

160 PAGINE!

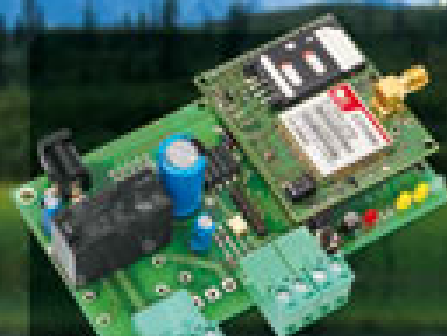
Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

€ 5⁰⁰

Elettronica In

www.elettronica.in.it

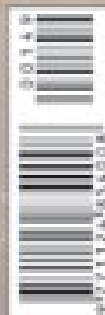
Anno XVI - n. 149
Agosto / Settembre 2010



**Termostato
ambiente con GSM**



Regolatore di carica per pannelli solari

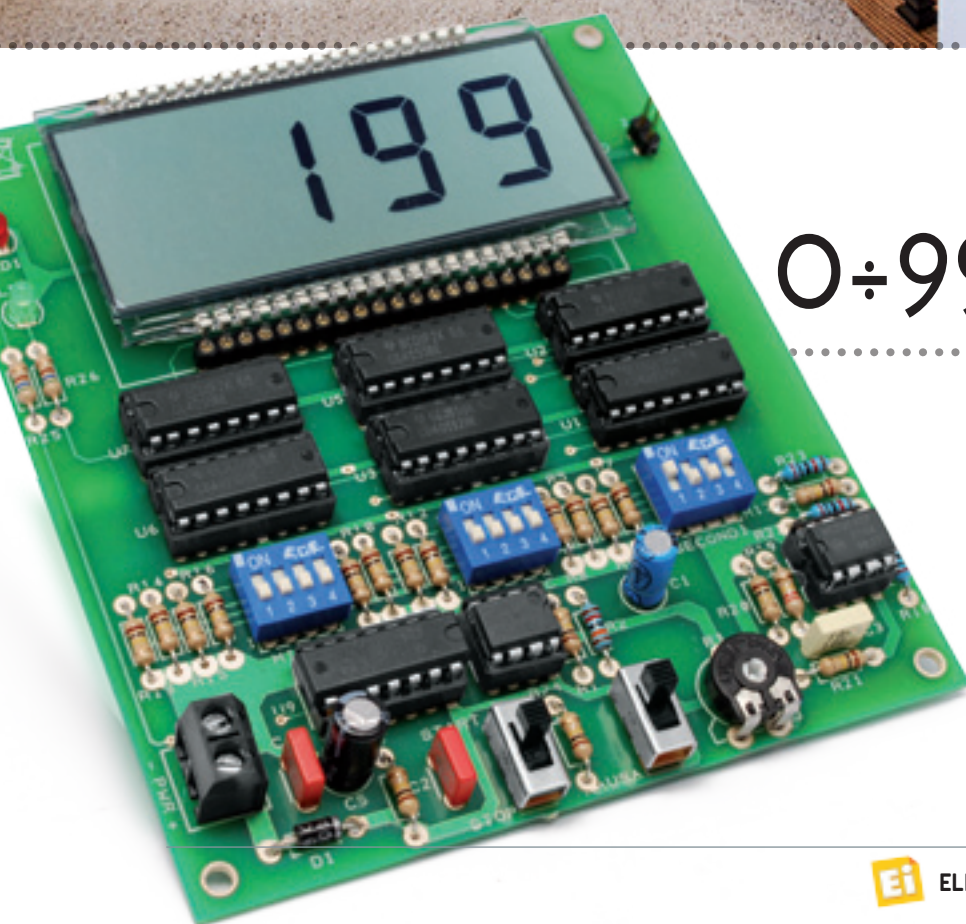


- Nanoterm, il piccolo terminale universale
- Timer 0 + 999 secondi
- Touch Sensing
- Tutorial PicoBlaze
- Consumi elettrici sotto controllo
- Photo Geotagging
- La serra domotica
- Realizziamo il Voip senza PC
- Fotocamera triggerabile
- Corso HW/SW Arduino



