit.
DEFINE ADC_CLOCK 3     ' Sorgente clock
                        ' (OSC/32= 2, rc = 3).
DEFINE ADC_SAMPLEUS 500 ' Velocità di conversione.
DEFINE OSC 4           ' Specifica clock a 4 MHz.

*****
Imposta registri interni e variabili
*****
OSCCON = $60 ' Imposta clock interno a 4MHz.
ADCON0 = %11000001 ' Abilita ADC.
ADCON1 = %10001110 ' In. Anal. giustificato a SX.
TRISA = %11111111 ' RA0-RA5 in.
TRISB = %00000000 ' RB0-RB7 out.
ISV VAR byte ' Corrente sul servo.
PWO VAR byte ' Durata impulso PWM.
int VAR byte ' Parte intera.
fra VAR byte ' Parte frazionaria.
Delay VAR byte ' Vel. ripetizione pulsante.
Speed VAR byte ' Velocità movimento servo.
TV VAR byte ' Velocità refresh display.
PWUP VAR byte ' Limite massimo movimento.
PWDW VAR byte ' Limite minimo movimento.
BO VAR byte ' Variabile ad uso interno.
ISVM VAR word ' Corrente massima sul servo.
i VAR byte ' Variabile di servizio.
TMP VAR byte ' Variabile di servizio.
symbol PEN = PORTA.1 ' Pulsante enter.
symbol PUP = PORTA.2 ' Pulsante incrementa.
symbol PDW = PORTA.3 ' Pulsante decrementa.
symbol SVR = PORTB.3 ' Uscita segnale servo.
PORTB = 0 ' Azzerà PORTB.

PWO = 150 ' Imposta PWO a 150.
PWUP = 200 ' Imposta PWUP a 200.
PWDW = 100 ' Imposta PWDW a 100.
ISV = 0 ' Azzerà
Delay = 0 ' variabili
TV = 0 ' di servizio
Speed = 0 ' interne.

*****
Intestazione Display
*****
pause 200 ' Power-UP.
lcdout $FE,1 ' Cancella display.
lcdout $FE,2 ' Corsore ad inizio.
lcdout "TDS V1" ' Scrive messaggio.
pause 2000 ' Pausa 2 secondi.
call Lcd_New ' Aggiornamento display.
*****
Programma principale
*****
Main
TV=0 ' Avvio loop principale.
if PEN=0 then Main ' Azzerà refresh display.
Ripeti
if Delay<5 then Delay=Delay+1 ' Prosegue se ENTER premuto.
TV=TV+1 ' Incrementa Delay.
if TV=5 then ' Incrementa contatore refresh.
TV=0 ' Se refresh = 5
call Lcd_New ' azzera contatore
endif ' e aggiorna il display.
if PUP=0 and Delay=5 then ' Se UP premuto e delay=5
if PWO < 250 then PWO=PWO+1 ' incrementa PWO
Delay=0 ' e azzera delay.
endif
if PDW=0 and Delay=5 then ' Se DW premuto e delay=5
if PWO > 50 then PWO=PWO-1 ' decrementa PWO
Delay=0 ' e azzera delay.
endif
ADCIN 0, ISV ' Legge corrente da ADC.
PULSOUT SVR, PWO ' Genera impulso sul servo.
pause 20 ' Pausa di 20 msec
if PEN=0 then Menu_PWUP ' Enter premuto: cambia display.
goto Ripeti ' Ripete ciclo lettura puls.
*****
Aggiornamento display
*****
Lcd_New
lcdout $FE,1 ' Routine aggiorna display.
lcdout $FE,2 ' Cancella display.
int=PWO/100 ' Corsore ad inizio.
fra=PWO//100 ' Calcola parte intera.
lcdout "Segnale=", #int, "." ' Calcola parte frazionaria.
if fra<10 then lcdout "0" ' Scrive parte frazionaria.
lcdout #fra, "mS" ' + unità "msec"
lcdout $FE, $CO, "Corrente=", #ISV, " mA" ' Scrive corrente.
return ' Terminato.

```

che minuto che i condensatori elettrolitici si scarichino completamente quindi inserite l'operazionale e il microcontrollore, quest'ultimo programmato con il codice esadecimale MF741.HEX (scaricabile anch'esso gratuitamente dal nostro sito) e il display. Per la programmazione del PIC, potete utilizzare un qualsiasi programmatore, come ad esempio i nostri FT284 e FT386. Riaccendete il circuito e, dopo avere regolato il contrasto tramite il trimmer R1 verificate

che sul display appaia la segnalazione visibile in Fig. 2. Il circuito, ora, è pronto per essere utilizzato.

Utilizzo in pratica

Il firmware (il cui sorgente completo è riportato in un riquadro a parte) non è molto complesso: esso deve semplicemente gestire in modo opportuno il segnale di comando del servocontrollo, insieme al display LCD e ai pulsanti.

Appena acceso il dispositivo, sul di-

splay compare il nome del firmware e la sua versione; la schermata successiva (Fig. 3, che appare dopo qualche istante) visualizza la durata dell'impulso di comando presente in uscita ed il valore di corrente assorbita dal servo. Quest'ultimo deve essere collegato al connettore SERVO rispettando le indicazioni riportate in Fig. 1.

Se il servocontrollo è lasciato libero e non si trova sotto sforzo, la corrente visualizzata corrisponde alla corrente

```

*****
Impostazione movimento massimo
*****
Menu_PWUP
  lcdout $FE,1, $FE,2      ' Prepara display.
  int=PWUP/100             ' Calcola parte intera.
  fra=PWUP//100            ' Calcola parte frazionaria.
  lcdout "Max=", #int, "." ' Scrive parte intera.
  if fra<10 then lcdout "0" ' Scrive parte frazionaria.
  lcdout #fra, "mS"        ' + unita' "msec".
Push_1
  pause 100                ' Aspetta rilascio pulsante.
  if PEN=0 then goto Push_1 ' Pausa di 100 msec.
  ' Ripete se Enter premuto.
Push_2
  button PUP,0,255,0,B0,1,incPWUP ' UP? Va a incPWUP.
  button PDW,0,255,0,B0,1,decPWUP ' DW? Va a decPWUP.
  Button PEN,0,255,0,B0,1,Menu_PWDW ' ENTER? Va a Menu_PWDW.
  goto Push_2
incPWUP
  if PWUP < 250 then PWUP = PWUP + 1
  goto Menu_PWUP
decPWUP
  if PWUP > PWDW then PWUP = PWUP - 1
  goto Menu_PWUP
  ' Riparte da Menu_PWUP.
*****
Impostazione movimento minimo
*****
Menu_PWDW
  lcdout $FE,1, $FE,2      ' Prepara display.
  int=PWDW/100             ' Calcola parte intera.
  fra=PWDW//100            ' Calcola parte frazionaria.
  lcdout "Min=", #int, "." ' Scrive parte intera.
  if fra<10 then lcdout "0" ' Scrive parte frazionaria.
  lcdout #fra, "mS"        ' + unita' "msec".
Push_3
  pause 100                ' Aspetta rilascio pulsante.
  if PEN=0 then goto Push_3 ' Pausa di 100 msec.
  ' Ripete se Enter premuto.
Push_4
  button PUP,0,255,0,B0,1,incPWDW ' UP? Va a incPWDW.
  button PDW,0,255,0,B0,1,decPWDW ' DW? Va a decPWDW.
  Button PEN,0,255,0,B0,1,Menu_Time ' ENTER? Va a Menu_Time.
  goto Push_4
incPWDW
  if PWDW < PWUP then PWDW = PWDW + 1
  goto Menu_PWDW
decPWDW
  if PWDW > 50 then PWDW = PWDW - 1
  goto Menu_PWDW
  ' Riparte da Menu_PWDW.
*****
Impostazione tempo
*****
Menu_Time
  lcdout $FE,1, $FE,2      ' Prepara display.
  if Speed=0 then lcdout "Slow" ' Test Lento.
  if Speed=1 then lcdout "Fast" ' Test Veloce.
Push_5
  pause 100                ' Aspetta rilascio pulsante.
  if PEN=0 then goto Push_5 ' Pausa di 100 msec.
  ' Ripete se Enter premuto.
Push_6
  if PUP=0 then
    Speed=1                ' Se UP premuto
    goto Menu_Time         ' imposta test Veloce
  endif
  if PDW=0 then
    Speed=0                ' Se DW premuto
    goto Menu_Time         ' imposta test Lento
  endif
  ' e riparte.

  if PEN=0 then Run_Servo ' Se Enter premuto avvia servo.
  goto Push_6              ' Ripete.
*****
Esecuzione test
*****
Run_Servo
  lcdout $FE,1, $FE,2      ' Prepara il display.
  lcdout "Run"             ' Scrive "Run".
Push_7
  pause 100                ' Aspetta rilascio pulsante.
  if PEN=0 then goto Push_7 ' Ripete se Enter premuto.
Speed0
  ISVM=0                   ' Test Lento.
  ' Azzera corrente massima.
  if Speed=0 then
    ' Test Lento?
    for i=PWDW to PWUP
      pulsout SVR, i
      pause 18
      ADCIN 0, TMP
      ISVM=ISVM+TMP
      ' Ciclo test da PWDW a PWUP.
      ' Genera impulso servo.
      if PEN=0 then Main
      PWO=I
      ' Pausa 18 msec.
      ' Misura la corrente.
      CALL Lcd_New
      ' Somma per calcolo media.
      ' Enter premuto: ferma il test.
      ' Legge durata impulso.
      ' Aggiorna display.
      ' Ripete.
    next i
    ISV=ISVM/(PWUP-PWDW)
    call Lcd_New
    ' Calcola corrente media.
    ' Scrive a display.
    ISVM=0
    ' Azzera corrente media.
    for i=PWUP to PWDW step -1
      ' Ciclo test da PWUP a PWDW.
      ' Genera impulso servo.
      pulsout SVR, i
      pause 18
      ADCIN 0, TMP
      ' Pausa 18 msec.
      ' Misura la corrente.
      ISVM=ISVM+TMP
      ' Somma per calcolo media.
      if PEN=0 then Main
      PWO=I
      ' Enter premuto: ferma il test.
      ' Legge durata impulso.
      CALL Lcd_New
      ' Aggiorna display.
      ' Ripete.
    next i
    ISV=ISVM/(PWUP-PWDW)
    call Lcd_New
    ' Calcola corrente media.
    ' Scrive a display.
    ' Ripete.
  endif
  ' Fine test Lento.
Speed1
  TV=0                     ' Test Veloce.
  ISVM=0                   ' Azzera variabile di servizio.
  ' Azzera corrente massima.
  repeat
    TV=TV+1
    pulsout SVR, PWDW
    pause 18
    ADCIN 0, TMP
    ISVM=ISVM+TMP
    ' Avvia test minimo.
    ' Incrementa TV.
    if pen=0 then Main
    ' Genera impulso minimo.
    ' Pausa di 18 msec.
    ' Misura la corrente.
    ISVM=ISVM+TMP
    ' Somma per calcolo media.
    ' Enter premuto: ferma il test.
    ' 100 ripetizioni.
    until TV>100
    ISV=ISVM/100
    PWO=PWDW
    ' Calcola corrente media.
    ' Legge durata impulso.
    call Lcd_New
    ' Aggiorna il display.
    ' Azzera TV.
    repeat
      TV=TV+1
      PULSOUT SVR, PWUP
      pause 18
      ADCIN 0, TMP
      ISVM=ISVM+TMP
      ' Avvia test massimo.
      ' Incrementa TV.
      ' Genera impulso massimo.
      ' Pausa 18 msec.
      ' Misura la corrente.
      ISVM=ISVM+TMP
      ' Somma per calcolo media.
      ' 100 ripetizioni.
      ' Calcola corrente media.
      ' Legge durata impulso.
      ' Aggiorna il display.
      ' Ripete il test.
    until TV>100
    ISV=ISVM/100
    PWO=PWUP
    ' Calcola corrente media.
    ' Legge durata impulso.
    call Lcd_New
    ' Aggiorna il display.
    ' Ripete il test.
  repeat
  ' Fine programma.

```

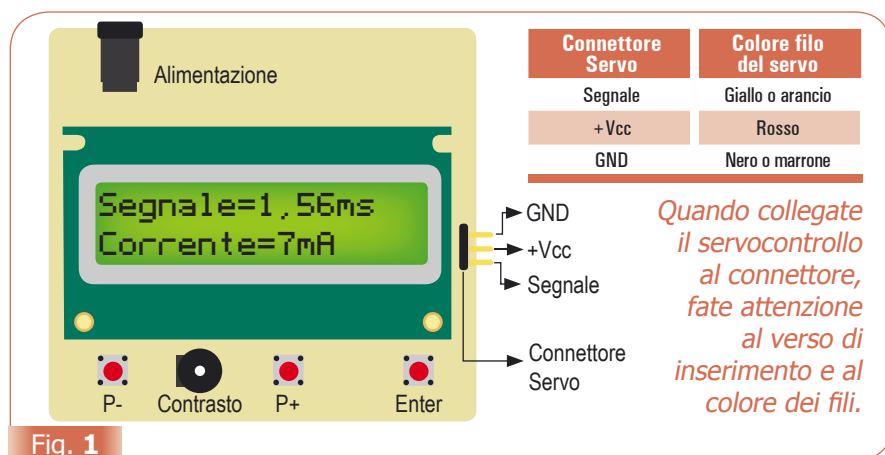


Fig. 1

“a riposo” (Fig. 3). Premendo i pulsanti *P+* e *P-* (fate sempre riferimento alla Fig. 1) è possibile modificare manualmente la durata dell’impulso sino agli estremi di 0,5 msec e 2,5 msec. L’estensione prevista (0,5÷2,5 msec) è quella massima utilizzabile, per cui alcuni servi potrebbero posizionarsi a fondo corsa prima del raggiungimento di questi limiti: in effetti il nostro tester serve anche per valutare l’ampiezza della rotazione del perno di un servocontrollo qualsiasi. Il ➤

TDS V1

Fig. 2

posizionamento manuale è comodo per le verifiche preliminari, ma, per rendere il nostro dispositivo più completo, abbiamo voluto dare la possibilità di programmare le escursioni automatiche agendo opportunamente sui pulsanti. Ipotizziamo che stiate controllando un servo i cui estremi d'escursione vengano raggiunti quando gli impulsi di controllo hanno una durata minima di 0,6 msec e massima di 2,30 msec.

A questo punto eseguiamo il test automatico: premendo il tasto *Enter*, appare una schermata (Fig. 4) che mostra la massima durata dell'impulso in uscita. Tale valore può essere modificato agendo sui pulsanti *P+* e *P-*: impostiamo il valore di 2,28 msec, che corrispondono quasi al nostro li-

Segnale=1,22msec
Corrente=8mA

Fig. 3

Max=2,28msec

Fig. 4

mite massimo. Premendo nuovamente *Enter*, si conferma l'impostazione del parametro visualizzato; compare, quindi, una seconda schermata nella quale viene chiesto il valore minimo

Min=0,62msec

Fig. 5

dell'impulso in uscita (Fig. 5). Tale valore deve essere sempre inferiore al valore massimo; agendo ancora su *P+* o *P-*, impostiamo il nostro limite minimo di 0,62 msec. Premendo *Enter*, viene memorizzato anche il valore minimo; a questo punto appare una nuova schermata (Fig. 6) in cui viene chiesto se il movimento del servo deve essere lento (tasto *P-*) oppure veloce (tasto *P+*). Ipotizziamo di scegliere il test lento, che verifica tutte le posizioni del servo comprese tra i valori di impulso minimo e massimo.

Premendo nuovamente *Enter*, si avvia la movimentazione automatica con i parametri impostati: per un attimo sul display compare la scritta "Run" che lascerà, dopo qualche istante, il posto

Slow

Fig. 6

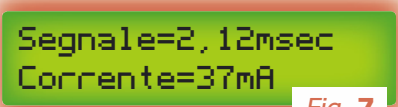
Segnale=2,12msec
Corrente=37mA

Fig. 7

alla visualizzazione dei parametri di funzionamento (Fig. 7). Il test viene terminato con una nuova pressione del tasto *Enter*.

Avendo scelto il test "Slow", il servo

inizia a muoversi lentamente e alternativamente tra i due estremi impostati, mentre il display mostra il valore dell'impulso in uscita in tempo reale e la corrente media assorbita dal servo ad ogni cambio di direzione.

Questa funzione permette di verificare il funzionamento sia a vuoto che a carico: quando il servocontrollo viene provato "al banco", il test consente di stabilire se gli ingranaggi sono perfettamente funzionanti e se il movimento è fluido e preciso. Quando, invece, il servo è installato a bordo di un modello, lo stesso test permette di verificare se la corsa è ottimale e correttamente eseguita oppure se, per effetto di problemi meccanici sui leveraggi, sono presenti dei punti di sovraccarico.

Scegliendo il test "Fast" (che mostra sempre la durata degli impulsi di controllo e la corrente media assorbita), al servo giungeranno alternativamente gli impulsi minimo e massimo senza valori intermedi, intervallati da un leggero ritardo, in modo da verificarne l'effettiva velocità di rotazione del perno. Quando il servo è installato in un modello, tramite questo test è possibile verificare se esso è adatto per l'applicazione: nel caso fosse troppo debole, il servo non riuscirà a riposizionarsi con la dovuta velocità e pertanto, non riuscendo a seguire il segnale di controllo, si muoverà con evidenti difficoltà e notevole sforzo da parte del motore interno; in gergo modellistico, si dice che "il servo va in flutter".

per il

MATERIALE

Il materiale utilizzato in questo progetto è facilmente reperibile presso qualsiasi negozio di elettronica. Il master del circuito stampato ed il firmware col quale programmare il microcontrollore possono essere scaricati dal sito della rivista (www.elettronica.in.it). Il microcontrollore già programmato (cod. MF741) può essere acquistato al costo di 12.00 Euro, IVA compresa.

Il materiale va richiesto a: Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)

Tel: 0331-799775 ~ Fax: 0331-792287 ~ <http://www.futurashop.it>

Datalogger USB firmware e software

di *Francesco Ficili*

Iniziamo ad utilizzare in pratica ed a prendere confidenza col datalogger presentato il mese scorso. Analizziamo in dettaglio le routine firmware più significative ed installiamo il dispositivo, insieme al software di controllo relativo, su un PC. Vediamo quindi come utilizzare il software di controllo in ambiente Windows. Ultima puntata.

Nella puntata precedente abbiamo presentato un datalogger USB a 12 canali (codice progetto FT617), soffermandoci principalmente sulla descrizione della parte hardware e mettendo in evidenza le scelte di progetto effettuate, i principi di funzionamento sui quali si basa il dispositivo ed i dettagli pratici per il montaggio della scheda. In questa seconda parte, conclusiva, si vuole porre in maggior risalto sia tutto ciò che riguarda la realizzazione del firmware con il quale è stato programmato il microcontrollore PIC18F4550 sia il software per ambienti Windows, appositamente svilup-

pato per effettuare l'acquisizione dei dati ed il controllo remoto del dispositivo. Per prima cosa, analizzeremo il diagramma di flusso del firmware che viene eseguito dal microcontrollore, evidenziando le parti più significative del codice. Successivamente ci occuperemo dell'utilizzo pratico dell'interfaccia software, che permette all'utente sia di visualizzare il risultato dell'acquisizione, che di impostare una serie di parametri di funzionamento, come i canali attivi, i guadagni dei canali analogici, gli allarmi di superamento soglia, la memorizzazione dei dati sulla EEPROM della scheda, ed altro ancora. Una ➤

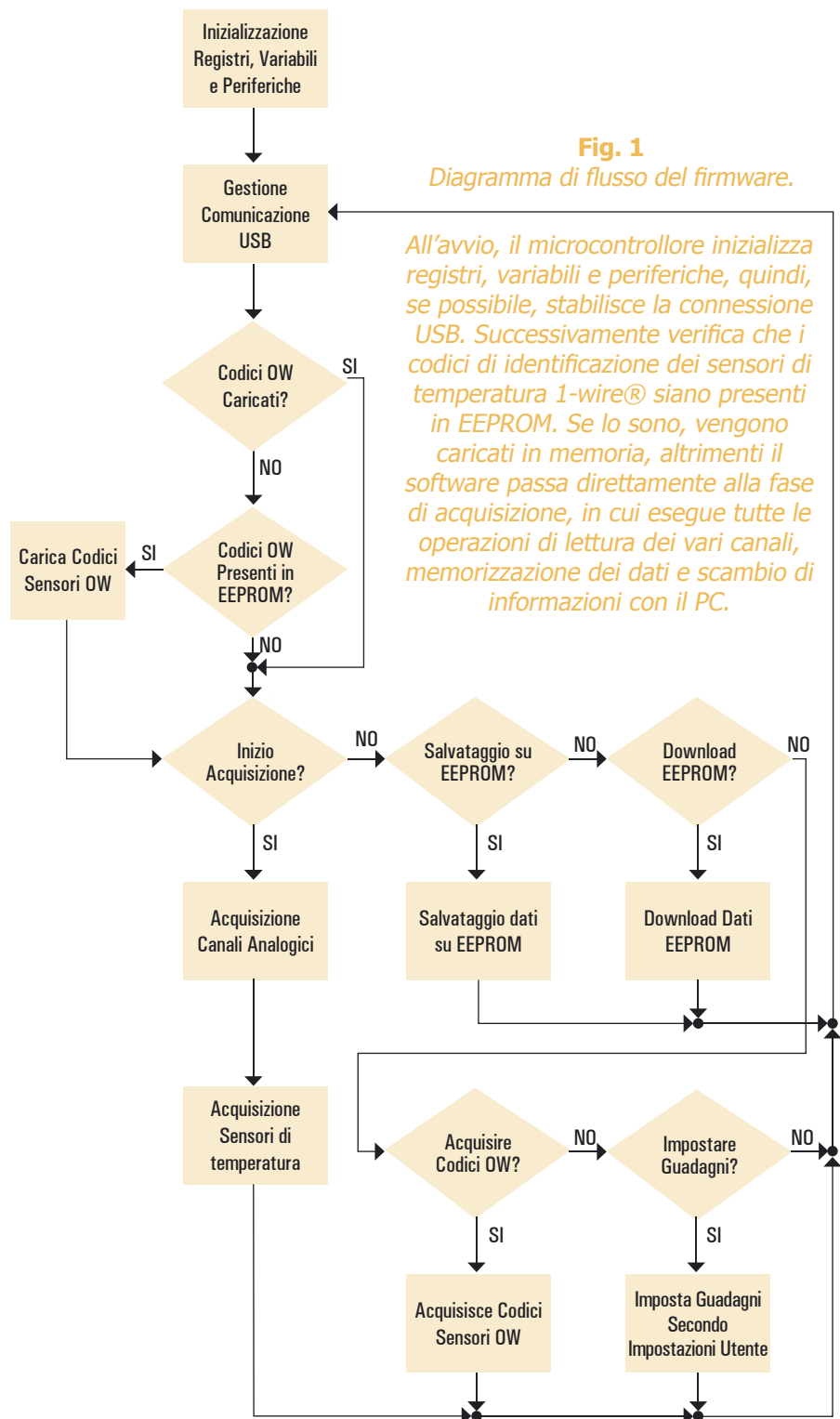
piccola parte dell'articolo è stata dedicata, inoltre, alla procedura di collegamento e configurazione del dispositivo con personal computer che dispongano del sistema operativo Windows XP.

Diagramma di flusso del firmware

Prima di iniziare a descrivere il firmware, è opportuno sottolineare che esso è stato scritto in linguaggio C, usando in combinazione l'ambiente di sviluppo proprietario Microchip (MPLab IDE) e il compilatore "C18 Student Edition": la sinergia di questi due tool fornisce un completo e potente ambiente di sviluppo integrato che, inoltre, ha il grande pregio (soprattutto per gli hobbisti) di essere completamente gratuito. La Fig. 3 mostra la finestra del file manager di MPLab IDE relativa al progetto del datalogger USB. Per analizzare in dettaglio il codice, occorre fare riferimento al diagramma di flusso di Fig. 1, che illustra in maniera schematica le varie fasi operative del programma all'interno del dispositivo.

All'avvio del programma, vengono inizializzati i registri di controllo delle varie periferiche utilizzate, come l'interfaccia PC, il convertitore analogico/digitale, l'interfaccia USB 2.0, oltre a tutte le porte di I/O generiche e ad altri registri speciali del microcontrollore. In questa fase vengono, inoltre, inizializzate tutte le variabili ed i contatori utilizzati all'interno del programma.

Terminata la fase di inizializzazione, la parte di firmware dedicata alla gestione del collegamento USB dà inizio al processo di *enumerazione*, che porta il dispositivo dallo stato 1 (collegato) allo stato 5 (configurato). Per quel che riguarda l'implementazione firmware del protocollo di comunicazione tra PC e datalogger, si è scelto di utilizzare la classe *USB CDC (Communication Device*



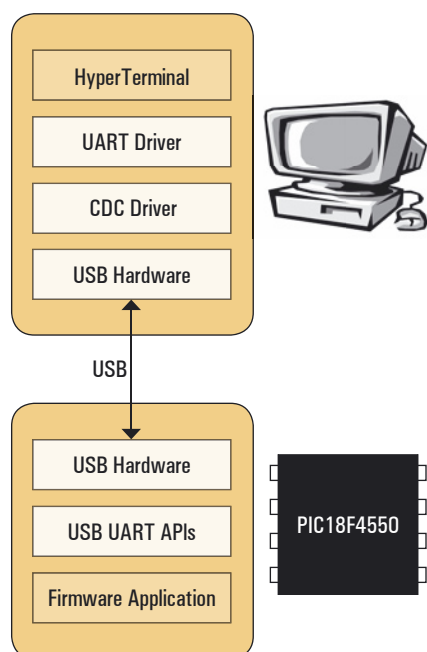
Class), che permette di creare "dispositivi" di comunicazione seriale (leggasi "porte seriali virtuali") generici basati sullo standard USB: ciò è particolarmente utile in quanto i sistemi operativi Microsoft Windows dispongono già al loro interno di un apposito driver per la gestione di dispositivi CDC.

Tale driver si interpone fra lo strato hardware USB (livello fisico) ed il driver UART (livello di link) del sistema operativo, trasformando, di fatto, la periferica USB in una emulazione di periferica seriale, come illustrato schematicamente in Fig. 2. Una volta terminato il processo di enumerazione, se il dispositivo

si trova allo stato “configurato”, il controllo del programma viene trasferito alla funzione *user.c*, che si occupa della gestione delle varie periferiche (convertitore A/D, I²C bus, porte di I/O digitale, eccetera), sfruttando le macroistruzioni USB messe a disposizione dalla funzione *cdc.h*, per inviare e ricevere dati dall’host (computer). Il firmware che si occupa della gestione del protocollo viene richiamato ciclicamente per mantenere attiva la comunicazione con l’host USB (PC).

L’insieme delle istruzioni cui è affidata la gestione della comunicazione USB è stato per semplicità indicato nel diagramma di Fig. 1 con il solo blocco “Gestione Comunicazione USB”, ma la sua implementazione è abbastanza complessa e lo testimonia il fatto che l’occupazione di memoria di questa porzione di codice supera i 3 kbyte. In questo articolo, comunque, non verrà analizzata in dettaglio questa parte del firmware, poiché una trattazione completa sarebbe eccessivamente pesante. Nel caso, però, gradite avere ulteriori delucidazioni, la sezione *USB Connectivity* del sito di Microchip Te-

Fig. 2 - Il collegamento USB tra scheda e PC avviene emulando una seriale virtuale.



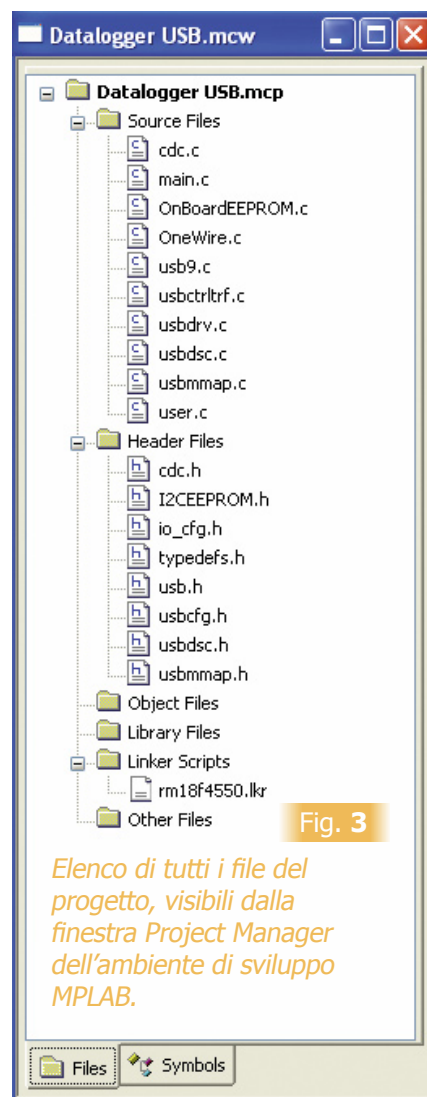
chnology è un eccellente punto di riferimento.

Come già accennato in precedenza, terminato il processo di enumerazione, se il dispositivo si trova nello stato “configurato”, esso è pronto per raccogliere dati dalle sue periferiche e trasferirle verso l’host USB al quale è collegato. La funzione che si occupa di effettuare queste operazioni è la *user.c*: essa è costituita, di fatto, dall’implementazione firmware di una macchina a stati finiti nella quale, ad ogni stato corrisponde, tipicamente, una prima fase in cui viene interrogata una determinata periferica ed una seconda fase nella quale il risultato dell’interrogazione viene trasferito verso l’host.

Esaminiamo, allora, più in dettaglio i vari stati (che non sono altro che le varie funzioni che può compiere il datalogger), facendo sempre riferimento alla Fig. 1.

Per prima cosa, vengono controllate le variabili contenenti i codici univoci identificativi dei sensori one-wire (OW), così da verificare che tali codici siano stati caricati correttamente nelle rispettive variabili. Se la verifica è positiva (codici caricati correttamente) il programma passa direttamente alla fase successiva, altrimenti verifica la presenza dei codici nella EEPROM interna del micro. Se i codici sono presenti in memoria vengono immediatamente caricati (la parte di firmware che si occupa dell’acquisizione e memorizzazione dei codici dei sensori verrà esaminata più in dettaglio in seguito). A questo punto il dispositivo è pronto per effettuare le diverse operazioni per cui è stato predisposto.

L’operazione che viene più spesso richiesta al firmware (tra le varie disponibili) consiste nell’effettuare un ciclo di acquisizione dei sei canali analogici e dei sei sensori di temperatura one-wire. Esaminiamo in dettaglio il firmware che implementa queste due funzioni, prendendo come esempio il canale analogico



AN0 ed il sensore numero 1. Il frammento di firmware che realizza l’acquisizione di un canale analogico è riportato nel Listato 1. La prima riga seleziona il canale attivo (in questo caso AN0), intervenendo sul primo registro di controllo del convertitore A/D (ADCON0).

La seconda riga serve ad attivare il convertitore A/D, settando il bit 0 del registro ADCON0 (ADON). La terza riga serve a dare effettivamente inizio alla conversione, ponendo a 1 il bit GO/DONE appartenente sempre al registro ADCON0. Questo bit, oltre a fungere da bit di start per il convertitore, è anche un flag (segnalatore) che indica lo stato della conversione, difatti viene resettato dal PIC a conversione ultimata. La riga 4 sfrutta proprio questa ➤

proprietà effettuando un controllo in *polling* (lettura ciclica) su tale bit, per mezzo di un ciclo *while*. Questa operazione serve a mettere il microcontrollore in attesa fino a quando la conversione non è ultimata. L'istruzione della quinta riga, infine, sposta il contenuto del registro ADRESH (nel quale è immagazzinato il risultato della conversione su 8 bit) all'interno del buffer di trasferimento, chiamato anch'esso ADRESH.

Vediamo ora più in dettaglio come viene realizzata, dal punto di vista firmware, la lettura dei sensori one-wire, facendo riferimento al Listato 2 che illustra la procedura per il sensore 1. La prima riga (che avvia la transazione 1) invoca la funzione *reset_owewire*, che fornisce un impulso di reset della durata di 500 µs, con il quale tutti i dispositivi one-wire

presenti sul bus vengono avvisati che sta per iniziare una nuova transazione dati. La seconda riga invoca la funzione *write_byte_owewire* (che come dice il nome stesso scrive un byte sul bus one-wire) per inviare sul bus il comando *Match ROM* (55h), che serve ad informare i dispositivi presenti che il master (ossia il microcontrollore) sta per inviare il codice identificativo a 64 bit con il quale è possibile indirizzare un determinato dispositivo, distinguendolo da tutti gli altri. Il ciclo *while* successivo invia gli 8 byte del codice ROM sul bus, sfruttando sempre la routine *write_byte_owewire*.

A questo punto lo scambio di dati prosegue solo tra il master e lo slave indirizzato, mentre tutti gli altri dispositivi si mettono in attesa del successivo impulso di reset. L'istru-

zione della riga 7 scrive il comando *Convert T* (44h) sul bus; ricevuto questo comando il DS18B20 dà inizio ad una conversione di temperatura. Il ciclo *while* successivo mette il microcontrollore in attesa finché lo slave non comunica che l'acquisizione è terminata. A questo punto il DS18B20 memorizza il dato di temperatura nella sua memoria di lavoro e ritorna nello stato iniziale (*idle state*) rimanendo in attesa di ricevere istruzioni dal microcontrollore. Nella transazione successiva (transazione 2), vengono ripetuti tutti i passi della precedente, ad eccezione della riga 15, dove, anziché richiedere una conversione di temperatura, il master richiede la lettura della memoria di lavoro (*Read Scratchpad*) nella quale è contenuto il risultato della conversione. Lo slave

Listato 1

```
/* ----- CANALE AN0 ----- */
ADCON0 = 0b00000000;           // Selezione canale AN0.
ADCON0bits.ADON = 1;           // Attivazione ADC.

/* Conversione AN0 */
ADCON0bits.GO_DONE = 1;         // Inizio conversione.

while (ADCON0bits.GO_DONE == 1); // Polling sul bit GO_DONE.
output_buffer[0] = ADRESH;      // Copia dato a 8-bit sul
                                // buffer di trasferimento.
```

Listato 2

```
/* ----- SENSORE OW1 ----- */

reset_owewire();                // Transazione 1.
write_byte_owewire(0x55);       // ROM command "Match ROM".

for (m=0; m<8; m++)            // Invia codice ROM a 64 bit.
{
    write_byte_owewire(ROM_code_owewire_1[m]);
}

write_byte_owewire(0x44);       // Function command "Convert T".
while(read_bit_owewire() == 0); // Attendi fino a quando la
                                // conversione non è ultimata.

reset_owewire();                // Transazione 2.
write_byte_owewire(0x55);       // ROM command "Match ROM".

for (m=0; m<8; m++)            // Invia codice ROM a 64 bit.
{
    write_byte_owewire(ROM_code_owewire_1[m]);
}

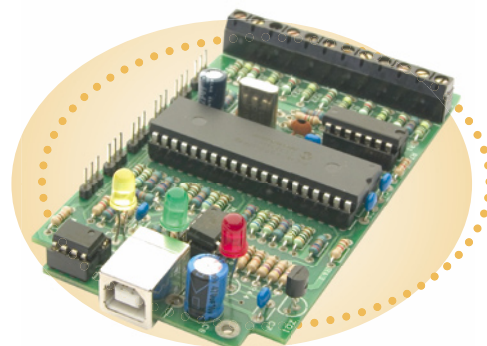
write_byte_owewire(0xBE);       // Function command.
                                // "Read Scratchpad".

temperature_1_LSB = read_byte_owewire(); // Leggi LSB.
temperature_1_MSB = read_byte_owewire(); // Leggi MSB.

output_buffer[6] = temperature_1_LSB;    // Sposta LSB nel
                                // buffer di trasf.

output_buffer[7] = temperature_1_MSB;    // Sposta MSB nel
                                // buffer di trasf.

/* ----- */
```



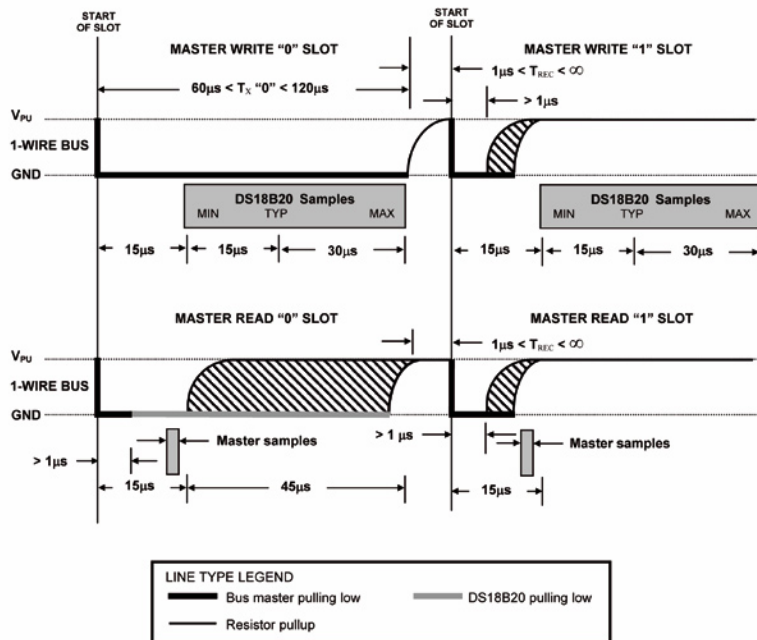
interpreta il comando ed invia i due byte contenenti il risultato sul bus. Le istruzioni delle righe 16 e 17 leggono i due byte, facendo uso della routine *read_byte_owewire*, e li memorizzano in due variabili temporanee, prima di copiarli sul buffer di trasferimento.

In Fig. 4 sono visualizzati i grafici delle tempistiche di lettura/scrittura sul bus one-wire, grafici sui quali ci si è basati per la realizzazione delle routine *write_byte_owewire* e *read_byte_owewire*.

All'interno di un ciclo di acquisizione, le operazioni di lettura dei canali analogici e dei sensori di temperatura one-wire vengono ripetute per i sei canali ed i sei sensori e, alla fine, il contenuto del buffer di trasferimento (*output_buffer*) viene inviato dal

Fig. 4 - Tempistiche di lettura/scrittura sul bus 1-wire®.

READ/WRITE TIME SLOT TIMING DIAGRAM



microcontrollore al PC facendo uso della macro *mUSBUSARTTxRam*, fornita dalla funzione *cdc.h*.

Vediamo adesso la porzione di firmware con la quale viene gestito il salvataggio dei dati sulla EEPROM

Listato 3

```

/* ----- EEPROM WRITE AN0 ----- */
// Scrittura byte dati su EEPROM I2C.
WriteI2CEEPROM (0xA0,I2CAddressH,I2CAddressL,I2CSample0);
for (i=0;i<2000;i++); // Ciclo di ritardo.
/* Incremento indirizzo */
if (I2CAddressL <= 255) // Se I2CAddressL è < 255 incrementalo.
{
    I2CAddressL++;
}
else
{
    I2CAddressL = 0; // Altrimenti I2CAddressL = 0 e incrementa
    I2CAddressH++; // I2CAddressH.
}
/* ----- */

```

Listato 4

```

if (mUSBUSARTIsTxTrfReady()) // Controlla se la periferica USB
{ // è pronta a trasmettere.
    /* ----- EEPROM READ ----- */
    // Lettura byte dati da EEPROM I2C.
    output_buffer[0] = ReadI2CEEPROM(0xA0,I2CAddTempH,I2CAddTempL);
    /* Incremento indirizzo */
    if (I2CAddTempL <= 255) // Se I2CAddressL è < 255 incrementalo.
    {
        I2CAddTempL++;
    }
    else
    {
        I2CAddTempL = 0; // Altrimenti I2CAddressL = 0 e incrementa
        I2CAddTempH++; // I2CAddressH.
    }
}
/* ----- */

```

montata a bordo della scheda. Per maggior chiarezza facciamo riferimento al frammento di codice riportato nel Listato 3, che illustra il procedimento di salvataggio nel caso del canale AN0.

Viene fatto uso della routine *WriteI2CEEPROM* che permette di scrivere un byte all'interno della EEPROM utilizzando il protocollo I²C bus. La routine prende 4 byte come argomento, un primo fisso (A0h), il secondo ed il terzo byte contengono l'indirizzo della cella di memoria puntata ed il quarto, infine, il byte dati da salvare. L'istruzione successiva è un ciclo *for* di ritardo per consentire il completamento dell'operazione di scrittura, mentre la struttura *if* che segue il *for* permette l'incremento dei byte relativi all'indirizzo.

Per quel che riguarda invece il download dei dati contenuti all'interno della EEPROM si faccia riferimento al Listato 4, che illustra il frammento di codice utilizzato per questa operazione.

Per prima cosa si controlla se il modulo USB è pronto alla trasmissione, per mezzo della macro *mUSBUSARTIsTxTrfReady*. L'invocazione di questa macro restituisce "0" se il modulo USB non è pronto a trasmettere e "1" nell'altro caso. Con l'istruzione successiva, si inizia a riempire il buffer di trasferimento con il contenuto della EEPROM all'indirizzo specificato dalla coppia *I2CAddTempL* e *I2CAddTempH* (due byte contenenti le parti alta e bassa dell'indirizzo della locazione EEPROM che si desidera leggere) per mezzo della funzione *ReadI2CEEPROM*. L'istruzione di lettura della EEPROM è molto più veloce di quella di scrittura e pertanto non sono necessarie istruzioni per forzare un ritardo. L'istruzione *if* successiva funziona esattamente come nel caso della scrittura su EEPROM, incrementando la parte alta e bassa dell'indirizzo. Una volta ➤

Fig. 5 - Codice ROM a 64 bit del DS18B20

64BIT LASERED ROM CODE

8-BIT CRC		48-BIT SERIAL NUMBER		8-BIT FAMILY CODE (28h)	
MSB	LSB	MSB	LSB	MSB	LSB

Listato 5

```
EEadr = 0;
reset_onewire();                // Reset del bus onewire.
write_byte_onewire(0x33);        // Invia comando ROM 'Read ROM'.

for(i=0;i<8;i++)                // Legge gli 8 byte del codice ROM.
{
    ROM_code_onewire_1[i] = read_byte_onewire();
}

for(i=0;i<8;i++)                // Scrive gli 8 byte del codice ROM
                                // sulla EEPROM.
{
    WriteEEdat(EEadr,ROM_code_onewire_1[i]);
    EEadr++; // Incrementa indirizzo
    while(EECON1bits.WR == 1);
}
```

Listato 6

La funzione Autorange

Il calcolo dell'autorange sui canali analogici è stato realizzato via software, sfruttando le potenzialità del Visual Basic 6.

Il principio che sta alla base della funzione che lo implementa è molto semplice: se nel menu *Impostazione Guadagni e Canali Attivi* viene selezionata la funzione *autorange* nel programma principale, allora viene attivata una routine che valuta (utilizzando un'istruzione *select case*) il valore assunto dalla variabile in cui è immagazzinato il valore del corrispondente canale analogico.

A seconda del valore letto si possono verificare tre casi:

1. *Valore < 5*: viene inviato al microcontrollore il comando per impostare il guadagno adatto al range 0-5V e si imposta il fattore moltiplicativo a 1.4;
2. *5 < Valore < 10*: viene inviato al microcontrollore il comando per impostare il guadagno adatto al range 0-10V e si imposta il fattore moltiplicativo a 2.8;
3. *Valore > 10*: viene inviato al microcontrollore il comando per impostare il guadagno adatto al range 0-30V e si imposta il fattore moltiplicativo a 7.

I fattori moltiplicativi sono calcolati in base alla configurazione assunta dallo stadio d'ingresso del canale considerato (partitore + amplificatore) e servono per risalire correttamente al valore analogico che viene applicato all'ingresso del partitore passivo. Di seguito è riportato il codice VB6 utilizzato per l'autorange del canale analogico 1.