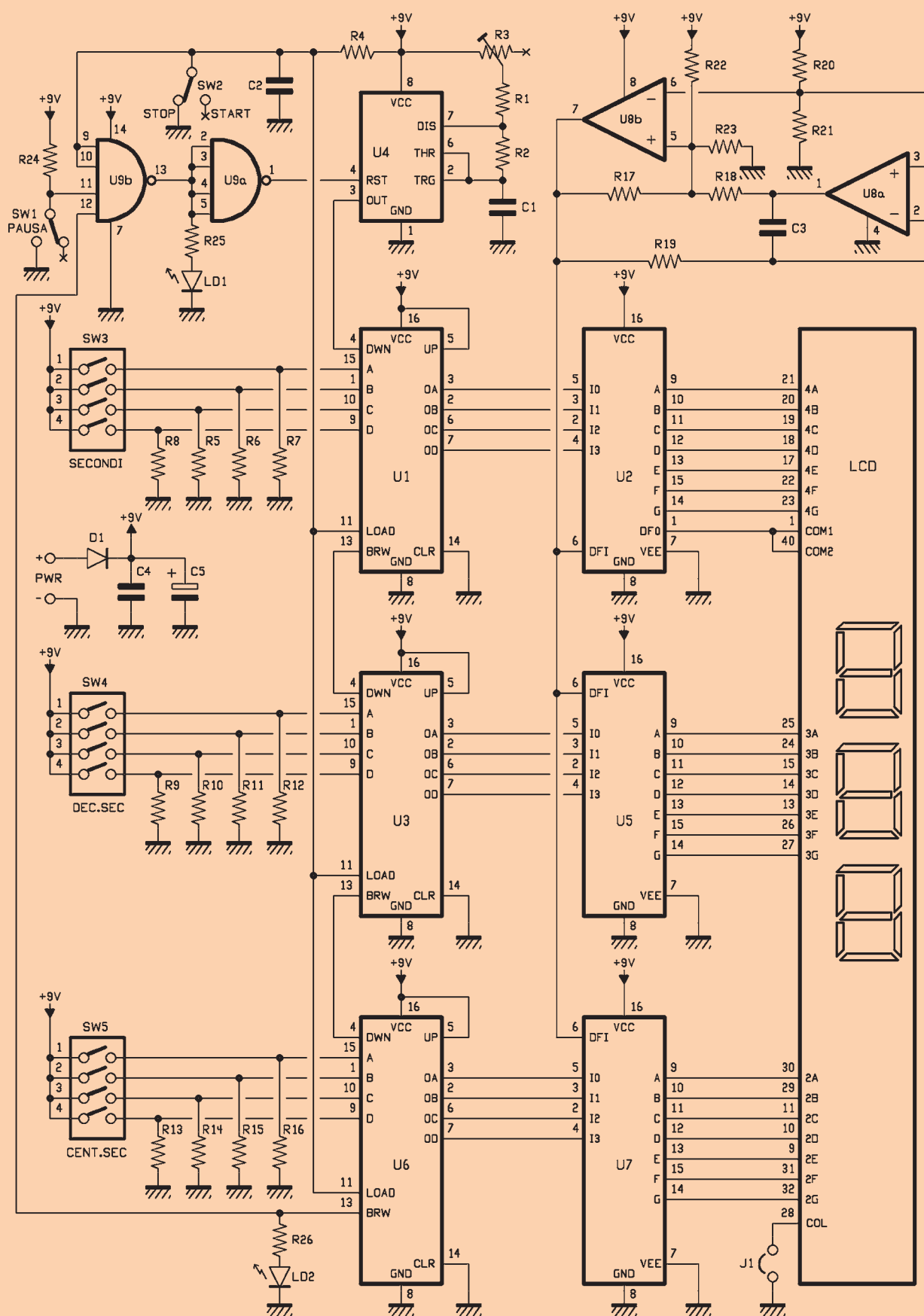
Corso LabVIEW

Temporizzatore elettronico programmabile tra 0 e 999 secondi, che segnala mediante LED la temporizzazione in corso e lo scadere del tempo; è provvisto di un display a cristalli liquidi che assiste nelle impostazioni e mostra l'avanzamento del tempo.



TIMER 0÷999 SECONDI

Le occasioni in cui serve accendere o spegnere a tempo un apparecchio elettronico, un motore elettrico, una lampadina o una macchina gestibile elettricamente, sono tantissime e variegate; è questa la ragione per cui, malgrado la sua semplicità, il timer risulta tuttora uno dei progetti più richiesti dai nostri lettori. Abbiamo dunque pensato di preparare un nuovo schema e proporvelo in queste pagine: si

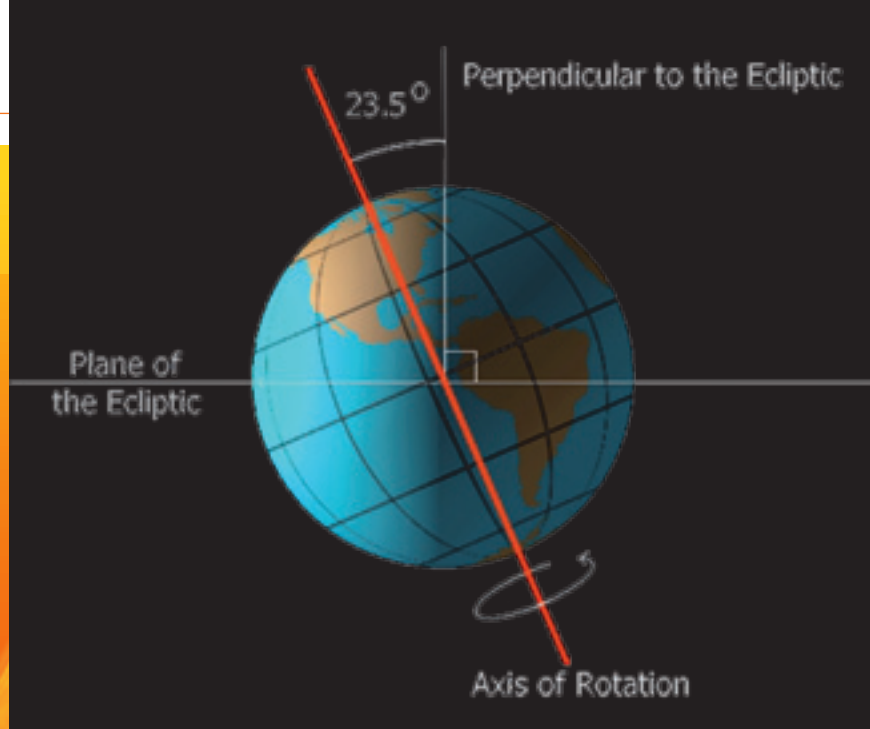


tratta di un temporizzatore generico capace di conteggiare un tempo fino a 999 secondi (circa un quarto d'ora) impostabile mediante dip-switch.

La particolarità di questo circuito è che viene realizzato completamente con logica discreta. Sì, è vero che in questi anni ci siamo distinti per aver proposto progetti in linea con la più recente tecnologia, basati su integrati di ultima generazione, ma anche per l'aver riproposto dispositivi tradizionali rivisitati in versione gestita da microcontrollore; però riteniamo giusto, ogni tanto, pubblicare qualche circuito che funzioni con l'elettronica tradizionale.

Questa scelta è per accontentare chi, tra i nostri lettori, non ama mettersi a programmare, ma rappresenta anche una buona occasione per spiegare un po' di elettronica vera e propria, dal momento che il funzionamento del circuito si può comprendere immediatamente dalla semplice analisi dello schema elettrico, senza andare a vedere listati e roba simile.

Ma lasciamo da parte questo discorso e parliamo un po' di questo timer: dispone di un visualizzatore LCD a 3 digit e mezzo pilotato direttamente da driver per display a 7 segmenti, che segnala il tempo impostato e, dopo l'avvio, aggiorna il tempo restante, effettuando un vero e proprio conto alla rovescia. Il tempo da contare si imposta mediante dip-switch, che sono quattro per i secondi, altrettanti per le decine di secondi e ancora quattro per le centinaia di secondi; un deviatore permette di avviare il conto alla rovescia o di fermare e resettare il timer. Un interruttore consente di mettere in pausa il conteggio ma senza azzerare il temporizzatore.



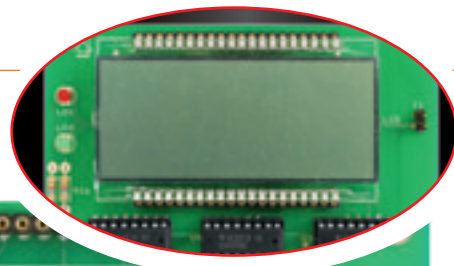
Come cambia il tempo...