```
If (autorange_ch1 = 1) Then      ' Esegue solo se autorange attivato.
    Select Case (Voltage_1)      ' Istruzione select su Voltage_1
        Case 0 To 5              ' Se voltage_1 è compreso tra 0 e 5
            MSComm1.Output = "0" ' seleziona range 0-5V e imposta
            Gain_Ch1 = 1.4        ' fattore moltiplicativo adeguato.
        Case 5 To 10             ' Se voltage_1 è compreso tra 5 e 10
            MSComm1.Output = "1" ' seleziona range 0-10V e imposta
            Gain_Ch1 = 2.8        ' fattore moltiplicativo adeguato.
        Case 10 To 30            ' Se voltage_1 è compreso tra 10 e 30
            MSComm1.Output = "2" ' seleziona range 10-30V e imposta
            Gain_Ch1 = 7          ' fattore moltiplicativo adeguato.
    End Select                  ' Fine istruzione select.
End If                          ' Fine IF relativo autorange canale 1.
```

immagazzinati i tutti dati dei canali analogici e dei sensori di temperatura all'interno del buffer di trasferimento, il contenuto di quest'ultimo viene inviato al PC utilizzando sempre la macro *mUSBUSARTTxRam*.

A questo punto, facendo sempre riferimento al diagramma di flusso di Fig. 1, si può osservare che il blocco successivo è il blocco di acquisizione dei codici dei sensori one-wire: esso è indispensabile per la corretta gestione dei sensori DS18B20 ed il compito del codice firmware con il quale è implementato è quello di richiedere, acquisire e memorizzare nella EEPROM interna i codici ROM a 64 bit di identificazione dei sei sensori.

Una volta eseguita la procedura di acquisizione dei codici, se non si decide di cambiare il blocco dei sensori, per i successivi utilizzi del datalogger il microcontrollore recupererà i codici direttamente dalla sua EEPROM in maniera trasparente per l'utilizzatore. In Fig. 5 è riportato il formato del codice ROM del DS18B20.

Analizziamo più in dettaglio il firmware con il quale è stato realizzato il blocco di acquisizione dei codici dei sensori, facendo riferimento al Listato 5, che si applica al sensore numero 1.

Come si può vedere, dopo la prima riga nella quale viene inizializzata a "0" la variabile di indirizzo della EEPROM, si procede con l'istruzione di reset del bus one-wire, obbligatoria per iniziare qualsiasi transazione dati. Successivamente si usa la funzione *write_byte_onewire* per scrivere sul bus il comando *Read ROM (33h)*, utilizzato per richiedere al dispositivo collegato di inviare il codice identificativo. Con il ciclo *for* successivo, si memorizza il codice all'interno dell'array *ROM_code_onewire*. A questo punto, utilizzando un altro ciclo *for*, si memorizza il contenuto dell'array sulla EEPROM interna, facendo uso della funzione

Tabella 1 - Il set di comandi

Nome Comando	Codice Comando	Funzione
Start Acquisizione	0x53	Avvia un ciclo di acquisizione dei canali analogici e dei sensori di temperatura. Al termine del ciclo spedisce un pacchetto dati sul bus USB.
Acquisisce Codice Sensore One-Wire	0x4F	Acquisisce il codice ROM del sensore collegato sul bus One-Wire e lo memorizza sulla EEPROM interna.
Cancella Codici dei Sensori One-Wire	0x43	Cancella i codici dei sensori One-Wire contenuti nella EEPROM interna.
Scrivi I ² CEEPROM	0x45	Avvia un ciclo di acquisizione dei canali analogici e dei sensori di temperatura e memorizza il risultato sulla EEPROM I ² C montata a bordo della scheda.
Legge I ² CEEPROM	0x4C	Legge il contenuto di un intero pacchetto dati memorizzato sulla EEPROM della scheda e lo invia sul bus USB. Al termine dell'operazione incrementa il contatore indirizzo.
Reset	0x52	Resetta, con un comando software, il datalogger USB.
Guadagno Ch0	0x4*	* =9 Seleziona guadagno per range 0-5V. * =A Seleziona guadagno per range 5-10V. * =B Seleziona guadagno per range 10-30V.
Guadagno Ch1	0x3*	* =0 Seleziona guadagno per range 0-5V. * =1 Seleziona guadagno per range 5-10V. * =2 Seleziona guadagno per range 10-30V.
Guadagno Ch2	0x3*	* =3 Seleziona guadagno per range 0-5V. * =4 Seleziona guadagno per range 5-10V. * =5 Seleziona guadagno per range 10-30V.
Guadagno Ch3	0x3*	* =6 Seleziona guadagno per range 0-5V. * =7 Seleziona guadagno per range 5-10V. * =8 Seleziona guadagno per range 10-30V.
Guadagno Ch4	0x4*	* =1 Seleziona guadagno per range 0-5V. * =2 Seleziona guadagno per range 5-10V. * =4 Seleziona guadagno per range 10-30V.
Guadagno Ch5	0x4*	* =6 Seleziona guadagno per range 0-5V. * =7 Seleziona guadagno per range 5-10V. * =8 Seleziona guadagno per range 10-30V.
Versione Firmware	0x56	Richiede l'invio di una byte contenente la versione del firmware caricato sul datalogger.

WriteEEdata. Rimane da analizzare l'ultimo blocco di Fig. 1, che provvede all'impostazione dei guadagni. La funzione svolta da questo blocco è la variazione dei guadagni di retroazione degli amplificatori che costituiscono gli stadi d'ingresso dei sei canali analogici. Come già illustrato nella puntata precedente la variazione dei guadagni degli stadi d'ingresso consente di avere sempre la miglior risoluzione di misura

in rapporto al range del segnale in ingresso.

Sono possibili tre guadagni differenti, ottenuti collegando a massa una delle tre resistenze di retroazione di ciascun amplificatore (mentre le altre due vengono poste in alta impedenza; vi suggeriamo di andare a riprendere la rivista 125 e di rileggere il paragrafo *Principio di funzionamento* a pagina 73).

La funzione svolta dal firmware, quindi, è quella di intercettare un comando impartito da PC (attraverso la routine *getsUSBUSART*, fornita dalla funzione *cdc.c*) e agire di conseguenza sui registri PORT e TRIS (registri del microcontrollore) delle resistenze interessate, in modo da modificarne la configurazione e ottenere così il valore di guadagno richiesto.

Il protocollo di comunicazione

La gestione remota del datalogger è molto semplice. Sebbene per effettuare la comunicazione tra datalogger e PC venga effettivamente utilizzato lo standard USB, l'utilizzo della classe CDC rende il dispositivo, dal punto di vista dell'host, praticamente identico ad una periferica seriale RS232.

Questa *astrazione* (termine che può essere letto come "conversione virtuale") rende più agevole la creazione di una interfaccia di comunicazione, poiché consente di riutilizzare tutto il codice eventualmente scritto per la gestione di dispositivi RS232 per mezzo di ambienti di programmazione differenti. Utilizzando, ad esempio, Visual Basic 6 per realizzare un'interfaccia di comunicazione (che per inciso è proprio il caso nostro) può essere utilizzato il ➤

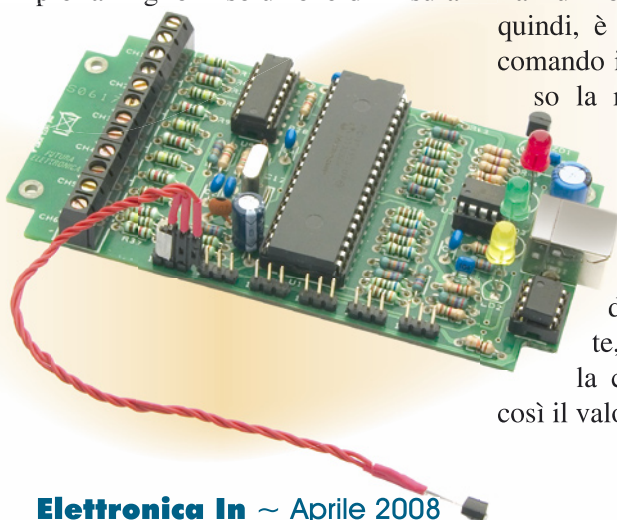


Fig. 6 - Formato del pacchetto dati inviato dal datalogger al PC.

Primo Byte Dati

Analog Channel 0	Analog Channel 1	Analog Channel 2	Analog Channel 3	Analog Channel 4	Analog Channel 5	Temp Sensor 0 LowByte	Temp Sensor 0 HighByte
------------------	------------------	------------------	------------------	------------------	------------------	-----------------------	------------------------

Temp Sensor 1 LowByte	Temp Sensor 1 LowByte	Temp Sensor 2 LowByte	Temp Sensor 2 LowByte	Temp Sensor 3 LowByte	Temp Sensor 3 LowByte	Temp Sensor 4 LowByte	Temp Sensor 4 LowByte
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------

Temp Sensor 5 LowByte	Temp Sensor 5 LowByte	Riservato	Riservato	Riservato	Riservato	Riservato	Riservato
-----------------------	-----------------------	-----------	-----------	-----------	-----------	-----------	-----------

Ultimo Byte Dati

Byte Riservati per usi futuri

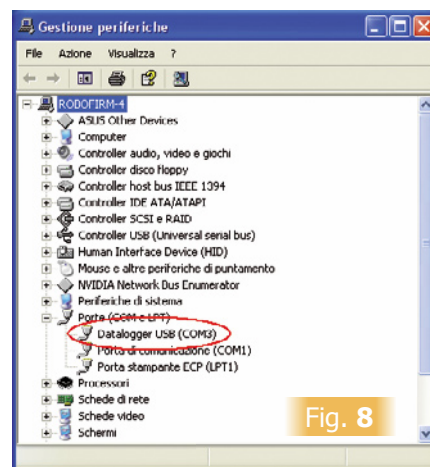


Fig. 8

controllo *MSComm* per inviare e ricevere dati dalla periferica USB. Tale controllo può essere facilmente aggiunto alla barra dei *tool* di *Visual Studio* e da questa al *form* (la finestra del programma che si sta sviluppando), dopodiché viene creato il controllo *MSComm1*, che può essere invocato per la gestione della porta COM virtuale.

In Tabella 1 è riportato il set di istruzioni supportato dal datalogger. Come si vede, è disponibile un totale di 24 istruzioni di alto livello (18 per la gestione dei guadagni), più

su 8 bit), 12 per i sensori di temperatura (campioni su 12 bit) ed infine 6 byte che abbiamo voluto riservare per eventuali usi futuri.

Collegamento e configurazione

Una volta programmato il PIC18F4550 con il firmware scaricabile gratuitamente dal sito della rivista, è possibile collegare il datalogger al PC attraverso una qualsiasi porta USB ed effettuare la configurazione del dispositivo.

Al primo collegamento, il sistema

dialogo richiede di specificare il percorso da cui il driver può essere scaricato: è possibile specificare tanto una cartella (di rete o nello stesso PC, Fig. 7b) quanto un indirizzo Internet.

Quando si utilizzano Windows XP e Vista, nella successiva finestra di dialogo compare un messaggio di notifica che il driver specificato non ha superato le prove WHQL (Windows Hardware Qualification Laboratories): cliccare comunque su *Continua*; il datalogger funzionerà correttamente.

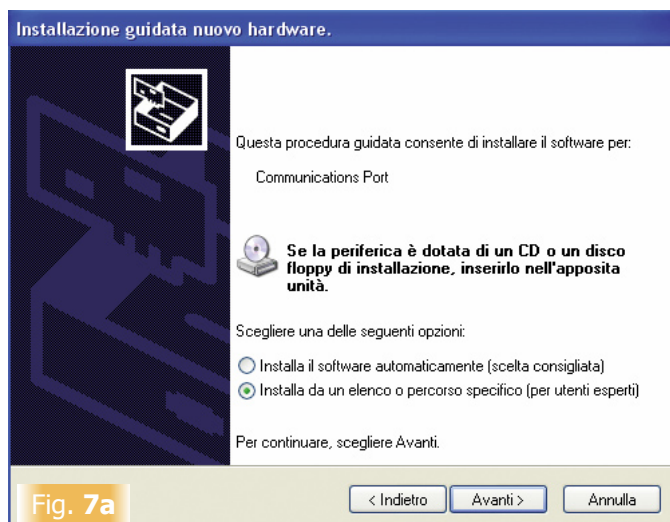


Fig. 7a

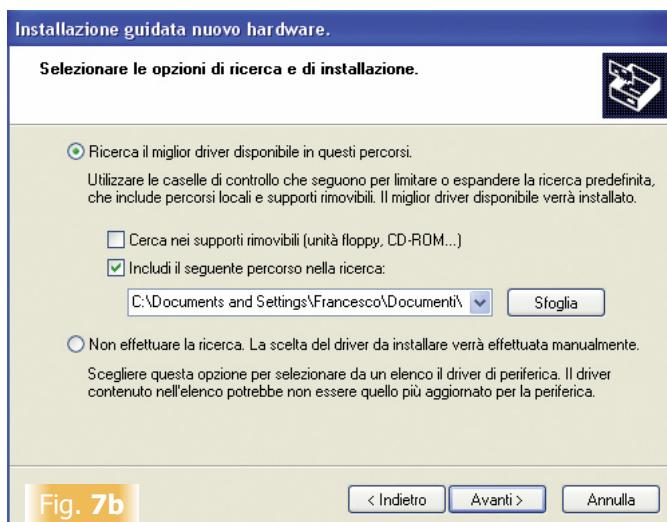


Fig. 7b

una per richiedere al sistema di fornire la versione del firmware. Il formato del pacchetto dati inviato dal PIC è rappresentato in Fig. 6: il pacchetto complessivo è composto da 24 byte trasmessi sequenzialmente: 6 per i canali analogici (campioni

operativo riconosce la nuova periferica e chiede all'utente di definire il percorso del driver della periferica, così come illustrato in Fig. 7a. Scegliete l'opzione "Installa da un elenco o percorso specifico (per utenti esperti)". La successiva finestra di

A questo punto Windows completa l'installazione del driver e la nuova periferica viene aggiunta all'elenco delle risorse di sistema (*Pannello di controllo / sistema / Hardware / gestione periferiche*) con il nome di Datalogger USB, sotto il gruppo

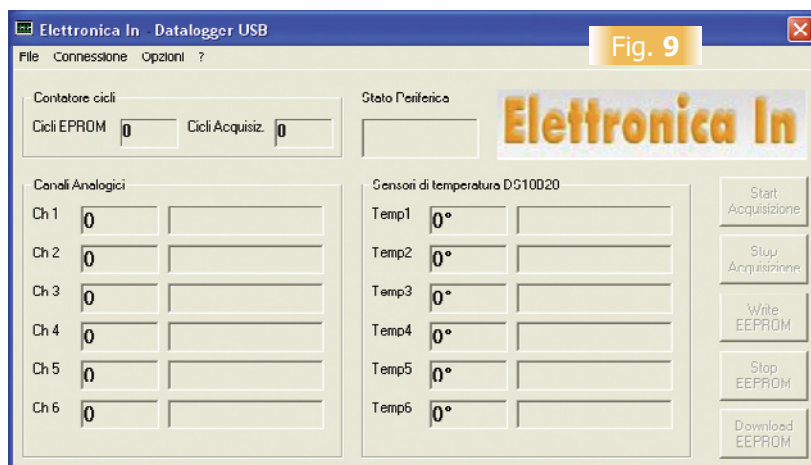


Fig. 9

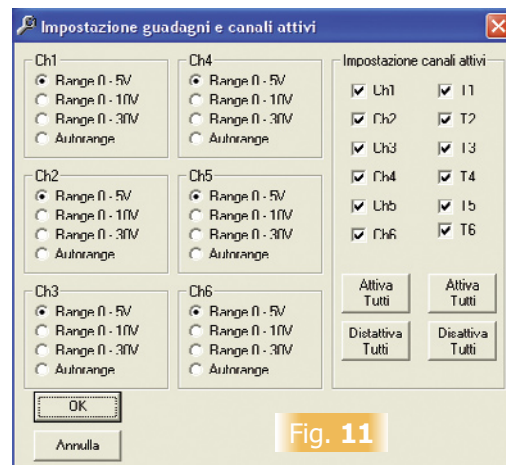


Fig. 11

Porte Com e LPT. Ciò avviene perché il datalogger viene riconosciuto come una seriale virtuale (Virtual Com Port, come da Fig. 8). Annotate su un foglietto di carta il numero della porta COM virtuale associata al dispositivo, poiché vi servirà quando utilizzerete l'interfaccia software.

Ad installazione ultimata, i due LED rosso e giallo del datalogger lampeggiano alternativamente: questo indica che il dispositivo è stato riconosciuto dal sistema e il collegamento USB è attivo.



Fig. 10

Il Software per Windows

Per la gestione remota del datalogger è stato sviluppato un apposito software di acquisizione dati e controllo del dispositivo scritto in Visual Basic 6, utilizzabile sotto ambienti operativi Microsoft Windows.

All'avvio, l'interfaccia software del datalogger si presenta come in Fig. 9. La prima operazione da compiere è la connessione del dispositivo al PC. Collegare dunque il connettore USB ad una qualsiasi porta: il sistema notifica il riconoscimento

dell'hardware con il classico 'tono' di Windows ed i due LED del datalogger iniziano a lampeggiare alternativamente. A questo punto bisogna accedere al menu a tendina *Connessione*, selezionare la porta seriale virtuale associata al dispositivo (quella che avevate annotato durante l'installazione dei driver) e cliccare sulla voce *Connetti*. Se tutto è stato eseguito correttamente la finestra di dialogo del software visualizza il messaggio "Periferica Connessa". Ora l'interfaccia software è connessa al datalogger ed è in grado di inviare e ricevere dati.

Se si sta utilizzando il dispositivo per la prima volta, prima di eseguire una nuova acquisizione, è necessario memorizzare i codici univoci a 64 bit delle sonde DS18B20 all'interno della memoria del microcontrollore, altrimenti quest'ultimo non è in grado di indirizzarle correttamente. L'operazione è possibile utilizzando il comando *Memorizza codici sensori ow*, presente sotto il menu a tendina *Opzioni*: una volta selezionato questo comando, compare la schermata di Fig. 10. Ora non dovete far altro che inserire una alla volta le sonde in uno qualsiasi dei connettori *header* a 3 poli e cliccare di volta in volta sul tasto *Memorizza codice sensore*.

Ad ogni pressione del tasto, il microcontrollore richiede il codice identificativo dell'unico sensore collegato, codice che viene poi sal-

vato sulla sua EEPROM interna.

Una volta terminata l'operazione, si potrà premere il tasto *Fine* per tornare alla schermata principale.

L'operazione appena descritta va eseguita solamente la prima volta che si accende il datalogger: al successivo collegamento il microcontrollore recupererà i codici dei sensori memorizzati in EEPROM in maniera trasparente per l'utente e tali codici rimarranno invariati fino a che non si sostituiranno uno o più sensori. Per cancellare i codici contenuti nella memoria del microcontrollore, è possibile selezionare la voce *Elimina Codici Sensori ow* presente sempre sotto il menu a tendina *Opzioni*.

Prima di iniziare una nuova acquisizione è opportuno selezionare i canali attivi ed i guadagni dei canali analogici. Per farlo, si deve selezionare dal menu *Opzioni* la voce *Impostazione Guadagni e Canali Attivi*.

Fatto ciò, compare la finestra di dialogo riportata in Fig. 11, tramite la quale è possibile selezionare i canali attivi (possono essere selezionati individualmente agendo sui vari *checkbox* oppure attivati e disattivati in blocco agendo sugli appositi pulsanti) ed i range di tensione d'ingresso dei sei canali analogici.

I range possibili sono 0-5V, 0-10V e 0-30V, ma è prevista la possibilità di selezionare la modalità *autorange*, in cui il software va a ricercare



automaticamente la gamma di lettura idonea a garantire la miglior risoluzione possibile del segnale in ingresso. Naturalmente, più è ampio il range di tensione minore è la precisione con cui viene effettuata la misura. È possibile accedere alla finestra *Impostazione Guadagni e Canali Attivi* solo ad acquisizione ferma (con il dispositivo connesso) quindi, se si desidera modificare uno di questi parametri, occorre interrompere l'acquisizione, effettuare i settaggi voluti e riavviare l'acquisizione.

Una volta selezionati i canali attivi e i range di tensione, prima di proseguire con una nuova acquisizione è possibile impostare degli allarmi su tutti i dodici canali d'ingresso. Per farlo, occorre selezionare, sempre

tramite il menu a tendina *Opzioni*, la voce *Imposta Allarmi*. Compare la finestra di Fig. 12 che permette di selezionare sia il canale sul quale si vuole impostare l'allarme sia le due soglie (una superiore ed una inferiore) al di fuori delle quali scatta la segnalazione.

Come per l'impostazione dei canali attivi, possono essere selezionati i singoli *checkbox* oppure si possono utilizzare i pulsanti di selezione rapida. Se durante l'acquisizione un segnale supera una delle due soglie, la barra di segnalazione del livello passa da blu a rosso e, contemporaneamente, viene riprodotto un suono associato all'allarme, che può essere modificato a piacere tramite la finestra di dialogo che appare cliccando *Seleziona suono allarme* dalla scher-

mata *Imposta Allarmi* (Fig. 12).

A questo punto il datalogger è pronto per eseguire una nuova acquisizione. Potete dunque collegare tutte le sonde ed i canali analogici che si desidera acquisire quindi premere il tasto *Start Acquisizione*, posizionato, insieme agli altri pulsanti di controllo acquisizione, sulla destra della finestra principale. Il contatore dei cicli di acquisizione inizia ad incrementarsi, ed il risultato dell'acquisizione dei canali attivi viene visualizzato dai vari indicatori. Per completare l'acquisizione corrente premere il tasto *Stop Acquisizione*. Se si desidera salvare su file il risultato dell'acquisizione, occorre selezionare dal menu a tendina *File* la voce *Salva su File*.

Compare quindi la finestra di dialogo di Fig. 13, che permette di scegliere sia il nome del file che la cartella dove effettuare il salvataggio. I dati vengono memorizzati su un file di testo *notepad* (*prova.txt*, *log.txt*...) secondo il formato

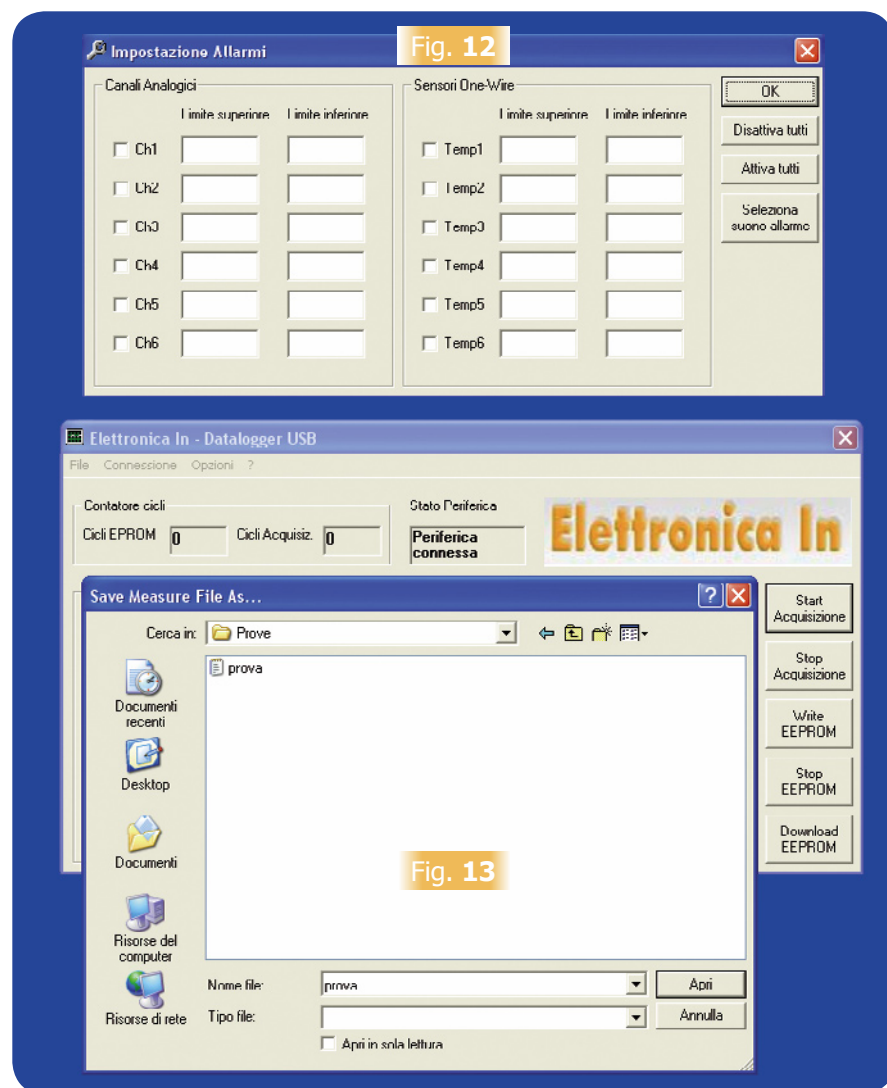
<data-ora>,<CH1>,<CH2>,<CH3>,<CH4>,<CH5>,<CH6>,<Temp1>,<Temp2>,<Temp3>,<Temp4>,<Temp5>,<Temp6><CR-LF>

Notate che ogni riga inizia con la data e l'ora, termina con CR-LF e contiene i valori dei dodici canali, separati tutti da una virgola.

Tale formato è facilmente esportabile su un foglio di calcolo come, ad esempio, *Microsoft Excel* (vedere Fig. 15). Il datalogger prevede anche la possibilità di funzionare in modalità autonoma, memorizzando i dati sulla EEPROM 24LC256 montata a bordo della scheda. Per utilizzare questa funzionalità, si deve premere il pulsante *Write EEPROM*.

Il contatore dei cicli di scrittura su EEPROM inizia, quindi, ad incrementarsi, indicando il salvataggio dei dati in memoria.

Per completare il salvataggio dei dati premere il pulsante *Stop EEPROM*. A questo punto, per scaricare su file il contenuto della EEPROM, occor-



re selezionare prima il percorso del file (come nel caso dell'acquisizione *real-time*) e poi premere il pulsante *Download EEPROM*.

Una barra di stato segnala la percentuale di memoria scaricata. Ad operazione ultimata, il file di testo conterrà i dati scaricati dalla EEPROM. Il software fornisce anche la possibilità di modificare la frequenza di campionamento, sia per quanto riguarda il salvataggio su file che per il salvataggio sulla EEPROM interna. Per farlo, selezionare, all'interno del menu *Opzioni*, la voce *Opzioni salvataggio dati*: compare la finestra di Fig. 14 dalla quale è possibile modificare le frequenze di salvataggio dei dati. Si tenga in considerazione il fatto che i sensori di temperatura sono dispositivi piuttosto lenti: essi,

ze di campionamento al di sotto di 1 campione ogni 5 secondi valgono solo quando sono attivi i soli canali analogici.

Importare in Excel il file di log

Il log file, normalmente, serve per effettuare un'analisi dei dati acquisiti, analisi dalla quale si possono generare grafici dell'andamento dei parametri misurati, valori medi e statistici di vario tipo, eccetera, utili per eventuali operazioni di controllo e "post-processing". Esistono molti metodi e molti programmi che possono essere usati per tale scopo, ma sicuramente, l'applicativo più diffuso è Excel. Per importare il log file in un foglio di calcolo XLS, è sufficiente avviare il programma Excel;

Windows: nella prima finestra occorre specificare che i campi sono delimitati da un separatore (clic su *Avanti*) e nella seconda, invece, che il separatore è la tabulazione (TAB). Si termina la procedura facendo clic su *Fine*, ottenendo la schermata visibile in Fig. 15.

Note conclusive

A questo punto sono stati esaminati tutti i vari aspetti del funzionamento ed i dettagli costruttivi del nostro datalogger USB, dispositivo che, sebbene non possa essere paragonato per efficienza e prestazioni ad una moderna scheda di acquisizione dati industriale, costituisce comunque una valida ed economica alternativa a tale categoria di prodotti, nel caso in cui l'applicazione finale non

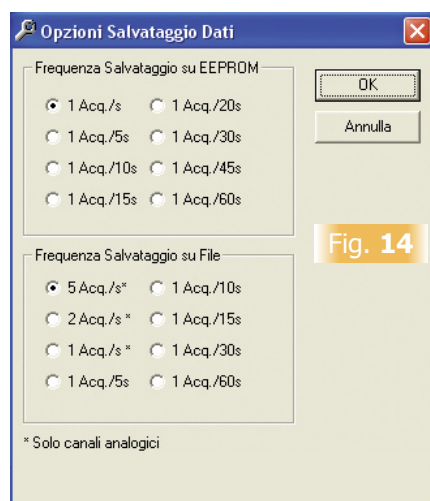


Fig. 14

	A	B	C	D	E	F	G	H	I	J	K	L	M
	Ref.	Ch1	Ch2	Ch3	Ch4	Ch5	Ch6	Temp1	Temp2	Temp3	Temp4	Temp5	Temp6
2	2008/02/22 23:55:00	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,6°	21,5°	20,6°	20,4°	20,4°	20,8°
3	2008/02/22 23:55:15	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,6°	21,5°	20,6°	20,4°	20,4°	20,8°
4	2008/02/22 23:55:30	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,5°	21,4°	20,6°	20,4°	20,4°	20,8°
5	2008/02/22 23:55:45	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,5°	21,4°	20,5°	20,4°	20,4°	20,8°
6	2008/02/22 23:56:00	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,4°	21,4°	20,5°	20,4°	20,4°	20,7°
7	2008/02/22 23:56:15	4,31V	4,35V	4,31V	4,31V	4,26V	4,31V	20,4°	21,4°	20,5°	20,3°	20,4°	20,6°
8	2008/02/22 23:56:30	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,4°	21,4°	20,5°	20,3°	20,4°	20,6°
9	2008/02/22 23:56:45	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,4°	21,4°	20,4°	20,3°	20,3°	20,6°
10	2008/02/22 23:57:00	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,3°	21,4°	20,4°	20,3°	20,3°	20,6°
11	2008/02/22 23:57:15	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,3°	21,3°	20,4°	20,3°	20,3°	20,6°
12	2008/02/22 23:57:30	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,3°	21,3°	20,4°	20,3°	20,3°	20,6°
13	2008/02/22 23:57:45	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,3°	21,3°	20,4°	20,3°	20,3°	20,5°
14	2008/02/22 23:58:00	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,3°	21,3°	20,4°	20,3°	20,3°	20,5°
15	2008/02/22 23:58:15	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,3°	21,3°	20,4°	20,3°	20,3°	20,5°
16	2008/02/22 23:58:30	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,3°	21,3°	20,4°	20,3°	20,3°	20,5°
17	2008/02/22 23:58:45	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,3°	21,2°	20,3°	20,3°	20,3°	20,4°
18	2008/02/22 23:59:00	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,2°	21,2°	20,3°	20,3°	20,3°	20,5°
19	2008/02/22 23:59:15	4,31V	4,35V	4,31V	4,31V	4,26V	4,31V	20,2°	21,1°	20,3°	20,3°	20,3°	20,5°
20	2008/02/22 23:59:30	4,31V	4,35V	4,31V	4,31V	4,31V	4,31V	20,2°	21,1°	20,3°	20,2°	20,3°	20,5°

infatti, necessitano di un tempo di acquisizione di 750 ms per acquisizioni a 12 bit, e quindi le frequen-

za da menu File/Apri, quindi, si seleziona il log file (l'esempio si chiama *prova.txt*). Appare ora un wizard

abbia specifiche troppo rigide sulla frequenza di campionamento e non necessiti di più di dodici canali.

per il

MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Il master del circuito stampato, il firmware col quale programmare il microcontrollore ed il software da caricare sul PC possono essere scaricati gratuitamente dal sito della rivista (www.elettronica.in). Il microcontrollore è disponibile anche già programmato (cod. MF617) al prezzo di 20,00 Euro.

Il materiale va richiesto a: Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)

Tel: 0331-799775 ~ Fax: 0331-792287 ~ <http://www.futurashop.it>

Progettare lo stadio di alimentazione

di Matteo Bosio

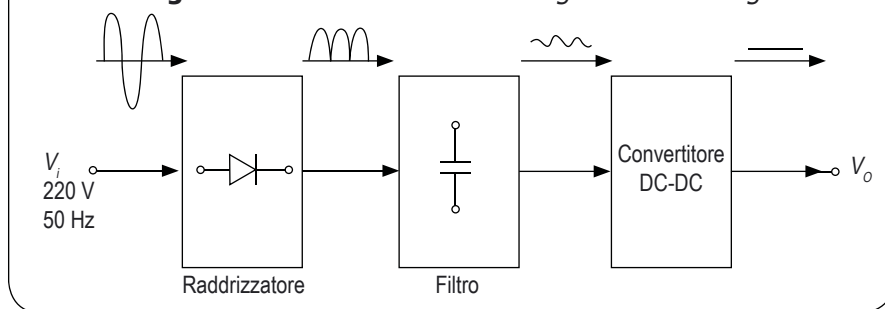
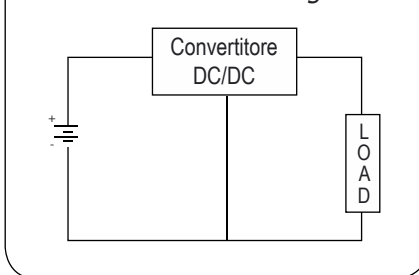
Impariamo a progettare e realizzare lo stadio di alimentazione di un circuito elettronico, analizzando le esigenze in termini di tensione, corrente, dissipazione e sistemi di raffreddamento. In questo articolo, approfondiamo la conoscenza degli alimentatori a commutazione e impariamo a dimensionare correttamente i dissipatori. Ultima puntata.

Rieccoci a parlare di alimentatori, dando seguito a quanto visto nella puntata precedente in cui, partendo dalle basi della conversione da corrente alternata a corrente continua, abbiamo approfondito soluzioni circuitali lineari di efficienza e prestazioni via via crescenti. Abbiamo detto lineari in quanto tutti i circuiti visti facevano uso di un dispositivo che regola e stabilizza la tensione in uscita dissipando l'energia in eccesso tramite un elemento attivo (mosfet o transistor) fatto lavorare in zona lineare. Nel corso di questo articolo conclusivo, invece, conosciamo meglio gli stabilizzatori di tensione a

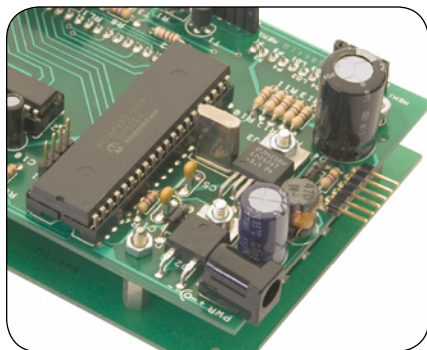
commutazione e terminiamo il discorso con un paragrafo dedicato alle procedure per dimensionare correttamente i dissipatori di calore.

Alimentatori stabilizzati switching

Gli alimentatori stabilizzati switching si prefiggono l'obiettivo di ottimizzare il rendimento dei circuiti utilizzando, al posto di regolatori lineari a dissipazione, dei regolatori a commutazione. Questi ultimi, infatti, lavorando come interruttori veloci invece che come resistenze variabili, permettono di migliorare il rendimento dell'ali- ➤

Fig. 21 - Alimentatori da rete con regolatore switching.**Fig. 22 - Alimentazione a batteria con convertitore switching.**

mentatore, riducendo contemporaneamente temperature di esercizio e consumi: in questo modo, apparati alimentati a batteria possono funzionare più a lungo. Gli alimentatori switching sono una classe particolare dei dispositivi chiamati *DC-DC converter*, termine inglese che identifica una vasta gamma di circuiti in grado di trasformare una tensione continua in un'altra sempre continua ma con parametri diversi; quindi, parlando idealmente, un DC-DC converter può convertire una tensione di 10 V a 1 A (potenza 10 W) in una di 5 V a 2 A (sempre 10 W). In pratica, però, anche può raggiungere un valore elevato, il rendimento non può mai raggiungere il 100%. Esistono, comunque, particolari soluzioni circuitali che garantiscono rendimenti del 95% (e oltre), equivalenti, nel nostro esempio, ad un trasferimento di 9,5 W partendo da 10 W. Pensate che se avessimo utilizzato un regolatore lineare, avremmo potuto solamente abbassare la tensione senza aumentarne la corrente, e quindi avremmo ottenuto in uscita 5 V a 1 A, con un rendimento reale solamente del 50 %.



Quando vengono utilizzati nei dispositivi a tensione di rete (Fig. 21), i DC-DC converter sono spesso inseriti direttamente dopo il gruppo “ponte raddrizzatore (che raddrizza direttamente la tensione di rete) e condensatore di filtro”. Vengono, inoltre, utilizzate delle particolari soluzioni che uniscono, ai pregi sopracitati, l'isolamento galvanico. Nei dispositivi portatili, il convertitore è collegato direttamente alla batteria (Figura 22): questo è l'esempio più tipico di applicazione pratica. Quale che sia l'architettura impiegata negli alimentatori, bisogna sempre tenere conto del fattore rendimento, il cui valore influenza in modo inversamente proporzionale la temperatura dell'elettronica: è importante, infatti, che i sistemi elettronici generino meno calore possibile. Ecco quindi, che i convertitori switching offrono una via sicura per raggiungere questo obiettivo. I moderni switching (sia in termini di chip che di prodotti finiti), infatti, sono piccoli, flessibili ed offrono rendimenti spesso molto elevati (in alcuni casi superano il 98%). Anche questi circuiti, però, hanno delle limitazioni e degli svantaggi rispetto ai regolatori lineari: innanzitutto sono circuitalmente più complessi e, pertanto, meno economici dei lineari; essi inoltre, lavorando a commutazione con elementi reattivi (condensatori ed induttanze), possono generare (se non progettati correttamente) dei disturbi radio elettromagnetici spesso talmente elevati da disturbare apparati adiacenti (in gergo, si parla di problemi

di *Compatibilità Elettromagnetica*). Il ripple residuo in uscita, inoltre, è tipicamente più elevato di quello presentato dai lineari.

Comunque sia, ogni qual volta si richieda un alto rendimento, spazi ridotti (per quanto possibile!), peso minimo o alta potenza, i regolatori switching non temono confronti: basti pensare che quando con potenze di centinaia di watt, i regolatori lineari necessiterebbero di dissipatori enormi e talvolta anche di ventole. Non dimentichiamo, comunque, il grande vantaggio di poter trasformare la tensione d'ingresso ad un qualunque valore desiderato in uscita e questo, in prima approssimazione, senza limitazioni di efficienza e indipendentemente dal fatto che l'uscita sia maggiore o minore dell'ingresso: sono, in pratica, dei veri e propri trasformatori in continua. Abbiamo detto, nel corso di questa breve introduzione, che nei DC-DC converter esiste un elemento (talvolta più d'uno) che lavora come interruttore veloce, ma non abbiamo ancora detto che può essere utilizzato in modi e circuiti differenti: vediamoli uno per uno.

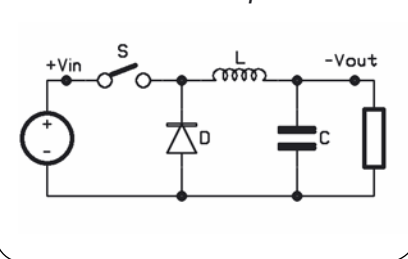
Fig. 23 - Convertitore switching buck o step-down.

Fig. 24 - Convertitore switching boost o step-up.

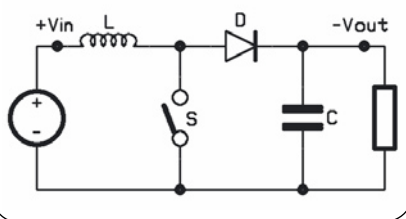


Fig. 26 - Convertitore switching invertente.

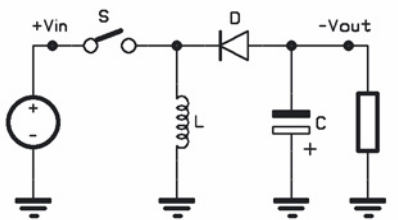
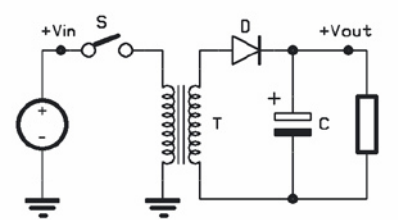


Fig. 27 - Convertitore switching Flyback (boost).



Buck o step-down

In questa tipologia di circuiti switching, la tensione di uscita può essere solo più bassa di quella d'ingresso. Come si può notare in Fig. 23, la sola presenza di componenti reattivi e di un interruttore con caratteristiche quasi ideali rende l'efficienza elevata. L'interruttore S (switch) viene comandato in modo On/Off con un segnale tipo PWM (Pulse Width Modulation). Con l'acronimo PWM (usato frequentemente da questo momento in poi) si intende un segnale di tipo digitale (0-1) a frequenza costante, ma di *duty-cycle* (rapporto tra la durata del livello logico 1 e il periodo del segnale) variabile. Quando l'interruttore S è chiuso la corrente in ingresso fluisce verso l'uscita tramite l'induttanza, che nel frattempo immagazzina energia magnetica. Quando l'interruttore S è aperto la corrente non si annulla immediatamente, ma continua a fluire grazie all'energia immagazzinata nell'induttore, sfruttando il diodo D che va in conduzione; il condensatore C riduce le fluttuazioni di tensioni uscita. Se le operazioni di chiusura

e apertura di S sono sufficientemente veloci la corrente sull'induttanza non si annulla mai. Un regolatore vero e proprio è completato da una rete di retroazione che controlla il processo agendo sui tempi di apertura e chiusura dell'interruttore (di solito un mosfet), ossia agendo sul *duty-cycle* del segnale di PWM.

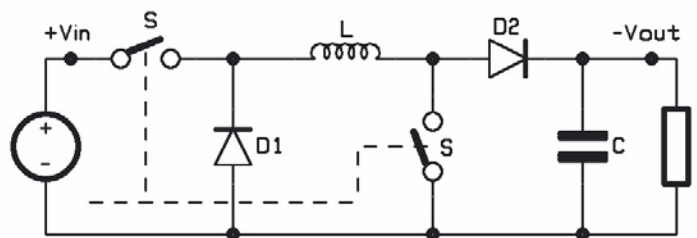
Boost o step-up

In questi switching (Fig. 24), la tensione di uscita è sempre maggiore di quella d'ingresso. L'interruttore S (switch) viene comandato in On/Off con un segnale tipo PWM. Ad ogni chiusura dell'interruttore, l'induttanza immagazzina energia magnetica, che viene rilasciata durante la fase di Off (interruttore aperto) tramite il diodo D verso il condensatore C. Data la brusca variazione di corrente sull'induttanza ai suoi capi si genera una tensione indotta di elevato valore, che carica il condensatore in uscita (C) ad una tensione maggiore di quella in ingresso.

Buck-boost

Questo tipo di convertitore a commutazione, schematizzato in Fig. 25,

Fig. 25
Convertitore switching buck-boost o step-up/step-down.