Secondo studi approfonditi, il terremoto che recentemente ha colpito il Cile ha accorciato la durata del giorno e spostato l'asse terrestre: il nostro pianeta gira più velocemente e le giornate si sono accorciate di 1,26 milionesimi di secondo, mentre l'asse di rotazione si è spostato di 2,7 millisecondi di arco, pari a 8 centimetri. La notizia arriva dalla NASA, la quale ha calcolato le variazioni tramite un modello matematico complesso. Secondo l'ideatore di questo modello, la zolla di Nazca, che subduce sotto quella sudamericana e ha generato il terremoto cileno, ha spostato masse di crosta terrestre verso l'interno del pianeta. Come avviene nei pattinatori quando, durante la trottola, portano le braccia al petto e aumentano la velocità di rotazione, così capita al nostro pianeta. Masse più vicine al centro della Terra determinano una maggiore velocità di rotazione e quindi un accorciamento della durata del giorno. Gli 1,26 microsecondi di accorciamento del giorno sono un cambiamento minimo ma permanente, che è addirittura sotto la soglia dell'osservazione diretta strumentale, che è di 5 microsecondi. Anche il terremoto di Sumatra del 26 dicembre 2004, che è stato più forte (9,1 gradi Richter) aveva causato importanti cambiamenti: una diminuzione della durata del giorno di 6,8 microsecondi e uno spostamento dell'asse terrestre di 2,32 millisecondi di arco, pari a circa 7 centimetri.

SCHEMA ELETTRICO

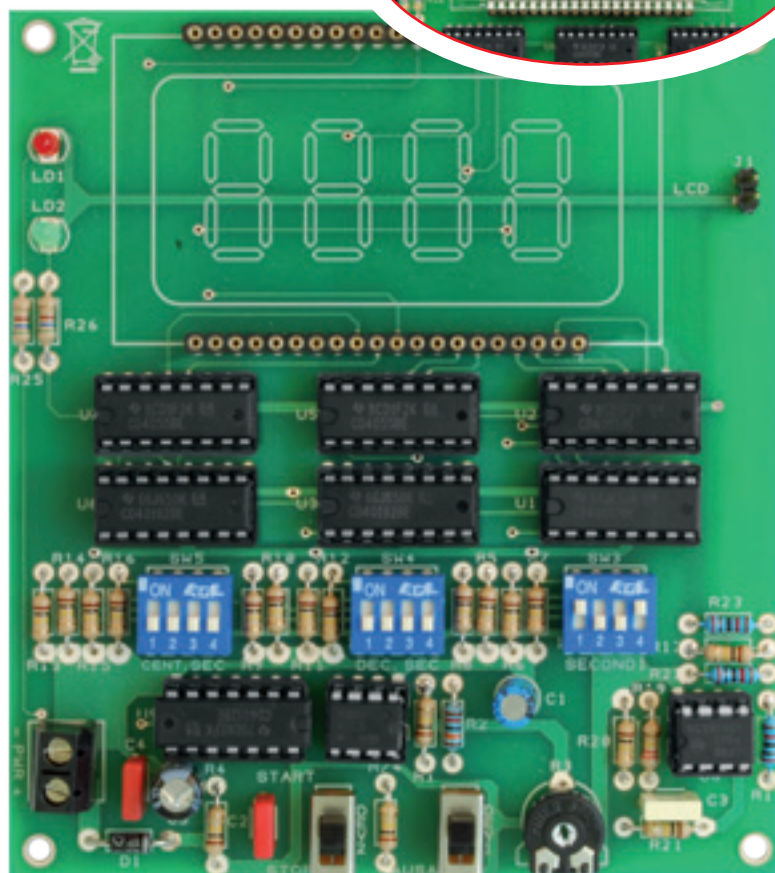
Diamo dunque uno sguardo al circuito esaminandone lo schema elettrico: il conteggio viene effettuato mediante una logica composta da tre contatori decimali UP/DOWN presettabili, collegati in cascata mediante l'uscita BRW (Borrow Out) e l'ingresso DOWN

(quindi impostati per contare all'indietro) il primo dei quali riceve il clock da un multivibratore astabile realizzato con un NE555 configurato, appunto, da astabile, con i componenti di temporizzazione (R1, R2, C1) calcolati per generare una frequenza di 1 Hz. Per l'esattezza, la frequenza