combina le caratteristiche viste nei due circuiti precedenti. Esso nasce dall'esigenza di poter prelevare una tensione in uscita regolata e stabilizzata ad un valore che può essere sia minore che maggiore di quello della tensione di alimentazione. Lo schema è ricavato unendo un convertitore tipo *buck* e uno di tipo *boost*. L'interruttore S (a doppio contatto) viene comandato, come già visto, con un segnale On/Off di tipo PWM, il cui *duty-cycle* definisce la tensione in uscita.

Invertente

Questa tipologia di circuiti switching (Fig. 26) permette di ottenere una tensione in uscita invertita rispetto a quella di ingresso. Quando il commutatore S viene chiuso, V_{in} si localizza ai capi dell'induttanza L, in cui fluisce una rampa di corrente lineare tramite la quale l'induttanza immagazzina energia magnetica. Quando l'interruttore viene aperto, la corrente continua a scorrere con il medesimo verso (per effetto dell'autoinduzione indotta dall'energia magnetica accumulata da L) e, attraverso il diodo, va a caricare C creando la tensione V_{out} con una polarità opposta rispetto l'ingresso. Si notino il verso dell'anodo del diodo e, di conseguenza, la polarità del condensatore (positivo a massa).

Flyback

Schematizzata in Fig. 27, questa soluzione permette di ottenere un isolamento galvanico tra ingresso e uscita. Utilizzando un trasformatore sulla ➤

rete di ingresso, pilotato dal commutatore, è possibile realizzare le tipologie Boost (Fig. 24), Buck (Fig. 23), o Invertente (Fig. 26) descritte poco sopra ed ottenere contemporaneamente un netto isolamento tra i due lati del convertitore.

Questa struttura garantisce, pertanto, una sicurezza intrinseca maggiore ed è, quindi, da preferire in tutte le applicazioni ad alta tensione quali gli AC-DC converter (DC-DC converter a tensione di rete).

Retroazione

Un regolatore switching è composto anche da una sezione di retroazione il cui compito è quello di regolare la tensione in uscita, per mantenerla stabile al variare del carico. La tecnica di controllo più diffusa, come già più volte detto, è quella del controllo in PWM. In pratica, viene realizzata secondo lo schema di principio di Fig. 28: la tensione in uscita viene riportata (opportunamente attenuata, se necessario, dal partitore R1-R2) all'ingresso di un circuito di controllo che attiva e disattiva il mosfet SW (usato come interruttore) a frequenza di commutazione costante e duty-cycle variabile. Se il carico aumenta, la tensione V_{out} tende a diminuire, riducendo, di conseguenza la tensione di reazione al circuito di controllo che, in questo caso aumenta il tempo di chiusura del mosfet (in pratica viene aumentato il duty-cycle). Analogamente, se il carico diminuisce, V_{out} tende ad aumentare così come

Fig. 28 - Regolatore switching retroazionato controllato in PWM.

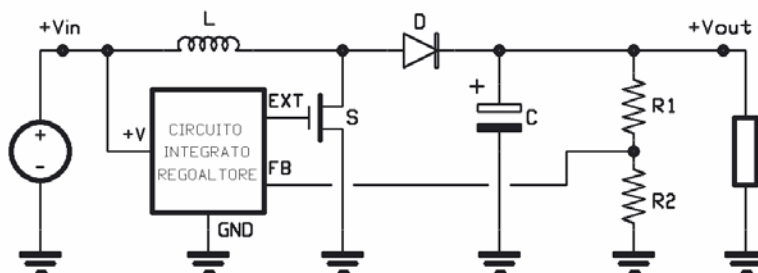


Fig. 29 - Circuito con LT1934ES6.

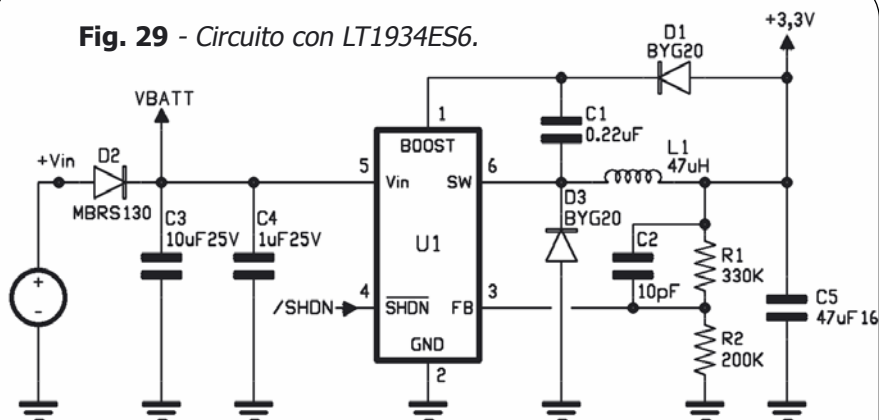
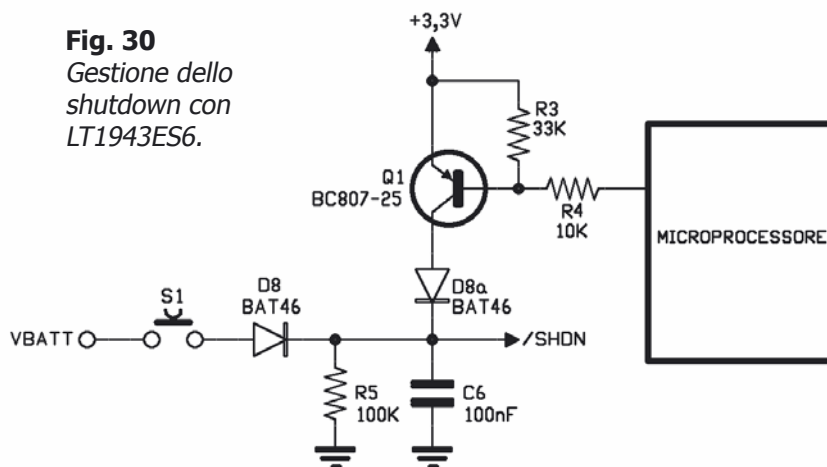


Fig. 30
Gestione dello
shutdown con
LT1934ES6.



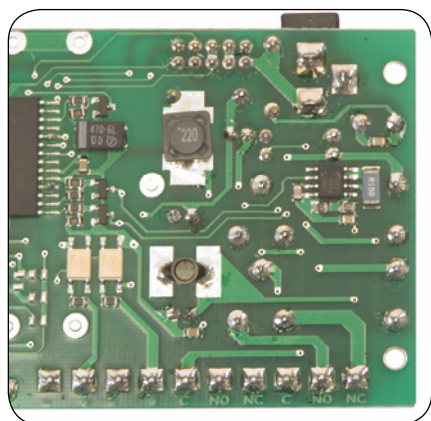
il segnale di retroazione: il generatore PWM, quindi, riduce il duty-cycle. Vediamo ora tre esempi pratici di circuiti di alimentazione switching, realizzati con circuiti integrati reperibili in commercio.

Regolatore LT1934ES6 (step-down)

L'integrato LT1934ES6 è un DC-DC converter di tipo step-down, che contiene un commutatore da 400

mA; esso, pertanto, non necessita di alcun mosfet esterno. È ideale per applicazioni a batteria, in quanto ha una *Quiescent Current* (corrente a riposo) di 12 μ A; dispone, inoltre, della funzione di *shutdown*, che permette, all'occorrenza, di scollegare il carico dalla sorgente (tipicamente la batteria), semplificando ed ottimizzando la gestione del risparmio di energia.

In Fig. 29 è mostrata una tipica ap-



plicazione del chip LT1934ES6, in cui la tensione proveniente da una batteria a 12 V viene abbassata e stabilizzata al valore di 3,3 V con una corrente massima di 200 mA. Come potete constatare, il circuito è più complesso di quello che si avrebbe utilizzando un classico regolatore lineare (ad esempio l'LM317 visto nella puntata precedente): esso infatti include un'induttanza, due diodi veloci (D1 e D3, più altri componenti discreti quali diodi, resistenze e condensatori. Vediamo le principali caratteristiche di questo integrato:

- Tensione d'ingresso: 3,2÷34 Vdc,
- Corrente a riposo: 12 µA,
- Corrente di shutdown: <1µA,

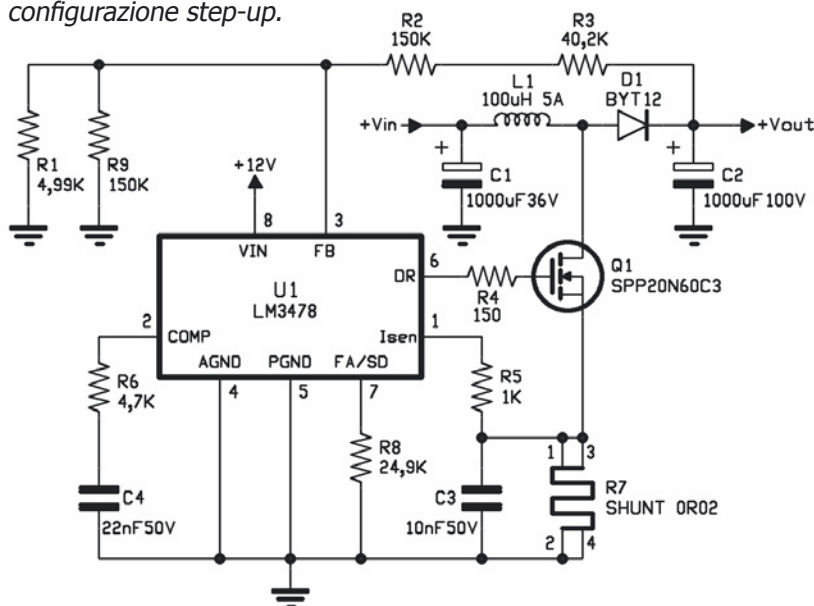
lotare il transistor NPN interno.

- GND: massa.
- FB: feedback riferito a 1,25 V. La tensione in uscita può essere calcolata con la formula:
 $V_{out} = 1,25 \times (1 + R1 / R2)$
- SHDN: pin di shutdown (spegnimento); se collegato a massa, fa sì che il chip disalimenti il carico.
- VIN: tensione di ingresso.
- SW: è l'uscita dello switch interno di potenza, che pilota un capo dell'induttanza.

Il pin di shutdown è largamente utilizzato in pratica, perché grazie ad esso e a un minimo di logica esterna, è possibile realizzare dispositivi che si accendano alla pressione di un pulsante e che si spengano su co-

tuazione iniziale in cui il sistema è spento. Premendo il tasto S1 si porta al pin SHDN la tensione di batteria che, se di valore superiore a 2,3 V, dà il consenso all'accensione del regolatore che, pertanto, alimenta il micro ed il resto del carico. In seguito il tasto viene rilasciato, ma il software del microcontrollore (che nel frattempo è partito) obbliga un pin di I/O del micro a polarizzare la base del transistor Q1 il quale, pertanto, forza il pin SHDN a livello logico alto. Ipotizzando di avere un'applicazione portatile a batteria, si può fare in modo che, dopo un certo periodo di inattività sui tasti, il microcontrollore decida di spegnere il dispositivo, facendo interdire il transistor Q1 e mandando a zero la linea SHDN. In questo modalità si ha un consumo irrisorio, inferiore a 1 µA.

Fig. 31 - LM3478 in configurazione step-up.



- Vce(sat) switch interno: 200 mV,
- Corrente massima: 250 mA.

Quanto appena visto fa del regolatore LT1934ES6 un dispositivo molto versatile sia per le dimensioni contenute che per i possibili impieghi, specialmente in dispositivi portatili alimentati a batteria.

Le funzioni dei sei pin sono:

- BOOST: alimentato tramite un diodo veloce dalla tensione in uscita, fornisce la tensione per pi-

mando del microcontrollore interno, come avviene ad esempio nei lettori MP3 e in molte macchine fotografiche digitali compatte, eccetera.

In Fig. 30 abbiamo riportato lo schema di un controllo di accensione e spegnimento digitale di massima, utilizzabile con il nostro chip LT1934ES6 e con altri dispositivi che abbiano lo shutdown attivo a massa.

Supponiamo di partire da una si-

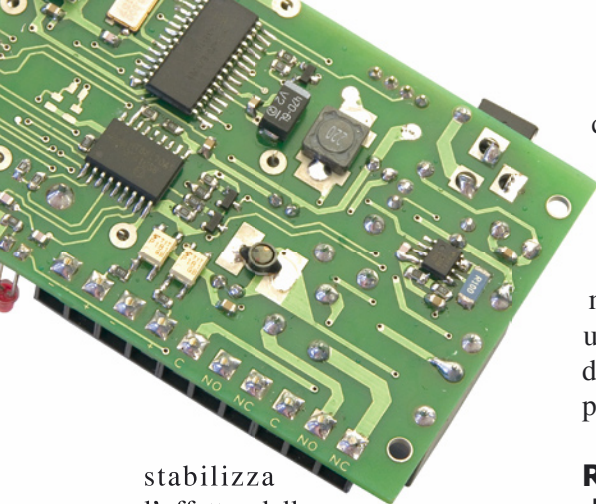
Regolatore LM3478 in modalità step-up

Questo tipo di regolatore switching, impiegato nello schema di Fig. 31, può essere utilizzato in tutte le modalità viste ossia buck, boost e fly-back. Vediamo le sue caratteristiche:

- tensione di ingresso: 2,97÷40 Vdc;
- corrente di shutdown: 10 µA;
- frequenza di commutazione: da 360 a 430 kHz in funzione della resistenza applicata tra il pin FA/SD e la massa (R8);
- controllo della temperatura e protezione dal cortocircuito in uscita integrati;
- protezione sulle sovra-tensioni.

L'integrato dispone di 8 pin:

- ISEN: ingresso del sensore di corrente; rileva la tensione generata con una resistenza di shunt esterna da 0,025 ohm, realizzata con una pista del c.s. opportunamente dimensionata o con una resistenza di precisione.
- COMP: pin di compensazione che, grazie ad un circuito RC, ➤



stabilizza
l'effetto della
retroazione.

- FB: pin di feedback, sul quale deve essere sempre applicata una tensione di 1,26 V, ricavata dalla tensione di uscita tramite un partitore di tensione; la formula per il calcolo della tensione in uscita è

$$V_{out} = 1,25 \times (1 + R_a / R_b)$$

dove

$$R_a = R_2 + R_3$$

$$R_b = R_1 // R_9$$

- AGND: massa analogica.
- PGND: massa di potenza.
- DR: pilota il gate del mosfet di potenza esterno.
- FA/SD: Frequency Adjust / Shutdown; collegato a massa tramite una resistenza di valore opportuno (ricavabile dal datasheet), imposta la frequenza di commutazione. Collegato direttamente a massa, spegne il chip.
- VIN: tensione di ingresso.

Vediamo ora l'esempio pratico di Fig. 31, in cui vogliamo ottenere una tensione d'uscita di 50 V, partendo da un ingresso di tensione variabile tra 2,97 V e 40 V.

Applicando la formula già vista nel nostro circuito otteniamo:

$$\begin{aligned} V_{out} &= \\ &= 1,26 \times [1 + (R_2 + R_3) / (R_1 // R_9)] = \\ &= 1,26 \times [1 + (150 + 40,2) / (4,99 // 150)] = \\ &= 1,26 \times [1 + 190,2 / 4,83] = \\ &= 1,26 \times [1 + 39,4] = \\ &= 1,26 \times 40,4 = 50,9 \text{ V} \end{aligned}$$

La resistenza di shunt R7 serve per misurare la corrente che passa nell'interruttore (Q1) un mosfet a canale N modello SPP20N60. Anche questo esempio evidenzia che la

complessità circuitale di uno stadio di alimentazione switching è superiore rispetto a quella di un circuito lineare.

Con un regolatore lineare, però, non è possibile prelevare tensioni in uscita superiori a quelle in ingresso, dato che esso funziona solamente per "caduta di tensione".

Regolatori buck sincroni della famiglia LM20k PowerWise®

La nuova famiglia LM20k (National Semiconductor) di regolatori buck sincroni di tipo PowerWise è costituita da dispositivi in grado di fornire un'alta efficienza, con un'ampia gamma di opzioni che consentono al progettista di scegliere l'integrato più adatto alle proprie esigenze. Questi dispositivi nascono per soddisfare un'esigenza particolarmente diffusa nei circuiti elettronici com-

plessi, ossia quella di alimentare diversi carichi (leggasi "sezioni di circuito") che necessitano di tensioni e correnti differenti. Una combinazione adeguata degli LM20k (dove k significa "kilo") permette di ottimizzare il circuito di alimentazione sia in termini di dimensioni che di efficienza. Tutti i dispositivi LM201xx (formato TSSOP-16) accettano una tensione di ingresso di $2,95 \div 5,5$ Vdc, con correnti fino a 5 A; la versione LM20242 (formato TSSOP 20), invece, accetta tensioni di ingresso di $4,5 \div 36$ Vdc, con correnti fino a 2 A. Ecco un elenco delle principali caratteristiche di questa famiglia di DC-DC converter:

- Ingresso "Precision enable", che può essere usato per attivare o disattivare l'uscita.
- Uscita "Power good", che indica che la tensione di uscita è regolata con una tolleranza accettabile; può

Fig. 32 - LM20k utilizzati in un alimentatore a tensioni multiple.

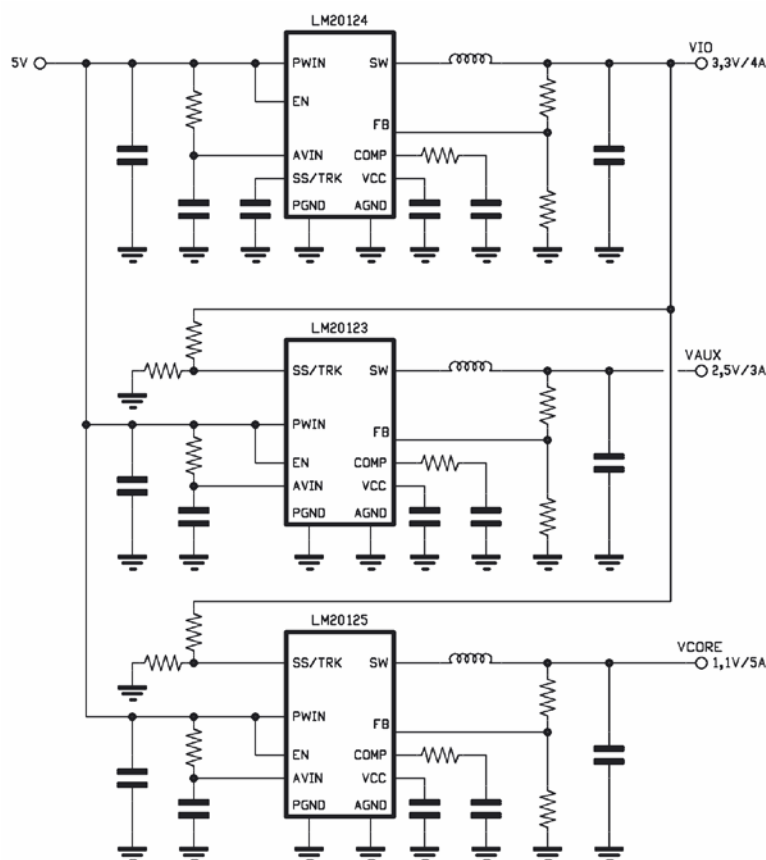


Fig. 33 - I parametri dell'alimentatore vengono inseriti on-line nel form del tool Webench.

WEBENCH Live Simulation!

LM20123 WEBENCH® Custom Design/Analyze/Build It/Test It

V in Lower **2.95** V **<=** V in Upper **5.00** V **<=** 5.50

V out **3.3** V **<=** 4.00

I out **0.5** A **<=** 3.00

Ambient Temperature **20** °C **<=** 100

Start Your Design

What are WEBENCH® tools?

Fig. 34 - Cliccando sul pulsante verde "Start Your Design", l'applicativo chiede il vostro indirizzo e-mail e password (forniti durante la registrazione) quindi avvia il dimensionamento del circuito. Dopo qualche istante, compare la tabella che riporta i componenti calcolati dal programma.

Part	Manufacturer	Part#	Attributes	Top View
Cavin	TDK	C2012K7R1E105K	1 uF	Select Alternate Part
Ccomp	AVX	08055A152JAT2A	0.0015 uF	Select Alternate Part
Cext	TDK	C2012K7R1E105K	1 uF	Select Alternate Part
Cin	TDK	CS750XS81C476M	47 uF, 0.004 Ohms	Select Alternate Part
Cout	HuRate	GRM31CR71A475K	47 uF, 0.009 Ohms	Select Alternate Part
Css	TDK	C2012C0G1E562J	0.0056 uF	Select Alternate Part
L1	Coiltronics	DR73-1R0-R	1 uH, 0.0102 Ohms	Select Alternate Part
Ravin	Panasonic	ERJ-6RQF1R0V	1 Ohms	Select Alternate Part
Rcomp	Vishey-Dale	CRCW0805768RFEA	768 Ohms	Select Alternate Part
Rfb1	Panasonic	ERJ-6ENF1002V	10000 Ohms	Select Alternate Part
Rfb2	Panasonic	ERJ-6ENF3092V	30900 Ohms	Select Alternate Part
U1	National Semiconductor	LM20123MH		Select Alternate Part

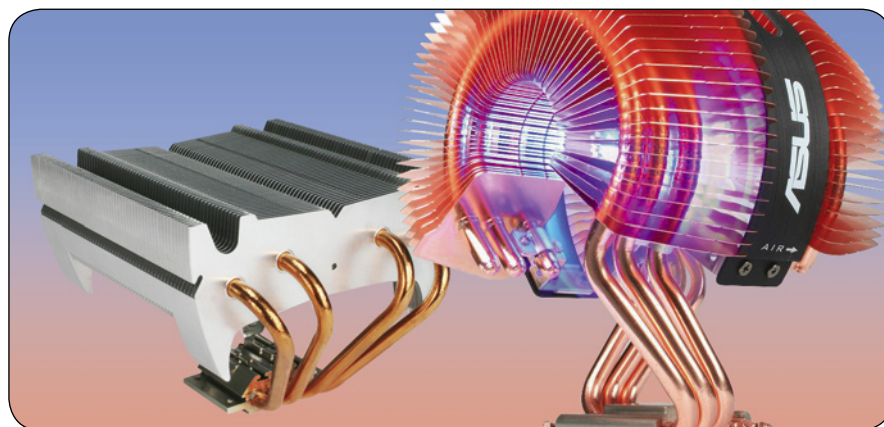
Quick Search Parametric Search See Our Disclaimer Product Tree Back To Webench

essere collegato al pin "Precision input" di un dispositivo in cascata, che deve essere acceso solo dopo il precedente.

- **Ingresso "Tracking"**, per controllare la tensione di uscita.
- **Uscita "Soft Start"**, che fornisce una rampa di tensione controllata, per proteggere i carichi da picchi di corrente in ingresso.
- **Ingresso "Sync-in"**, che permette la sincronizzazione con un dispositivo esterno.
- **Uscita "Sync-out"**: fornisce un clock sfasato di 180° che può essere usato per il current-sharing (stabilizzatori che lavorano in parallelo per aumentare la corrente massima in uscita) o per operazioni multi-fase (stabilizzatori indipendenti che generano tensioni in uscita differenti, ma ottenute con segnali di commutazione in fase tra loro).
- Possibilità di correggere la frequenza di lavoro.
- **Protezione** contro sovra-correnti, sovra-tensioni e sotto-tensioni, per aumentare l'affidabilità del sistema. Questi integrati sono forniti in formato TSSOP, con un'area di dissipazione nella parte inferiore del chip, che deve essere saldata al PCB. La famiglia è stata concepita per essere utilizzata in architetture con canali di alimentazione multipli, nella modalità riportata nella Fig. 32. Prendiamo adesso in considerazione un particolare integrato di questa

famiglia, l'LM20123, e vediamo di realizzare un piccolo stabilizzatore in grado di ricavare, a partire da una serie di 3 batterie di tipo AA da 1,5 V, una tensione di 3,3 V con correnti di 500 mA, per alimentare un microprocessore e qualche periferica collegata. L'integrato è un regolatore buck a

Inserendo le specifiche del proprio progetto nel form on-line (Fig. 33), si ottiene come risultato l'elenco dei componenti da utilizzare (Fig. 34). Nel nostro caso è stata inserita una tensione minima di 3,9 V (si considerano tre batterie da 1,5 V scariche a 1,3 V) e una massima di 5 V, con corrente di 500 mA.



1,5 MHz, di tipo sincrono, in grado di fornire fino a 3 A in continua. È ottimizzato per funzionare con tensioni di ingresso di 2,95÷5,5 V, ha un'efficienza del 96% (a 1,5 MHz) e viene supportato da un eccellente sistema di dimensionamento dei componenti: stiamo parlando dell'"ONLINE WEBENCH", cui National Semiconductor dà accesso a seguito di una registrazione (gratuita) su www.national.com. Per lavorare con l'LM20123 è necessario andare alla pagina www.national.com/pf/LM/LM20123.html.

Dimensionamento dei dissipatori

Quale che sia l'architettura scelta (lineare o switching), nella progettazione di un alimentatore bisogna tenere conto, oltre che dei parametri elettrici, anche della dissipazione degli elementi di potenza quali i transistor/mosfet degli switching o dell'integrato lineare in uso, che si scaldano proporzionalmente alla corrente che devono erogare. Occorre pertanto intervenire al fine di evitare che il componente si surriscaldi fino alla distruzione, cosa che ➤

avviene più frequentemente negli alimentatori lineari. Per far questo è necessario dimensionare un dissipatore da montare sul componente in questione, così da agevolare la dispersione del calore.

Prima di iniziare la fase di dimensionamento dei dissipatori vera e propria, è necessario conoscere almeno le basi del fenomeno fisico della dissipazione termica, applicato ai componenti elettronici.

Nei componenti passivi (resistori) e a semiconduttore (sia discreti che integrati) il passaggio della corrente elettrica provoca - per effetto Joule - un innalzamento della temperatura dei dispositivi, che può essere calcolato attraverso la relazione

$$(1) T_w - T_a = P_d \times R_{th}$$

dove

- T_w è la temperatura raggiunta dal dispositivo per effetto della dissipazione,
- P_d è la potenza applicata,
- T_a è la temperatura dell'ambiente circostante,
- R_{th} è una costante, chiamata *resistenza termica*, che rappresenta la difficoltà di smaltimento del calore verso l'ambiente circostante; è equivalente all'inverso della *conduttanza termica* dei materiali e dei fluidi che smaltiscono il calore prodotto e si misura in grado/watt ($^{\circ}\text{C}/\text{W}$).

Poiché il costruttore di qualsiasi componente elettronico riporta sul datasheet relativo il valore della temperatura massima di funzionamento T_{max} , è possibile calcolare la massima potenza P_{max} dissipabile dal dispositivo stesso ad una temperatura ambiente $T_{a_{max}}$ sfruttando la relazione già vista, riscritta come segue:

$$(2) T_{max} - T_{a_{max}} = P_{max} \times R_{th}$$

che spiega, inoltre, perché, una volta raggiunta la massima temperatura specificata, un'ulteriore incremento della potenza applicata porterebbe ad un ulteriore aumento della temperatura, con conseguente possibile

danneggiamento del dispositivo.

È chiaro, quindi, che i limiti di potenza specificati per un certo dispositivo dipendono strettamente sia dai limiti di temperatura cui può funzionare sia dalle condizioni in cui viene fatto funzionare.

Visti i limiti di temperatura - e quindi di potenza - imposti dal costruttore e obbligatoriamente da rispettare per la "salvaguardia" del dispositivo, come è possibile smaltire adeguatamente il calore prodotto durante il funzionamento?

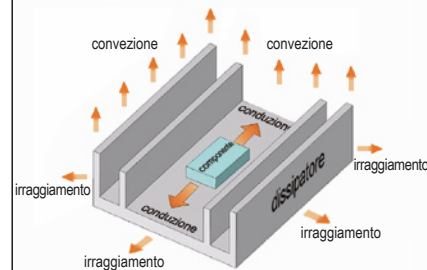
La risposta è semplice: mediante un dissipatore di calore, ossia un elemento che sia un buon conduttore di calore (tipicamente un metallo o un materiale ceramico) che, posto a contatto con il nostro dispositivo (incollato o fissato meccanicamente), permetta un migliore smaltimento del calore.

Come opera un dissipatore? In pratica esso estende le dimensioni fisiche del dispositivo, permettendo quindi un miglior smaltimento del calore per effetto Joule: il calore in eccesso viene trasferito all'ambiente circostante, e ciò impedisce il superamento della temperatura limite.

Dal punto di vista dei numeri, cosa cambia nella formula (2)? Semplice: poiché la presenza del dissipatore serve per aumentare la conduttanza termica fra il nostro dispositivo e l'ambiente, si può affermare che la presenza del dissipatore porta ad una riduzione della resistenza termica, che è appunto l'inverso della conduttanza termica. La riduzione della R_{th} nella formula (2) permette quindi di diminuire la temperatura raggiunta dal dispositivo oppure - volendo sfruttare comunque la massima temperatura di lavoro - disporre di una potenza dissipabile più elevata.

Per meglio comprendere non solo il meccanismo di smaltimento del calore, ma anche come ottimizzare i vantaggi derivabili dall'impiego di un dissipatore, analizziamo brevemente le tre modalità di propagazione del calore.

Fig. 35 - Meccanismi di propagazione del calore.



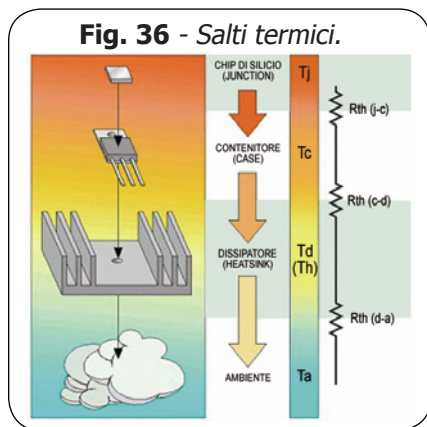
mente le tre modalità di propagazione del calore.

- **Conduzione:** avviene all'interno di un corpo o fra due corpi in contatto fra di loro a causa dell'agitazione termica degli atomi, ed è il meccanismo più efficiente di trasmissione del calore. Per tale motivo il dispositivo che produce calore va fissato saldamente a contatto con il dissipatore, che deve essere costituito da un materiale con una elevata conducibilità termica; il materiale più utilizzato, grazie al costo contenuto e alla facilità di produrre profilati ottenuti per estrusione, è l'alluminio.

- **Convezione:** è il moto all'interno di un fluido, per cui si sfrutta questo fenomeno ad esempio per asportare il calore del dissipatore (o del componente stesso) attraverso il movimento dell'aria, che può essere naturale (dovuto al fenomeno di espansione termica) oppure forzato (tramite ventole). Per le elevate potenze, il fluido refrigerante può essere anche un liquido, che scorre in un condotto all'interno del dissipatore e viene poi raffreddato o per sostituzione (acqua corrente) o tramite un opportuno radiatore.

- **Irraggiamento:** il dissipatore e lo stesso componente irradiano calore nell'ambiente circostante per mezzo della radiazione infrarossa, ed anche questo fenomeno contribuisce allo smaltimento del calore, anche se alle basse temperature

Fig. 36 - Salti termici.



rappresenta una frazione trascurabile del totale.

Una esemplificazione visiva dei tre meccanismi è mostrata in Fig. 35.

Va inoltre precisato che l'efficienza di un dissipatore dipende da svariati fattori e parametri meccanici quali, ad esempio, il materiale utilizzato (conducibilità termica), le sue dimensioni (valore della resistenza termica), la forma (efficacia della convezione dell'aria), il colore (effetto "corpo nero" e quindi efficienza nell'irraggiamento), la finitura superficiale (resistenza di contatto con il componente elettronico e valore dell'irraggiamento), le condizioni di ventilazione (se interno o esterno all'apparecchiatura), la posizione di montaggio (verticale ad alette verticali per una migliore convezione dell'aria) e infine la posizione di montaggio del carico sul dissipatore stesso (centrale oppure verso il bordo).

Detto ciò, il parametro che caratterizza l'efficienza di un dissipatore e che riassume gli altri parametri appena citati, è la sua **resistenza termica** $R_{th\ da}$ (si legge "resistenza termica tra dissipatore e ambiente") che viene specificata dal costruttore sul relativo foglio tecnico del dissipatore, e che viene definita come l'incremento di temperatura subito a causa dell'applicazione di una potenza elettrica (attiva) di 1 Watt, secondo quanto specificato dalla relazione (1). Formule a parte, in linea di massima, maggiori sono le

dimensioni del dissipatore e la sua superficie utile, minore è la sua resistenza termica.

La scelta del dissipatore

Per scegliere un dissipatore occorre conoscere i seguenti parametri:

- potenza massima dissipata dal dispositivo da raffreddare;
- temperatura massima d'esercizio e resistenza termica del dispositivo riscaldante;
- massima temperatura ambiente;
- tipo di flusso di aria in corrispondenza del dissipatore.

Per tener conto di questi parametri, la formula (2) può essere riscritta nel modo seguente, facendo riferimento ad un dispositivo a semiconduttore:

$$(3) T_{j\ max} - T_{a\ max} = Pd_{\ max} \times (R_{th\ jc} + R_{th\ cd} + R_{th\ da})$$

dove

- $T_{j\ max}$ è la massima temperatura di funzionamento della giunzione (ossia del chip al silicio), specificata dal costruttore;
- $T_{a\ max}$ è la massima temperatura ambiente raggiungibile, imposta in fase di progetto;
- $Pd_{\ max}$ è la potenza massima dissipabile, determinata dalle esigenze del carico;
- $R_{th\ jc}$ è la resistenza termica fra giunzione (leggasi "chip") e contenitore, riportata sul datasheet del dispositivo;
- $R_{th\ cd}$ è la resistenza termica fra

contenitore e dissipatore, dovuta al tipo di contatto fra i due e alla eventuale presenza di fogli isolanti o di grasso silconico;

- $R_{th\ da}$ è la resistenza termica dissipatore-ambiente, ricavata dal catalogo dei dissipatori.

La Fig. 36 mostra dove sono posizionate le varie resistenze termiche partendo dal chip e arrivando all'aria che circonda l'insieme. Notate la similitudine con una serie di tre resistenze elettriche. Talvolta, però, non è necessario utilizzare un dissipatore (come avviene spesso nei mosfet degli switching). In questo caso, la formula (2) viene riscritta come

$$(4) T_{j\ max} - T_{a\ max} = Pd_{\ max} \times R_{th\ ja}$$

nella quale si utilizza la sola resistenza termica giunzione-ambiente $R_{th\ ja}$ ricavabile anch'essa dal datasheet del componente. Nella Fig. 37 sono riportati i valori di resistenza termica $R_{th\ jc}$ ed $R_{th\ ja}$ per alcuni dei più comuni contenitori impiegati in elettronica per i dispositivi a semiconduttore.

Un esempio numerico

Un dispositivo a semiconduttore in grado di operare a una temperatura massima di giunzione di 150 °C ha la necessità di dissipare una potenza di 5 watt, ed è incapsulato in un contenitore TO-3. La massima temperatura ambiente prevista è di 50 °C e ➤

Fig. 37 - Caratteristiche termiche di alcuni semiconduttori.

Package	$R_{th\ ja}$ (°C/W)	$Pd_{\ max}$ $T_a = 25^\circ C$	$R_{th\ jc}$ (°C/W)	$Pd_{\ max}$ $T_c = 25^\circ C$	$I_{c\ max}$ DC
TO-92 TO-18	200	0,6 W	80	1,6 W	1 A
TO-5	160	0,9 W	80	1,6 W	1 A
TO-126 TO-127	90 80	1,4 W 1,6 W	2,8 1,3	45 W 100 W	6 A 15 A
TO-220	65	2 W	1,2	125 W	35 A
TO-3 TO-3 spesso	50 38	3 W 3,3 W	0,7 0,5	175 W 350 W	60 A 7 A
TO-3P TO-218 SOT-93 TO-247	50	2,5 W	1	150 W	75 A

si intende raffreddare il dissipatore a convezione naturale in aria libera. Le specifiche di progetto, inoltre, impongono di fissare meccanicamente il dissipatore direttamente sul contenitore TO-3, in modo da poter ridurre al massimo la resistenza termica contenitore-dissipatore: il contatto diretto metallo-metallo determina una $R_{th_{cd}}$ di 0,3 °C/W.

Dovendo scegliere il tipo di dissipatore necessario, l'incognita è la $R_{th_{da}}$, che può essere ricavata sostituendo i valori noti nella relazione (3):

$$\begin{aligned} R_{th_{da}} &= \\ &= [(T_{j_{max}} - T_{a_{max}}) / (P_{d_{max}})] - R_{th_{jc}} - R_{th_{cd}} = \\ &= [(150 - 50) / 5] - 0,7 - 0,3 = \\ &= 19 \text{ °C/W} \end{aligned}$$

con $R_{th_{jc}} = 0,7 \text{ °C/W}$ ricavato dalla Fig. 37. A questo punto abbiamo calcolato il valore massimo ammesso, per la resistenza termica tra dissipatore e ambiente, ma conviene scegliere sul catalogo un dissipatore con una resistenza termica inferiore a tale valore. Si tenga presente, inoltre, che il valore della resistenza termica ottenuta per il dissipatore può essere ridotta con un'opportuna ventilazione (sia in aria libera che forzata) che permette di ridurre il valore della $R_{th_{da}}$.

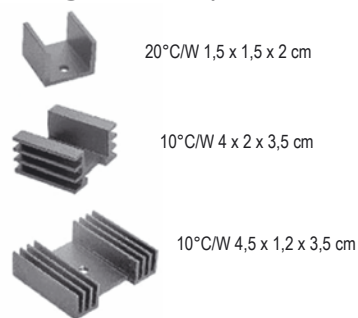
Come si è detto, la scelta del tipo di dissipatore va fatta in base al valore della sua resistenza termica, valore che viene indicata dal costruttore insieme alle dimensioni: ricordate sempre che, come già detto, maggiori sono le dimensioni del dissipatore, minore è la sua resistenza termica verso l'ambiente. Nel caso dei dissipatori in profilato d'alluminio, la resistenza termica diminuisce con l'aumentare della lunghezza del profilato, con un andamento pressoché iperbolico asintotico (ossia che tende a zero per una lunghezza infinita). Si noti infatti che, al di sopra di una certa lunghezza del dissipatore (ad esempio sopra i 20 centimetri) il valore di $R_{th_{da}}$ cala solo leggermente, in quanto l'ecces-

siva distanza dal punto di fissaggio del componente che produce il calore provoca un calo della temperatura dell'elemento metallico al punto da rendere meno efficiente lo scambio termico con l'ambiente. I fogli tecnici dei dissipatori precisano inoltre che il grado di finitura superficiale dei profilati è in grado di influenzare la dissipazione per irraggiamento.

I dissipatori sono normalmente in alluminio anodizzato nero opaco. Nel caso di superfici lucide o grezze l'efficienza del dissipatore si riduce e la resistenza termica aumenta di circa il 10 %. Si tenga inoltre conto che i valori di $R_{th_{da}}$ vengono specificati per un montaggio verticale sia del dissipatore che delle sue alette. Ogni altro tipo di montaggio porta ad un peggioramento della resistenza termica. In particolare, per il montaggio orizzontale si deve considerare un aumento di R_{th} pari al 20 %. Nel caso di dissipatori rettangolari, è consigliabile il montaggio col lato minore in verticale, in modo da evitare l'effetto di eccessivo riscaldamento della porzione di metallo sovrastante il punto di fissaggio del componente che produce calore.

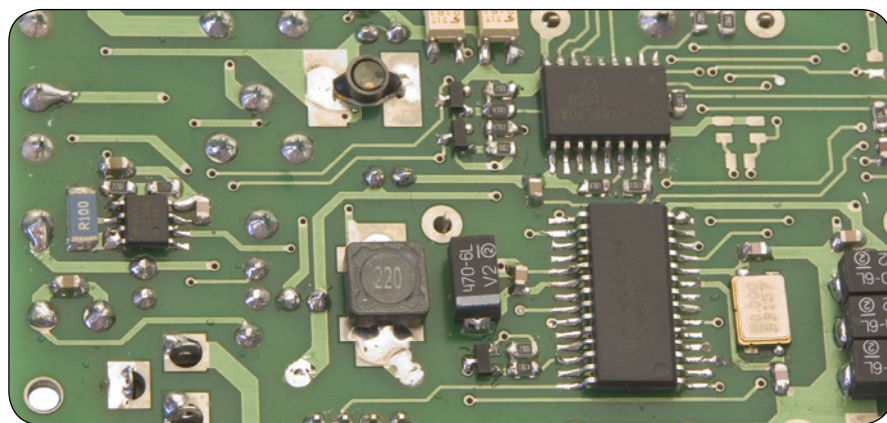
Può capitare, talvolta, che si debbano utilizzare componenti di dimensioni e potenze ragguardevoli (in contenitore TO-220 o TO-3, per esempio) per commutare potenze molto elevate solo per brevi istanti. In tal caso, il più delle volte, è sufficiente appoggiare il contenitore direttamente sul circuito stampato, senza aletta di

Fig. 38 - Dissipatori.



raffreddamento. Si tenga però conto che, nel frequente caso in cui la potenza applicata al dispositivo non sia continua bensì impulsiva, il calcolo della potenza da dissipare è differente, in quanto, dopo ciascun impulso applicato, la giunzione ha modo di raffreddarsi con una velocità che dipende sia dalla resistenza termica che dal duty-cycle e dalla frequenza della serie di impulsi. Ipotizziamo, per esempio, che un transistor commuti/dissipi una potenza istantanea di 10 Watt per 10 millisecondi e che, in questa situazione, la temperatura della sua giunzione passi da 25 a 60 °C con un successivo raffreddamento. Se la durata dell'impulso di potenza dovesse aumentare (ad esempio 50 msec) anche la temperatura di giunzione salirebbe, forse a 80 °C e forse di più.

L'esempio appena fatto non è reale: serve solo per farvi capire che la temperatura di un dispositivo dipende sia dalla potenza che deve dissipare sia dal tempo in cui tale potenza viene dissipata: minore è



Convertitore AC-DC switching da circuito stampato con ingresso e uscita isolati galvanicamente.



il tempo di applicazione, minore è l'innalzamento della temperatura, anche se il picco di potenza è elevato. Proprio per tener conto di questo fenomeno, che dipende da tanti fattori tra i quali l'inerzia termica del dispositivo, molti costruttori forniscono un grafico che evidenzia la variazione della resistenza termica al variare della durata degli impulsi applicati, in funzione di vari valori di duty-cycle, di frequenza e di potenza commutata.

Alcuni esempi pratici

Facciamo ora un esempio pratico di calcolo utilizzando le formule già viste. Il datasheet del transistor 2N1711 - che ha un contenitore TO-5 visto in Fig. 37 - indica che esso può dissipare una potenza massima di 1 watt, che possiede una resistenza termica $R_{th_{ja}}$ di 220 °C/W ed ammette una temperatura massima di giunzione di 195 °C. Se supponiamo ad esempio che in un certo circuito esso dissipi una potenza di 0,8 W, la temperatura raggiunta dal chip di

Alimentatore switching da laboratorio: uscita 0÷40 Vdc, 0÷5 A, rumore residuo di 10 mV.



silicio potrà essere ricavata dalla relazione (2):

$$T_j - 25 = 0,8 \times 220$$

da cui si ricava che la temperatura di giunzione T_j è di ben 151 °C.

Nella formula (2), però, l'incognita non è necessariamente sempre e solo la temperatura di giunzione: da essa è possibile anche ricavare la massima potenza dissipabile conoscendo la temperatura massima e la resistenza termica, e viceversa.

Ad esempio, se il nostro transistor si trovasse su di un circuito stampato collocato all'interno di un contenitore in cui la temperatura a regime è di 75 °C a causa degli altri componenti presenti, nella formula (2) la "temperatura ambiente" per il transistor diventerebbe 75 °C (invece che 25 °C) e ciò limiterebbe la massima potenza dissipabile, che in questo caso sarebbe:

$$T_{j_{max}} - T_a = P_{max} \times R_{th_{ja}}$$

$$P_{max} = (T_{j_{max}} - T_a) / R_{th_{ja}} = (195 - 75) / 220 = 545 \text{ mW}$$

Per dissipare una potenza maggiore si impone quindi l'uso di un dissipatore.

I componenti senza dissipatore possono, infatti, dissipare potenze modeste, tipicamente non più di qualche watt. Con potenze maggiori, è necessario "aiutarli" fissandoli a un dissipatore, in grado di smaltire il calore in eccesso. Non tutti i componenti, però, permettono il fissaggio a un dissipatore, per cui - per potenze superiori al watt - è necessario utilizzare componenti appositi, provvisti di flange metalliche forate, come ad esempio i TO-126, TO-127, TO-220, TO-3, eccetera, dotati di fori di fissaggio. Esistono anche dei dissipatori che possono essere fissati a componenti sprovvisti di fissaggio meccanico: tipicamente sono a molla, di forma radiale, ed avvolgono il contenitore del componente, ma non sono in grado di decuplicare la potenza massima dissipabile; ottimisticamente parlando, ci si può aspet-

tare un dimezzamento della $R_{th_{ja}}$, ma non di più.

Quale dissipatore?

Facciamo un altro esempio: supponiamo di utilizzare il transistor TIP131, ospitato nel contenitore TO-220 (Fig. 37). Il costruttore dichiara che esso può dissipare una potenza massima di 40 W (a patto che il contenitore non superi i 25 °C), ammette una T_j massima di 150 °C ed è caratterizzato da una $R_{th_{jc}}$ pari a 3,1 °C/W. La precisazione "a patto che il contenitore non superi i 25 °C" ci fa immaginare che, per poter garantire tale condizione, dobbiamo raffreddare bene il nostro transistor altrimenti il contenitore si scalderebbe sicuramente e addio potenza di 40 watt: il dissipatore, quindi, è indispensabile!

Supponiamo invece che il nostro transistor si trovi in un circuito in cui debba dissipare una potenza di soli 5 W, racchiuso in un contenitore (il suo "ambiente") con una temperatura di 40 °C. Servirà il dissipatore? Se sì, come dovrà essere? Per scoprirlo è sufficiente utilizzare la relazione (4), in cui ipotizziamo di fissare il contenitore al dissipatore con una $R_{th_{cd}}$ di 0,9 °C/W, decidendo di sfruttare al massimo la giunzione di silicio, facendogli raggiungere i suoi 150 °C massimi. La formula diventa quindi

$$150 - 40 = 5 \times (3,1 + 0,9 + R_{th_{da}})$$

da cui si ricava una resistenza termica dissipatore-ambiente di 18 °C/W. Dai fogli tecnici del costruttore di dissipatori si possono scegliere vari tipi di alette di raffreddamento, a seconda dell'ingombro voluto. Immaginiamo di scegliere un tipo a "C" con alette, il secondo di Fig. 38. Esso misura 4 x 2 x 3,5 cm, è idoneo per il TO-220 e presenta una resistenza termica di 10 °C/W, parecchio inferiore ai 18 richiesti.

È sempre preferibile che la resistenza termica del dissipatore sia inferiore al valore calcolato: in tal modo ➤

Alimentatore switching da PC: i 120 W che può erogare vengono prelevati direttamente dalla tensione di rete raddrizzata.



si ha la certezza che il dissipatore conduca il calore meglio del dovuto, limitando ulteriormente il surriscaldamento del transistor! Si parla in questo caso di dissipatore “sovradimensionato”.

Grazie ai 10 °C/W di resistenza termica, il dissipatore scelto può permetterci di dissipare un po' più dei 5 watt suddetti; inserendo infatti questo valore nella (4) si ha:

$$150 - 40 = P_d \times (3,1 + 0,9 + 10)$$

da cui si ottiene una potenza massima dissipabile di 7,9 W e ciò ci fa ritenere che i nostri 5 W siano dissipabili in tutta sicurezza!

Non solo, ma a pensarci bene, possiamo anche intuire che dissipando solo 5 W anziché i 7,9 ammissibili, il transistor lavorerà ad una temperatura inferiore ai 150 °C massimi. Sì, ma a quanto? Lo possiamo scoprire sempre usando opportunamente la relazione (4), utilizzando come incognita proprio la temperatura di giunzione T_j :

$$T_j - 40 = 5 \times (3,1 + 0,9 + 10)$$

dalla quale scopriamo che, grazie al dissipatore leggermente sovradimensionato, la temperatura interna del transistor raggiungerà i 110 °C anziché i 150 massimi, a tutto vantaggio della sicurezza di funzionamento e quindi della durata del componente.

Utilizzando in modo appropriato la formula (4), si possono ottenere anche altre informazioni utili come, ad esempio, la temperatura raggiunta dal dissipatore. Come si fa? È sem-

plice: basta riscrivere la formula relativamente al solo salto termico fra ambiente e dissipatore, in modo da trasformarla nella seguente:

$$T_d - T_{amb} = P_d \times R_{th_{da}}$$

e sostituire in essa i valori noti, con incognita proprio T_d :

$$T_d - 40 = 5 \times 10$$

dalla quale si ricava che il dissipatore raggiungerà i 90 °C. Tale informazione è utile, ad esempio, se si intende monitorare la temperatura raggiunta dal dissipatore tramite dei sensori termici, spesso impiegati al fine di limitare la potenza dissipata (e quindi la temperatura raggiunta) ad esempio con dei contatti bimetallici (come quello utilizzato nell'alimentatore con PIC, pubblicato sul fascicolo 123) o con dei termistori collegati a dei relé i quali, superata la soglia voluta, spengono il circuito. Si noti che lo stesso risultato può essere ottenuto anche effettuando il percorso inverso, ovvero partendo dalla temperatura del chip e dalla somma delle resistenze termiche per raggiungere il dissipatore, ossia:

$$T_j - T_d = P_d \times (R_{th_{jc}} + R_{th_{cd}})$$

dove, sostituendo i valori, si ottiene:

$$110 - T_d = 5 \times (3,1 + 0,9)$$

ottenendo comunque i 90 °C calcolati in precedenza.

I valori della resistenza termica dei dissipatori indicati dal costruttore, comunque, sono ricavati attraverso prove di laboratorio effettuate nelle seguenti condizioni:

1. *dissipatore anodizzato nero opaco*: la finitura superficiale del dissipatore influenza principalmente la dissipazione per irraggiamento. I dati riportati dai costruttori si riferiscono - ove non diversamente specificato - a dissipatori ossidati neri opachi.
2. *montaggio verticale in aria libera con convezione naturale*: i moti convettivi dell'aria sono determinanti per l'asportazione del calore.
3. *componente da raffreddare posi-*

zionato in un'area centrale del dissipatore: questo per sfruttare al massimo la conducibilità termica del metallo e rendere più uniforme possibile la sua temperatura.

4. *contatto diretto tra componente e dissipatore, con interposizione di grasso termico*: al fine di ottenere la minima resistenza termica fra componente e dissipatore si deve garantire la massima superficie di contatto possibile, e ciò significa che occorre far sì che la superficie di contatto sia la più levigata possibile, onde massimizzarne l'area effettiva. Poiché, però, non è possibile garantire la totale assenza di aria interposta (che aumenta la resistenza termica) si può usare del grasso al silicone (a bassa resistenza termica) meglio se riempito con ossido di zinco.

Note conclusive

Siamo giunti al termine di questo nostro minicorso sulla progettazione degli alimentatori e sul dimensionamento dei dissipatori. Siamo certi che, con le nozioni acquisite, da oggi sarete in grado di dimensionare al meglio gli stadi di alimentazione utilizzati nei vostri circuiti. Noi, però, non ci fermiamo qui: continuate a seguirci perché, nei prossimi numeri, pubblicheremo due progetti realizzati applicando in pratica quanto spiegato in queste due puntate. Vi anticipiamo che si tratta di un alimentatore PWM/DC per fermodellismo e di un piccolo DC-DC converter (il cui prototipo è visibile qui sotto) in grado di accettare tensioni in ingresso di 5÷30 Vdc e di fornire in uscita 3,6 Vdc 1,5 A, in 35 x 9 x 6 mm di ingombro.



Cartellino orario con fingerprint

di Boris Landoni

Progettato per tenere traccia delle entrate e delle uscite dal luogo di lavoro, per il riconoscimento della persona impiega un lettore di impronta digitale professionale che rende superfluo l'utilizzo di schede, tessere e transponder. I dati relativi alle timbrature per un massimo 250 utenti vengono memorizzati in una comune pen-drive USB o possono essere scaricati da un PC remoto collegato via radio o via Ethernet. Prima puntata.

Alcuni anni or sono iniziammo a parlare dei transponder, dispositivi attivi a semiconduttore dotati di un'induttanza avvolta su molte spire, quest'ultima utilizzata sia per prelevare l'alimentazione indotta via radio sia come antenna per trasmettere un piccolo quantitativo di informazioni memorizzate. Come nostra consuetudine, anche in quel caso avevamo pensato a un progetto pratico che potesse sfruttare appieno le potenzialità di tali dispositivi; nasceva così uno dei progetti più impegnativi proposti in passato, ossia il cartellino orario a transponder, dispositivo usato nelle azien-

de per registrare l'orario di entrata e di uscita del personale, rivisitato in versione elettronica. In esso, i vetusti cartellini cartacei erano stati rimpiazzati con dei badge a transponder e la registrazione avveniva non su carta ma sotto forma di dati digitali salvati in un banco di memorie. L'FT449 (questo era il codice di tale progetto, pubblicato su *Elettronica In* numero 71 e 72) ha prestato egregiamente servizio in molte aziende e tuttora lo sta facendo con onore.

La ricerca nel mondo dell'elettronica, però, ha continuato a progredire, proponendo nuove tecnologie sempre più

Collegamenti al PC

XTR-7020A



Radio Modem



Il collegamento via radio viene effettuato tramite un modulo XTR7020A che comunica con un radiomodem equivalente (FT737), collegato al PC remoto.



Pen Drive

Il cartellino orario può operare sia in modalità autonoma, salvando i dati relativi ai vari eventi su una pen drive, sia in modalità assistita, cioè inviando i dati ad un PC, per mezzo di un collegamento radio (riquadro in alto) o Ethernet (riquadro in basso). Può anche operare in entrambe le modalità, salvando gli accessi su pen drive ed inviandoli contemporaneamente al PC remoto.

Computer + Software



Switch



EM202



Porta rete PC



Il collegamento via Ethernet viene effettuato tramite un convertitore seriale-Ethernet Tibbo EM202, cui qualsiasi PC dotato di interfaccia di rete può accedere.

sicure, economiche e prestanti: proprio per questo abbiamo deciso di mandare in pensione il vecchio cartellino orario e di presentarne uno più compatto e, contemporaneamente, più performante.

Questa nuova versione (codice progetto FT736) nemmeno necessita più di alcun badge: la registrazione del movimento del personale, infatti, viene effettuata riconoscendo l'im-

pronta digitale tramite un fingerprint professionale, il noto FPS che abbiamo già utilizzato nell'apriporta (fascicolo 115). Gli immensi banchi di memoria, poi, sono stati rimpiazzati da una comunissima pen-drive USB, oggi reperibile in commercio a pochi Euro al gigabyte. Lo scarico dei dati, infine, può essere fatto rimuovendo la pen-drive e copiandone il contenuto su un qualsiasi PC, oppure in remo-

to da PC, via radio o, in alternativa, via Ethernet. Visto lo spessore del progetto, abbiamo preferito trattarlo in tre puntate: in questo primo articolo analizzeremo dettagliatamente la struttura del sistema, cui faranno seguito le note relative alla realizzazione e al collaudo. Nelle due puntate successive, invece, parleremo dell'utilizzo in pratica e analizzeremo il diagramma di flusso, le routine ➤

- Identificazione utente: tramite riconoscimento dell'impronta digitale.
- Impronte digitali memorizzabili: 250
- Tempo di accesso in lettura: circa 130 ms per impronta.
- Numero timbrature memorizzabili su pen drive: circa 100 milioni.
- Interfaccia utente: display, pulsanti entrata/uscita, tastiera configurazione.
- Gestione timbrature: salvataggio su pen drive USB e/o invio a PC operatore.
- Dimensione pen drive: 4 Gbyte massimi.
- Tipo di connessione al PC dell'operatore (una esclude l'altra):
 - via radio tramite radiomodem XTR-7020A-4 (434 MHz)
e XTR-7020A-4 (868 MHz);
 - via Ethernet tramite convertitore EM202.
- Orario di riferimento: tramite modulo real-time-clock con batteria tampone.
- Tensione di alimentazione e assorbimento: 9 ÷ 12 Vdc, 300 mA massimi.
- Operazioni da remoto:
 - impostazione di data e ora, scarico file timbrature
 - tramite software Windows e set di comandi proprietari.
- Operazioni in locale: configurazione completa e accesso al file timbrature mediante rimozione e lettura della pen drive.
- Modi di funzionamento per il riconoscimento di utenti ed eventi:
 - Modo 0, pressione pulsante entrata/uscita e lettura impronta;
 - Modo 1, come modo 0 con trasmissione dell'evento all'operatore;
 - Modo 2, attesa continua di un'impronta;
 - Modo 3, come modo 2, con trasmissione dell'evento all'operatore.

firmware più significative e l'applicativo Windows per la gestione da remoto dei dati memorizzati.

Entriamo ora nel vivo della discussione e scopriamo tutti i segreti di questo interessante progetto.

La struttura

La struttura del cartellino orario è visibile nello schema a blocchi di Fig. 1 in cui, per semplificare la comprensione, abbiamo voluto raggruppare tra loro i vari blocchi funzionali a seconda delle funzioni svolte da essi. Iniziamo la descrizione dall'elettronica di gestione: essa include il microcontrollore, un PIC18F458 in formato 40 pin DIL, che sovrintende tutte le funzioni delle varie periferiche. Al microcontrollore fanno capo tutte le periferiche, a partire dal modulo RTC (Real Time Clock, del quale abbiamo già pubblicato un articolo sul fascicolo 117) il cui compito è quello di fornire al sistema data e ora valide. Grazie alla batteria di back-up di

cui dispone, l'orologio e il calendario possono continuare a funzionare anche in caso di mancanza prolungata dell'alimentazione. Il collegamento tra micro e RTC è bidirezionale via seriale, in quanto l'RTC deve poter essere sia letto (per prelevare data e ora correnti) che scritto (per impostare data e ora)

Il secondo blocco è rappresentato dal convertitore seriale-USB Host, tramite il quale il microcontrollore può facilmente accedere sia in lettura che in scrittura, ad una qualsiasi pen drive USB, utilizzata come memoria per l'archiviazione dei dati relativi alle timbrature e agli accessi. Tale blocco è rappresentato da un modulo VDIPI, prodotto da FTDI (www.ftdichip.com) e basato sul Vinculum VNC1L (www.vinculum.com) un chip progettato dalla stessa FTDI che consente di aggiungere molto facilmente una periferica USB Host (ossia come quella del PC) a qualsiasi piattaforma a microcontrollore di

natura non PC, a patto che tale piattaforma disponga di almeno una porta seriale standard, di una porta SPI o di una parallela. Qualora vogliate saperne di più (o semplicemente vogliate rinfrescare la memoria) potete andare a rileggere l'articolo teorico relativo al Vinculum, pubblicato sul fascicolo 119. Come visibile dalla Fig. 1, il microcontrollore ha anche il compito di gestire l'interfaccia utente, rappresentata dal lettore di impronta digitale (modello FPS), dal display alfanumerico a due righe per sedici caratteri e, infine, dalla tastiera, che serve per comunicare al sistema l'entrata o l'uscita (qualora il modo operativo impostato lo preveda, ma questo verrà descritto più avanti) e gestire la configurazione attraverso i vari menu.

Abbiamo detto che lo scarico dei dati contenuti nella pen drive può essere fatto rimuovendo temporaneamente quest'ultima dalla propria sede e collegandola ad un qualsiasi PC, oppure interrogando via remoto il cartellino orario. Per lo scarico dati remoto, però, è necessario che il cartellino disponga di un'interfaccia operatore PC, rappresentata da un convertitore seriale-Ethernet modello EM202 (prodotto da Tibbo) oppure, in alternativa, da un radiomodem seriale modello XTR-7020-Ax, prodotto da Aurel nella versione a 434 MHz (-A4) o 868 MHz (-A8) e descritto dettagliatamente nell'articolo pubblicato sul fascicolo 125 di *Elettronica In*. Tenete presente che non è possibile installare entrambi i dispositivi mentre è invece possibile utilizzare il cartellino orario senza di essi.

Rimane da descrivere l'ultimo blocco, ossia l'alimentatore: esso è costituito da un regolatore stabilizzatore DC-DC a 5 V, seguito da un regolatore lineare "low-drop-out" a 3,3 V. Quest'ultimo serve per fornire alimentazione sia al radiomodem Aurel che al fingerprint, unici dispositivi che necessitano di una tensione differente dai 5 V. L'alimentatore richie-

de una tensione d'ingresso compresa tra 9 e 12 volt, con una corrente massima di 300 mA.

Flessibilità di impiego

Nello strutturare il sistema, ci siamo resi conto che l'impiego di moduli specializzati conferisce ad esso una grande flessibilità di utilizzo, così da poter adattare il cartellino orario a qualsiasi necessità. Noi abbiamo, comunque, ipotizzato che si possano verificare tre situazioni di impiego tipiche.

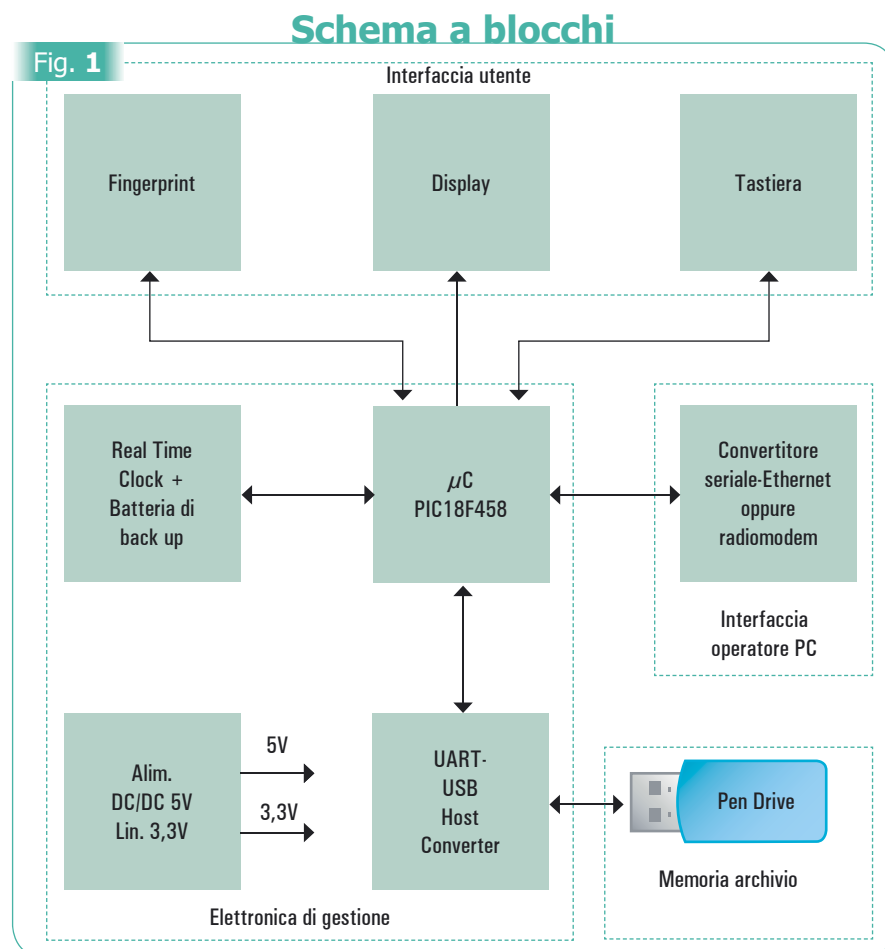
- *Funzionamento stand-alone*: gli accessi sono salvati in un database contenuto in una pen drive; in questo caso vengono montati i moduli RTC, fingerprint e VDIP1.

- *Funzionamento remote-mode*: i dati non vengono memorizzati in locale, ma vengono spediti al PC operatore; in questo caso devono essere montati i moduli RTC, fingerprint e, uno in alternativa all'altro, il radiomodem o il convertitore seriale-Ethernet.

- *Funzionamento full-feature*: gli accessi vengono salvati su pen drive e, in funzione delle impostazioni, eventualmente inviati in automatico al PC dell'operatore che, comunque, può accedere al database degli accessi in qualsiasi momento. In questo caso vengono montati i moduli RTC, fingerprint, VDIP1 e, a scelta (in base alle proprie necessità), il radiomodem oppure il convertitore seriale-Ethernet.

Schema elettrico

L'utilizzo di moduli specializzati quali il VDIP1, l'RTC, l'FPS, l'EM202 e l'XTR-7020-Ax (per ognuno di essi abbiamo previsto un riquadro a parte che ne riassume le caratteristiche salienti) ha permesso di ottenere un dispositivo estremamente sofisticato e performante, ma con uno schema elettrico molto semplice. Iniziamo la descrizione dello schema partendo dal microcontrollore che, durante la fase di avvio del programma, inizializza le proprie linee di I/O in modo

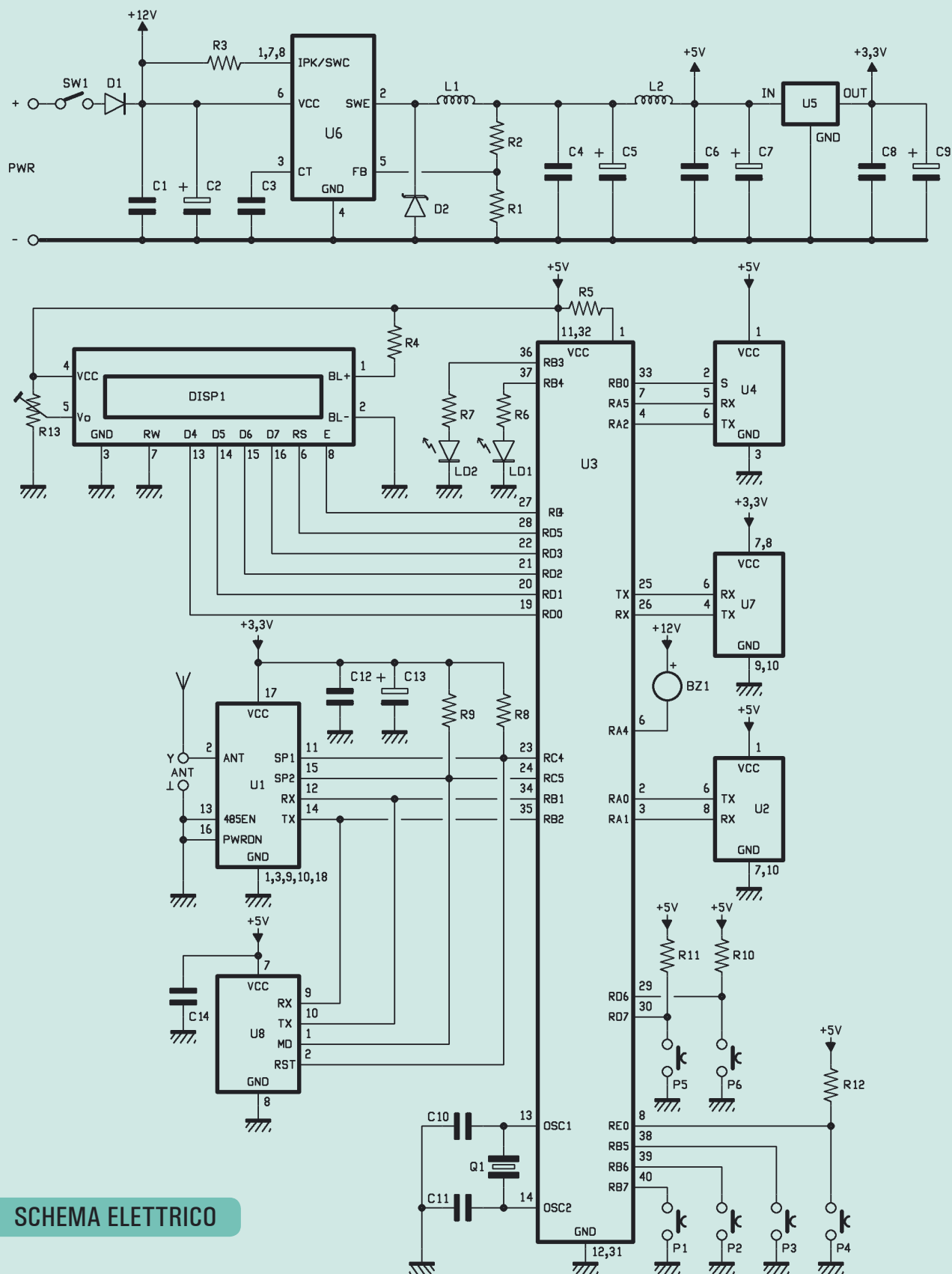


opportuno. Le linee RA5 e RA2, facenti capo ai pin 7 e 4 del PIC (U3), realizzano la porta seriale necessaria per la comunicazione tra il micro e l'RTC U4; la linea RB0 (pin 33), utilizzata come ingresso e collegata all'uscita S, verifica costantemente il funzionamento dell'RTC, configurato dal micro per far pulsare tale uscita. La comunicazione seriale, come già accennato, è bidirezionale e consente sia la lettura che l'aggiornamento di data e ora dell'orologio.

Le linee TX e RX (pin 25 e 26) realizzano il collegamento seriale con il lettore di impronte digitali. Anche in questo caso il collegamento è bidirezionale in quanto il fingerprint deve poter essere sia programmato per riconoscere solo le impron-

te volute che letto durante la fase di attesa di un'impronta. Le linee RA0 e RA1 (pin 2 e 3) vengono impiegate per ottenere una terza porta seriale, necessaria per il collegamento al convertitore seriale-USB host U2 (VDIP1). Anche in questo caso la bidirezionalità è necessaria in quanto il cartellino orario può sia leggere che scrivere dati sulla pen drive esterna, collegata al VDIP1.





SCHEMA ELETTRICO

Le linee RB1, RB2, RC4 e RC5 consentono al microcontrollore di entrare in comunicazione con il radiomodem

XTR7020 (U1) o con il convertitore seriale-Ethernet EM202 (U8): esse realizzano una quarta porta seriale

che, oltre ai classici TX (RB2) e RX (RB1), dispone anche di due pin di controllo (RC4 e RC5), che servono

per impostare la velocità della seriale del radiomodem oppure per resettare e avviare il modo "aggiornamento firmware" nel convertitore seriale-Ethernet.

Il display, il cui contrasto viene regolato con il trimmer R13, viene gestito nella classica modalità a "4 bit + Enable e RS" tramite le linee da RD0 a RD6.

Rimangono infine i quattro pulsanti per la configurazione del dispositivo

i due pulsanti per comunicare al dispositivo l'entrata (P5, collegato alla linea RD7) e l'uscita (P6, collegato a RD6), i due led di segnalazione (LD1 e LD2), gestiti dalle porte RB4 e RB3, ed il buzzer, collegato a RA4.

Lo stadio di alimentazione, è realizzato con un regolatore switching a 5 V (U6, un MC33063AP) seguito da un regolatore lineare low-drop-out modello LD1086-3.3

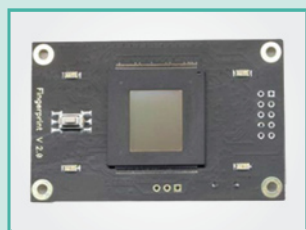
Il secondo, il cui calore (generato du-

a U, alimenta i soli moduli fingerprint e radiomodem. Il primo, invece, alimenta tutto il resto del circuito.

Realizzazione pratica

L'impiego di moduli specializzati ha permesso non solo di ottenere uno schema elettrico semplice, ma anche di realizzare un circuito stampato facile da montare: esso infatti è di dimensioni abbondanti e non particolarmente denso di componenti, ma

Il Fingerprint FPS



Si tratta di un lettore biometrico professionale di impronte digitali, di dimensioni molto contenute. È dotato di un sensore fingerprint e di un microcontrollore a 32 bit RISC della serie FR di casa Fujitsu; può essere controllato facilmente tramite qualsiasi mi-

crocontrollore che disponga almeno di una porta seriale standard a livello TTL 3,3 V. Il modulo FPS è dotato di due uscite ausiliari TTL 3,3 V completamente gestibili da seriale; può, inoltre, essere programmato per funzionare in modalità Stand-Alone, modalità in cui non necessita di alcun controllo esterno. Questo dispositivo rappresenta la soluzione ottimale per la realizzazione di sistemi biometrici per impronte digitali.

Caratteristiche tecniche

- Sensore Fingerprint:
 - Dispositivo capacitivo allo stato solido;
 - Risoluzione 500 dpi (50 μ m);
 - Area sensibile: 1,28 cm x 1,50 cm (0,5" x 0,6");

- 256 x 300 sensor array;
- Alta precisione dell'identificazione biometrica;
- Vetro di protezione ad elevate caratteristiche di resistenza.
- Modulo Hardware:
 - Microcontrollore 32 bit RISC serie FR di Fujitsu;
 - SDRAM 4 Mbyte;
 - Flash memory da 2 Mbyte per sistema codifica;
 - Porta seriale;
 - 2 uscite ausiliari TTL;
 - 4 LED + 1 tasto frontale.
- Librerie Software:
 - Tempo medio di verifica: < 1 secondo;
 - Database interno con capacità di 500 impronte digitali;
 - Uscite ausiliari configurabili in modo che vengano gestite dal modulo senza alcun intervento esterno.
- Alimentazione: 3,3 V.
- Consumo corrente: 0,8 W.
- Temperatura operativa: 0÷70 °C.
- Temperatura di stoccaggio: -20÷+85 °C.
- Dimensioni: 39 x 62 x 7 mm.
- Peso: 15 g.

Il modulo RTC FT674

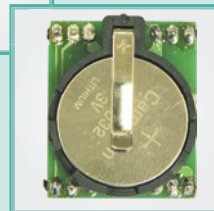
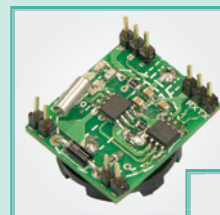
Con un ingombro di pochi centimetri quadri e un consumo di corrente irrisorio, genera data e ora di sistema.

Facilmente utilizzabile in qualsiasi apparato digitale, comunica con l'esterno tramite una porta seriale e dispone anche di un'uscita di clock programmabile. Grazie ad un sistema di back-up interno, può mantenere i dati in memoria anche in assenza di corrente.

Caratteristiche tecniche

- Chip RTC: DS1307.

- Back up: batteria da 3 Vdc, CR2032.
- Frequenza base: 32,768 kHz.
- Configurazione: I²C bus.
- Comunicazione: microcontrollore IC12F675-SO programmato come convertitore seriale-I²C bus.
- Utilizzo: tramite un set di comandi seriali.
- Alimentazione: 5 Vdc.
- Assorbimento: 1,5 mA.
- Dimensioni: 27 x 23 mm.



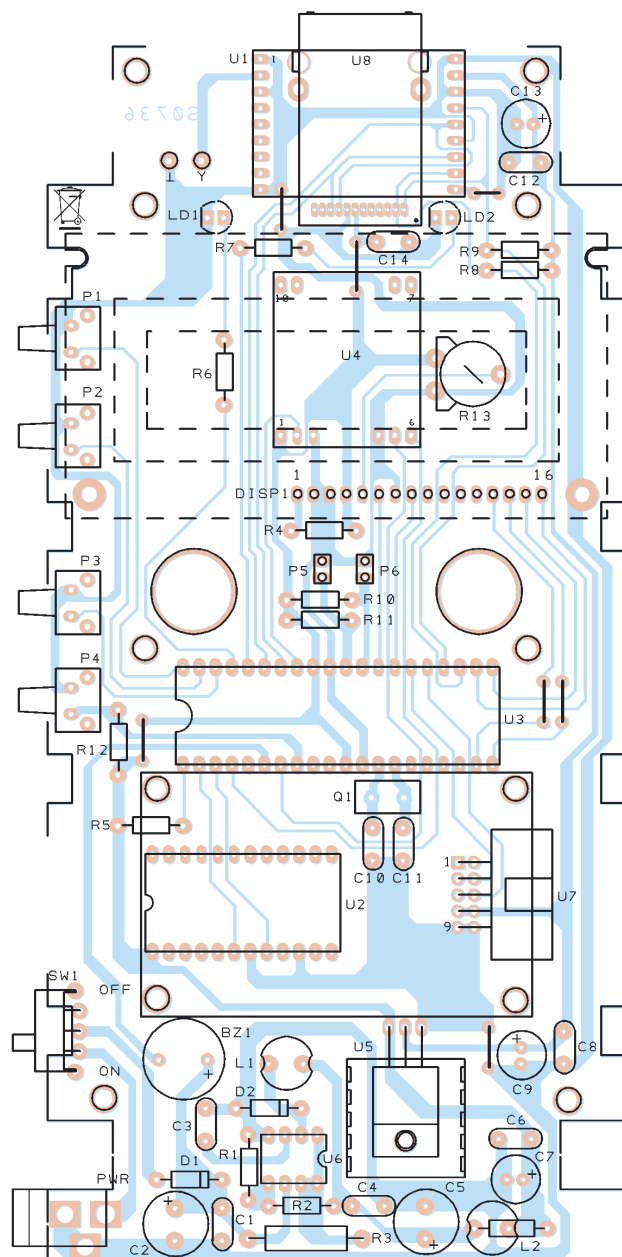
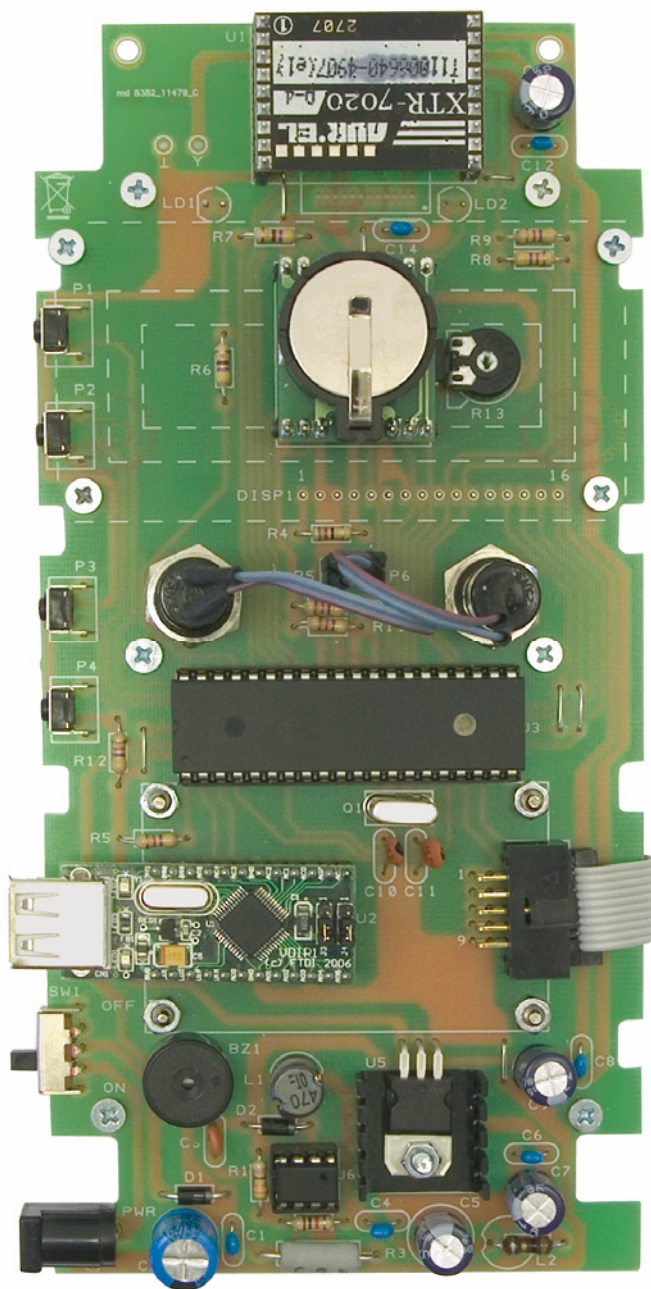
(P1÷P4), collegati alle linee RB7, RB6, RB5 e RE0 (quest'ultima necessita di una resistenza di pull-up),

rante il normale funzionamento) viene opportunamente dissipato tramite una piccola aletta di raffreddamento

procediamo per gradi.

Innanzitutto dovete realizzare il circuito stampato: a tal proposito, pote- ➤

La realizzazione del cartellino orario non è particolarmente difficile in quanto vengono impiegati componenti ad inserzione e moduli già assemblati. La maggior parte degli elementi è posizionata sul lato componenti del circuito (monofaccia). Sul lato saldature vanno montati solamente il display, i due led di segnalazione



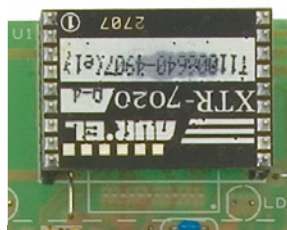
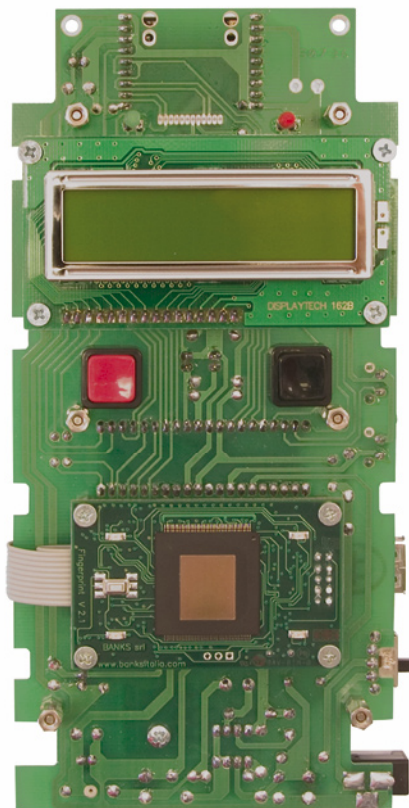
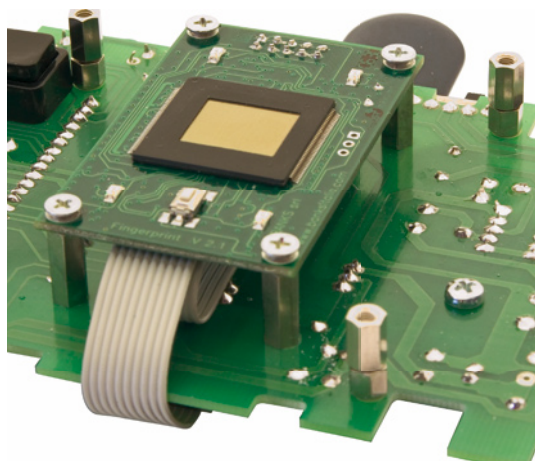
ELENCO COMPONENTI:

R1: 330 ohm
 R2: 1 kohm
 R3: 0,1 ohm 1W
 R4: 82 ohm
 R5, R8+R12: 4,7 kohm
 R6, R7: 470 ohm
 R13: trimmer MO 4,7 kohm
 C1, C4, C6, C8: 100 nF multistrato
 C2: 470 µF 25 VL elettrolitico
 C3: 150 pF ceramico

C5, C7, C9, C13: 470 µF 16 VL elettrolitico
 C10, C11: 15 pF ceramico
 C12, C14: 100 nF multistrato
 D1: 1N4007
 D2: 1N5819
 LD1: led 3 mm rosso
 LD2: led 3 mm verde
 U1: XTR-7020 (vedi testo)
 U2: VDIP1
 U3: PIC18F458 (MF736)
 U4: Modulo RTC (FT674)
 U5: LD1086-3.3

U6: MC33063A
 U7: Modulo Finger FPS
 U8: EM202 (vedi testo)
 SW1: Deviatore a slitta 90°
 L1: Bobina 47 µH
 L2: Induttanza 1 µH
 BZ1: Buzzer 12V con elettronica
 P1+P4: Microswitch 90°
 P5: Pulsante NA rosso
 P6: Pulsante NA nero
 DISP1: Display LCD 16x2
 Q1: Quarzo 20 MHz

(rosso e verde), i due pulsanti che determinano l'entrata e l'uscita ed il fingerprint. Prestate attenzione al corretto verso di inserimento di tutti i moduli. Il VDIP1 va inserito con il connettore USB rivolto verso l'esterno. Il modulo RTC può essere inserito in un solo modo, dato che i suoi pin non sono disposti in modo simmetrico. Riguardo i moduli radio e Ethernet, il primo va inserito sull'apposito connettore con le scritte capovolte, mentre il secondo, può essere inserito in un solo modo, ma va saldato. Notate che non è (volutamente) possibile montare sul circuito stampato entrambi i moduli. I due pulsanti rosso e nero possono essere fissati direttamente allo stampato oppure montati all'esterno, sull'eventuale contenitore plastico. Il Fingerprint è fissato meccanicamente sul lato piste tramite quattro torrette da 2,5 MA da 13 mm; il collegamento elettrico viene operato tramite un flat cable a 10 conduttori.



Varie:

- Plug alimentazione
- Zoccolo 4+4 passo 7,62 mm
- Zoccolo 12+12 passo 15,24 mm
- Zoccolo 20+20 passo 15,24 mm
- Antenna accordata (Vedi testo)
- Dissipatore (TE19)
- Connettore maschio POD 10 vie 90°
- Connettore POD 10 vie a crimpare (2 pz.)
- Flat cable
- Strip maschio 16 vie
- Strip femmina 16 vie

- Strip femmina 9 vie (2 pz.)
- Strip femmina 3 vie (2 pz.)
- Strip femmina 2 vie (2 pz.)
- Vite 8 mm 3 MA
- Vite 6 mm 3 MA (20 pz.)
- Vite 6 mm 2,5 MA (8 pz.)
- Dato 3 MA
- Torretta 13 mm 3 MA F/F (6 pz.)
- Torretta 13 mm 2,5 MA F/F (4 pz.)
- Torretta 10 mm 3 MA F/F (4 pz.)
- circuito stampato (S0736)

te scaricare gratuitamente il master in formato 1:1 dal nostro sito (www.elettronica.in.it), da stampare su lucido così da poter realizzare il c.s. con la tecnica della fotoincisione. Lo stampato non è problematico da realizzare e tanto le piste quanto le piazzole sono di dimensioni generose. Nel caso, comunque, non ve la sentiate di procedere con l'autocostruzione, potete richiederlo a un'azienda che realizza gli stampati (come ad esempio la MD S.R.L., www.mdsrl.it). Quale che sia la scelta effettuata, ora che avete a disposizione il circuito stampato, potete iniziare il montaggio partendo, come sempre, dai componenti a basso profilo quali resistenze, diodi, condensatori non polarizzati, il trimmer R13 e gli zoccoli DIL 20+20 e 4+4 per il microcontrollore e per il DC-DC converter. Proseguite, quindi, saldando i connettori pin-strip a passo 2,54 mm, necessari per poter ospitare il radiomodem (U1), l'RTC (U4) e il VDIP1 (U2). L'EM202 (U8), invece, non necessita di alcun connettore in quanto può solo essere saldato direttamente sullo stampato pertanto, qualora abbiate optato per la connessione al PC remoto tramite Ethernet, aspettate a montare il convertitore. Saldate ora l'interruttore e i quattro pulsanti a 90° (SW1 e P1÷P4), il quarzo, i condensatori elettrolitici, la bobina L1 del DC-DC converter, il buzzer il connettore per l'alimentazione e lo stabilizzatore lineare, quest'ultimo montato a 90° sul circuito e provvisto, come visibile sulle illustrazioni del piano di montaggio, di un minuscolo dissipatore a U. Montate per ultimo il connettore U7, cui andrà collegato il flat-cable del fingerprint. Girate, ora, il circuito e montate sul lato componenti i due led (quello verde è a sinistra, tenendo il PCB come visibile nelle illustrazioni del piano di montaggio). I due pulsanti P5 (nero) e P6 (rosso) possono essere montati sia direttamente sui fori presenti sullo stampato e collegati elettricamente ai due connettori

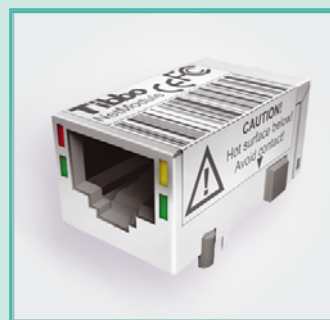


Dispositivo in grado di effettuare la conversione da seriale a Ethernet. Integrato all'interno di un connettore RJ45, è particolarmente compatto e dispone di quattro led di segnalazione posti sul connettore. La seriale, a livello TTL, è in grado di funzionare sia in full-duplex che in half-duplex, con velocità di trasmissione programmabile fino a 115,2 kbps. È compatibile sia con tutti gli altri moduli di casa Tibbo che con i relativi software applicativi. La porta Ethernet supporta entrambi i formati 10BaseT (10 Mbit/s) e 100BaseT (100 Mbit/s).

Caratteristiche tecniche

- Dimensioni: 32,3 x 19 x 16 mm.
- Alimentazione / consumo: 5 Vdc / 230 mA.
- Accesso rete: 10/100BaseT, connettore RJ45 + trasformatore.
- Porta seriale: full-duplex e half-duplex, a livello TTL (la modalità full-duplex è indicata per funzionare con porte RS232 e RS422, mentre, in modalità half-duplex, la trasmissione dei dati mediante i protocolli RS232/422/485 non è supportata).
- Baudrate: configurabile fino a 115.200 bps.

- Parità supportate: nessuna, pari, dispari, mark, space.
- Formato dati: 7 e 8 bit.
- Criteri seriali supportati: RTS, CTS, DTR e DSR; ciascuna linea può essere utilizzata per in controllo remoto di I/O.