Elenco Componenti:

R1, R24: 100 kohm	U8: MC33202
R2: 13 kohm 1%	U9: CD4012
R3: Trimmer 47 kohm MO	SW1÷SW3: Dip-Switch 4 vie
R4÷R17: 100 kohm	DEV1: Deviatore a slitta
R18: 20 kohm 1%	DEV2: Deviatore a slitta
R19: 220 kohm	D1: 1N4007
R20, R21: 100 kohm	LD1: Led 3 mm rosso
R22, R23: 20 kohm 1%	LD2: Led 3 mm verde
R25, R26: 560 ohm	LCD: Display LCD (LCD-S401C71TR)
C1: 10 µF 16 VL elettrolitico	
C2: 100 nF 63 VL poliestere	
C3: 10 nF 63 VL poliestere	Varie:
C4: 100 nF 63 VL poliestere	- Morsettiera 2 poli
C5: 220 µF 16 VL elettrolitico	- Zoccolo 4+4 (2 pz.)
U1, U6: CD40192	- Zoccolo 7+7
U2, U5, U6: CD4055	- Zoccolo 8+8 (6 pz.)
U3: CD40192	- Jumper
U4: NE555	- Strip maschio 2 poli
	- Strip femmina torniti 20 poli (2 pz.)
	- Circuito stampato



prodotta dall'U4 è pari:

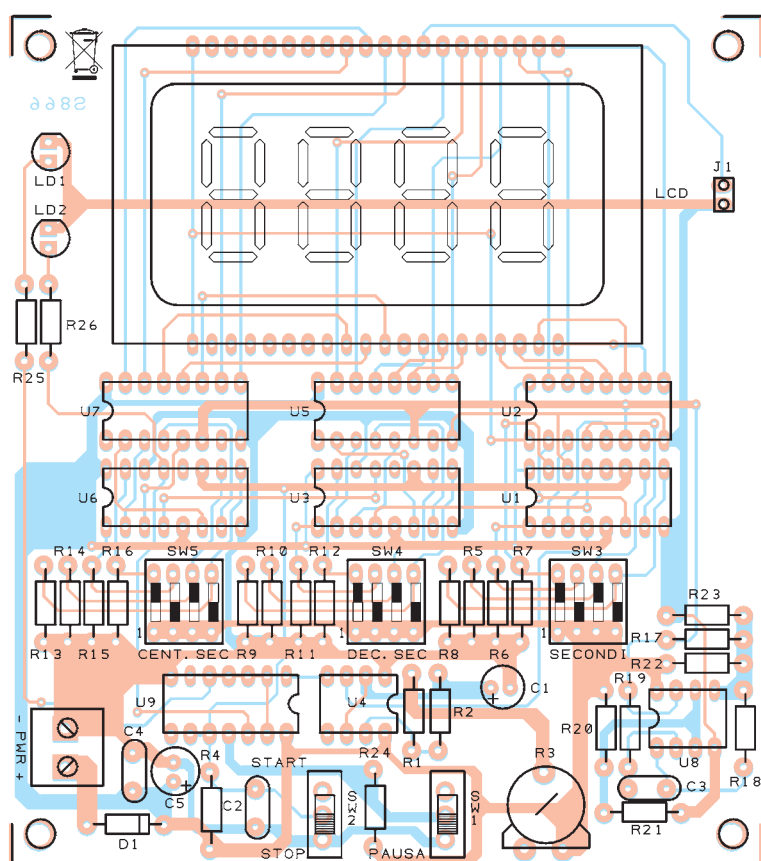
$$f = 1,44 / [(R1+R3+2 \times R2) C1]$$

e risulta in Hz se le resistenze sono espresse in Mohm e il condensatore in microfarad. Essendo R3 un trimmer, la frequenza può essere modificata a piacimento, entro certi limiti. Diciamo comunque che per far funzionare il timer come previsto la frequenza deve essere 1 Hz esatto.