- Indicazione dello stato di funzionamento: mediante 4 LED.
- Buffer dati: 2 da 12 kbit.
- EEprom interna.
- Protocolli di rete supportati: UDP, TCP, ARP, ICMP (PING) e DHCP.
- Setup del modulo: tramite porta seriale, protocollo TCP/IP, UDP/IP o Telnet.
- Aggiornamento firmware: sia tramite porta seriale che rete (TCP/IP).



P5 e P6 attraverso due cavetti, sia sul pannello di un eventuale contenitore plastico. A titolo informativo, sappiate che noi abbiamo progettato lo stampato affinché possa essere inserito in un contenitore modello HS.3 prodotto da TEK0 ed acquistabile presso www.futurashop.it (codice HS). Date un'ultima occhiata al circuito, alla ricerca di eventuali errori e "baffi di stagno" quindi passate alla fase successiva che consiste nella messa in opera.

Verifica e messa in opera

Per prima cosa, è necessario controllare che le tensioni siano corrette: innestate sul circuito (su cui non devono essere ancora inseriti né il microcontrollore né il display né alcuno degli altri moduli) il solo DC-DC converter quindi fornite alimentazione al circuito (9 Vdc, 200 mA, tramite un qualsiasi alimentatore, anche non stabilizzato) e verificate che la tensione di 5 V sia presente sui pin

- 4 del display,
- 11 e 32 del micro,
- 1 dell'RTC,
- 1 del VDIP1,
- 7 dell'EM202.

Verificate ora che i 3,3 V provenienti dallo stabilizzatore lineare siano pre-

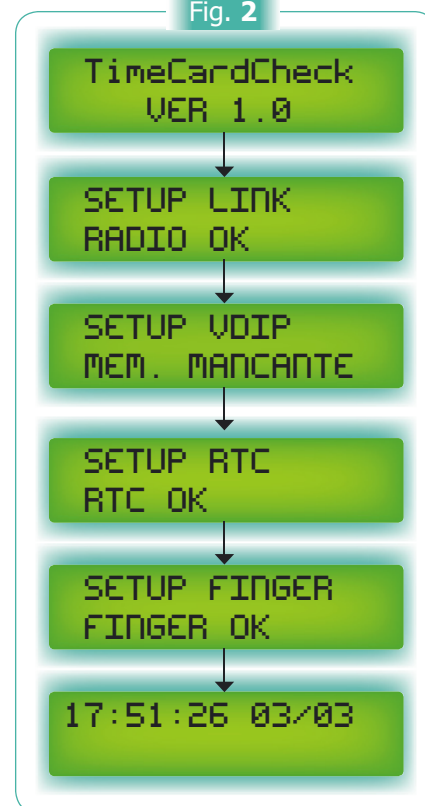
senti sui pin:

- 17 del radiomodem XTR-7020
- 7 e 8 del connettore del fingerprint.

Spegnete il circuito e attendete per qualche minuto che tutti i condensatori si siano scaricati, quindi inserite nell'apposito zoccolo il microcontrollore. Quest'ultimo deve essere programmato esternamente, tramite un programmatore idoneo (quale il nostro FT386) con il programma MF736.HEX, scaricabile anch'esso gratuitamente dal sito della rivista. Inserite infine il display e riaccendete il cartellino orario: per prima cosa dovete regolare il contrasto del display. Su quest'ultimo potete leggere per qualche secondo il nome del dispositivo e la versione FW, seguita da una serie di segnalazioni di errore nell'inizializzazione dei vari moduli attualmente non montati, segnalazioni di cui non vi dovete preoccupare. Una volta verificato che le tensioni siano quelle richieste e che il micro funzioni correttamente, potrete spegnere il dispositivo ed inserire nelle rispettive sedi i componenti mancanti, ossia il modulo RTC, il radiomodem e il convertitore seriale-USB, aiutandovi per il corretto posizionamento con le figure riportate nel piano di montaggio. Relativamente al

fingerprint, esso non viene montato su di uno zoccolo come tutti gli altri moduli, bensì su di quattro torrette esagonali da 2,5MA lunghe 13 mm, poste sullo stesso lato in cui è posi-

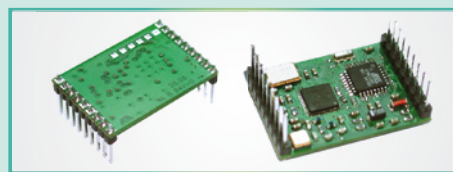
Fig. 2



zionato il display. Il collegamento alla base viene effettuato per mezzo di un connettore flat cable della lun-

Il modulo XTR-7020A

Caratteristiche	XTR-7020A-4	XTR-7020A-8	Unità
Tensione di alimentazione	2,7 ÷ 3,6	2,7 ÷ 3,6	Vdc
Corrente consumata (RX mode)	26	26	mA
Corrente consumata (Tx mode @ -8 dBm)	20	20	mA
Corrente consumata (Tx mode @ 10 dBm)	31	31	mA
Corrente consumata (Stand-by mode)	8 ÷ 10	18 ÷ 20	μA
Tipo di modulazione	FSK	FSK	-
Sensibilità in Rx (velocità RF @ 115200 bps)	-100	-100	dBm
Potenza di trasmissione (Tx)	-8 ÷ +10	-8 ÷ +10	dBm
Bit rate seriale	9,6 - 19,2 - 38,4 57,6 - 115,2	9,6 - 19,2 - 38,4 57,6 - 115,2	kbps
Outdoor range	300	300	m
Frequenza di lavoro	434	868	MHz
Canali/canalizzazione	10 / 153,6	7 / 153,6	-/kHz



Il modulo XTR-7020A è un radiomodem bidirezionale, prodotto da Aurel (www.aurelwireless.com) e disponibile nelle versioni a 434 (7020A-4) e 868 (7020A-8) MHz, completamente compatibili sia come pin-out che come comandi software; sono, inoltre, pin-to-pin compatibili con il modulo XTR-903A-4. Il dispositivo comunica con qualsiasi tipo di microcontrollore attraverso una seriale bidirezionale a livello TTL (0÷3V) e invia i dati via radio modulando la portante RF in FSK. La velocità della seriale può essere

variata da 9,6 a 115,2 kbps sia tramite due pin (SP1 e SP2) che tramite un registro software (S5). È configurabile con un set di comandi AT che permettono di scegliere il livello di potenza RF, il canale di trasmissione e differenti modi di funzionamento.

ghezza di 15 cm che, partendo dal fingerprint, giunge al connettore U7. Riaccendete il cartellino orario e verificate che appaiano, in sequenza, le schermate riportate in Fig. 2. Il setup

del “File Nomi”, descritto al paragrafo successivo, fallisce (SETUP VDIIP MEM. MANCANTE) in quanto nel VDIIP1 non è attualmente inserita alcuna pen drive contenente, appunto,

il file dei nomi delle persone che devono essere riconosciute, comunque questo argomento verrà spiegato nel prossimo paragrafo. Tenete presente che ora è necessario riavviare il dispositivo in quanto, durante la fase di setup dei vari moduli, vengono inviati ad ognuno di essi dei comandi di configurazione che hanno effetto solo dopo il riavvio. A questo punto potete preparare il file NOMI.TXT.

Il modulo VDIIP1



Prodotto dalla FTDI, il VDIIP1 è il primo modulo USB Host disponibile sul mercato: interfacciato ad un microcontrollore dotato di porta seriale, SPI (Serial Peripheral Interface) o parallela, permette di controllare diversi tipi di periferiche. Il dispositivo è basato sul chip VNC1L-1A della FTDI, chip che è stato programmato per gestire in modo semplice e trasparente dispositivi come pen drive USB e convertitori USB/seriali. Il firmware implementa un set di istruzioni (ricco ma contemporaneamente semplice) che consentono di gestire la struttura della FAT8 della pen drive, permettendo così di creare direttori, cartelle e file esattamente come si farebbe tramite un PC.

Caratteristiche tecniche

- Connettore USB: di tipo ‘A’ (quello del PC).
- Seconda porta USB: disponibile sui pin.
- Tipo di interfaccia: UART, parallela FIFO, o SPI (Serial Peripheral Interface), configurabile tramite jumper.
- Alimentazione: 5 Vdc.
- Uscita tensione supplementare: 3,3 Vdc, 200 mA massimi.
- Indicazioni luminose: Led per alimentazione e Led dati USB.
- Aggiornamento firmware: via USB Flash Disk o interfaccia UART / parallela FIFO / SPI.
- Conforme alle direttive RoHS.
- Schemi e firmware: scaricabili dal sito www.vinculum.com.

I File NOMI e TIMBRA

A fronte del riconoscimento di un’impronta valida, il cartellino orario può attribuire ad essa, e visualizzare a display, un breve testo quale ad esempio il nome della persona cui appartiene l’impronta.

Per fare ciò, è necessario che il dispositivo veda, attraverso il convertitore seriale-USB host VDIIP1, una pen drive di qualsiasi tipo che contenga il “File Nomi”, ossia un file in formato testo, chiamato “NOMI.TXT” e creato da PC. Tale file stabilisce la corrispondenza tra identificativo numerico dell’utente e testo assegnato ad esso secondo il formato

<#ID><spazio><Nome persona><,;><CRLF>

in cui <#ID> è un numero compreso tra 1 e 250, <spazio> è un carattere ➤

re di spazio (barra della tastiera del PC), <Nome persona> non necessita di commenti, <,> è la virgola e <CRLF> corrisponde a “a capo”. Un esempio di file nomi potrebbe essere il seguente

001 Mario Rossi,
002 Franco Neri,
004 Giuseppe Verdi,
005 Filippo Bianchi,
006 Amedeo Rosa,
007 ...,

in cui potete notare che gli identificativi numerici 1, 2, 4, 5 e 6 sono assegnati ad un nome. Nel corso delle prossime puntate vi mostreremo come assegnare lo stesso identificativo alfanumerico (nome) a differenti identificativi numerici, in modo da poter memorizzare più impronte digitali per utente.

Durante il normale funzionamento, se il modulo VDIP1 e la pen drive sono presenti, il cartellino orario va ad aggiornare (eventualmente creare,

se non dovesse esistere) un altro file di testo, chiamato TIMBRA.TXT. Questo file altro non è che la registrazione degli eventi di accesso (entrata – uscita) relativi ad ogni utente. Ogni evento, che occupa una linea, viene aggiunto in coda al file ed è descritto con una sintassi molto semplice:

<AA/MM/GG><spazio><OO:MM:SS><spazio><#ID><spazio><Tipo evento><CRLF>

Il primo campo è la data, espressa in formato anno/mese/giorno, formato scelto perché consente di ordinare facilmente gli eventi in ordine crescente o decrescente quando il file viene importato, ad esempio, in Excel; segue l’ora, separata da uno spazio. Quindi l’identificativo numerico e il tipo di evento, separati ancora da uno spazio. Termina la sequenza di dati il campo <CRLF> che, mandando a capo il testo, identifica l’inizio di un nuovo evento. Relativamente al campo <Tipo evento>, esso può valere E, U o X, a seconda del modo di

funzionamento impostato: questo argomento, però, verrà approfondito nel corso delle due puntate successive.

In questa sede, invece, vi anticipiamo che se il cartel-

lino è programmato nei modi 0 o 1 (entrata/uscita), tale campo può valere E per entrata e U per uscita; nei modi 2 e 3 (lettura continua), invece, esso vale sempre X. Un file TIMBRA.TXT tipico assomiglia allo stralcio riportato qui di seguito.

08/03/03 08:29:09 028 X
08/03/03 11:55:09 001 X
08/03/03 12:01:52 001 U
08/03/03 12:02:39 001 E
08/03/03 12:02:55 001 U
08/03/03 12:05:03 001 X
08/03/03 14:07:26 033 X

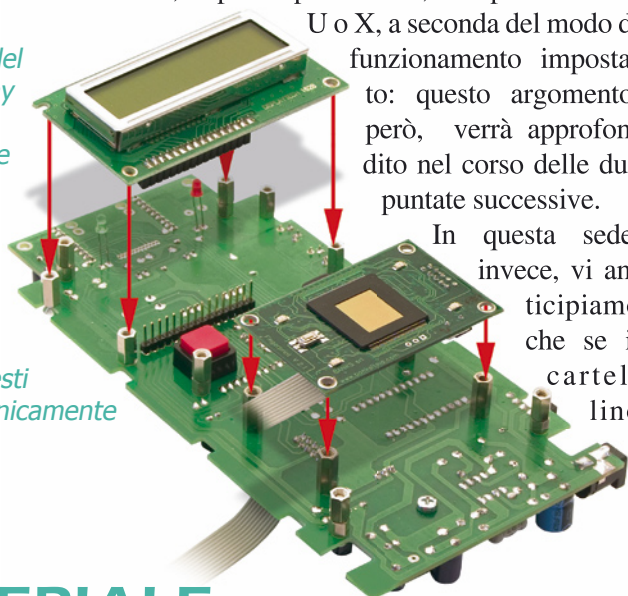
Come potete notare, è piuttosto facile affermare che “l’impronta dell’utente 28 è stata riconosciuta il giorno 3 marzo 2008 alle ore 8:29”.

A questo punto diamo per scontato che abbiate preparato il file NOMI.TXT, che l’abbiate posto nella pen drive e che la pen drive sia inserita nell’apposito connettore del VDIP1. Spegnete il cartellino e riaccendetelo: ora il setup del file nomi ha esito positivo.

Note conclusive

Per oggi ci fermiamo qui: avete realizzato e collaudato il cartellino orario che, a questo punto, è pronto per essere utilizzato. Nella prossima puntata vi mostreremo come configurare il dispositivo, come impostarlo in base alle vostre necessità, come usarlo in locale e come importare in Excel il file TIMBRA.TXT. Nella terza (e ultima) puntata, infine, analizzeremo i tratti salienti del firmware e l’utilizzo tramite il programma Windows.

Fig. 3 - Particolare del montaggio del display e del fingerprint. Notate che le torrette del display (10 mm) sono più corte di quelle del lettore di impronta (13 mm). Il flat cable va collegato al modulo finger prima che questi venga fissato meccanicamente al circuito stampato.



per il

MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Il master del circuito stampato ed il firmware col quale programmare il microcontrollore possono essere scaricati gratuitamente dal sito della rivista (www.elettronica.in.it). Il microcontrollore che sovrintende a tutte le funzioni del dispositivo è disponibile anche già programmato (cod. MF736) al prezzo di 25,00 Euro.

Il materiale va richiesto a: Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)

Tel: 0331-799775 ~ Fax: 0331-792287 ~ <http://www.futurashop.it>

Tastiera PS/2 programmabile per PC

dell'Ing. *Renzo Morelli*

Interposto tra la tastiera e il PC, è in grado di intercettare e memorizzare fino a sedici sequenze di codici provenienti dalla tastiera. Tali sequenze possono essere riprodotte, all'occorrenza, tramite la semplice pressione di uno dei pulsanti della sua tastiera a matrice. Viene alimentato direttamente dalla porta PS/2 e dispone di una porta seriale per il controllo e la configurazione.

Alcune settimane fa, nelle note conclusive dell'articolo relativo al protocollo PS/2 (pubblicato sul fascicolo N. 124), avevamo fornito alcune anticipazioni riguardo un progetto pratico: in tale occasione avevamo detto che stavamo lavorando ad una piccola tastiera PS/2 programmabile, in grado di memorizzare e riproporre, con la semplice pressione di un pulsante, delle sequenze di tasti (anche lunghe) che, normalmente, dovrebbero essere digitate tramite la tastiera del PC.

Bene, in queste pagine vi presentiamo il progetto completo del dispositivo, il cui codice è FT740.

Si tratta di un tastierino a matrice 4x4 collegabile a un PC sia da solo che in unione a una tastiera standard; abbiamo deciso di definire "tastierino" questo dispositivo perché, tipicamente, con tale termine si intende una tastiera PS/2 dotata di un numero di tasti estremamente ridotto: in Fig. 4, per esempio, viene mostrato un tastierino che ripropone solo la sezione numerica di una tastiera standard. Tornando al nostro progetto, la sua peculiarità sta nel fatto che i sedici tasti di cui dispone sono liberamente programmabili dall'utente per poter riprodurre ciascuno una sequenza che va da un minimo di uno a un

massimo di 15 codici (Scan Code) di tasti. Scopriamo insieme come è stato realizzato il progetto.

Caratteristiche tecniche

Come dicevamo, la caratteristica principale di questo tastierino è rappresentata dalla sua capacità di memorizzare e riprodurre sedici stringhe di Scan Code: la programmazione avviene in autoapprendimento semplicemente selezionando il tasto di interesse e, successivamente, digitando sulla tastiera del PC l'esatta sequenza di codici che vogliamo associare a quel tasto. Tali codici vengono poi salvati su EEPROM affinché possano rimanere memorizzati anche in assenza di tensione di alimentazione e senza la necessità di una batteria di backup. L'alimentazione viene prelevata direttamente dalla porta PS/2 del PC pertanto il nostro tastierino non necessita di alcun adattatore di rete aggiuntivo. Dispone, inoltre, di una porta seriale a livello RS232 tramite la quale è possibile effettuare il "monitor in linea" della scheda: in altre parole, con l'ausilio di qualsiasi programma di emulazione di terminale (Hyper Terminal, Procomm Plus) e pochi semplici comandi, è possibile leggere, scrivere e cancellare la EEPROM, verificare le sequenze di scan code memorizzati, eccetera.

Un dispositivo come il nostro presenta molteplici possibilità di utilizzo: pensate, ad esempio a tutte le volte in cui la registrazione on-line su alcuni siti Internet ha prodotto, come conferma di registrazione avvenuta, delle coppie nome utente / password impossibili da ricordare a causa della presenza di numeri e lettere (queste ultime sia maiuscole che minuscole) generate in ordine causale. Spesso, dopo il primo accesso, è possibile modificare i dati personali per renderli più facili da ricordare, ma talvolta ciò non è possibile (ad esempio con alcuni siti per la ricarica on-line di carte di credito

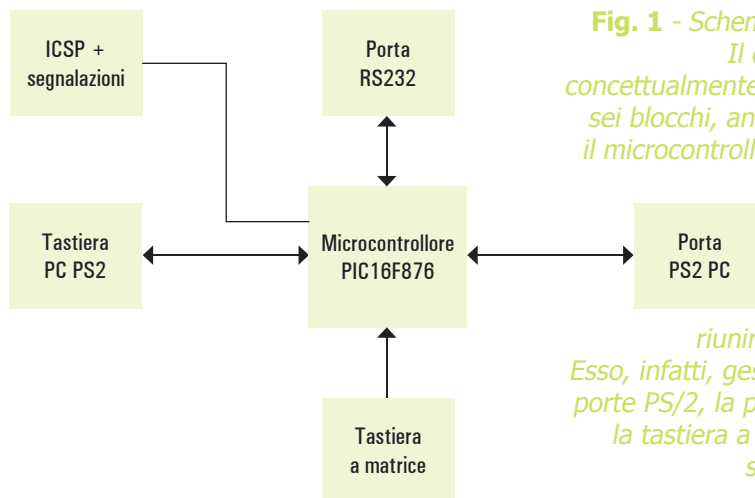


Fig. 1 - Schema a blocchi.
Il dispositivo è concettualmente divisibile in sei blocchi, anche se poi è il microcontrollore stesso a

riunirli tutti in sé. Esso, infatti, gestisce le due porte PS/2, la porta seriale, la tastiera a matrice e le segnalazioni.

- **Microcontrollore:** PIC 16F876
- **Interfacce:**
 - **Connettore MiniDin:** per il collegamento via cavo alla porta PS/2 di un PC;
 - **Connettore MiniDin:** per il collegamento alla tastiera PS/2;
 - **Connettore DB9:** per collegamento e gestione tramite la seriale di un PC con programma di emulazione di terminale.
 - **Connettore ICSP:** per la programmazione del micro.
- **Segnalazioni:** un led verde e uno rosso.
- **Input:** tastiera a matrice 4x4 (16 tasti) e porta seriale.
- **Output:** 1 ÷ 16 stringhe di 1 ÷ 15 scan code.
- **Alimentazione:** 5 Vdc ricavati da porta PS2, 100 mA massimi.

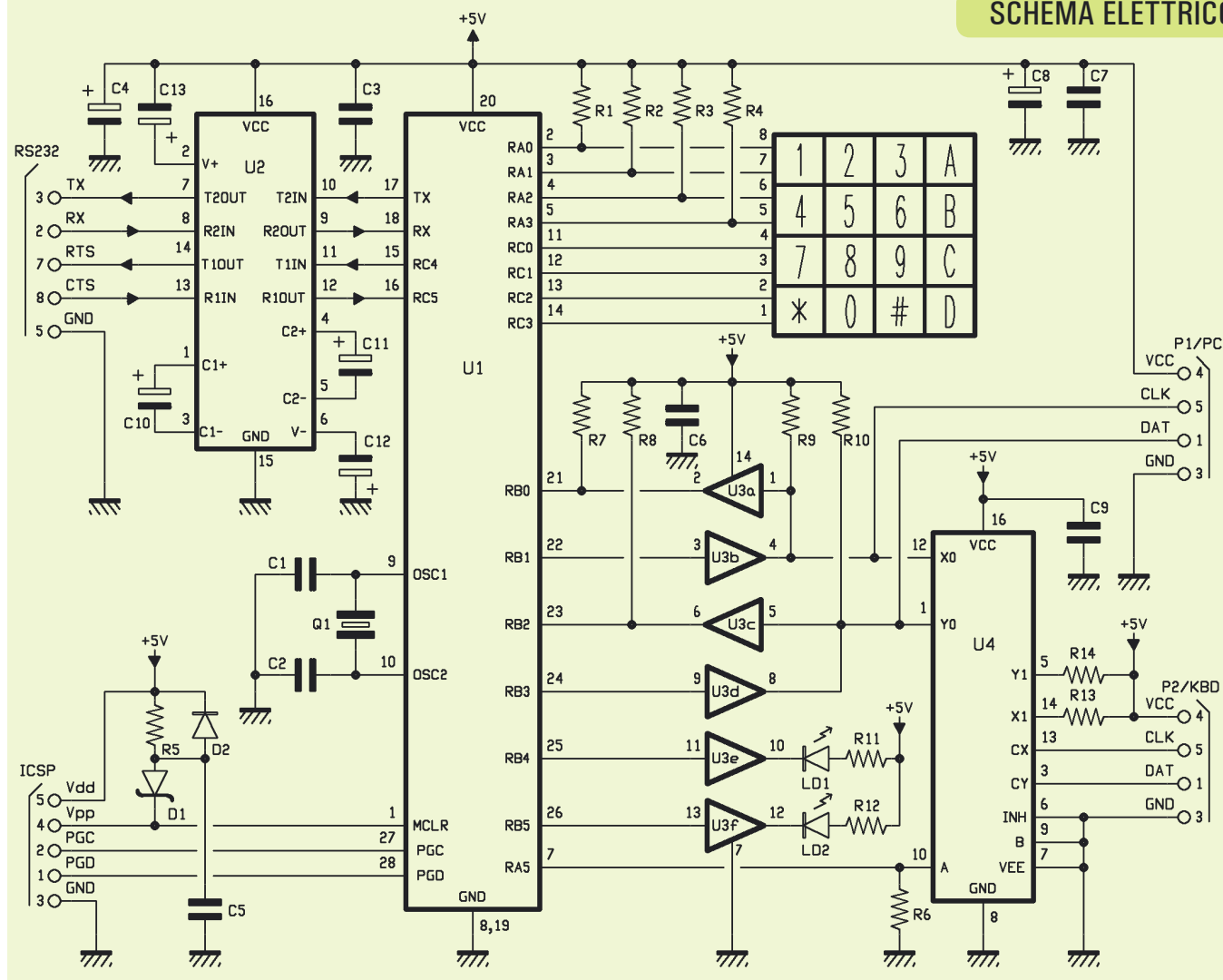
Specifiche tecniche

prepagate, oppure con alcuni forum di discussione). Ebbene, sia il login che la password potrebbero venire memorizzati nel tastierino e, all'occorrenza, richiamati con la semplice pressione di due tasti.

Un'altra valida applicazione può essere la memorizzazione di particolari sequenze di comando riconosciute dal software con cui si lavora normalmente. Il programma con cui noi della Redazione impaginiamo gli articoli è un eccellente esempio per l'utilizzo del nostro tastierino: la verifica preliminare del file viene eseguita con il comando ALT+SHIFT+CNTRL+F; la creazione di una copia di back-up, invece, si effettua con ALT+CNTRL+S. Il comando per avanzare diretta-

mente all'ultima pagina, infine, è SHIFT+CNTRL+PGDWN. Come potete constatare, non sono comandi immediati da ricordare.

Si possono presentare applicazioni varie anche a livello industriale. Un nostro collega ci diceva di avere visto da un suo parente una macchina per ricamo dotata di un PC industriale: tale macchina poteva essere gestita completamente tramite una tastiera PS/2 standard (in metallo) collocata sul pannello principale, e parzialmente (comandi come "solleva tutti gli aghi", "riposiziona i rulli", "allinea il carro"...), tramite un tastierino posizionato sul lato opposto della macchina, utilizzato solo nelle procedure di manutenzione ed avvio. Come avete constatato,



le possibilità di impiego, per quanto specifiche, sono molteplici.

Schema a blocchi ed elettrico

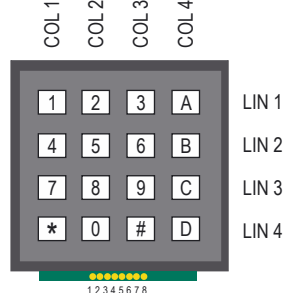
In Fig. 1 è schematizzata la struttura del tastierino; tenete presente che,

nonostante la suddivisione in blocchi per facilitare la comprensione del dispositivo, in realtà è il microcontrollore stesso ad occuparsi di tutto. Come avrete certamente intuito il blocco principale è rappresentato dal micro U1, un PIC16F876A

programmato per poter gestire contemporaneamente la comunicazione con le tre porte (due PS/2 e una seriale). Gestisce, inoltre la tastiera a matrice ed il blocco di segnalazione e riprogrammazione.

Iniziamo la descrizione dello schema elettrico partendo dalla tastiera: si tratta di una configurazione classica a matrice "4 linee x 4 colonne" (vedere Fig. 2), gestibile dal micro tramite quattro ingressi (RA0, RA1, RA2 e RA3, che rilevano lo stato delle linee della matrice) e quattro uscite (RC0, RC1, RC2 e RC3, che gestiscono la scansione delle colonne). Il principio di funzionamento è molto semplice: il microcontrollore attiva una colonna portando a zero logico lo stato dell'uscita corrispon-

Fig. 2 - Pin-out della tastiera e corrispondenza linea-colonna per ogni tasto.



OUTPUT PIN	SIMBOLO
1	COL 1
2	COL 2
3	COL 3
4	COL 4
5	LIN 1
6	LIN 2
7	LIN 3
8	LIN 4

dente ad essa, quindi controlla lo stato logico degli ingressi da RA0 a RA3, per verificare se una delle quattro linee è a zero logico (un tasto è premuto). Se nessun tasto risulta premuto, tutti gli ingressi si trovano a livello logico alto (determinato dalle resistenze di pull-up da R1 a R4). Vediamo un esempio pratico facendo riferimento alla Fig. 2: ipotizziamo che il micro abbia portato a zero logico la linea RC0 e che veda la linea RA2 allo stesso livello; in questa situazione è attiva la combinazione “Colonna 4 / Linea 2” che corrisponde al tasto B.

Esaurito l'argomento “tastiera a matrice” vediamo ora il collegamento seriale: dicevamo che il nostro dispositivo può essere controllato tramite un PC e un programma di emulazione di terminale. Il collegamento si effettua attraverso le quattro linee TX/RX (trasmissione e ricezione) e RC4/RC5 (controllo di flusso RTS/CTS, necessario per far sì che tanto la tastiera quanto il PC si accorgano dell'esistenza del collegamento), opportunamente adattate di livello per mezzo del chip U2, il classico driver RS232 modello MAX232 contenente due buffer di trasmissione, due di ricezione nonché il DC-DC converter che genera le tensioni di +10 e -10 Vdc partendo dai 5 V di alimentazione della linea PS/2.

Nella parte dello schema in basso a sinistra sono presenti il connettore ICSP e i componenti discreti R5-D1-D2-C5, utilizzati per la programmazione In-Circuit del PIC.

Abbiamo lasciato volutamente per ultima la descrizione del gruppo interfacce PS/2 perché necessita di un'analisi più approfondita. Identifichiamo, innanzitutto le possibili situazioni che si possono verificare dal punto di vista del traffico dati:

- 1) la tastiera invia dati al PC e il tastierino è in ascolto (come monitor o come apprendimento);
- 2) il PC invia dati alla tastiera e il tastierino è in ascolto;

3) il tastierino invia dati al PC e la tastiera viene temporaneamente isolata.

Notate come ognuno dei tre dispositivi possa, in qualche modo, comunicare con gli altri: questa situazione non è gestibile con un collegamento parallelo delle linee Clock e Data (che ricordiamo essere entrambe bidirezionali open collector), in quanto il rischio di conflitti sul bus PS/2 (punto-punto) è troppo elevato, specialmente quando al PC sono collegati sia la tastiera che il tastierino.

Abbiamo pertanto dovuto inserire un circuito di commutazione, in grado di isolare la tastiera quando il tastierino deve prendere il controllo. Tale circuito è rappresentato dal chip U4, un doppio multiplexer analogico bidirezionale “quattro in uno” modello 4052, concettualmente identico ad un commutatore meccanico *quattro posizioni due vie*.

Le linee CX e CY (il comune dei due multiplexer) portano Clock e Data alla tastiera, le linee X0 e Y0 (canale 1) vanno sia alla porta del PC che al microcontrollore; le linee X1 e Y1, vengono collegate al positivo tramite dei pull-up. Infine, la linea

collegate direttamente alle linee X0 e Y0; in questo modo esiste un collegamento diretto bidirezionale tra tastiera PS/2 e PC.

Nel caso in cui il tastierino debba inviare dati al PC (caso 3), per prima cosa il micro porta la linea RA5 a livello alto: vengono così creati i collegamenti CX-X1 e CY-Y1 che deviano le linee Clock e Data della sola tastiera alle resistenze di pull-up R14 e R13, isolandola di fatto dal computer. A questo punto la porta del PS2 del PC vede solamente il tastierino che può, pertanto, scambiare dati con essa. Terminato l'invio da parte del tastierino, la linea RA5 viene nuovamente portata a zero, ripristinando così il collegamento diretto tra tastiera e PC.

Non rimane, ora, che descrivere il circuito che collega il micro al bus PS/2. Sappiamo che le linee Clock e Data sono bidirezionali e che la direzione viene determinata dal modo in cui le due linee vengono attivate prima dell'inizio di una trasmissione. A questo punto, per intercettare dati e clock in transito, abbiamo a disposizione due possibili metodi hardware: utilizzare due linee del

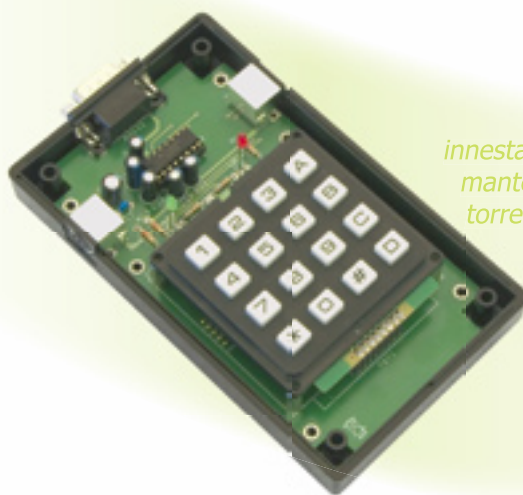


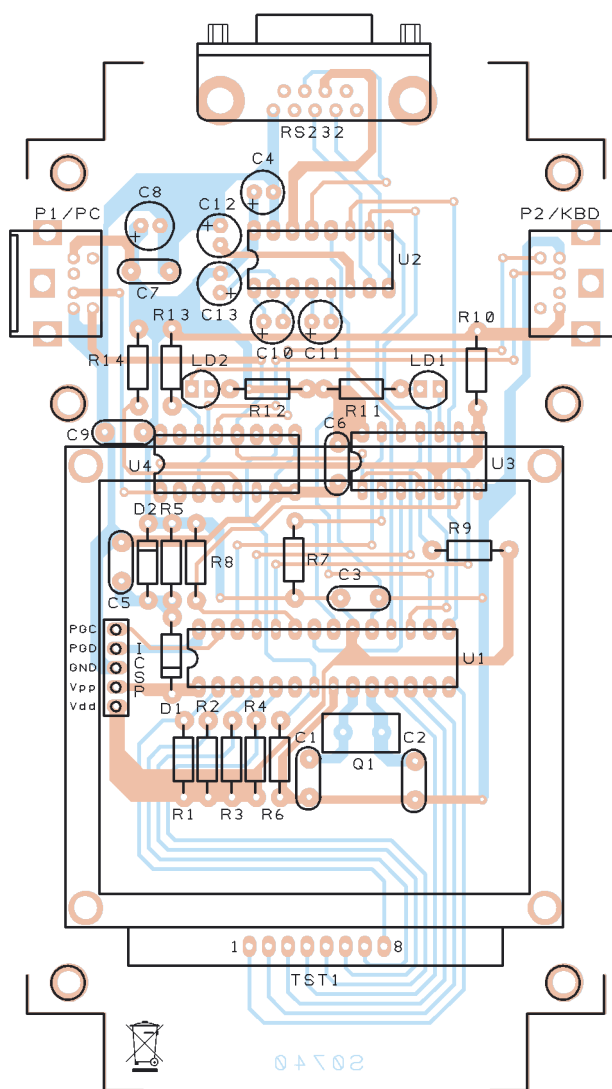
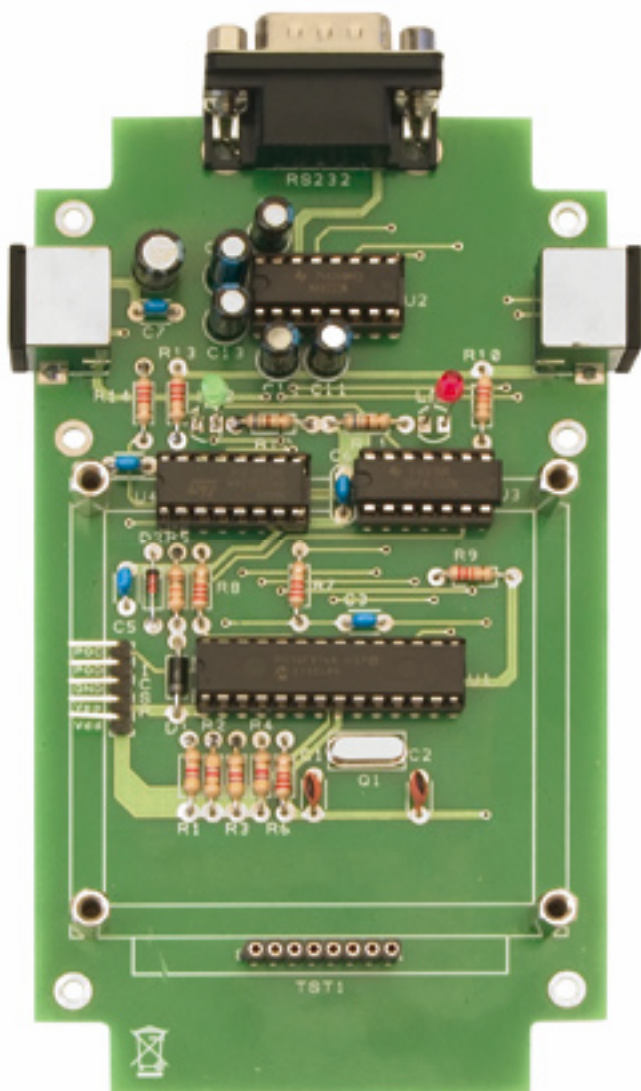
Fig. 3 - Il dispositivo inserito nel contenitore Teko.

La tastiera a matrice, una volta innestata nell'apposito connettore, viene mantenuta in posizione tramite quattro torrette esagonali da 15 mm di altezza e relative viti da 3MA.

I due led devono essere saldati a circa 14 mm di altezza, in modo che le due teste possano fuoriuscire dal coperchio del contenitore, dopo che sarà stato opportunamente forato.

A (che seleziona il canale dei multiplexer) viene pilotata direttamente con il pin RA5 del microcontrollore. In condizione di normale funzionamento (i casi 1 e 2 visti poche righe sopra), la linea RA5 è a zero logico, situazione che fa commutare il 4052 in modo che le linee CX e CY siano

micro e creare una routine software molto complessa in grado di variarne dinamicamente il verso, oppure utilizzare quattro linee (due in ingresso e due in uscita) con l'ausilio di hardware esterno, che, complicando di poco il circuito, alleggerisce di molto il firmware.



ELENCO COMPONENTI:

R1÷R4: 4,7 kohm
 R5: 10 kohm
 R6÷R8, R13÷R15: 2,2 kohm
 R9, R10: 22 kohm
 R11, R12: 680 ohm
 C1, C2: 22 pF ceramico
 C3, C5÷C7, C9: 100 nF multistrato
 C4: 1 µF 100 VL elettrolitico
 C8: 10 µF 16 VL elettrolitico
 C10÷C13: 1 µF 100 VL elettrolitico

U1: PIC16F876A (MF740)
 U2: MAX232
 U3: SN7407
 U4: CD4052
 D1: 1N5819
 D2: 1N4148
 LD1: led 3 mm rosso
 LD2: led 3 mm verde
 Q1: Quarzo 4 MHz
 TST1: Tastiera matrice 4x4

Varie:
 - Connettore PS2 femmina da CS (2 pz.)
 - Connettore DB9 maschio
 - Zoccolo 7+7
 - Zoccolo 8+8 (2 pz.)
 - Zoccolo 14+14
 - Strip maschio 5 poli
 - Strip maschio 8 poli
 - Strip femmina 8 poli
 - Circuito stampato (S0740)

Abbiamo optato per questa seconda soluzione: notate la presenza dei buffer open collector (modello SN7407) U3a-U3b sulla linea Clock e U3c-U3d sulla linea Data in configurazione “antiparallelo” (ossia parallelo ingresso con uscita), possibile proprio perché le uscite sono

open collector. Questa soluzione permette al micro, opportunamente programmato, di gestire una linea di input ed una linea di output che si affaccino fisicamente su uno stesso filo (Clock o Data che sia), nel pieno rispetto del protocollo PS/2. Rimangono a disposizione due buf-

fer che sono stati utilizzati per pilotare i led di segnalazione LD1 e LD2.

Realizzazione pratica

Realizzare in pratica questo tastierino è abbastanza facile. Dopo che avrete sviluppato il c.s. (ricordiamo ➤

che il master doppia faccia in scala 1:1 può essere scaricato gratuitamente dal nostro sito www.elettro-nicain.it, forato opportunamente le varie piazzole e saldato le varie “via” (ponticello passante che collega due piste posizionate su facce opposte), potrete iniziare a saldare i componenti a basso profilo (resistenze, condensatori non polarizzati, zoccoli per i circuiti integrati e quarzo). Proseguite quindi, con i condensatori elettrolitici e i diodi (aiutatevi con le illustrazioni del piano di montaggio per identificare il verso corretto), lasciando per ultimi i connettori PS/2 e RS232 (DB9), i led e il connettore ICSP (pin strip a cinque poli, passo 2,54 mm). Controllate che il montaggio sia stato eseguito perfettamente, quindi inserite i tre chip nei rispettivi zoccoli. Il micro deve essere programmato con il codice esadecimale MF740.hex (scaricabile anch’esso gratuitamente da Internet), collegando al connettore ICSP un programmatore idoneo come i nostri FT284 e FT386. Potete infine inserire il tastierino in un contenitore di dimensioni adeguate. Sappiate che noi abbiamo progettato il circuito per essere inserito in un contenitore modello Teko 660 (Fig. 3).

Utilizzo in pratica

Il tastierino va collegato dal lato del connettore P1 all’ingresso tastiera del PC e dall’altro (connettore P2) ad una eventuale tastiera standard PS/2 (Fig. 5); può funzionare in tre



Fig. 4 - Tastierino alfanumerico PS2 commerciale

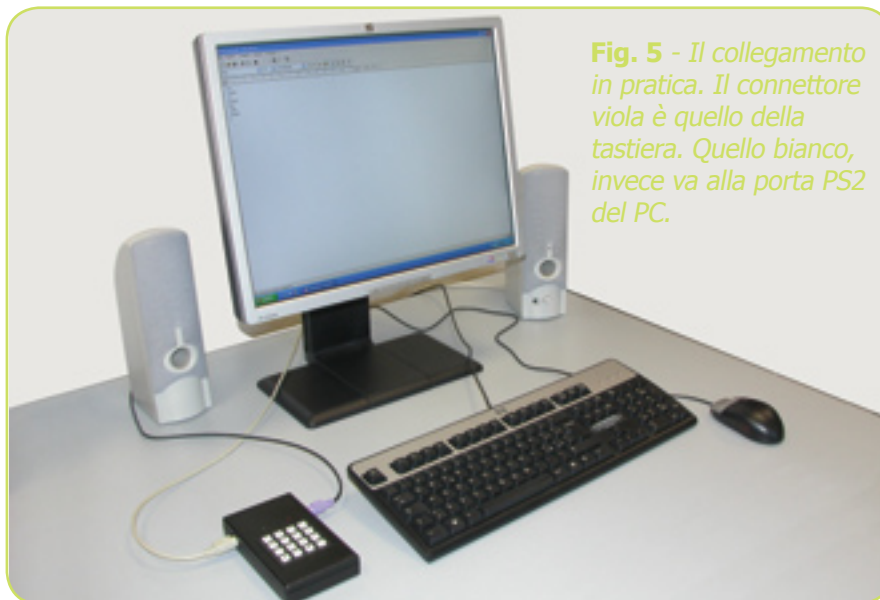


Fig. 5 - Il collegamento in pratica. Il connettore viola è quello della tastiera. Quello bianco, invece va alla porta PS2 del PC.

modi differenti, scelti dall’utente premendo un particolare tasto durante la fase di inizializzazione di dieci secondi, quest’ultima segnalata dal lampeggio di entrambi i led.

Modo autoapprendimento

È la modalità utilizzata per programmare le sequenze di codici che vogliamo associare ai vari tasti; viene avviata premendo e rilasciando il tasto A durante l’inizializzazione ed è segnalata dalla combinazione luminosa “led verde acceso, led rosso spento”. La programmazione delle sequenze si effettua in tre fasi

- 1) pressione del pulsante da programmare,
- 2) pressione dei tasti da tastiera (al massimo 15),
- 3) pressione dello stesso pulsante, a chiusura della programmazione.

Ipotizziamo di voler programmare il tasto 1, che premiamo brevemente: il led verde inizia a lampeggiare, comunicando che il micro è pronto per registrare la sequenza di codici da associare al tasto. Da ora in poi, ogni tasto premuto sulla tastiera PS/2 viene memorizzato: la ricezione e successiva memorizzazione di uno scan code viene confermata da un lampeggio del led rosso. Terminiamo la programmazione del tasto 1 premendolo nuovamente: il

led verde smette di lampeggiare. È possibile programmare tutti gli altri tasti eseguendo la stessa procedura. Per uscire dalla modalità autoapprendimento, entrando nella modalità di funzionamento normale, occorre premere per sei volte il tasto C. In questa condizione entrambi i led sono spenti.

Modo normale

Si ha quando sia la tastiera che il tastierino sono collegati al PC; si attiva automaticamente (senza pertanto agire su alcun tasto) allo scadere dei dieci secondi che scandiscono la fase di inizializzazione. Come già visto, viene segnalata dai led spenti. Premendo uno qualsiasi dei tasti programmati, il tastierino esclude temporaneamente la tastiera ed invia al PC la sequenza dei codici programmati per quel tasto. Se il tasto non è stato ancora programmato, viene inviato al PC solo lo scan code predefinito (1 per il tasto 1, 2 per il tasto 2, eccetera). La risposta ad eventuali comandi inviati dal PC, viene data solo dalla tastiera standard.

Modo master

Al PC è collegato il solo tastierino; avviata premendo 0 durante l’inizializzazione, viene segnalata dalla combinazione “led verde spento e ➤

Listato 1

Il tastierino comunica con il PC e con la tastiera esterna tramite i segnali di Clock (CLOCKTX, CLOCKRX) e di Dati (DATITX, DATIRX), che vengono collegati alle relative linee della tastiera (CLKKBD, DATIKBD su P2) e del Pc (CLOCK, DATI su P1) come segue:

- Il tastierino è inattivo: <ENKBD=0, Tastiera esterna abilitata> <CLOCK=CLKKBD, clock generato da tastiera esterna> <DATI=DATIKBD, trasmissione dati da tastiera esterna a PC e viceversa>.
- Il tastierino invia uno o più scan code o un comando al PC: <ENKBD=1, Tastiera esterna disabilitata> <CLOCK=CLOCKTX, clock generato dal tastierino> <DATI=DATITX, dati generati dal tastierino sul fronte di salita di CLOCKTX> <CLKKBD=Vcc e DATIKBD=Vcc, tastiera PS2 isolata>.
- Il tastierino è in ricezione dalla tastiera esterna (Modo Autoapprendimento): <ENKBD=0, Tastiera esterna abilitata> <CLOCKRX=CLKKBD, clock generato dalla tastiera esterna> <DATIRX=DATIKBD, dati trasmessi da tastiera esterna e letti dal tastierino sul fronte di discesa di CLOCKRX> <CLOCK=CLKKBD, clock generato tastiera esterna> <DATI=DATIKBD, dati generati da tastiera esterna e ricevuti dal PC>.
- Il tastierino sta ricevendo un comando da PC (Modo Master): <ENKBD=1, Tastiera esterna disabilitata> <CLOCK=CLOCKTX, clock generato dal tastierino> <DATIRX=DATI, il PC trasmette i dati che vengono letti dal tastierino sul fronte di discesa di CLOCKTX> <CLKKBD=Vcc e DATI KBD=Vcc, tastiera esterna isolata>.

Riportiamo di seguito il codice sorgente relativo alla sola gestione del flusso dati PS2. Tenete presente che il sorgente completo può essere scaricato gratuitamente, come sempre, dal nostro sito www.elettronica.in.it.

```

/*****
Tx bit b via PS2
*****/
void TxBitPS2(UCHAR b)
{
#define CLKL 40 //Durata livello Low = 40[usec].
#define CLKH 20 //Semi durata livello High = 20[usec].
DATITX=b; //Trasmette il bit b.
Delay_us(CLKH); //Seconda metà impulso CLOCKTX=1.
CLKTX=0; //Fronte discesa CLOCKTX.
//Il PC legge DATITX.
Delay_us(CLKL); //Impulso CLOCKTX=0.
CLKTX=1; //Fronte salita CLOCKTX: il tastierino
//può trasmettere il bit successivo.
Delay_us(CLKH); //Prima metà impulso CLOCKTX=1.
}

/*****
Tx byte d via PS2
*****/
UCHAR TxBytePS2(UCHAR d)
{
UCHAR f0k; //
UCHAR BitPar; //
BitPar=CalcoloParityOdd(d); //BitPar=1 se d ha un
//numero pari di 1
while (!DATIRX || !CLKRX); //DATIRX=CLKRX=1, Pc Libero.
DisableClkKbd(); //DATIKBD=CLKKBD=Vcc
//Tastiera Disabilitata
DATITX=1; //
CLKTX=1; //
TxBitPS2(0); //Tx Start Bit = 0
TxBitPS2(d.F0); //Tx Bit 0 (LSB) di d
TxBitPS2(d.F1); //Tx Bit 1 di d
TxBitPS2(d.F2); //Tx Bit 2 di d
TxBitPS2(d.F3); //Tx Bit 3 di d
TxBitPS2(d.F4); //Tx Bit 4 di d
TxBitPS2(d.F5); //Tx Bit 5 di d
TxBitPS2(d.F6); //Tx Bit 6 di d
TxBitPS2(d.F7); //Tx Bit 7 (MSB) di d
TxBitPS2(BitPar); //Tx Bit Parity Odd
TxBitPS2(1); //Tx Stop Bit = 1

while (CLKRX);while (!CLKRX); //Attesa fronte discesa
//CLOCKRX: Pronto Bit ACK da Host.
if (!DATIRX) f0k=0; else f0k=1; //DATIRX=0>PC richiede
//ritrasmissione (RESEND)>f0k=0.
//DATIRX=1>ACK da Pc> f0k=1
EnableClkKbd(); //DATIKBD=DATI e
//CLKKBD=CLOCK;
//Tastiera Riabilitata.
return(f0k);
}

/*****
Tx Codice KeyCode di un tasto premuto
*****/
void TxKeyCodePS2(UCHAR KeyCode)
{
TxBytePS2(KeyCode); //Tx KeyCode = Scan Code tasto.
}

/*****
Tx Codice KeyCode di un tasto rilasciato
*****/
void TxBreakKeyCodePS2(UCHAR KeyCode)
{
TxBytePS2(BREAK_CODE); //Tx Break Code = 0xF0
TxBytePS2(KeyCode); //Tx ScanCode tasto premuto.
}

/*****
Rd come Keyboard bit Tx da Pc >
Tx Clock e legge DATIRX su fronte di salita di CLOCKTX
*****/
UCHAR RxBitPS2daHost(void)
{
CLKTX=0;
Delay_us(CLKL); //Impulso CLOCKTX=0.
CLKTX=1; //Fronte salita CLOCKTX: legge bit tx da PC.
Delay_us(CLKH); //Prima metà impulso CLOCKTX=1
return(DATIRX); //DATIRX=bit tx da Pc
}

/*****
Rd come Keyboard Byte Tx da Pc
*****/
void RxBytePS2daHost(void)
{
d=0x00;
d.F0=RxBitPS2daHost; //Rx Bit 0 (LSB)
d.F1=RxBitPS2daHost; //Rx Bit 1
d.F2=RxBitPS2daHost; //Rx Bit 2
d.F3=RxBitPS2daHost; //Rx Bit 3
d.F4=RxBitPS2daHost; //Rx Bit 4
d.F5=RxBitPS2daHost; //Rx Bit 5
d.F6=RxBitPS2daHost; //Rx Bit 6
d.F7=RxBitPS2daHost; //Rx Bit 7 (MSB)
ParityPs2= RxBitPS2daHost; //Rx Bit Parity (Odd)
StopPs2=RxBitPS2daHost; //Rx Stop Bit
TxBitPS2(0); //Tx Bit ACK = 0
DATITX=1; //Linea Dati Idle
return(d);
}

/*****
Rd come Host bit Tx da Keyboard esterna >
Rx Clock e legge DATIRX su fronte di discesa di CLOCKRX
*****/
UCHAR RxBitPS2daKbd(void)
{
while (!CLKRX);while (CLKRX); //Attesa discesa CLOCKRX
return(DATIRX); //DATIRX=bit Tx tast. Esterna
}

/*****
Rd come Host Byte Tx da Keyboard Esterna
*****/
UCHAR RxBytePS2daKbd(void)
{
d=0x00;
d.F0=RxBitPS2daKbd; //Rx Bit 0 (LSB)
d.F1=RxBitPS2daKbd; //Rx Bit 1
d.F2=RxBitPS2daKbd; //Rx Bit 2
d.F3=RxBitPS2daKbd; //Rx Bit 3
d.F4=RxBitPS2daKbd; //Rx Bit 4
d.F5=RxBitPS2daKbd; //Rx Bit 5
d.F6=RxBitPS2daKbd; //Rx Bit 6
d.F7=RxBitPS2daKbd; //Rx Bit 7 (MSB)
ParityPs2= RxBitPS2daKbd; //Rx Bit Parity (Odd)
StopPs2=RxBitPS2daKbd; //Rx Stop Bit
return(d);
}

/*****
Routine di risposta all'interrupt provocato da una
variazione di livello (fronte) di CLOCKRX, ogni volta che
vogliamo leggere un byte tx da PC o da Keyboard esterna.
*****/
void RxPS2Isr(void)
{
if (!DATIRX) //0>Bit Start>inizio rx byte
{
if (fHost) //Se fHost=1 Modo Host quindi
//tx tast. esterna.
BufPs2[BufPs2IdxWr]=RxBytePS2daKbd();
Else //Rx Clock e Rd DATIRX sulla discesa di CLOCKRX
//HA!trimenti Modo normale:
//dati tx da Pc.
BufPs2[BufPs2IdxWr]=RxBytePS2daHost();
//Tx Clock e Rd DATIRX sulla salita di CLOCKTX

BufParityPs2[BufPs2IdxWr++]=ParityPs2;
if (BufPs2IdxWr>MAXBUFPS2) BufPs2IdxWr=0;

ClearInterruptRBO //Reset flag Interrupt
//CLOCKRX (RBO)
}
}

```

Tabella 1 - I comandi seriali

Funzione	Comando	Risposta tastierino
Richiesta status	I	Trasmette lo Status: Nome + Release + Errori. Ad esempio: FT740 - V1.00 - OK
Eco tasti	E	Abilita la trasmissione su seriale degli scan code dei tasti premuti e rilasciati sulla tastiera PS/2. Abilita la trasmissione su seriale della sequenza codici del tasto premuto o rilasciato su FT740.
Fine eco tasti	Q	Disabilita la trasmissione su seriale degli scan code dei tasti premuti e rilasciati sulla tastiera PS/2. Disabilita la trasmissione su seriale della sequenza codici del tasto premuto o rilasciato su FT740.
Lettura EEPROM	R	Legge le locazioni di memoria EEPROM che contengono le sequenze di scan code memorizzati e le ripropone sulla seriale.
Cancellazione EEPROM	D	Cancella la zona della memoria EEprom che contiene le sequenze dei codici associate ai tasti.
Scrittura sequenze predefinite	W	Ricarica la configurazione tasti predefinita, corrispondente allo scan code di ogni tasto (1 per il tasto 1, 2 per il tasto 2, eccetera).
Autoapprendimento	A	Avvia manualmente la modalità di autoapprendimento, senza dover passare attraverso la fase di avvio e inizializzazione.

led rosso acceso". Premendo un tasto qualsiasi viene inviata al PC la sequenza dei codici tastiera associati a quel tasto (anche in questo caso se il tasto non è programmato, viene inviato lo scan code predefinito). La risposta ad eventuali comandi inviati dal PC viene data direttamente dal tastierino.

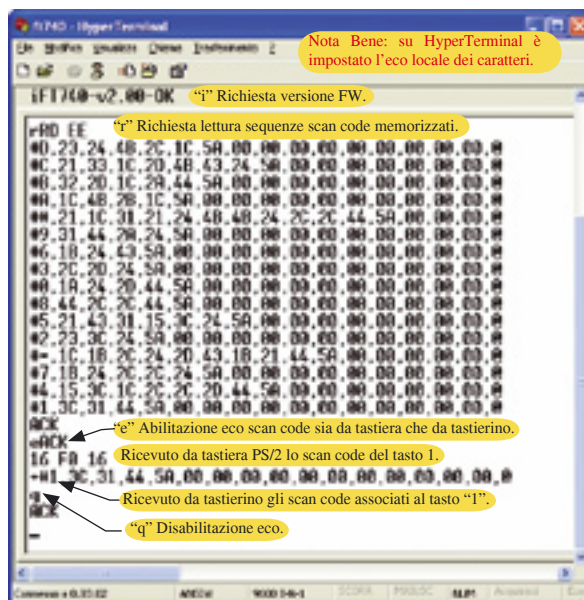
Monitor seriale in RS232

È possibile effettuare il controllo in linea del tastierino collegandolo alla seriale di un PC tramite un classico cavo seriale incrociato (Null-Modem) ed inviando, tramite qualsiasi software di emulazione di terminale (ad esempio HyperTerminal o Procom Plus o altro ancora) dei semplici comandi, riepilogati nella Tabella 1: il collegamento deve essere aperto a 9600 bps, 8, N, 1, con controllo di flusso RTS-CTS. In Fig. 6 è visibile un esempio di log ottenuto tramite la

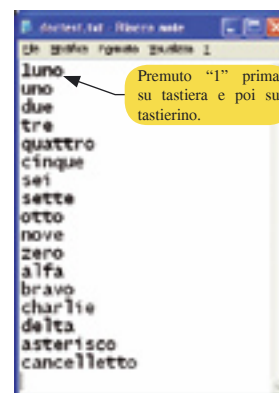
seriale: potete notare che la pressione dei tasti ha generato una stringa differente per ogni singolo tasto del

tastierino e che il tasto "1" premuto sulla tastiera è stato ricevuto dalla sessione "Blocco note" aperta.

Fig. 6 - Monitor da seriale. I comandi "i" e "r" richiedono la versione firmware e le stringhe di scan code memorizzate. "e" e "q" abilitano/disabilitano l'eco degli scan code inviati da tastiera e dal tastierino. Nel



documento di esempio sono stati premuti i tasti "1" di entrambi i dispositivi. Ad ogni tasto corrisponde una stringa di scan code.



per il

MATERIALE

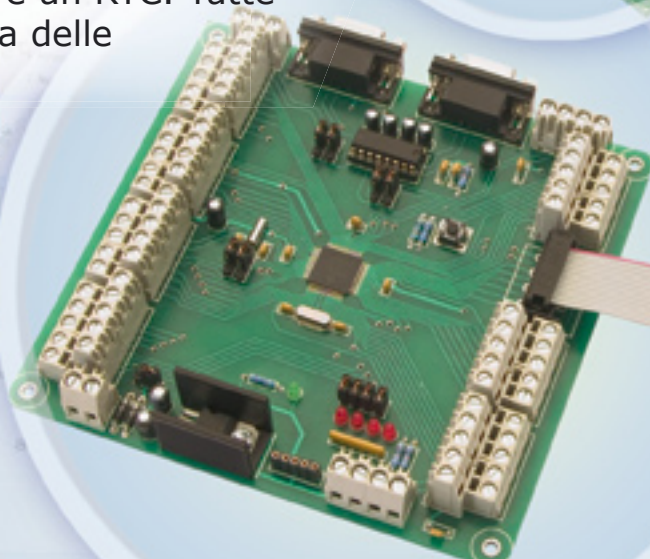
Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Il master del circuito stampato ed il firmware col quale programmare il microcontrollore possono essere scaricati gratuitamente dal sito della rivista (www.elettronica.in.it). Il microcontrollore PIC18F876A è disponibile anche già programmato (cod. MF740) al prezzo di 20,00 Euro, IVA inclusa.

Il materiale va richiesto a: Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)
Tel: 0331-799775 ~ Fax: 0331-792287 ~ <http://www.futurashop.it>

Piattaforma di sviluppo per PIC

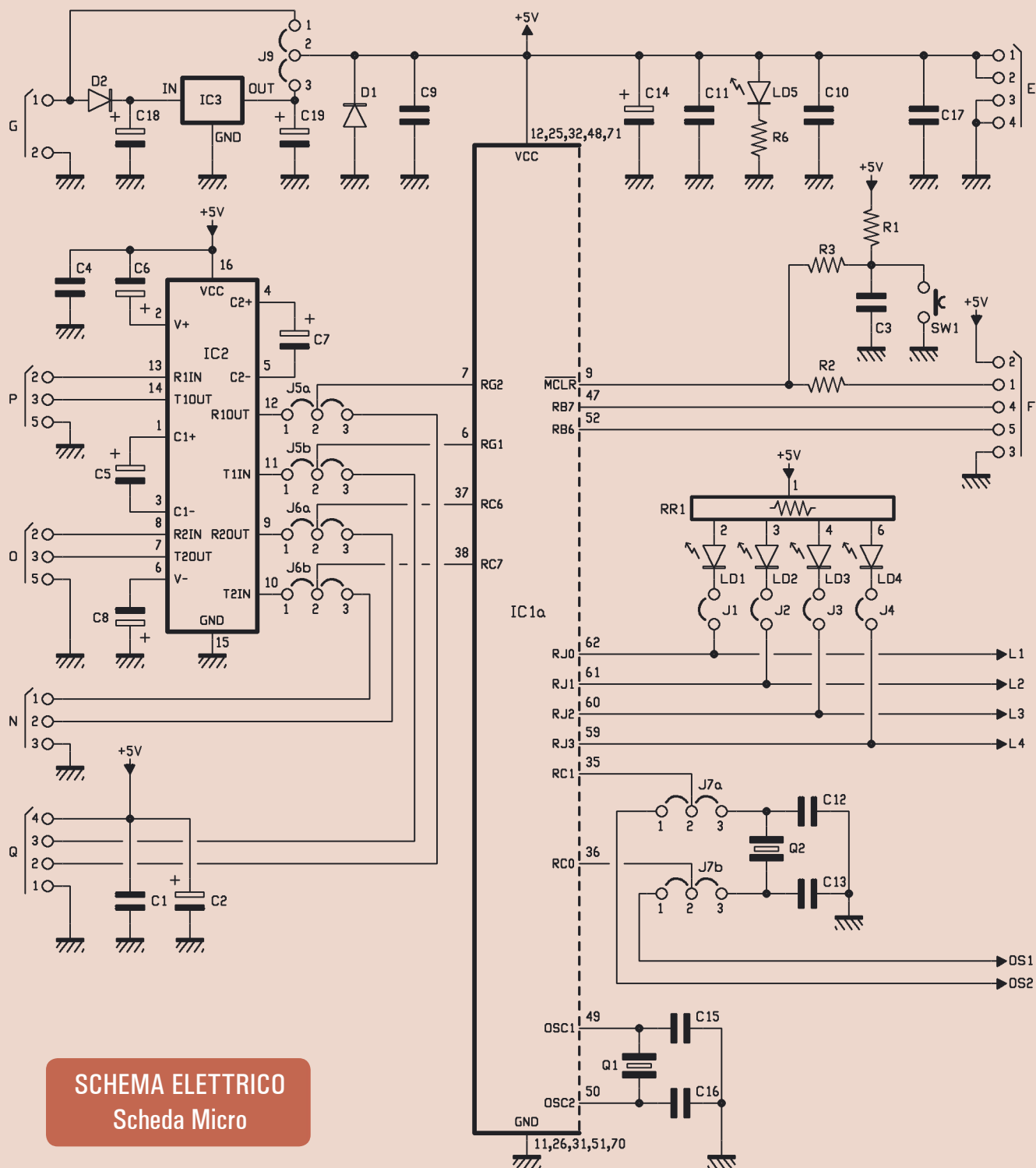
dell'Ing. Vincenzo Ligorio

Piattaforma per lo sviluppo hardware e software di sistemi basati su microcontrollori della famiglia PIC18F8xxx di casa Microchip. È costituito da una scheda base contenente il micro e le interfacce seriali di comunicazione, più un gestore di alimentazione, un display, una pulsantiera e un RTC. Tutte le linee di I/O del micro giungono a delle morsettiere. Prima puntata.



I microcontrollori Microchip sono stati sempre apprezzati dai progettisti per la semplicità d'uso e di programmazione che li contraddistingue, tanto da essere diventati i più usati nella realizzazione di applicazioni sia dilettantistiche che professionali, semplici e di media complessità. Da qualche tempo la casa americana ha deciso di sviluppare dispositivi più prestanti e destinati a un pubblico che necessita microcontrollori caratterizzati da una discreta potenza di calcolo e un nutrito set di porte e funzionalità integrate; così sono nati componenti come il PIC16F8723, un prestante micro da 80 pin (20

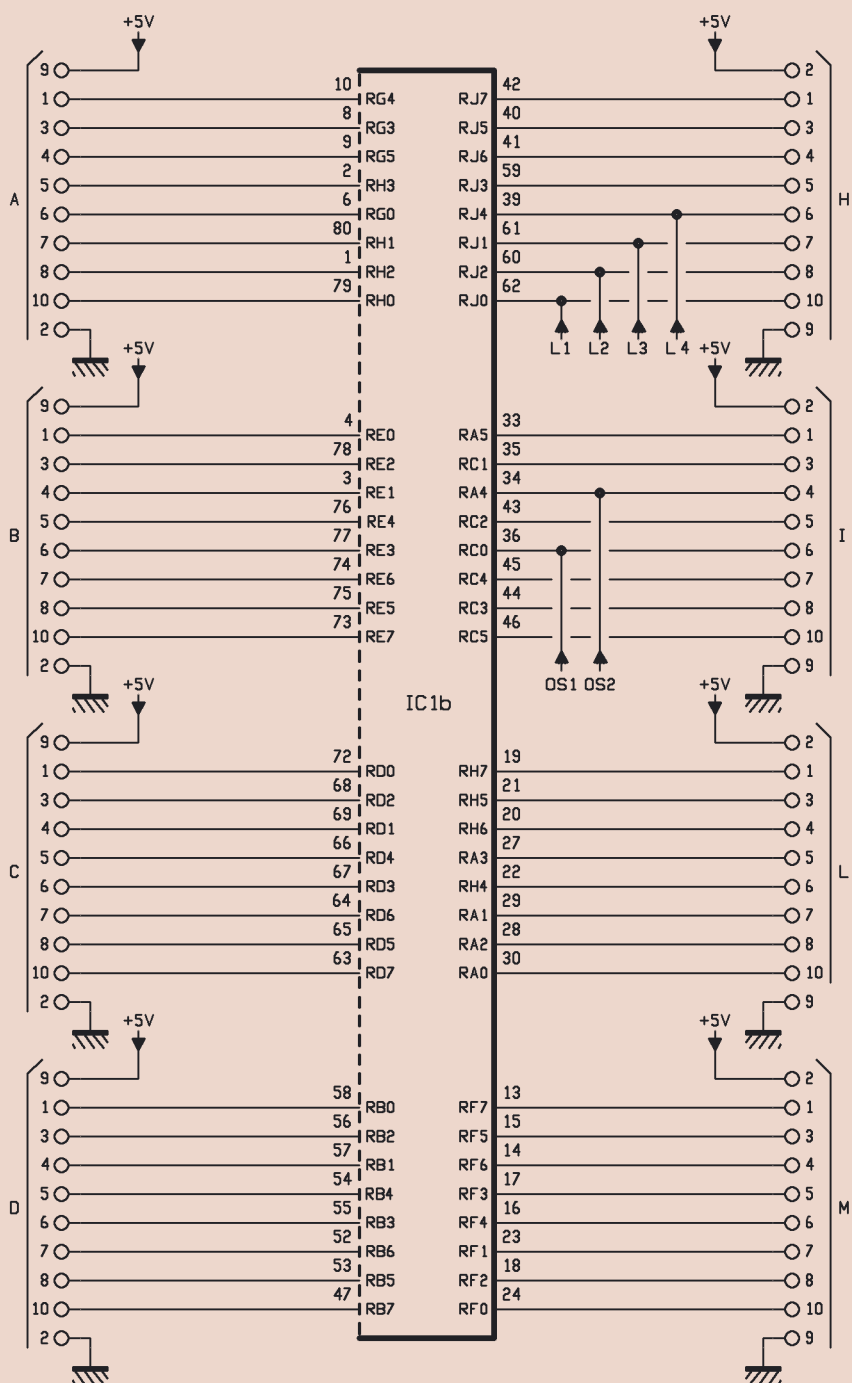
per lato) provvisto di doppio UART e A/D converter a 12 bit assegnabile a un massimo di 16 I/O, oltre che di una memoria di programma da ben 128 kword (ogni word è da 14 bit) innovativa nei riguardi di quelle finora adottate dalla Microchip, perché non segmentata in banchi o pagine differenti. Ciò significa che il programma di gestione può girare più rapidamente perché può chiamare parti che si trovano nella stessa pagina, senza la necessità (e i limiti che ciò comporta) di istruzioni che carichino dati da altre pagine. Un ulteriore incremento della velocità di funzionamento lo si ottiene anche grazie all'oscillatore >



interno a PLL che, opportunamente pilotato con un quarzo esterno da 10 MHz, consente al micro di operare fino a 40 MHz, garantendo così una potenza di calcolo di 10 MIPS. Esiste, comunque, un secondo ingresso clock a 32 kHz che può essere utilizzato per pilotare uno dei timer

di sistema. Qualora non dovesse servire, i due pin richiesti dal quarzo a 32 kHz possono sempre essere utilizzati come I/O generici. Intuendo le grandi possibilità di impiego del PIC18F8723, top gamma della famiglia PIC18F, abbiamo deciso di progettare e proporvi una piattaforma di

sviluppo per tale microcontrollore (che può, però, essere utilizzata con tutti i micro della famiglia) dando a corredo una libreria software molto interessante perché provvista di tutta una serie di routine esemplificative, ciascuna delle quali realizza una delle funzioni più richieste dai progetti-



alle varie porte, che si rivelano molto utili in pratica quando occorre verificare la funzionalità di determinate applicazioni.

Le unità esterne interfacciabili mediante le porte del micro sono: un modulo Real Time Clock, un display a cristalli liquidi ed un gestore dell'alimentazione, utile per applicazioni in cui si richiede che il micro possa spegnersi da solo per risparmiare il massimo di energia. Completa il set di circuiti ausiliari una pulsantiera, assegnabile via software a determinate linee di I/O.

L'unità principale

Delle unità esterne parleremo in un secondo momento; per ora vediamo di analizzare la scheda di base del sistema di sviluppo. In essa prende posto il microcontrollore, del quale portiamo all'esterno tutte le linee di I/O mediante morsettiere e connettori; anzi, per dirla giusta l'unità principale è sostanzialmente il microcontrollore cablato per portarne all'esterno tutte le linee di I/O, più alcuni componenti di contorno che servono ad alimentarlo e a utilizzarlo al meglio. Le linee programmabili per più di un uso, possono essere commutate, a seconda delle esigenze, mediante jumper, così da poterle usare, in base all'applicazione emulata, come meglio credete. Un esempio sono i pin 35 e 36, utilizzati per il clock secondario, che, tramite il doppio jumper J7 possono essere applicati al quarzo oppure portati all'esterno della scheda per essere impiegati come comuni linee di I/O (quando non serve il clock secondario).

Sempre in tema di linee commutabili, notate i pin 6, 7, 37 e 38, che sono i canali dati seriali corrispondenti alle due UART interne; più esattamente, 6 è l'RX della prima porta e 7 il suo TX, mentre 37 è il TX della seconda porta e 38 l'RX corrispondente. Quando serva utilizzare le porte di comunicazione, si devono chiudere i jumper J5 e J6 sui contatti 1 e 2, così ➤

sti; disponendo già delle routine, per simulare una certa operazione non bisogna fare altro che prendere quella corrispondente e sostituire in essa i parametri. Nulla di più facile.

Ma del software applicativo parleremo nella seconda puntata; qui vediamo l'hardware, che consta,

come detto, di un'unità principale e di alcune periferiche già sviluppate. L'unità principale è sostanzialmente una scheda contenente il microcontrollore, del quale rende accessibili all'esterno tutte le linee di I/O e le periferiche integrate. Le periferiche sono semplici circuiti da connettere

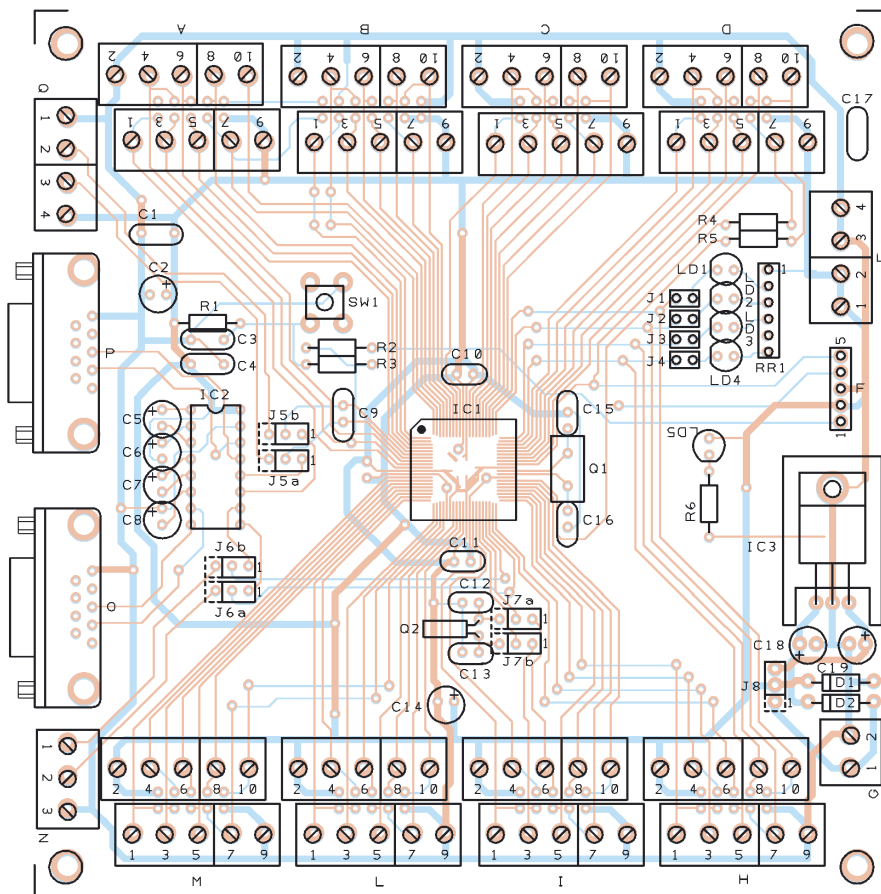
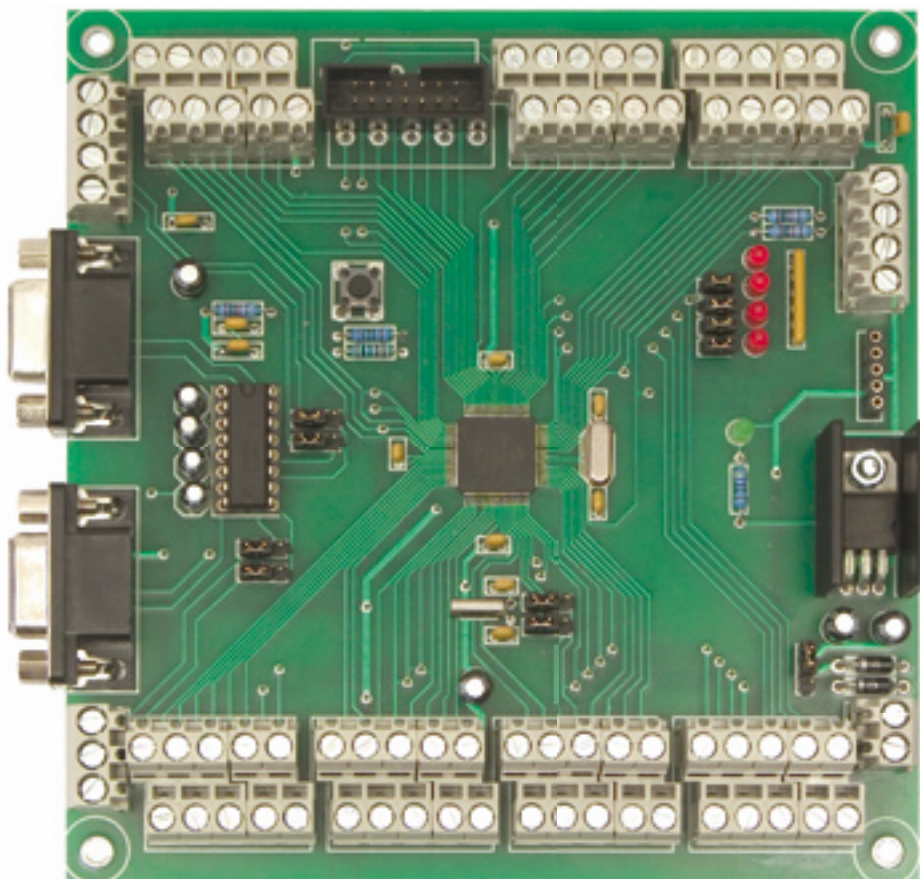
La scheda base trasporta tutti i pin di I/O del micro ad una serie di morsettiere esterne, in modo da rendere estremamente facile il cablaggio di un prototipo sperimentale. Le morsettiere e i jumper di configurazione, infatti, rendono superfluo l'utilizzo del saldatore per collegare la scheda ad altri circuiti stampati. Durante il montaggio, occorre prestare attenzione al micro, che deve essere saldato tramite un saldatore da 30 W a punta fine e abbondante fluxante in gel.

ELENCO COMPONENTI:

R1, R4, R5: 10 kohm
 R2: 160 ohm
 R3: 1 kohm
 R6: 470 ohm
 RR1: Rete resistiva 6x470 ohm + C
 C1, C3, C4: 100 nF multistrato
 C2, C14: 22 µF 25 VL elettrolitico
 C5÷C8: 10 µF 25 VL elettrolitico
 C9÷C11: 100 nF multistrato
 C12, C13, C15, C16: 22pF ceramico
 C17: 100 nF multistrato
 C18, C19: 22 µF 35 VL elettrolitico
 IC1: PIC18F8723-I/PT
 IC2: MAX232
 IC3: 7805
 D1, D2: 1N4007
 LD1÷LD4: led 3 mm rosso
 LD5: led 3 mm verde
 SW1: Microswitch
 Q1: Quarzo 10 MHz
 Q2: Quarzo 32768 Hz

Varie:

- Zoccolo 8+8
- Morsettieria componibile 2 livelli 2+2 vie (8 pz.)
- Morsettieria componibile 2 livelli 3+3 vie (8 pz.)
- Morsettieria 2 poli (5 pz.)
- Connettore DB9 femmina da CS
- Strip maschio 2 vie (4 pz.)
- Strip maschio 3 vie (7 pz.)
- Strip femmina 5 vie
- Jumper (11 pz.)
- Dissipatore (ML26)
- Vite 10 mm 3 MA
- Dado 3 MA
- Circuito stampato



da collegare l'UART ai convertitori TTL/RS232; in caso contrario, le linee in questione possono essere utilizzate diversamente, portandole ai connettori "N" e "Q".

A riguardo bisogna notare che i piedini 6, 7, 37 e 38 del micro possono essere portati direttamente all'esterno sia per utilizzarli allo scopo di gestire periferiche, sia per realizzare comunque linee di comunicazione seriale, solo a livello TTL.

Sull'unità di base, oltre al microcontrollore si trovano altri componenti, utili ad interfacciare meglio il PIC e a ricavare l'alimentazione per esso; vediamoli ordinatamente, iniziando con il converter TTL/RS232: si tratta di un doppio convertitore (contiene due sezioni di conversione da TTL a RS232 ed altrettante da RS232 a TTL) integrato di tipo MAX232, qui utilizzato quando si voglia, ad esempio, collegare il sistema di sviluppo a un dispositivo dotato di porte di comunicazione a standard RS232, quindi anche ad un Personal Com-

puter. Come vedrete più avanti, ciò permette, grazie al software di emulazione previsto, di operare, tramite un programma di terminale, direttamente nel micro, per apportare modifiche e verificare l'esecuzione di procedure (debug via porta seriale). Le linee corrispondenti ai piedini 59, 60, 61 e 62, all'occorrenza possono essere dotate di resistori di pull-up esterni; se serve, basta chiudere i jumper J1, J2, J3 e J4, così da inserire gli elementi della rete resistiva RR1. Ciò non l'abbiamo previsto per gli altri I/O, in quanto i restanti possono essere dotati di resistore di pull-up interno, programmabile da software. Per il microcontrollore è stata prevista la programmazione In-Circuit, da effettuare mediante programmatori quali l'IC2D o lo stesso PICKIT2 originale Microchip, tramite il connettore "F" (Programming); la linea corrispondente al pin 1 è utilizzata altresì come reset e può essere comandata dal pulsante SW1.

L'alimentazione per l'intera unità

base viene applicata alla morsettiera "G"; sono richieste una tensione di 5 Vdc ben stabilizzata e una corrente di circa 200 mA. Se si dispone di un alimentatore con tali requisiti, si può chiudere il ponticello J9 tra 1 e 2, mentre se la tensione disponibile è più alta, occorre inserire il regolatore IC3, che provvede esso stesso a stabilizzarla a 5 volt. In quest'ultimo caso J9 va chiuso fra i punti 2 e 3. Notate che se si usa il regolatore, occorre fornire almeno 8,5 Vdc, dato che circa 3 V cadono tra i terminali di ingresso e uscita (drop-out). Dunque, o si dispone di 5 volt stabilizzati, oppure sono richiesti almeno 8,5 V.

I moduli aggiuntivi

Per consentire la simulazione delle più svariate applicazioni, abbiamo voluto corredare l'unità principale di alcuni moduli fatti per ottenere determinate funzioni, moduli non indispensabili, che ognuno potrà realizzare solo all'occorrenza.

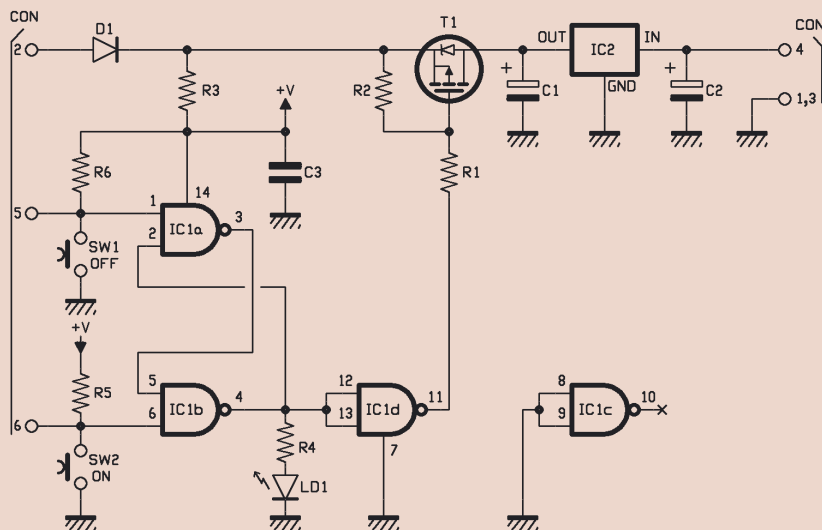
Le unità complementari del sistema di sviluppo che descriviamo sono: un modulo di accensione e spegnimento (controllo dell'alimentazione), un RTC, eventualmente abbinabile al modulo di accensione e spegnimento, una pulsantiera e un display alfanumerico a cristalli liquidi provvisto di controller. Vediamoli dettagliatamente.

Il modulo di accensione/spegnimento

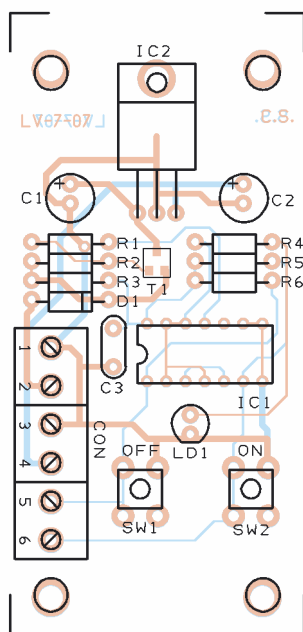
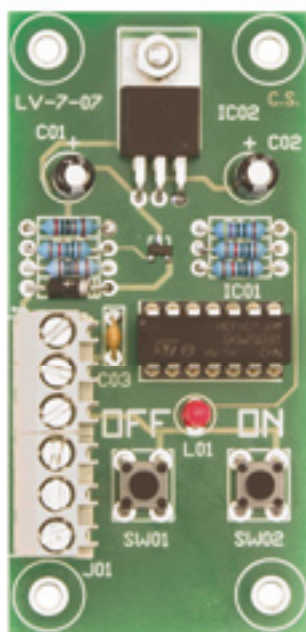
Si tratta di un piccolo circuito contenente una logica molto semplice che comanda un interruttore statico a mosfet e un regolatore integrato di tipo 7805; il modulo va solitamente posto a monte della morsettiera "G" dell'unità principale (in tal caso occorre chiudere il J9 della stessa unità fra 1 e 2, perché i 5 volt stabilizzati li dà il modulo...) e può essere utilizzato quando si desidera comandare da pulsanti o mediante altri dispositivi che diano segnali logici TTL-compatibili l'attività del microcontrollore. ➤



Il sistema di sviluppo completo. Notate, in basso a destra, la scheda che realizza un A/D converter ad alte prestazioni, scheda che sarà l'argomento di un prossimo articolo.



Il modulo di alimentazione consente di accendere e spegnere la scheda base sia manualmente (agendo su SW1 e SW2) sia elettronicamente, in questo caso con l'ausilio di circuiti aggiuntivi (come il nostro RTC) collegati ai pin 5 e/o 6 della morsettiere. Il multivibratore bistabile composto da IC1a e IC1b garantisce spegnimenti e accensioni esenti da condizioni di incertezza.



ELENCO COMPONENTI:

R1, R2, R5, R6: 10 kohm 1%
 R3: 1 kohm 1%
 R4: 499 ohm 1%
 C1, C2: 22 µF 25 VL elettrolitico
 C3: 100 nF multistrato
 D1: 1N4004
 T1: IRLML6402
 IC1: CD4093

IC2: 7805
 SW1, SW2: Microswitch
 LD1: Led 3 mm rosso

Varie:
 - Morsettiere 2 poli (3 pz.)
 - Zoccolo 7+7
 - circuito stampato

Un esempio è il test di quelle routine che prevedono lo spegnimento automatico totale del micro e il risveglio tramite un comando dall'esterno: queste routine, molto utilizzate in pratica, sono finalizzate al risparmio di energia elettrica (batterie).

Per come è strutturato, il modulo potrebbe venire comandato direttamente dall'unità centrale e quindi dal microcontrollore, mediante una linea che deve essere posta a zero logico quando il PIC desidera porsi in spegnimento. Ciò apparirà chiaro osservando lo schema elettrico del modulo: alla morsettiere CON (pin 2) va applicata una tensione continua (meglio se stabilizzata) di almeno 8,5 V, che servirà per alimentare sia la logica su scheda che l'unità principale del sistema di sviluppo; la corrente richiesta è di poche centinaia di microampere per la logica e di quel che serve all'unità di controllo, ossia circa 200 mA. In condizioni di riposo, le due NAND facenti capo ai piedini 1, 2, 3 e 4, 5, 6 (che realizzano un multivibratore bistabile) presentano in uscita rispettivamente 1 e 0 logico; se così non fosse, per portarle in tale stato, che corrisponde allo spegnimento, basta premere il tasto OFF (SW2) che dà un impulso a zero logico al piedino 1. Al contrario, premendo ON (SW1) o inviando un impulso a livello basso al pin 6, ci si porta in accensione.

In condizioni di riposo i piedini 12 e 13 dell'IC1d sono a zero logico, quindi l'11 è a 1; il mosfet T1, che è di tipo a canale P, è interdetto e non conduce fra drain e source, quindi la corrente che attraversa il diodo D1 (inserito per proteggere il modulo dall'inversione di polarità) è solo quella piccola parte che serve a mantenere in funzione le porte logiche. Se giunge un impulso a livello basso sul piedino 6 o si preme per un istante il pulsante ON, l'uscita della NAND IC1b (pin 4) passa a 1 logico e porta questa condizione sul piedino 2; essendo anche l'1 a livello alto (a

ciò provvede il resistore di pull-up) il 3 si porta a zero e blocca il bistabile nella situazione attuale. Il livello alto al piedino 4 forza a zero il pin 11 e determina la piena conduzione del mosfet T1, il quale alimenta il regolatore IC2 fin quando non viene premuto il tasto OFF o al contatto 5 della morsettiera non giunge un impulso a zero logico. Fino ad allora, mediante i contatti 4 (positivo) e 1 e 3 (massa) della morsettiera CON il modulo dà i 5 volt all'unità di controllo e tiene alimentato il microcontrollore. Tale condizione viene evidenziata dall'accensione del led LD1, polarizzato (tramite R4) per effetto dell'1 logico all'uscita della porta IC1b.

Come accennato, il dispositivo di controllo dell'alimentazione può essere usato per testare alcune routine che prevedono lo spegnimento automatico del micro; allo scopo, bisogna impostare una linea di I/O del PIC perché dia, quando si desidera lo spegnimento, zero logico, quindi collegare tale linea al punto 5 della morsettiera CON del modulo di controllo dell'alimentazione.

Per il risveglio, ossia la nuova accensione, serve un impulso a zero logico sul contatto 6; è ovviamente improponibile chiedere al micro (che è spento) di dare detto impulso. Solitamente si può prevedere di destare il PIC dal suo "sonno" quando si verifica un evento su una certa periferica, quale, ad esempio, il presentarsi di una tensione, l'arrivo di una chiamata telefonica (tramite un rilevatore dell'alternata di chiamata) lo spostamento di un microswitch o, perché no, lo scadere di un certo intervallo di tempo o il raggiungimento di un'ora. In quest'ultimo caso di solito si ricorre ad un modulo orologio, come quello che abbiamo previsto.

Il modulo RTC

Nel set di dispositivi complementari spicca un modulo real Time Clock, che potete impiegare sia per gestire l'orologio di sistema nelle applica-

zioni che lo richiedono, sia per programmare ed effettuare il risveglio del micro quando lo stesso si pone automaticamente in stato di riposo (spento).

Come il circuito appena visto, anche questo dispone di una morsettiera a sei poli che ne permette il collegamento all'unità principale del sistema di sviluppo o ad altri circuiti; a bordo ha un RTC molto preciso (un PCF8563, siglato IC1) incapsulato in un contenitore plastico dip a 4 pin per lato, che per funzionare si avvale di un quarzo da 32 kHz collegato fra i pin 1 e 2 e di un condensatore connesso tra il piedino 1 e massa.

Il chip che realizza l'RTC è programmabile mediante un bus I²C che fa capo ai contatti 6 e 7 della morsettiera (rispettivamente, piedini 5 e 6 del componente) e dispone di un'uscita abbinabile ad un evento, secondo, appunto, la programmazione effettuata. Quest'uscita è localizzata al piedino 3 e nel nostro circuito la usiamo per pilotare, direttamente (jumper J3 chiuso fra 2 e 3) o tramite l'adattatore a transistor formato da T1 e T2 (jumper J3 chiuso fra i contatti 1 e 2) il punto 5 della morsettiera di interconnessione. Eventualmente è possibile prevedere il pull-up, mediante R5, chiudendo il jumper J2 fra i punti 1 e 2 (centrale). Dell'adattatore a transistor, si noti che esso non inverte la condizione logica del piedino 3; consente, inoltre, di erogare più corrente di quella prelevabile da quest'ultimo e trasla il livello logico 1 (5 V) dell'RTC ad un livello di 8,5÷15 Vdc, livello idoneo per pilotare relé o altri utilizzatori che assorbano fino a una cinquantina di milliampere o che operino a tensione più alta, come il circuito di accensione. La particolare circuitazione adottata, infine permette di non consumare alcuna corrente quando l'uscita deve essere a riposo; infatti, se il piedino 3 è a 1 logico T1 è interdetto e così pure T2 e non fluisce alcuna corrente nelle



CD3+
Disturbatore Cellulare
All Bands
23W totali min.

Vendita soggetta a limitazioni

NEW



PD-1
Rilevatore tascabile
Cellulare e microspie
GSM, DCS, UMTS
1÷3000 Mhz
Portata 5÷10m
Vibra-beep-led
Batteria ricaricabile



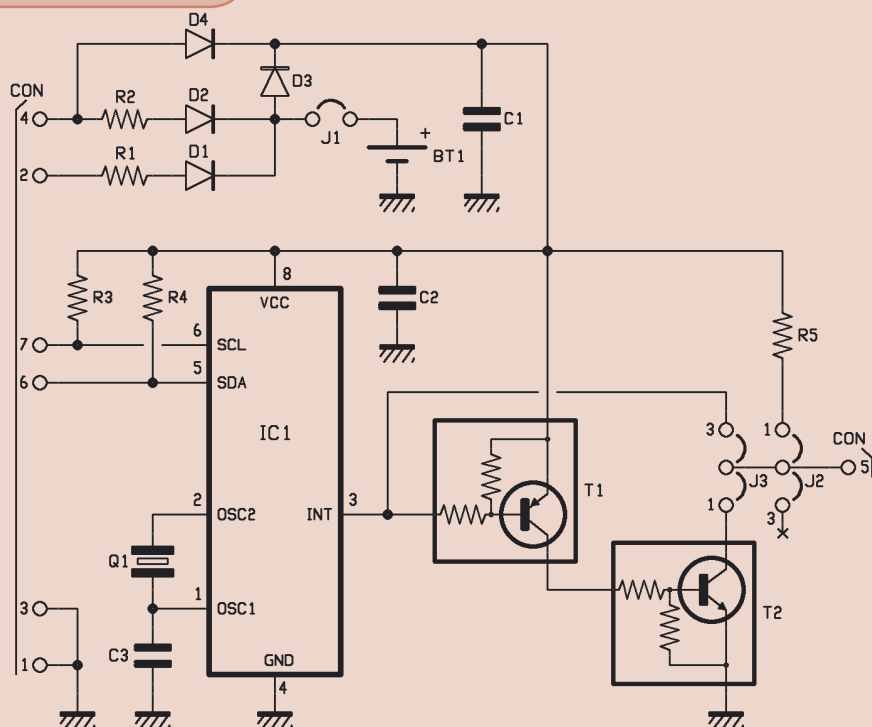
DESPY

Ideale per la ricerca di micro spie fino a 3 Ghz

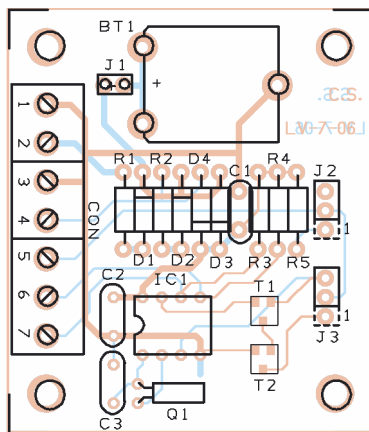
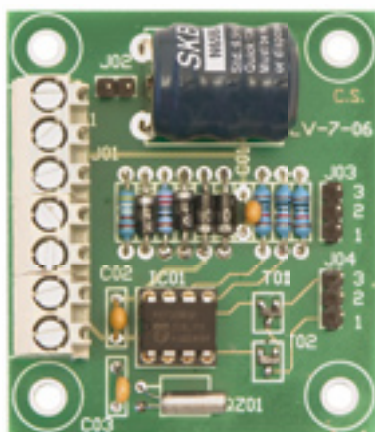
Bias s.r.l.

Strada del Lavoro 119-sub2
Zona Industriale Gualdicciolo
47894-Chiesanuova
Repubblica di San Marino
Tel. 0549-999408 - Fax. 0549-999431

www.bias.sm



Il chip RTC (IC1) viene configurato dalla scheda micro tramite i pin 6 e 7 (I²C Bus) della morsettiera. La batteria tampone (con relativo circuito di ricarica) garantisce la ritenzione di data e ora in assenza di alimentazione. L'uscita INT (pin 5 della morsettiera) può essere utilizzata per comandare circuiti esterni (come la scheda di alimentazione) sia direttamente sia amplificata in corrente da T1 e T2, a seconda dello stato dei jumper J2 e J3.



ELENCO COMPONENTI:

R1: 4,99 kohm 1%
 R2: 220 ohm 1%
 R3+R5: 10 kohm 1%
 C1, C2: 100 nF multistrato
 C3: 22 pF ceramico
 D1+D4: 1N4004
 T1: PDTA144ET
 T2: PDTCT144ET
 Q1: Quarzo 32 kHz
 IC1: PCF8563P

BT1: Batteria 3,6 V / 65 mA

Varie:

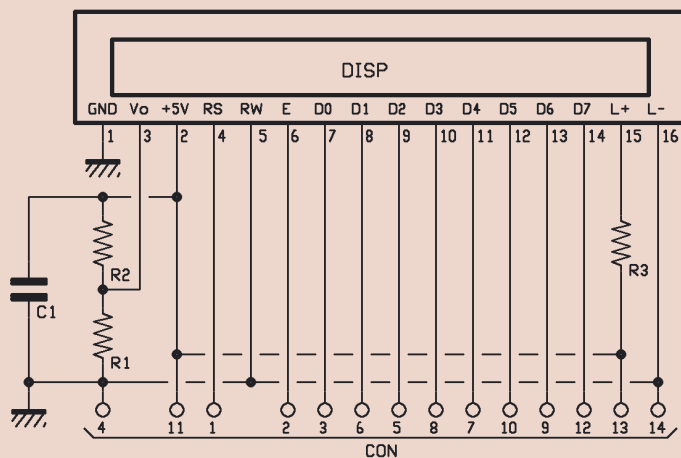
- Morsettiera 2 poli (2 pz.)
- Morsettiera 3 poli
- Strip maschio 2 poli
- Strip maschio 3 poli (2 pz.)
- Jumper (3 pz.)
- Zoccolo 4+4
- circuito stampato

resistenze di polarizzazione di entrambi.

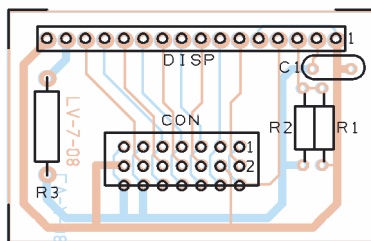
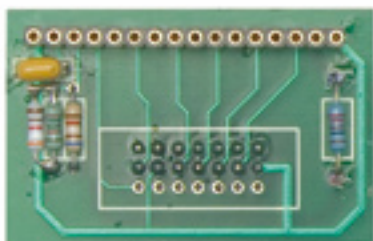
Per evitare la perdita dell'ora in caso di black-out, è stata prevista una batteria tampone da 3,6 volt (65 mA/h) "a barilotto", la quale serve anche a mantenere in funzione il modulo qualora la sua alimentazione principale venga interrotta insieme a quella del PIC. L'alimentazione per il modulo è previsto sia a 5 V stabilizzati (occorrono meno di 10 mA) e va applicata diversamente a seconda dell'impiego; tipicamente i 5 volt vanno forniti tra la massa (contatti 1 e 3) e il punto 4 della morsettiera. In tal caso la batteria viene ricaricata tramite R2 e D2 (quest'ultimo serve ad evitare che, mancando l'alimentazione al punto 4, la batteria si scarichi tramite R2...) mentre D4 provvede a far funzionare il resto del circuito. In caso di black-out, l'accumulatore (se J1 è stato chiuso) alimenta il resto del modulo tramite il diodo D3, componente che nel normale funzionamento (in presenza dei 5 volt dell'alimentazione principale) non conduce e lascia che la corrente di carica della batteria fluisca attraverso la rete R2/D2. Notate che esiste una seconda via per caricare la batteria: il contatto 2 della morsettiera CON; è stata prevista per particolari applicazioni nelle quali il modulo RTC viene alimentato insieme al microcontrollore dell'unità principale tramite il modulo di controllo dell'alimentazione. In una simile evenienza, se i cicli di accensione e spegnimento sono tali che la batteria fatica a caricarsi, è opportuno provvedere a mantenerla in carica mediante la debole corrente che passa da R1 e D1; allo scopo, il piedino 2 va costantemente sottoposto a una tensione di 5 volt, sempre riferita alla massa (1 e 3 della morsettiera CON).

Il modulo LCD

Bene, ora possiamo passare ad esaminare il display LCD: si tratta di un



Si tratta di un adattatore hardware passivo, che permette di interfacciarsi più semplicemente ad un display alfanumerico, come il classico 16x2.



ELENCO COMPONENTI:

R1: 3,92 kohm 1%
R2: 10 kohm 1%
R3: 221 ohm 1% 1/2W
C1: 100 nF multistrato
DISP: Display LCD 16x2

Varie:
- Strip femmina 16 poli
- Strip maschio 16 poli
- Strip maschio 7 poli (2 pz.)
- circuito stampato

linee di controllo E ed RS; R/W è fissa a zero logico, perché nel nostro modulo desideriamo solamente scrivere o impartire comandi (modo "Write") e non leggere criteri o stati dal suo buffer (modo "Read"). Rammentate che R/W decide se il display deve ricevere i dati, ovvero inviarne al dispositivo che lo pilota. In altre parole, la logica del controller HD44780 può sia ricevere i dati relativi ai caratteri da mostrare e ai comandi per farlo, che mandare verso il dispositivo di interfaccia eventuali informazioni di stato o altro ancora. La linea RS serve per indicare al display se i dati in arrivo sul bus vanno interpretati come comandi o informazioni da visualizzare: per comando si intende un byte che fa muovere il cursore o azzerare l'LCD, mentre i dati sono, ovviamente, i caratteri da mostrare. Infine, E corrisponde all'Enable del display: va posta a livello basso quando il componente deve operare, mentre deve essere lasciata a riposo (stato 1) se il visualizzatore non deve essere utilizzato per alcuna operazione (scrittura, spostamento cursore, azzeramento). Il modulo display può essere applicato a una qualsiasi delle porte previste nella scheda di base: sarà poi cura del progettista andare a definire, nella routine corrispondente in

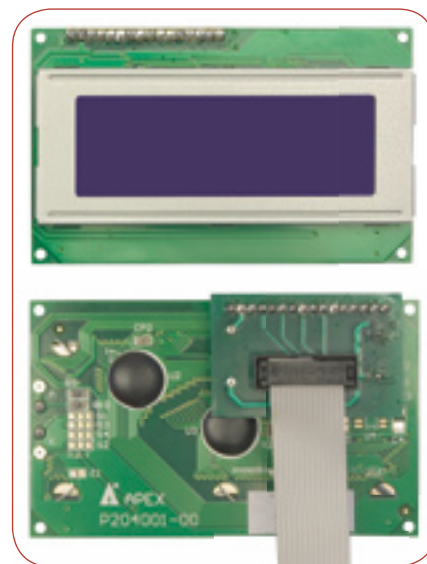
>

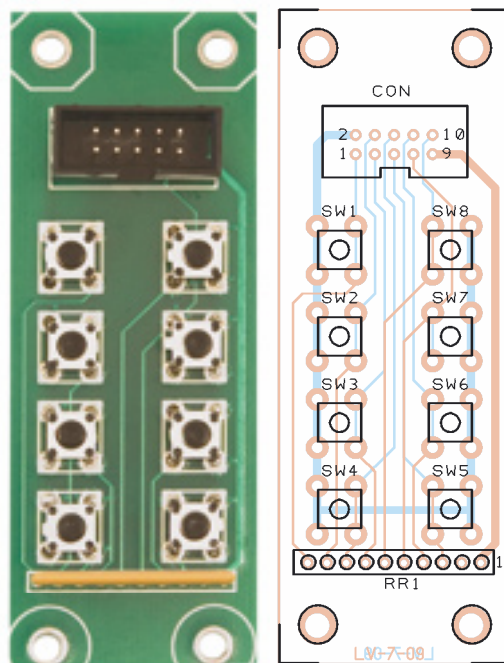
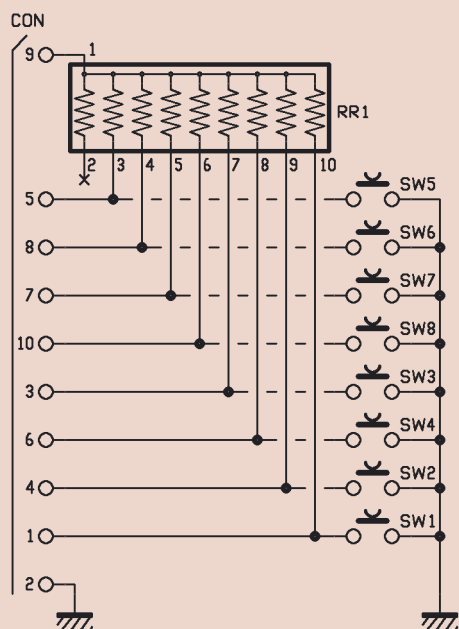
modulo visualizzatore basato su un display a interfaccia standard, gestito da un controller Hitachi HD44780 o compatibile, del tipo da due righe per sedici caratteri (ma anche 2x20, 4x20, 4x16...). Anzi, più esattamente il modulo consta del display, di un piccolo circuito stampato adattatore, di due resistori (R1 e R2) per stabilire il contrasto ed uno (R3) per dare corrente al retroilluminatore a led, il tutto connesso tramite una morsetteria (siglata, al solito, CON).

L'interfaccia di comunicazione prevede l'impiego di tutti gli otto bit disponibili nel display: una configu-

razione un po' insolita, per i nostri lettori che sovente hanno visto impiegare LCD del genere con un bus di comando a soli 4 bit di dati; in questo caso abbiamo optato per tale configurazione per dare ad ognuno la possibilità di scegliere il formato dei dati che più si addice alle applicazioni sviluppate: ricordate, infatti, che la gestione di un display tramite un bus a otto bit consuma quattro linee di I/O in più ma dimezza il tempo di esecuzione delle letture e delle scritture.

Oltre al bus dei dati e dei comandi, dalla morsetteria passano le due





Nella tastiera sono stati previsti otto tasti in modo da occupare un intero byte di una porta del microcontrollore. Il collegamento alla base viene effettuato con un flat cable.

ELENCO COMPONENTI:

RR1: Rete resistiva
9x10 kohm + C
SW1÷SW8: Microswitch

Varie:

- Connettore POD 10 vie
- circuito stampato

corporata nel software di emulazione, la porta effettivamente usata.

La pulsantiera

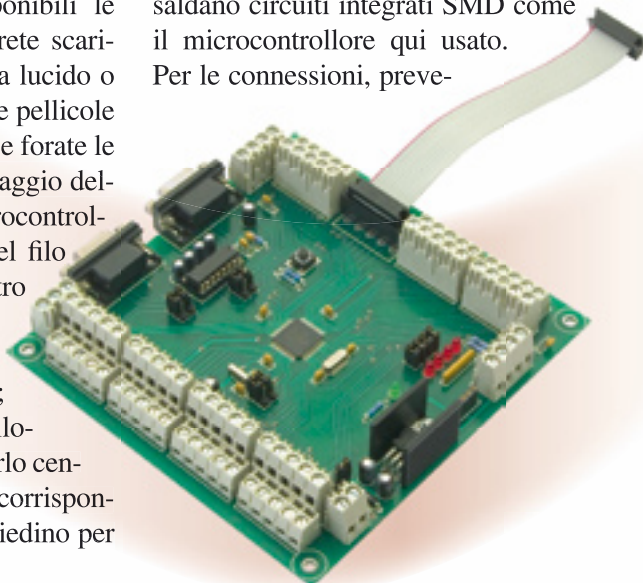
L'ultimo dei moduli esterni è quello che monta i tasti: si tratta, sostanzialmente, di un circuito contenente un set di otto pulsanti normalmente aperti collegati fra le altrettante linee di connessione (facenti capo a una morsettiere a 10 poli siglata CON) e la massa; ciascun tasto è dotato di un resistore di pull-up, utile a mantenere la linea corrispondente a livello logico alto quando è rilasciato. La pulsantiera è stata prevista per l'emulazione di quelle applicazioni in cui il microcontrollore deve prendere ordini, appunto, da pulsanti o impulsi logici a livello basso; può essere collegata a una qualsiasi delle porte previste nell'unità di controllo, con la solita premessa: bisogna definire, nella routine di lettura "preconfezionata" nel software di emulazione, la porta in uso ogni volta. Abbiamo utilizzato otto pulsanti per utilizzare un byte intero (leggasi anche "una porta completa") ma nulla vieta di realizzare tastiere

differenti, in grado di gestire un numero maggiore di pulsanti.

Realizzazione pratica

Giunti a questo punto, non ci resta che analizzare i dettagli costruttivi del sistema di sviluppo. La prima cosa da fare è preparare i circuiti stampati dell'unità base (quella contenente il microcontrollore) e dei moduli esterni che eventualmente vi servono per emulare le vostre applicazioni; per tutti, nel nostro sito Internet (www.elettronica.in.it) sono disponibili le tracce lato rame, che dovrete scaricare e stampare su carta da lucido o acetato al fine di ottenere le pellicole per la fotoincisione. Incise e forate le basette, partite con il montaggio della scheda principale (microcontrollore), attrezzandovi con del filo di lega saldante del diametro di 0,5 mm e un saldatore (da non più di 30 watt di potenza) con punta sottile; appoggiate il microcontrollore sul c.s. cercando di tenerlo centrato rispetto alle piazzole corrispondenti, quindi stagnate un piedino per

lato, in modo da bloccare il chip. Procedete quindi stagnando i pin restanti, sempre uno per lato, così da non surriscaldare il componente. Saldare lo zoccolo del MAX232, quindi sistemate tutti gli altri componenti, procedendo in ordine di altezza. Anche se lo abbiamo citato più volte in molto articoli, vi rammentiamo ancora una volta di fare uso del flussante in gel, di una pinzetta a becchi e di una lampada con lente di ingrandimento, attrezzi questi, indispensabili quando si saldano circuiti integrati SMD come il microcontrollore qui usato. Per le connessioni, preve-



Anticipazioni sul software

dedite delle morsettiere componibili a due livelli, che sono quelle più pratiche per realizzare i collegamenti con le unità usate per l'emulazione. Sistemata la scheda del micro, procedete in maniera analoga con gli stampati delle periferiche (RTC, controllo alimentazione...) ricordando di stagnare un pin per lato nei componenti SMD (dove presenti) e andando in ordine di altezza (prima i chip, poi i jumper e infine le morsettiere). Per quel che riguarda il modulo display, collegate l'LCD alla basetta di adattamento utilizzando una strip maschio da 16 contatti, che salderete nello stampato del display, ed una femmina (sempre da 16 vie) che prenderà posto sulla basetta e permetterà l'inserzione. Per rendere l'insieme più robusto e pratico, potete fissare le varie schede su una tavola di legno mediante delle colonnine esagonali di altezza adeguata, facendo attenzione a mantenere tra esse una spaziatura sufficiente per consentirvi di effettuare senza fatica i vari cablaggi.

Conclusioni

Ora che avete preparato la piattaforma hardware di questo mini sistema di sviluppo, se non volete attendere fino al mese prossimo, potete scaricare dal nostro sito sia il codice sorgente completo (che, come già detto, racchiude una serie di routine di uso comune preconfezionate) sia il semplice programma di debug (operante in ambiente Windows tramite la porta seriale) ed iniziare a dare un'occhiata

FIG. 1 - L'emulatore di terminale viene utilizzato per accedere ad una serie di registri del microcontrollore, durante il funzionamento in modalità "debug". L'accesso al micro avviene tramite porta seriale.

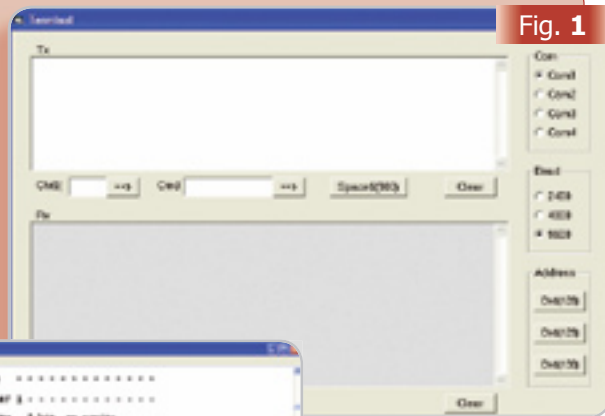


Fig. 1