L'uscita di ciascuno dei contatori esprime un valore numerico tra 0 e 9 in formato binario, su 4 bit, che viene inviato all'ingresso di un driver BCD per display a 7 segmenti, che pilota un digit del display a cristalli liquidi (LCD). L'uscita BRW dell'ultimo contatore (U6) va a finire su uno degli ingressi di una NAND a 4 ingres-

si (U9b) della quale altri due input vengono gestiti dal deviatore SW2, cui è affidato il compito di avviare o arrestare e resettare il timer. L'ultimo ingresso della NAND viene pilotato dal deviatore SW1, che serve a mettere in pausa il timer. L'uscita (pin 13) della prima NAND viene invertita dalla seconda porta NAND contenuta nello stesso integrato (un CD4012) così da ottenere uno stato logico alto sul piedino di reset dell'NE555 quando SW1 o SW2 sono chiusi a massa; ciò sortisce l'effetto di bloccare il clock, però va notato che nel primo caso il gruppo di contatori resta "congelato" e mantiene il conteggio effettuato al momento, mentre nel secondo il blocco dell'NE555 è accompagnato dal reset del gruppo di conta-

tori, che si deve alla forzatura a zero logico della linea LOAD di U1, U3 e U6. Attenzione che in realtà il reset non corrisponde all'azzeramento dei contatori ma al semplice ripristino del tempo iniziale, in quanto portando a massa le linee LOAD ognuno dei CD40192 carica nuovamente il valore dei dip-switch. L'arresto del timer (messa in pausa o stop con ripristino del tempo iniziale) viene segnalato dall'accensione del LED rosso (LD1) mentre il LED verde si illumina quando il temporizzatore è in fase di conteggio o deve iniziare una nuova temporizzazione. A tempo esaurito naturalmente (il display segna 000) si spegne il led verde e si illumina quello rosso. Quando il timer viene messo in pausa mediante SW1, LD2 non



subisce alcuna alterazione e resta com'era. Scendiamo adesso nel dettaglio e vediamo come funziona esattamente il conteggio: ognuno dei contatori lavora con l'ingresso di CLEAR (pin 14) a zero logico, quindi viene fatto decrementare solo dal clock ricevuto al piedino 4; quest'ultimo è l'ingresso di clock per il conteggio all'indietro, mentre l'input di clock per il conteggio in avanti è posto a livello alto, come previsto dal costruttore del chip. Per entrambi i piedini di input, si ottiene l'avanzamento o arretramento di un'unità in corrispondenza di ogni transizione 1/0 logico. La

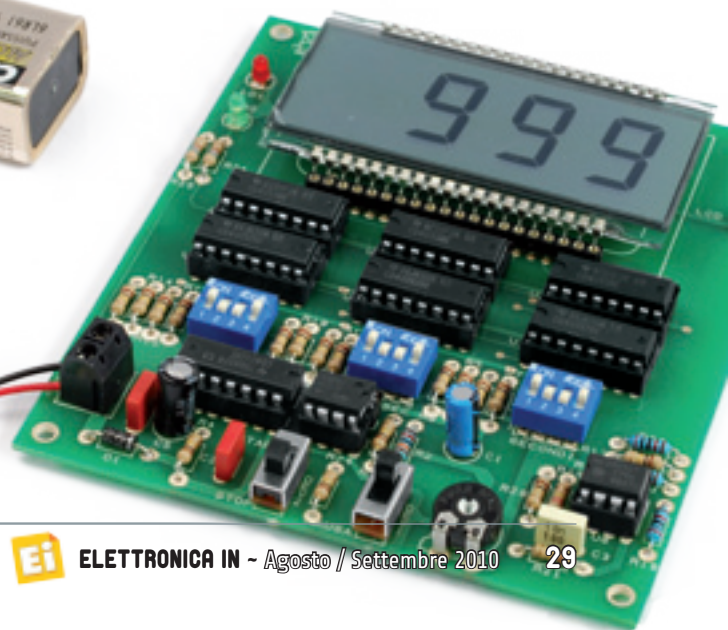
particolarità dei contatori UP/DOWN CD40192 è che, oltre a poter contare avanti o indietro, possono farlo a partire da un determinato valore (preset)

impostabile agli ingressi digitali (attivi ad 1 logico, nel senso che 5 V valgono 1 logico) A, B, C, D, corrispondenti rispettivamente ai piedini 15, 1, 10, 9; tali linee hanno peso binario pari a 1, 2, 4, 8 ed il loro stato viene acquisito tramite un latch interno all'integrato che carica i relativi dati quando la linea LOAD (piedino 11) passa da zero ad 1 logico e il pin 14 si trova a livello basso. La possibilità di presetare i contatori individualmente ci permette proprio di programmare il timer mediante semplici dip-switch, tenendo presente che dip chiuso corrisponde ad 1 logico sulla linea corrispondente del CD40192; in pratica il valore impostato mediante i dip è quello da cui parte il conteggio del rispettivo contatore. Ad esempio, se vogliamo che U1 parta da 8, dobbiamo chiudere il solo dip 4 (8 binario) del dip-switch corrispondente, mentre per partire da 5 bisogna che chiudiamo i soli dip 1 e 3 (1+4).