```
***** ROUTINE GESTIONE UART *****  
***** Routine di init UART *****  
initUART:      ; Inizializza il registro di controllo di flusso  
              ; 2 bit di stop, no controllo di flusso  
              ; Mod rate 9600 con 10 MHz  
              ; (MCR1) = 0x00  
              ; MCR2 = 0x00  
              ; MCR3 = 0x00  
              ; MCR4 = 0x00  
              ; MCR5 = 0x00  
              ; MCR6 = 0x00  
              ; MCR7 = 0x00  
              ; MCR8 = 0x00  
              ; MCR9 = 0x00  
              ; MCR10 = 0x00  
              ; MCR11 = 0x00  
              ; MCR12 = 0x00  
              ; MCR13 = 0x00  
              ; MCR14 = 0x00  
              ; MCR15 = 0x00  
              ; MCR16 = 0x00  
              ; MCR17 = 0x00  
              ; MCR18 = 0x00  
              ; MCR19 = 0x00  
              ; MCR20 = 0x00  
              ; MCR21 = 0x00  
              ; MCR22 = 0x00  
              ; MCR23 = 0x00  
              ; MCR24 = 0x00  
              ; MCR25 = 0x00  
              ; MCR26 = 0x00  
              ; MCR27 = 0x00  
              ; MCR28 = 0x00  
              ; MCR29 = 0x00  
              ; MCR30 = 0x00  
              ; MCR31 = 0x00  
              ; MCR32 = 0x00  
              ; MCR33 = 0x00  
              ; MCR34 = 0x00  
              ; MCR35 = 0x00  
              ; MCR36 = 0x00  
              ; MCR37 = 0x00  
              ; MCR38 = 0x00  
              ; MCR39 = 0x00  
              ; MCR40 = 0x00  
              ; MCR41 = 0x00  
              ; MCR42 = 0x00  
              ; MCR43 = 0x00  
              ; MCR44 = 0x00  
              ; MCR45 = 0x00  
              ; MCR46 = 0x00  
              ; MCR47 = 0x00  
              ; MCR48 = 0x00  
              ; MCR49 = 0x00  
              ; MCR50 = 0x00  
              ; MCR51 = 0x00  
              ; MCR52 = 0x00  
              ; MCR53 = 0x00  
              ; MCR54 = 0x00  
              ; MCR55 = 0x00  
              ; MCR56 = 0x00  
              ; MCR57 = 0x00  
              ; MCR58 = 0x00  
              ; MCR59 = 0x00  
              ; MCR60 = 0x00  
              ; MCR61 = 0x00  
              ; MCR62 = 0x00  
              ; MCR63 = 0x00  
              ; MCR64 = 0x00  
              ; MCR65 = 0x00  
              ; MCR66 = 0x00  
              ; MCR67 = 0x00  
              ; MCR68 = 0x00  
              ; MCR69 = 0x00  
              ; MCR70 = 0x00  
              ; MCR71 = 0x00  
              ; MCR72 = 0x00  
              ; MCR73 = 0x00  
              ; MCR74 = 0x00  
              ; MCR75 = 0x00  
              ; MCR76 = 0x00  
              ; MCR77 = 0x00  
              ; MCR78 = 0x00  
              ; MCR79 = 0x00  
              ; MCR80 = 0x00  
              ; MCR81 = 0x00  
              ; MCR82 = 0x00  
              ; MCR83 = 0x00  
              ; MCR84 = 0x00  
              ; MCR85 = 0x00  
              ; MCR86 = 0x00  
              ; MCR87 = 0x00  
              ; MCR88 = 0x00  
              ; MCR89 = 0x00  
              ; MCR90 = 0x00  
              ; MCR91 = 0x00  
              ; MCR92 = 0x00  
              ; MCR93 = 0x00  
              ; MCR94 = 0x00  
              ; MCR95 = 0x00  
              ; MCR96 = 0x00  
              ; MCR97 = 0x00  
              ; MCR98 = 0x00  
              ; MCR99 = 0x00  
              ; MCR100 = 0x00  
              ; MCR101 = 0x00  
              ; MCR102 = 0x00  
              ; MCR103 = 0x00  
              ; MCR104 = 0x00  
              ; MCR105 = 0x00  
              ; MCR106 = 0x00  
              ; MCR107 = 0x00  
              ; MCR108 = 0x00  
              ; MCR109 = 0x00  
              ; MCR110 = 0x00  
              ; MCR111 = 0x00  
              ; MCR112 = 0x00  
              ; MCR113 = 0x00  
              ; MCR114 = 0x00  
              ; MCR115 = 0x00  
              ; MCR116 = 0x00  
              ; MCR117 = 0x00  
              ; MCR118 = 0x00  
              ; MCR119 = 0x00  
              ; MCR120 = 0x00  
              ; MCR121 = 0x00  
              ; MCR122 = 0x00  
              ; MCR123 = 0x00  
              ; MCR124 = 0x00  
              ; MCR125 = 0x00  
              ; MCR126 = 0x00  
              ; MCR127 = 0x00  
              ; MCR128 = 0x00  
              ; MCR129 = 0x00  
              ; MCR130 = 0x00  
              ; MCR131 = 0x00  
              ; MCR132 = 0x00  
              ; MCR133 = 0x00  
              ; MCR134 = 0x00  
              ; MCR135 = 0x00  
              ; MCR136 = 0x00  
              ; MCR137 = 0x00  
              ; MCR138 = 0x00  
              ; MCR139 = 0x00  
              ; MCR140 = 0x00  
              ; MCR141 = 0x00  
              ; MCR142 = 0x00  
              ; MCR143 = 0x00  
              ; MCR144 = 0x00  
              ; MCR145 = 0x00  
              ; MCR146 = 0x00  
              ; MCR147 = 0x00  
              ; MCR148 = 0x00  
              ; MCR149 = 0x00  
              ; MCR150 = 0x00  
              ; MCR151 = 0x00  
              ; MCR152 = 0x00  
              ; MCR153 = 0x00  
              ; MCR154 = 0x00  
              ; MCR155 = 0x00  
              ; MCR156 = 0x00  
              ; MCR157 = 0x00  
              ; MCR158 = 0x00  
              ; MCR159 = 0x00  
              ; MCR160 = 0x00  
              ; MCR161 = 0x00  
              ; MCR162 = 0x00  
              ; MCR163 = 0x00  
              ; MCR164 = 0x00  
              ; MCR165 = 0x00  
              ; MCR166 = 0x00  
              ; MCR167 = 0x00  
              ; MCR168 = 0x00  
              ; MCR169 = 0x00  
              ; MCR170 = 0x00  
              ; MCR171 = 0x00  
              ; MCR172 = 0x00  
              ; MCR173 = 0x00  
              ; MCR174 = 0x00  
              ; MCR175 = 0x00  
              ; MCR176 = 0x00  
              ; MCR177 = 0x00  
              ; MCR178 = 0x00  
              ; MCR179 = 0x00  
              ; MCR180 = 0x00  
              ; MCR181 = 0x00  
              ; MCR182 = 0x00  
              ; MCR183 = 0x00  
              ; MCR184 = 0x00  
              ; MCR185 = 0x00  
              ; MCR186 = 0x00  
              ; MCR187 = 0x00  
              ; MCR188 = 0x00  
              ; MCR189 = 0x00  
              ; MCR190 = 0x00  
              ; MCR191 = 0x00  
              ; MCR192 = 0x00  
              ; MCR193 = 0x00  
              ; MCR194 = 0x00  
              ; MCR195 = 0x00  
              ; MCR196 = 0x00  
              ; MCR197 = 0x00  
              ; MCR198 = 0x00  
              ; MCR199 = 0x00  
              ; MCR200 = 0x00  
              ; MCR201 = 0x00  
              ; MCR202 = 0x00  
              ; MCR203 = 0x00  
              ; MCR204 = 0x00  
              ; MCR205 = 0x00  
              ; MCR206 = 0x00  
              ; MCR207 = 0x00  
              ; MCR208 = 0x00  
              ; MCR209 = 0x00  
              ; MCR210 = 0x00  
              ; MCR211 = 0x00  
              ; MCR212 = 0x00  
              ; MCR213 = 0x00  
              ; MCR214 = 0x00  
              ; MCR215 = 0x00  
              ; MCR216 = 0x00  
              ; MCR217 = 0x00  
              ; MCR218 = 0x00  
              ; MCR219 = 0x00  
              ; MCR220 = 0x00  
              ; MCR221 = 0x00  
              ; MCR222 = 0x00  
              ; MCR223 = 0x00  
              ; MCR224 = 0x00  
              ; MCR225 = 0x00  
              ; MCR226 = 0x00  
              ; MCR227 = 0x00  
              ; MCR228 = 0x00  
              ; MCR229 = 0x00  
              ; MCR230 = 0x00  
              ; MCR231 = 0x00  
              ; MCR232 = 0x00  
              ; MCR233 = 0x00  
              ; MCR234 = 0x00  
              ; MCR235 = 0x00  
              ; MCR236 = 0x00  
              ; MCR237 = 0x00  
              ; MCR238 = 0x00  
              ; MCR239 = 0x00  
              ; MCR240 = 0x00  
              ; MCR241 = 0x00  
              ; MCR242 = 0x00  
              ; MCR243 = 0x00  
              ; MCR244 = 0x00  
              ; MCR245 = 0x00  
              ; MCR246 = 0x00  
              ; MCR247 = 0x00  
              ; MCR248 = 0x00  
              ; MCR249 = 0x00  
              ; MCR250 = 0x00  
              ; MCR251 = 0x00  
              ; MCR252 = 0x00  
              ; MCR253 = 0x00  
              ; MCR254 = 0x00  
              ; MCR255 = 0x00  
              ; MCR256 = 0x00  
              ; MCR257 = 0x00  
              ; MCR258 = 0x00  
              ; MCR259 = 0x00  
              ; MCR260 = 0x00  
              ; MCR261 = 0x00  
              ; MCR262 = 0x00  
              ; MCR263 = 0x00  
              ; MCR264 = 0x00  
              ; MCR265 = 0x00  
              ; MCR266 = 0x00  
              ; MCR267 = 0x00  
              ; MCR268 = 0x00  
              ; MCR269 = 0x00  
              ; MCR270 = 0x00  
              ; MCR271 = 0x00  
              ; MCR272 = 0x00  
              ; MCR273 = 0x00  
              ; MCR274 = 0x00  
              ; MCR275 = 0x00  
              ; MCR276 = 0x00  
              ; MCR277 = 0x00  
              ; MCR278 = 0x00  
              ; MCR279 = 0x00  
              ; MCR280 = 0x00  
              ; MCR281 = 0x00  
              ; MCR282 = 0x00  
              ; MCR283 = 0x00  
              ; MCR284 = 0x00  
              ; MCR285 = 0x00  
              ; MCR286 = 0x00  
              ; MCR287 = 0x00  
              ; MCR288 = 0x00  
              ; MCR289 = 0x00  
              ; MCR290 = 0x00  
              ; MCR291 = 0x00  
              ; MCR292 = 0x00  
              ; MCR293 = 0x00  
              ; MCR294 = 0x00  
              ; MCR295 = 0x00  
              ; MCR296 = 0x00  
              ; MCR297 = 0x00  
              ; MCR298 = 0x00  
              ; MCR299 = 0x00  
              ; MCR300 = 0x00  
              ; MCR301 = 0x00  
              ; MCR302 = 0x00  
              ; MCR303 = 0x00  
              ; MCR304 = 0x00  
              ; MCR305 = 0x00  
              ; MCR306 = 0x00  
              ; MCR307 = 0x00  
              ; MCR308 = 0x00  
              ; MCR309 = 0x00  
              ; MCR310 = 0x00  
              ; MCR311 = 0x00  
              ; MCR312 = 0x00  
              ; MCR313 = 0x00  
              ; MCR314 = 0x00  
              ; MCR315 = 0x00  
              ; MCR316 = 0x00  
              ; MCR317 = 0x00  
              ; MCR318 = 0x00  
              ; MCR319 = 0x00  
              ; MCR320 = 0x00  
              ; MCR321 = 0x00  
              ; MCR322 = 0x00  
              ; MCR323 = 0x00  
              ; MCR324 = 0x00  
              ; MCR325 = 0x00  
              ; MCR326 = 0x00  
              ; MCR327 = 0x00  
              ; MCR328 = 0x00  
              ; MCR329 = 0x00  
              ; MCR330 = 0x00  
              ; MCR331 = 0x00  
              ; MCR332 = 0x00  
              ; MCR333 = 0x00  
              ; MCR334 = 0x00  
              ; MCR335 = 0x00  
              ; MCR336 = 0x00  
              ; MCR337 = 0x00  
              ; MCR338 = 0x00  
              ; MCR339 = 0x00  
              ; MCR340 = 0x00  
              ; MCR341 = 0x00  
              ; MCR342 = 0x00  
              ; MCR343 = 0x00  
              ; MCR344 = 0x00  
              ; MCR345 = 0x00  
              ; MCR346 = 0x00  
              ; MCR347 = 0x00  
              ; MCR348 = 0x00  
              ; MCR349 = 0x00  
              ; MCR350 = 0x00  
              ; MCR351 = 0x00  
              ; MCR352 = 0x00  
              ; MCR353 = 0x00  
              ; MCR354 = 0x00  
              ; MCR355 = 0x00  
              ; MCR356 = 0x00  
              ; MCR357 = 0x00  
              ; MCR358 = 0x00  
              ; MCR359 = 0x00  
              ; MCR360 = 0x00  
              ; MCR361 = 0x00  
              ; MCR362 = 0x00  
              ; MCR363 = 0x00  
              ; MCR364 = 0x00  
              ; MCR365 = 0x00  
              ; MCR366 = 0x00  
              ; MCR367 = 0x00  
              ; MCR368 = 0x00  
              ; MCR369 = 0x00  
              ; MCR370 = 0x00  
              ; MCR371 = 0x00  
              ; MCR372 = 0x00  
              ; MCR373 = 0x00  
              ; MCR374 = 0x00  
              ; MCR375 = 0x00  
              ; MCR376 = 0x00  
              ; MCR377 = 0x00  
              ; MCR378 = 0x00  
              ; MCR379 = 0x00  
              ; MCR380 = 0x00  
              ; MCR381 = 0x00  
              ; MCR382 = 0x00  
              ; MCR383 = 0x00  
              ; MCR384 = 0x00  
              ; MCR385 = 0x00  
              ; MCR386 = 0x00  
              ; MCR387 = 0x00  
              ; MCR388 = 0x00  
              ; MCR389 = 0x00  
              ; MCR390 = 0x00  
              ; MCR391 = 0x00  
              ; MCR392 = 0x00  
              ; MCR393 = 0x00  
              ; MCR394 = 0x00  
              ; MCR395 = 0x00  
              ; MCR396 = 0x00  
              ; MCR397 = 0x00  
              ; MCR398 = 0x00  
              ; MCR399 = 0x00  
              ; MCR400 = 0x00  
              ; MCR401 = 0x00  
              ; MCR402 = 0x00  
              ; MCR403 = 0x00  
              ; MCR404 = 0x00  
              ; MCR405 = 0x00  
              ; MCR406 = 0x00  
              ; MCR407 = 0x00  
              ; MCR408 = 0x00  
              ; MCR409 = 0x00  
              ; MCR410 = 0x00  
              ; MCR411 = 0x00  
              ; MCR412 = 0x00  
              ; MCR413 = 0x00  
              ; MCR414 = 0x00  
              ; MCR415 = 0x00  
              ; MCR416 = 0x00  
              ; MCR417 = 0x00  
              ; MCR418 = 0x00  
              ; MCR419 = 0x00  
              ; MCR420 = 0x00  
              ; MCR421 = 0x00  
              ; MCR422 = 0x00  
              ; MCR423 = 0x00  
              ; MCR424 = 0x00  
              ; MCR425 = 0x00  
              ; MCR426 = 0x00  
              ; MCR427 = 0x00  
              ; MCR428 = 0x00  
              ; MCR429 = 0x00  
              ; MCR430 = 0x00  
              ; MCR431 = 0x00  
              ; MCR432 = 0x00  
              ; MCR433 = 0x00  
              ; MCR434 = 0x00  
              ; MCR435 = 0x00  
              ; MCR436 = 0x00  
              ; MCR437 = 0x00  
              ; MCR438 = 0x00  
              ; MCR439 = 0x00  
              ; MCR440 = 0x00  
              ; MCR441 = 0x00  
              ; MCR442 = 0x00  
              ; MCR443 = 0x00  
              ; MCR444 = 0x00  
              ; MCR445 = 0x00  
              ; MCR446 = 0x00  
              ; MCR447 = 0x00  
              ; MCR448 = 0x00  
              ; MCR449 = 0x00  
              ; MCR450 = 0x00  
              ; MCR451 = 0x00  
              ; MCR452 = 0x00  
              ; MCR453 = 0x00  
              ; MCR454 = 0x00  
              ; MCR455 = 0x00  
              ; MCR456 = 0x00  
              ; MCR457 = 0x00  
              ; MCR458 = 0x00  
              ; MCR459 = 0x00  
              ; MCR460 = 0x00  
              ; MCR461 = 0x00  
              ; MCR462 = 0x00  
              ; MCR463 = 0x00  
              ; MCR464 = 0x00  
              ; MCR465 = 0x00  
              ; MCR466 = 0x00  
              ; MCR467 = 0x00  
              ; MCR468 = 0x00  
              ; MCR469 = 0x00  
              ; MCR470 = 0x00  
              ; MCR471 = 0x00  
              ; MCR472 = 0x00  
              ; MCR473 = 0x00  
              ; MCR474 = 0x00  
              ; MCR475 = 0x00  
              ; MCR476 = 0x00  
              ; MCR477 = 0x00  
              ; MCR478 = 0x00  
              ; MCR479 = 0x00  
              ; MCR480 = 0x00  
              ; MCR481 = 0x00  
              ; MCR482 = 0x00  
              ; MCR483 = 0x00  
              ; MCR484 = 0x00  
              ; MCR485 = 0x00  
              ; MCR486 = 0x00  
              ; MCR487 = 0x00  
              ; MCR488 = 0x00  
              ; MCR489 = 0x00  
              ; MCR490 = 0x00  
              ; MCR491 = 0x00  
              ; MCR492 = 0x00  
              ; MCR493 = 0x00  
              ; MCR494 = 0x00  
              ; MCR495 = 0x00  
              ; MCR496 = 0x00  
              ; MCR497 = 0x00  
              ; MCR498 = 0x00  
              ; MCR499 = 0x00  
              ; MCR500 = 0x00  
              ; MCR501 = 0x00  
              ; MCR502 = 0x00  
              ; MCR503 = 0x00  
              ; MCR504 = 0x00  
              ; MCR505 = 0x00  
              ; MCR506 = 0x00  
              ; MCR507 = 0x00  
              ; MCR508 = 0x00  
              ; MCR509 = 0x00  
              ; MCR510 = 0x00  
              ; MCR511 = 0x00  
              ; MCR512 = 0x00  
              ; MCR513 = 0x00  
              ; MCR514 = 0x00  
              ; MCR515 = 0x00  
              ; MCR516 = 0x00  
              ; MCR517 = 0x00  
              ; MCR518 = 0x00  
              ; MCR519 = 0x00  
              ; MCR520 = 0x00  
              ; MCR521 = 0x00  
              ; MCR522 = 0x00  
              ; MCR523 = 0x00  
              ; MCR524 = 0x00  
              ; MCR525 = 0x00  
              ; MCR526 = 0x00  
              ; MCR527 = 0x00  
              ; MCR528 = 0x00  
              ; MCR529 = 0x00  
              ; MCR530 = 0x00  
              ; MCR531 = 0x00  
              ; MCR532 = 0x00  
              ; MCR533 = 0x00  
              ; MCR534 = 0x00  
              ; MCR535 = 0x00  
              ; MCR536 = 0x00  
              ; MCR537 = 0x00  
              ; MCR538 = 0x00  
              ; MCR539 = 0x00  
              ; MCR540 = 0x00  
              ; MCR541 = 0x00  
              ; MCR542 = 0x00  
              ; MCR543 = 0x00  
              ; MCR544 = 0x00  
              ; MCR545 = 0x00  
              ; MCR546 = 0x00  
              ; MCR547 = 0x00  
              ; MCR548 = 0x00  
              ; MCR549 = 0x00  
              ; MCR550 = 0x00  
              ; MCR551 = 0x00  
              ; MCR552 = 0x00  
              ; MCR553 = 0x00  
              ; MCR554 = 0x00  
              ; MCR555 = 0x00  
              ; MCR556 = 0x00  
              ; MCR557 = 0x00  
              ; MCR558 = 0x00  
              ; MCR559 = 0x00  
              ; MCR560 = 0x00  
              ; MCR561 = 0x00  
              ; MCR562 = 0x00  
              ; MCR563 = 0x00  
              ; MCR564 = 0x00  
              ; MCR565 = 0x00  
              ; MCR566 = 0x00  
              ; MCR567 = 0x00  
              ; MCR568 = 0x00  
              ; MCR569 = 0x00  
              ; MCR570 = 0x00  
              ; MCR571 = 0x00  
              ; MCR572 = 0x00  
              ; MCR573 = 0x00  
              ; MCR574 = 0x00  
              ; MCR575 = 0x00  
              ; MCR576 = 0x00  
              ; MCR577 = 0x00  
              ; MCR578 = 0x00  
              ; MCR579 = 0x00  
              ; MCR580 = 0x00  
              ; MCR581 = 0x00  
              ; MCR582 = 0x00  
              ; MCR583 = 0x00  
              ; MCR584 = 0x00  
              ; MCR585 = 0x00  
              ; MCR586 = 0x00  
              ; MCR587 = 0x00  
              ; MCR588 = 0x00  
              ; MCR589 = 0x00  
              ; MCR590 = 0x00  
              ; MCR591 = 0x00  
              ; MCR592 = 0x00  
              ; MCR593 = 0x00  
              ; MCR594 = 0x00  
              ; MCR595 = 0x00  
              ; MCR596 = 0x00  
              ; MCR597 = 0x00  
              ; MCR598 = 0x00  
              ; MCR599 = 0x00  
              ; MCR600 = 0x00  
              ; MCR601 = 0x00  
              ; MCR602 = 0x00  
              ; MCR603 = 0x00  
              ; MCR604 = 0x00  
              ; MCR605 = 0x00  
              ; MCR606 = 0x00  
              ; MCR607 = 0x00  
              ; MCR608 = 0x00  
              ; MCR609 = 0x00  
              ; MCR610 = 0x00  
              ; MCR611 = 0x00  
              ; MCR612 = 0x00  
              ; MCR613 = 0x00  
              ; MCR614 = 0x00  
              ; MCR615 = 0x00  
              ; MCR616 = 0x00  
              ; MCR617 = 0x00  
              ; MCR618 = 0x00  
              ; MCR619 = 0x00  
              ; MCR620 = 0x00  
              ; MCR621 = 0x00  
              ; MCR622 = 0x00  
              ; MCR623 = 0x00  
              ; MCR624 = 0x00  
              ; MCR625 = 0x00  
              ; MCR626 = 0x00  
              ; MCR627 = 0x00  
              ; MCR628 = 0x00  
              ; MCR629 = 0x00  
              ; MCR630 = 0x00  
              ; MCR631 = 0x00  
              ; MCR632 = 0x00  
              ; MCR633 = 0x00  
              ; MCR634 = 0x00  
              ; MCR635 = 0x00  
              ; MCR636 = 0x00  
              ; MCR637 = 0x00  
              ; MCR638 = 0x00  
              ; MCR639 = 0x00  
              ; MCR640 = 0x00  
              ; MCR641 = 0x00  
              ; MCR642 = 0x00  
              ; MCR643 = 0x00  
              ; MCR644 = 0x00  
              ; MCR645 = 0x00  
              ; MCR646 = 0x00  
              ; MCR647 = 0x00  
              ; MCR648 = 0x00  
              ; MCR649 = 0x00  
              ; MCR650 = 0x00  
              ; MCR651 = 0x00  
              ; MCR652 = 0x00  
              ; MCR653 = 0x00  
              ; MCR654 = 0x00  
              ; MCR655 = 0x00  
              ; MCR656 = 0x00  
              ; MCR657 = 0x00  
              ; MCR658 = 0x00  
              ; MCR659 = 0x00  
              ; MCR660 = 0x00  
              ; MCR661 = 0x00  
              ; MCR662 = 0x00  
              ; MCR663 = 0x00  
              ; MCR664 = 0x00  
              ; MCR665 = 0x00  
              ; MCR666 = 0x00  
              ; MCR667 = 0x00  
              ; MCR668 = 0x00  
              ; MCR669 = 0x00  
              ; MCR670 = 0x00  
              ; MCR671 = 0x00  
              ; MCR672 = 0x00  
              ; MCR673 = 0x00  
              ; MCR674 = 0x00  
              ; MCR675 = 0x00  
              ; MCR676 = 0x00  
              ; MCR677 = 0x00  
              ; MCR678 = 0x00  
              ; MCR679 = 0x00  
              ; MCR680 = 0x00  
              ; MCR681 = 0x00  
              ; MCR682 = 0x00  
              ; MCR683 = 0x00  
              ; MCR684 = 0x00  
              ; MCR685 = 0x00  
              ; MCR686 = 0x00  
              ; MCR687 = 0x00  
              ; MCR688 = 0x00  
              ; MCR689 = 0x00  
              ; MCR690 = 0x00  
              ; MCR691 = 0x00  
              ; MCR692 = 0x00  
              ; MCR693 = 0x00  
              ; MCR694 = 0x00  
              ; MCR695 = 0x00  
              ; MCR696 = 0x00  
              ; MCR697 = 0x00  
              ; MCR698 = 0x00  
              ; MCR699 = 0x00  
              ; MCR700 = 0x00  
              ; MCR701 = 0x00  
              ; MCR702 = 0x00  
              ; MCR703 = 0x00  
              ; MCR704 = 0x00  
              ; MCR705 = 0x00  
              ; MCR706 = 0x00  
              ; MCR707 = 0x00  
              ; MCR708 = 0x00  
              ; MCR709 = 0x00  
              ; MCR710 = 0x00  
              ; MCR711 = 0x00  
              ; MCR712 = 0x00  
              ; MCR713 = 0x00  
              ; MCR714 = 0x00  
              ; MCR715 = 0x00  
              ; MCR716 = 0x00  
              ; MCR717 = 0x00  
              ; MCR718 = 0x00  
              ; MCR719 = 0x00  
              ; MCR720 = 0x00  
              ; MCR721 = 0x00  
              ; MCR722 = 0x00  
              ; MCR723 = 0x00  
              ; MCR724 = 0x00  
              ; MCR725 = 0x00  
              ; MCR726 = 0x00  
              ; MCR727 = 0x00  
              ; MCR728 = 0x00  
              ; MCR729 = 0x00  
              ; MCR730 = 0x00  
              ; MCR731 = 0x00  
              ; MCR732 = 0x00  
              ; MCR733 = 0x00  
              ; MCR734 = 0x00  
              ; MCR735 = 0x00  
              ; MCR736 = 0x00  
              ; MCR737 = 0x00  
              ; MCR738 = 0x00  
              ; MCR739 = 0x00  
              ; MCR740 = 0x00  
              ; MCR741 = 0x00  
              ; MCR742 = 0x00  
              ; MCR743 = 0x00  
              ; MCR744 = 0x00  
              ; MCR745 = 0x00  
              ; MCR746 = 0x00  
              ; MCR747 = 0x00  
              ; MCR748 = 0x00  
              ; MCR749 = 0x00  
              ; MCR750 = 0x00  
              ; MCR751 = 0x00  
              ; MCR752 = 0x00  
              ; MCR753 = 0x00  
              ; MCR754 = 0x00  
              ; MCR755 = 0x00  
              ; MCR756 = 0x00  
              ; MCR757 = 0x00  
              ; MCR758 = 0x00  
              ; MCR759 = 0x00  
              ; MCR760 = 0x00  
              ; MCR761 = 0x00  
              ; MCR762 = 0x00  
              ; MCR763 = 0x00  
              ; MCR764 = 0x00  
              ; MCR765 = 0x00  
              ; MCR766 = 0x00  
              ; MCR767 = 0x00  
              ; MCR768 = 0x00  
              ; MCR769 = 0x00  
              ; MCR770 = 0x00  
              ; MCR771 = 0x00  
              ; MCR772 = 0x00  
              ; MCR773 = 0x00  
              ; MCR774 = 0x00  
              ; MCR775 = 0x00  
              ; MCR776 = 0x00  
              ; MCR777 = 0x00  
              ; MCR778 = 0x00  
              ; MCR779 = 0x00  
              ; MCR780 = 0
```




Corso di programmazione modulo Tibbo EM1000

A CURA DI **SAVINO GIUSTO**

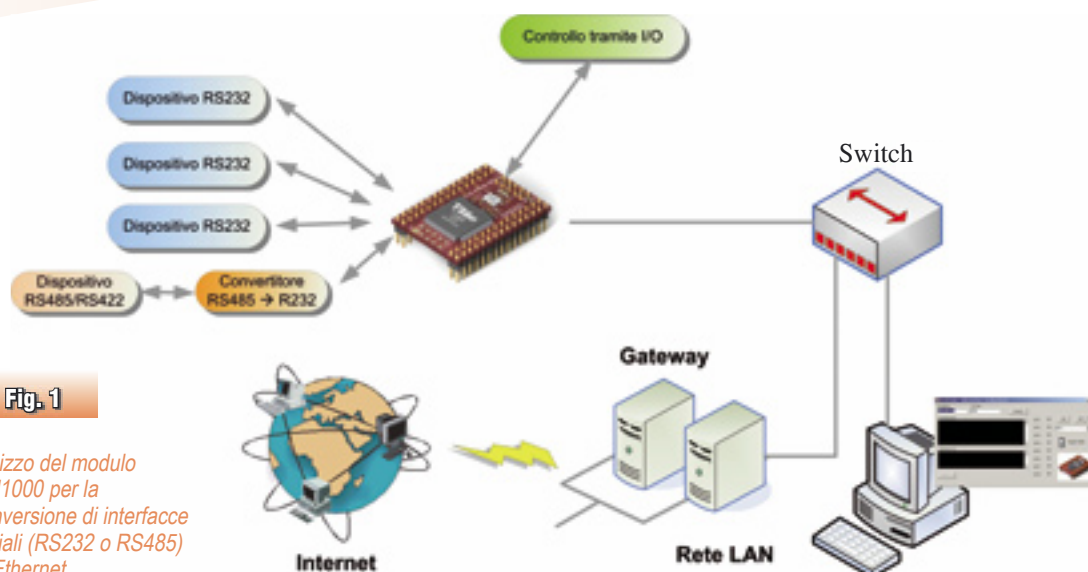
Corso di programmazione e utilizzo del modulo EM1000 di casa Tibbo, un potente Ethernet Server completamente programmabile con funzioni di Web Server, Mail Server e molto altro ancora. In questa puntata realizziamo un semplice convertitore seriale-Ethernet tramite il quale qualsiasi dispositivo dotato di interfaccia seriale può accedere ad una rete LAN. Quinta puntata.

Nella puntata precedente, avevamo realizzato il primo semplice programma tramite il quale lo stato di alcuni ingressi del modulo EM1000 veniva veicolato attraverso la rete Ethernet e visualizzato su di un'interfaccia grafica, sviluppata in Delphi ed installata su un PC: in quell'occasione si era visto che il set di comandi TibboBasic messo a disposizione dall'ambiente di sviluppo Taiko consentiva di scrivere

il sorgente dell'applicazione utilizzando poche macro-istruzioni. Bene, oggi vediamo come con tali istruzioni si possa realizzare molto facilmente un convertitore seriale-Ethernet.

Come la definizione stessa lascia presagire, il convertitore (che chiameremo anche *bridge*) seriale-Ethernet permette di collegare dispositivi dotati di interfaccia RS232 ad una rete Ethernet. Esso può essere utilizzato in tutti i casi in cui sia





necessario remotizzare un dispositivo dotato solo di una porta seriale, sfruttando a tal proposito sia il cablaggio di rete già esistente che il protocollo TCP/IP. La connessione alla rete tramite questo tipo di convertitore, inoltre, reca un altro vantaggio: poiché il *bridge* converte, di fatto, una connessione di tipo punto-punto (cioè RS232) in una di tipo multi-drop (cioè Ethernet), la nuova risorsa (ossia “il dispositivo collegato alla rete tramite il *bridge*”) potrà essere condivisa da tutti i PC connessi alla rete.

La Fig. 1 mostra un tipico esempio applicativo, in cui l'EM1000 viene utilizzato per convertire e remotizzare su di una rete LAN fino a quattro interfacce seriali. Qualora nella rete fosse presente un gateway per l'accesso ad Internet, il dispositivo seriale potrà essere controllato non solo in locale, ma anche in remoto, estendendo indefinitamente il limite di 15 m del cavo, imposti dallo standard RS232.

Il discorso fatto è valido anche per remotizzare interfacce diverse da RS232, quali la RS485 e la

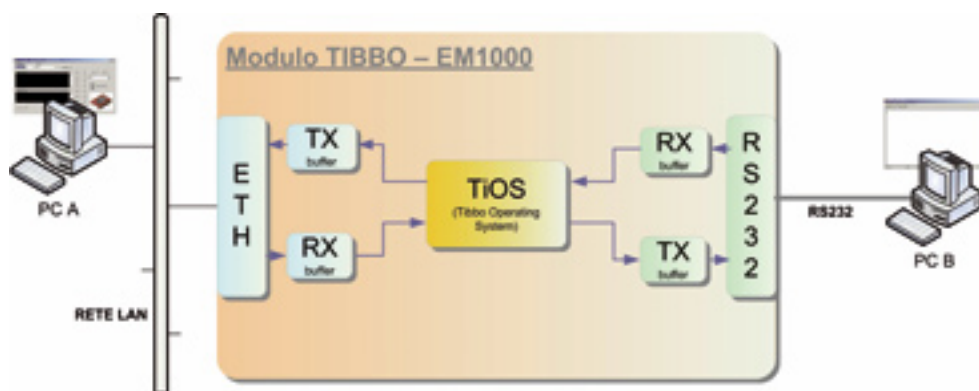
RS422. In questo caso sarà sufficiente interporre un secondo convertitore, questa volta da RS485/RS422 a RS232, tra l'apparato da controllare e l'EM1000, come mostrato in Fig. 1.

L'applicativo firmware

L'esempio presentato in queste pagine fornisce le basi per comprendere il funzionamento della conversione seriale-Ethernet: si tratta di una trasmissione bidirezionale tra un dispositivo dotato di porta seriale (RS232) ed uno di interfaccia Ethernet. In questo caso il modulo Tibbo riveste sostanzialmente il ruolo di *bridge* tra il mondo seriale e quello Ethernet: questo significa che il dispositivo ha il compito di estrarre il *payload* (ossia i dati) dal pacchetto TCP/IP, rendendolo disponibile sulla porta RS232 e viceversa: lo schema in Fig. 2 consente di identificare quali blocchi interni siano stati utilizzati per tale conversione. Con ETH e RS232 si sono indicati i blocchi che gestiscono il livello fisico (ossia, il segnale elettrico trasmesso); inoltre, sono state evidenziate quattro aree di

Fig. 2

Architettura per il test del convertitore seriale-Ethernet.



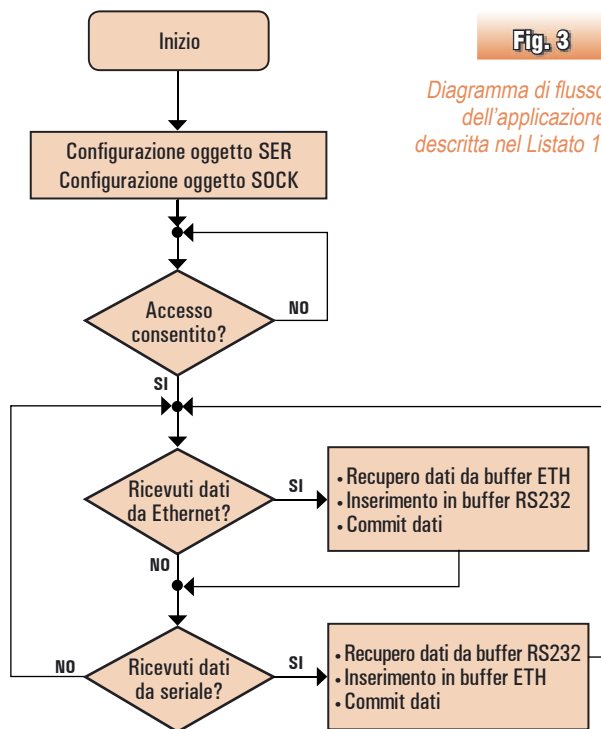
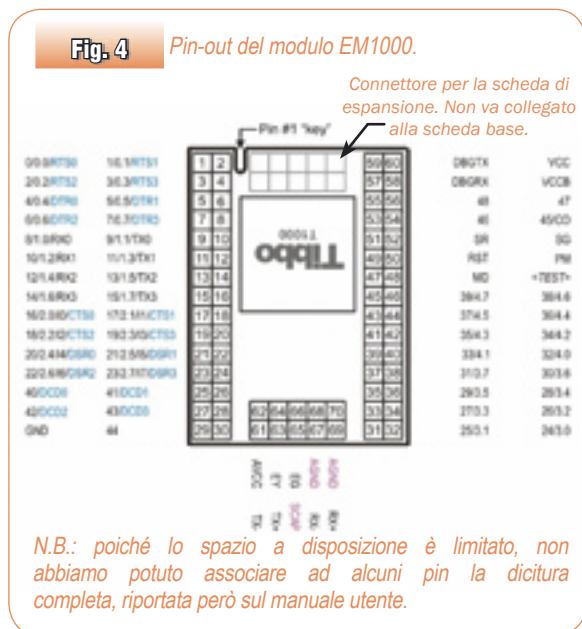


Tabella 1 - Le linee dell'interfaccia seriale

Linea	Descrizione	Direzione
TX	Trasmissione	Output
RX	Ricezione	Input
RTS	Request To Send	Output
CTS	Clear To Send	Input
DTR	Data Terminal Ready	Output
DSR	Data Set Ready	Input



memoria differenti, che rappresentano i buffer di trasmissione e ricezione di entrambi i livelli fisici (seriale ed Ethernet). Come mostreremo nel seguito dell'articolo, tali locazioni di memoria non sono fisse: il programmatore può deciderne la dimensione (compatibilmente con la memoria totale) e la velocità di trasferimento dei dati. Il blocco TiOS, infine, rappresenta il sistema operativo, che ha il compito di sovrintendere alle operazioni di configurazione e gestione. Il codice sorgente a cui si farà riferimento è riportato nel Listato 1 e può essere suddiviso nelle seguenti parti:

1. Configurazione iniziale oggetti sock e ser.
2. Verifica accesso consentito, tramite l'immissione di una password.
3. Attesa di dati provenienti dall'interfaccia Ethernet e relativo invio verso la porta seriale.
4. Attesa di dati provenienti dalla porta seriale e relativo invio verso l'interfaccia Ethernet.

Per rendere più intuitiva la comprensione dell'algoritmo utilizzato, si è riportato in Fig. 3 il diagramma di flusso di tale applicazione.

Tornando per un istante alla Fig. 2, essa mostra, oltre alla struttura della connessione, anche come testare il firmware: è possibile utilizzare il PC A (che deve essere dotato di interfaccia 10/100BaseT) per la trasmissione dati su rete LAN ed il PC B (che deve essere dotato di porta RS232) per il collegamento del dispositivo seriale. Quest'ultimo può essere simulato tramite il conosciuto emulatore di terminale HyperTerminal. I caratteri così inviati dal PC B potranno essere visualizzati sul PC A tramite il software *client* già descritto nella puntata precedente. Di riflesso, il collegamento descritto in Fig. 2 può essere realizzato anche con un singolo PC, purché dotato sia di porta Ethernet che di porta seriale. Qualora quest'ultima non fosse disponibile è possibile impiegare un convertitore seriale-USB: come vedete, la flessibilità di impiego e di test è massima.

Vediamo ora nel dettaglio i vari *oggetti* (ricordatevi sempre questo termine, che definisce gli elementi firmware attivi, ossia che svolgono dei compiti) utilizzati nel convertitore.

L'oggetto Ser

Ogni porta seriale (delle quattro presenti, come da Fig. 4) è costituita dall'hardware per il controllo delle relative linee e dai buffer che consentono di memorizzare i dati bidirezionali in transito. L'oggetto seriale che garantisce il controllo della relativa interfaccia è chiamato *Ser*. Esso contiene tutte le proprietà, i metodi e gli eventi relativi ai

EIA RS232C e ITU-T V24/V28

L'EIA RS-232, equivalente all'Europeo ITU-T (ex CCITT) V.24/V.28, è uno standard che definisce una serie di protocolli meccanici, elettrici ed informatici tramite i quali realizzare un'interfaccia seriale a bassa velocità per lo scambio di dati tra dispositivi digitali. In origine, esso includeva le caratteristiche elettriche dei segnali (chiamati talvolta "criteri"), la struttura e le tempistiche dei dati seriali, la definizione dei protocolli per il controllo del flusso (hardware RTS/CTS) di dati seriali su un canale telefonico operante a velocità più bassa di quella della porta seriale, la meccanica del connettore e la disposizione dei suoi pin ed infine il tipo e la lunghezza massima dei possibili cavi di collegamento. Negli anni lo standard si è evoluto pur mantenendosi in larga parte invariato: la revisione è riconoscibile dall'ultima lettera della sigla e l'ultima, del 1997, è la EIA RS-232F. Probabilmente la versione più diffusa è la RS232C del 1969: essa corrisponde alle specifiche europee CCITT (quando ancora non era ITU-T) raccomandazione V.24/V.28. Pur essendo un protocollo piuttosto vecchio, attualmente la EIA RS-232C è ancora largamente utilizzata per la comunicazione a bassa velocità tra microcontrollori, dispositivi industriali ed altri circuiti relativamente semplici che non necessitano di particolare velocità; è invece praticamente scomparsa in ambito desktop, ambito nel quale lo standard veniva utilizzato per far comunicare un computer ed un modem fonico esterno durante una connessione dati. Attualmente tale soluzione è rimpiazzata dai router Ethernet-ADSL.



buffer e ai parametri della seriale stessa. Le linee utilizzate per la connessione seriale sono riportate in Tabella 1. Mentre le linee TX ed RX corrispondono a degli I/O ben precisi, quelle di RTS e CTS possono essere *rimappate* (ossia spostate su linee diverse da quelle predefinite) a piacimento. Per modificarle si utilizzano i metodi *ser.ctsmap* e *ser.rtsmap*. Il *TX buffer* contiene i dati che devono essere inviati dall'EM1000 verso un qualsiasi apparato seriale; i dati presenti in questa locazione, però, sono in attesa di essere effettivamente inviati: l'invio vero e proprio avviene solo quando si richiama dal firmware il metodo *send*. In *RX buffer* sono posizionati, invece, tutti i dati in ingresso dalla seriale; in questo modo il modulo è in grado di sostenere flussi dati anche molto elevati. Disponendo di tale memoria tampone, è possibile avere una velocità di ricezione sulla seriale superiore a quella di elaborazione dei dati ricevuti. Le dimensioni di entrambi i buffer (RX e TX) possono essere selezionate in maniera indipendente (come abbiamo fatto noi) all'avvio dell'applicazione, tramite i metodi *ser.rxbufirq* e *ser.txbufirq*.

Tabella 2 - Valore della proprietà *ser.baudrate* per valori tipici di baudrate

Baudrate	Ser.baudrate =
1200	<i>ser.div9600</i> * 8
2400	<i>ser.div9600</i> * 4
4800	<i>ser.div9600</i> * 2
9600	<i>ser.div9600</i>
19200	<i>ser.div9600</i> / 2
38400	<i>ser.div9600</i> / 4

Essi hanno come parametro di ingresso il numero di pagine da allocare; ogni pagina corrisponde a 256 byte (il discorso è analogo a quello fatto nella puntata precedente per l'oggetto *Sock*). La selezione del baudrate (la velocità di trasmissione e ricezione dei dati) è effettuata mediante un opportuno divisore. Per tale scopo è utilizzata la proprietà di sola lettura *div9600* dell'oggetto *ser*, che consente di calcolare la velocità della seriale in maniera indipendente dalla piattaforma. Tale proprietà fornisce il corretto divisore per la configurare la porta a 9600 baud, quindi per qualunque altro valore si utilizzerà la seguente formula:

$$\text{ser.baudrate} = (9600 * \text{ser.div9600}) / \text{baudrate}$$

Ad esempio, per impostare una velocità pari a 38.400 baud si deve scrivere:

$$\text{ser.baudrate} = (9600 * \text{ser.div9600}) / 38400 = \text{ser.div9600} / 4$$

In Tabella 2 sono riassunti alcuni tipici valori di baudrate in funzione della proprietà *ser.baudrate* secondo la formula appena vista. La porta seriale può essere configurata in modalità full-duplex o half-duplex, secondo le impostazioni della proprietà *ser.interface*. I valori possibili da assegnare sono

PL_SER_SI_FULLDUPLEX
PL_SER_SI_HALFDUPLEX.

La modalità full-duplex consente la trasmissione e ricezione contemporanea dei dati ed particolarmente indicata per comunicazioni di tipo RS232, RS422 o RS485 a 4 fili. In half-duplex è possibile inviare o ricevere dati alternativamente, come avviene nelle comunicazioni RS485 su 2 fili.

Si è detto che il modulo EM1000 dispone di ben quattro porte seriali: è buona norma di program-

mazione specificare nuovamente la porta selezionata con il comando `ser.num = <porta selezionata>` ogni volta che si eseguono operazioni su una delle quattro seriali. In questo modo si evita di effettuare modifiche indesiderate su altre porte UART. Questa regola, però, vale anche in quei casi in cui si utilizza un'unica seriale.

L'evento `on_ser_data_arrival`

Questo evento viene generato automaticamente dal sistema ogni volta che almeno un byte è arrivato nel buffer di ricezione: detto in termini "da softwarista", ogni volta che la proprietà `ser.rxlen` risulta essere maggiore di zero. Quando l'evento è richiamato, il TiOS imposta automaticamente la proprietà `ser.num` sul valore della seriale che ha ricevuto i caratteri; in questo modo il programmatore ha modo di accedere automaticamente alla porta interessata. Viene richiamato per processare (in toto o in parte) i dati presenti nel buffer. Se per qualche ragione, non sono stati processati tutti i dati, all'uscita dalla sub-routine, l'evento sarà immediatamente richiamato. Per recuperare i dati presenti nel buffer RX, si deve utilizzare il metodo `ser.getdata` secondo il prototipo di funzione

`ser.getdata(num_char as word) as string`

dove `num_char` indica il numero dei caratteri da recuperare. Per quanto si è detto prima, tale valore non deve essere superiore a `ser.rxlen`. Tale funzione restituisce la stringa estratta dal buffer di ricezione. È importante sottolineare che, una volta estratti, tali dati sono cancellati in modo permanente dal buffer.

Nel Listato 1, all'interno dell'evento `on_ser_data_arrival` è utilizzata un'istruzione particolarmente compatta ed efficiente per trasferire i dati ricevuti da seriale verso l'interfaccia Ethernet

`sock.setdata(ser.getdata(sock.txfree))`

È interessante notare come con un sola istruzione si realizzi la conversione RS232>ETH. Può essere pensata (ed in realtà lo è) come la somma di tre istruzioni:

1. `sock.txfree`. Proprietà dell'oggetto `Sock` per la gestione dell'interfaccia Ethernet. Serve per capire la capacità del buffer di trasmissione.
2. `ser.getdata(...)`. Recupera i dati dal buffer seriale, compatibilmente con la dimensione del buffer TX dell'Ethernet.
3. `sock.setdata(...)`. Prepara i dati del punto 2 per la trasmissione su Ethernet.

Infine, i dati così preparati sono inviati sulla LAN

tramite la funzione `sock.send`.

Quanto appena descritto realizza il convertitore seriale-Ethernet solo in un verso (RS232 > ETH). Come realizzare il trasferimento nel verso opposto sarà argomento del paragrafo successivo.

L'evento `on_sock_data_arrival`

Poiché è già stato ampiamente descritto nella quarta puntata del corso, verrà dato maggiore rilievo alla parte che riguarda l'interazione con la seriale. Anche in questo caso si è utilizzata l'istruzione molto compatta

`ser.setdata(sock.getdata(ser.txfree))`

che svolge da sola tre istruzioni distinte:

1. `ser.txfree`. Restituisce la quantità di spazio libero nel buffer di trasmissione della UART.
2. `sock.getdata(...)`. Estrae i dati ricevuti da LAN, compatibilmente alla dimensione del buffer di trasmissione seriale.
3. `ser.setdata`. Inserisce i dati estratti nel buffer di trasmissione della seriale.

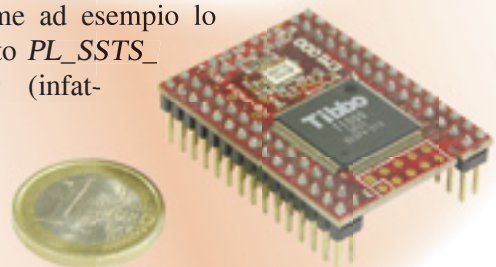
Anche in questo caso è necessario effettuare il `commit` (nel gergo del softwarista significa "l'invio") dei dati tramite il metodo `ser.send`.

L'evento `on_sock_event`

Questo evento ha il compito di notificare al sistema operativo (TiOS) che è avvenuta una variazione di stato del socket. Il prototipo della funzione è il seguente

`on_sock_event(newstate as pl_sock_state, newstatesimple as pl_sock_state_simple)`

Le variabili `newstate` e `newstatesimple` contengono lo stato del socket al momento della generazione dell'evento. È un uso diverso da quello delle proprietà `sock.state` e `sock.statesimple`, le quali restituiscono invece lo stato corrente. In `newstate` è riportato lo stato dettagliato del socket, mentre `newstatesimple` contiene una rappresentazione semplificata al momento della generazione dell'evento; la Tabella 3 riassume i possibili stati semplificati in cui può trovarsi l'oggetto `sock`. Alcuni di questi stati sono propri del protocollo TCP, come ad esempio lo stato `PL_SST_PO` (infat-



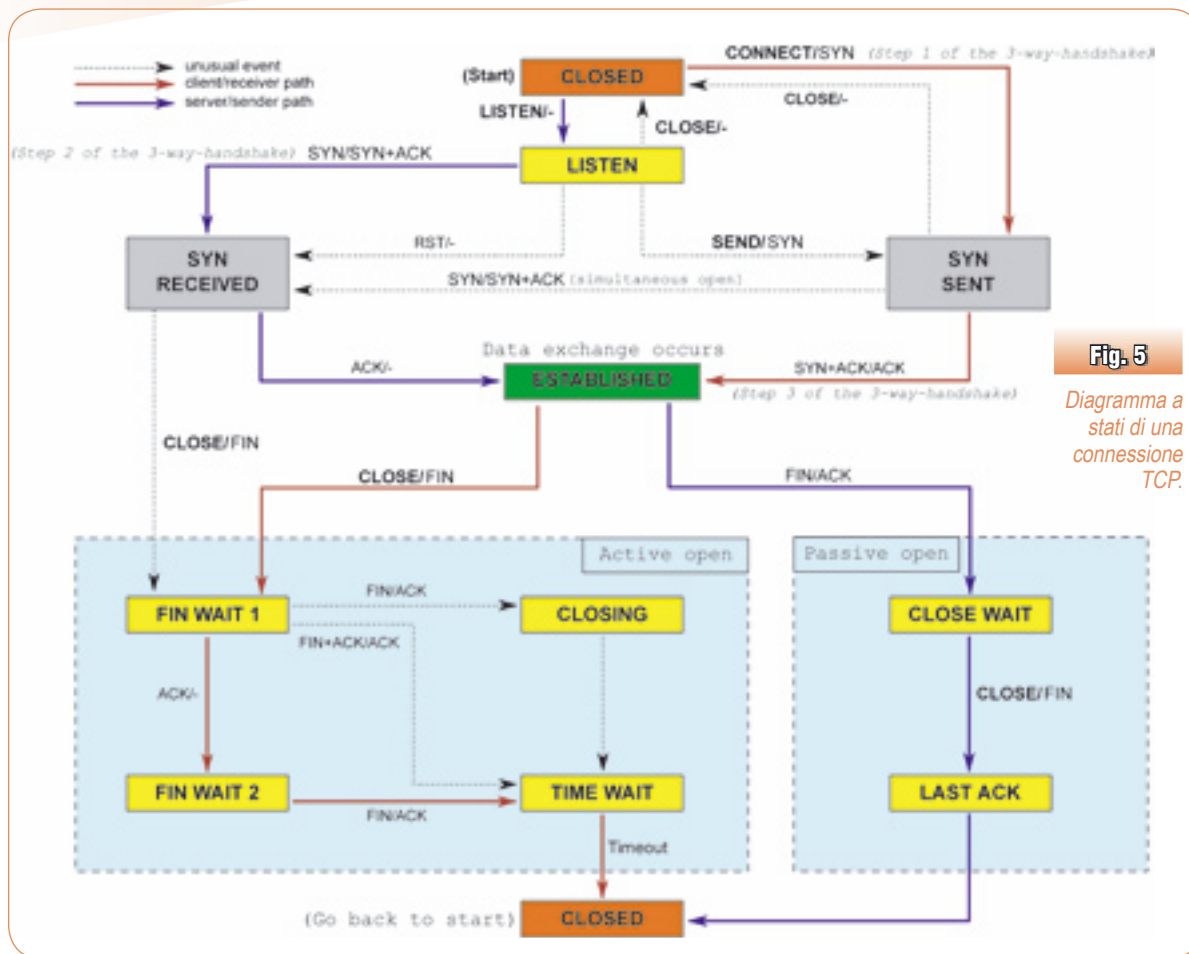


Fig. 5

Diagramma a stati di una connessione TCP.

ti, a differenza dell'UDP, il TCP è un protocollo di tipo "affidabile"). Per maggiori chiarimenti si faccia riferimento alla Fig. 5, la quale riporta il diagramma degli stati di una tipica connessione TCP. Ritornando alla nostra applicazione, l'evento appena descritto è stato utilizzato per segnalare quando viene stabilita la connessione. Poiché è stato definito un solo socket, non è necessario discernere quale socket ha subito la variazione. Secondo il codice riportato nel Listato 1, il led di stato verde rimane acceso a luce fissa se il modulo Tibbo è connesso, altrimenti effettuerà un doppio lampeggio. Allo stesso modo è possibile dare una

segnalazione visiva differente per ogni stato riportato nella Tabella 3.

Accesso con protezione

Nella scorsa puntata, parlando della proprietà `sock.inconmode` (quella che consente di impostare gli IP che possono accedere alla connessione con l'EM1000) si è detto che è possibile prevedere un accesso protetto al modulo. Questa problematica è particolarmente sentita in ambito LAN, e ancor più quando al dispositivo vi si può accedere da Internet: in questi casi si possono verificare "attacchi" non desiderati in grado di compromettere

Tabella 3 - Descrizione dei possibili stati dell'oggetto sock – rappresentazione semplificata

Valore	Descrizione
0 - PL_SSTS_CLOSED (default)	Connessione chiusa (UDP e TCP)
1 - PL_SSTS_ARP	Risoluzione indirizzo ARP in corso (TCP e UDP)
2 - PL_SSTS_PO	Connessione in corso – passive open (solo TCP)
3 - PL_SSTS_AO	Connessione in corso – active open (solo TCP)
4 - PL_SSTS_EST	Connessione stabilita (TCP e UDP)
5 - PL_SSTS_PC	Connessione chiusa – passive open (solo TCP)
6 - PL_SSTS_AC	Connessione chiusa – active open (solo TCP)

Listato 1

Riportiamo di seguito il codice sorgente completo dell'applicazione. Esso esegue quattro operazioni: all'accensione, configura gli oggetti *SER* e *SOCK*, che gestiscono rispettivamente la seriale e l'interfaccia Ethernet. Verifica quindi l'accesso al sistema tramite il controllo di una password. Infine, se la password è corretta, sovrintende al trasferimento dei dati da seriale a Ethernet e viceversa. Come potete notare, l'efficienza del TibboBasic è tale da permettere di scrivere programmi semplici per eseguire operazioni molto complesse.

```

=====
const pass_length = 8      ' La password è di otto
                           ' caratteri ed ha valore
const pass = "password"    ' predefinito password.
dim sbloccato as boolean   ' Definisce variabile.
=====
sub on_sys_init              ' Evento generato all'
                             ' avvio del dispositivo.
    dim w as word           ' Definisce variabile w.
    sbloccato = false       ' Imposta sbloccato.
    io.num=PL_IO_NUM_0      ' linea RTS seriale a 0
    io.enabled=YES          ' ossia configurata come
                             ' uscita.
    net.ip="192.168.1.95"    ' Indirizzo IP di Tibbo.
    ' Cambiarlo in accordo alla propria subnet!
    sock.num=0              ' Imposta socket 0.
    sock.rxbufrrq(4)        ' 4*256-16=1008 bytes
    ' per il buffer di ricezione sul socket 0.
    sock.txbufrrq(4)        ' 4*256-16=1008 bytes
    ' per il buffer di trasmissione sul socket 0.
    sock.num=0              ' E buona pratica, anche
                             ' se ridondante.
    ser.rxbufrrq(4)         ' 4*256-16=1008 bytes
    ' per il buffer di ricezione su seriale.
    ser.txbufrrq(4)         ' 4*256-16=1008 bytes
    ' per il buffer di trasmissione su seriale.
    sys.buffalloc           ' Allocations memoria
    sock.num=0              ' E buona pratica, anche
                             ' se ridondante.
    sock.protocol=PL_SOCK_PROTOCOL_TCP 'Impostazione
    ' protocollo TCP.
    sock.localportlist="2000,3000" 'Porta TCP su cui
    ' effettuare la connessione!
    sock.inconmode=PL_SOCK_INCONMODE_ANY_IP_ANY_PORT
    ' Sono accettate connessioni da qualsiasi host.
    sock.num=0              ' E buona pratica, anche
                             ' se ridondante.
    ser.flowcontrol=DISABLE 'Controllo di flusso
    ' disabilitato.
    ser.baudrate=ser.div9600/4 'Velocità seriale
    ' pari a 38400bps.
    ser.enabled=YES         ' Seriale abilitata.
    sock.setdata("Digitare password:") 'Richiesta
    ' password.
    sock.send               ' Invia dati su socket.
    pat.play("B-B-B-----",YES) ' Lampeggio del led
    ' verde e rosso per tre volte:
    ' indica avvenuta inizializzazione
end sub                     ' Fine routine avvio.
=====
sub on_sock_data_arrival    ' Dati ricevuti sull'
                             ' interfaccia Ethernet.
    ' Trasferimento verso la porta seriale
    if sbloccato then       ' Password già inserita.
        ser.setdata(sock.getdata(ser.txfree)) '
        ' Preleva dati da socket
        ser.send            ' e invia a seriale.
    else                    ' Password non inserita.
        if (sock.getdata(pass_length) = pass) then '
            ' Password inserita correttamente.
            sbloccato = true ' Accesso consentito.
            sock.setdata("ACCESSO CONSENTITO") '
            ' Conferma ricezione password con risposta.
            sock.send        ' Invia risposta.
        else                 ' Password sbagliata.
            sock.setdata("ACCESSO NEGATO")
            sock.setdata("Digitare password:")
            sock.send        ' Comunica che l'accesso
            ' viene negato e chiede nuovamente password.
        end if
    end if
end sub                     ' Fine sub.
=====
sub on_ser_data_arrival     ' Dati ricevuti su
                             ' seriale: trasferimento verso Ethernet
    sock.setdata(ser.getdata(sock.txfree))
    ' preleva i dati dal buffer seriale
    sock.send              ' e li invia al socket.
end sub                   ' Fine sub dati seriali.
=====
sub on_sock_event(newstate as pl_sock_state,
newstatesimple as pl_sock_state_simple)
    ' NOTA BENE: a capo inserito solo per fini
    ' editoriali. La definizione
    ' sub on_sock_event(newstate... deve rimanere su
    ' una sola riga.

    sock.num=0             ' Sono analizzati gli stati
    ' connesso e non connesso.
    if sock.statesimple=PL_SSTS_EST then
        pat.play("GGGGGGGGGGGGGGGG",NO)
        ' Se connessione stabilita
        ' LED verde acceso fisso
    else
        pat.play("-----G-G",NO)
        ' connessione non riuscita e
        ' LED verde lampeggia due volte.
    end if
end sub                   ' Fine evento connessione.
' Fine sub on_sock_event.

```

il funzionamento del modulo.

Per evitare tali problemi si è pensato di prevedere un accesso con password: oltre a specificare l'indirizzo IP, la porta e fare clic sul tasto connessi (dall'interfaccia client) sarà necessario immettere una password di accesso. È per questo motivo che nel Listato 1 sono state definite due costanti *pass* e *pass_length*, rispettivamente la password e la sua lunghezza. Oltre a ciò, è stata definita la variabile globale *sbloccato*, di tipo *boolean*: quando *sbloccato* è *false* (valore di default all'avvio), la conversione *Ethernet*>*seriale* è disabilitata, impedendo a chiunque non abbia i permessi di accesso di inviare dati sulla porta seriale.

Inoltre, anche la funzione *on_sock_data_arrival* è stata opportunamente modificata. Solo quando l'utente avrà immesso la password specificata nell'omonima costante, allora sarà abilitata

l'istruzione

ser.setdata(sock.getdata(ser.txfree))

il cui significato e funzionamento è già stato ampiamente spiegato nel corso dell'articolo.

Prove pratiche

Per verificare il codice riportato nel Listato 1, è necessario seguire le istruzioni seguenti:

- 1) Creare un nuovo progetto, come descritto sia nella terza che nella quarta puntata, e inserire in esso il codice riportato nel Listato 1 (scaricabile gratuitamente dal nostro sito www.elettronica.in.it). Compilarlo e trasferire il file oggetto nel modulo EM1000.
- 2) Aprire il software HyperTerminal sul PC B (secondo quanto riportato in Fig. 2) impostando i parametri di configurazione riportati in Fig. 6. ➤

3) Avviare l'interfaccia client sul PC A (Fig. 2) già descritta nella precedente puntata, quindi inserire negli appositi campi l'indirizzo IP che si è assegnato all'EM1000 (nell'esempio 192.168.1.95) e la porta di comunicazione (nell'esempio 2000). Attenzione: potrebbe essere necessario configurare il firewall di Windows per sbloccare la porta 2000. Ultimata la procedura di configurazione, fare clic su *Connetti*.

4) Per verificare la conversione Ethernet>RS232, scrivere del testo nel campo *Dati Inviati* del client. Ad esempio, in Fig. 7 è stato inserito il testo "Prova di trasmissione dati da Ethernet a Seriale". Fare clic sul pulsante *Invia* del client. Tale testo sarà immediatamente visualizzato HyperTerminal, come mostrato in Fig. 8.

5) In modo analogo, per testare la conversione RS232>Ethernet, scrivere del testo da HyperTerminal e verificare esso venga visualizzato all'interno del campo *Dati Ricevuti* del client.

Con riferimento allo schema in Fig. 2, quello che è stato fatto è inserire del testo all'interno di un pacchetto TCP (sulla porta 2000) e trasferirlo dal PC A al modulo Tibbo. Quest'ultimo ha fatto scattare l'evento *on_sock_data_arrival* e ha estratto l'infor-

mazione inserita nel buffer seriale, trasmettendola verso la porta seriale. I dati sono stati ricevuti dal PC B e visualizzati tramite HyperTerminal.

Note conclusive

Giunti a questo punto, le nuove nozioni acquisite vi permettono di estendere le potenzialità del modulo Tibbo, utilizzandolo come convertitore seriale-Ethernet. Ancora una volta si sono apprezzate le caratteristiche di compattezza e flessibilità del linguaggio di programmazione.

L'accesso con password ha consentito di aumentare la sicurezza, contro eventuali accessi non voluti. Nel corso delle prossime puntate sfrutteremo caratteristiche ancora più avanzate, che consentiranno di utilizzare l'EM1000 come web-server o mail-server. L'interfaccia client in Delphi descritta in queste due puntate pratiche, lascerà il posto ad una più flessibile interfaccia web per l'interazione con il dispositivo. Vedremo come caricare nella memoria di Tibbo una o più pagine HTML e visualizzarle col browser.

Da ora si fa sul serio! ...e sarà molto divertente! Appuntamento, quindi al mese prossimo. Non mancate.

Fig 6 - Parametri di configurazione del software HyperTerminal per il corretto invio di caratteri dal PC B alla seriale dell'EM1000.

Fig. 7 - Invio dati da Ethernet a seriale tramite il programma client. Notate l'impostazione dell'indirizzo IP e della porta TCP.

Fig. 8 - Ricezione dati e visualizzazione tramite HyperTerminal di quanto trasmesso dalla finestra *Dati Inviati* del client.