Il dip-switch associato ad U1 imposta i secondi, quello relativo a U3 definisce le decine di secondi e l'ultimo (collegato ad U6) consente di impostare le centinaia di secondi. Una volta impostati i valori di partenza dei tre contatori, avviando il timer il conteggio



Il display si monta su uno zoccolo ottenuto da due file di 20 contatti. Una volta completato, il timer si alimenta con una pila da 9 V.



Secondi	dip 1	dip 2	dip 3	dip 4
100	ON	OFF	OFF	OFF
200	OFF	ON	OFF	OFF
300	ON	ON	OFF	OFF
400	OFF	OFF	ON	OFF
500	ON	OFF	ON	OFF
600	OFF	ON	ON	OFF
700	ON	ON	ON	OFF
800	OFF	OFF	OFF	ON
900	ON	OFF	OFF	ON

Il tempo del timer si imposta mediante i tre dip-switch SW3, SW4, SW5: il primo definisce i secondi, il secondo le decine di secondi ed il terzo le centinaia di secondi. Ogni dip-switch compone in formato binario il valore corrispondente. Le tabelle illustrano l'impostazione di ciascuno. Nella figura, vedete come impostare i dip per ottenere, ad esempio, 199 secondi.



Secondi	dip 1	dip 2	dip 3	dip 4
1	ON	OFF	OFF	OFF
2	OFF	ON	OFF	OFF
3	ON	ON	OFF	OFF
4	OFF	OFF	ON	OFF
5	ON	OFF	ON	OFF
6	OFF	ON	ON	OFF
7	ON	ON	ON	OFF
8	OFF	OFF	OFF	ON
9	ON	OFF	OFF	ON

Secondi	dip 1	dip 2	dip 3	dip 4
10	ON	OFF	OFF	OFF
20	OFF	ON	OFF	OFF
30	ON	ON	OFF	OFF
40	OFF	OFF	ON	OFF
50	ON	OFF	ON	OFF
60	OFF	ON	ON	OFF
70	ON	ON	ON	OFF
80	OFF	OFF	OFF	ON
90	ON	OFF	OFF	ON

procederà decrementando, per ognuno dei CD40192, a partire dal valore iniziale definito dai dip-switch.

Il collegamento a cascata fa in modo che quando un contatore è arrivato a zero, l'uscita BRW fornisca un impulso a zero logico all'ingresso di clock DWN del contatore successivo; dopo l'azzeramento, ogni contatore riprende il conteggio e lo fa a partire da 9, fino a scendere a 10. Ciò vuol dire che al primo conto alla rovescia parte dal valore di preset definito dal dip-switch abbinatogli, mentre dopo essere arrivato a zero riprende da 9 e conta dieci passi, dopodiché fornisce un nuovo impulso al contatore successivo.

L'impulso a zero logico dovuto all'arrivo a zero dell'U1 fa da clock all'U3, che quindi arretrerà

di un'unità rispetto al valore di preset, ogni arrivo a zero dello stesso U1; quando U3 arriva a zero, riprende il conteggio da 9 indipendentemente dal suo valore di preset, fornendo contemporaneamente un impulso a livello basso all'input DWN dell'U6. Quando questo, partendo dal valore di preset impostato dal relativo dip-switch, raggiunge zero, il blocco di contatori si arresta, perché lo zero logico sull'uscita BRW si presenta sul piedino 12 della NAND U9b, forzando ad 1 logico l'uscita di quest'ultima e portando a zero il piedino 1 dello stesso integrato (uscita della U9a). Tale condizione ferma l'NE555, la cui uscita si porta a livello logico alto; inoltre fa spegnere il LED verde (che si illumina da quando SW2 viene messo in posizione START). Una volta giunti alla fine del conteggio, per ripartire con una nuova sessione di temporizzazione bisogna riportare SW2 in posizione di stop, di modo che il gruppo di contatori possa essere ripristinato. Ogni contatore fornisce lo stato del proprio conteggio su quattro linee in formato binario

(OA, OB, OC, OD) che valgono 1, 2, 4, 8; il numero binario corrispondente viene letto da U2 per U1, U5 per U3 e U7 per U6. U2, U5 e U7 sono driver per display a cristalli liquidi a 7 segmenti, di tipo CD4055, che qui utilizziamo per pilotare i digit interi del display siglato LCD (una linea per segmento). Il clock di riferimento per l'LCD viene generato dai CD4055, però in questo caso usiamo l'uscita del solo integrato U2, che comanda i piedini COM1 e COM2 del display mediante la linea DF0. Il discorso funziona perché non ci sono problemi di coerenza temporale tra le temporizzazioni dei singoli digit. Notate, infine, il jumper J1, collegato tra il piedino 28 del display e massa: permette di attivare i due punti posti tra il secondo e il terzo digit; potete scegliere se usarlo o meno, sebbene in realtà il contatore proceda in modo decimale e non sessagesimale, quindi in linea di massima sia inutile usare i due punti. Concludiamo l'analisi dello schema elettrico con l'alimentazione, che si applica ai contatti + e - PWR: il circuito richiede 9

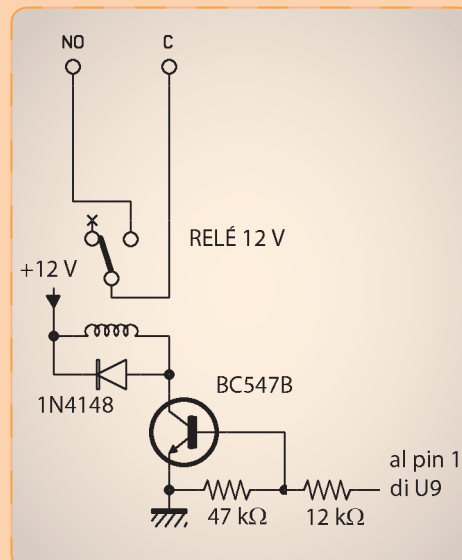
volt in continua e una corrente di una quarantina di milliamper (funziona bene anche con una pila a secco da 9 V); il diodo D1 lo protegge dall'inversione della polarità dei collegamenti di alimentazione e lascia passare la corrente solo quando la polarità è positiva sul contatto + rispetto al -. La tensione presente a valle del diodo viene filtrata dai condensatori C4 e C5, necessari ad abbattere disturbi impulsivi e l'eventuale ripple presente se il circuito si alimenta con un alimentatore da rete.

REALIZZAZIONE PRATICA

Bene, adesso che conoscete il funzionamento, possiamo vedere come costruire il timer; il tutto si realizza partendo dal circuito stampato a doppia faccia che potete ottenere per fotoincisione dopo aver scaricato le relative tracce lato rame dal nostro sito web www.elettronicaain.it. Per il montaggio dei componenti, partite con le resistenze, il diodo D1, gli zoccoli per gli integrati e i dip-switch, quindi proseguite con il trimmer, i condensatori (sistemando prima quelli non polarizzati e dopo gli elettrolitici) e i deviatori miniatura da c.s. Poi montate i LED, rammentando che il loro catodo è in corrispondenza del lato smussato del contenitore (l'anodo è usualmente il terminale più lungo) e completate l'opera con la morsettiera di alimentazione e due strip femmina a passo 2,54 mm da 20 contatti, che disporrete in corrispondenza delle piazzole riservate al display. Le due file di contatti faranno da zoccolo per l'inserimento dell'LCD. In tutte le fasi del montaggio tenete d'occhio la disposizione componenti che trovate illustrata in queste pagine: vi aiuterà a non sbagliare il verso di inserimento

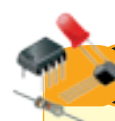
Sebbene il temporizzatore qui descritto non disponga di uscite vere e proprie ma solo di LED usati per segnalare l'inizio e la fine del conteggio, è possibile collegargli un relé con il quale gestire un utilizzatore elettrico. Per il comando del relé (che può essere a 12 V anche se il circuito funziona a 9÷10 volt) potete usare lo schema qui accanto, che impiega un transistor e pochi altri componenti. Con questa soluzione il relé verrà eccitato quando inizierà il conteggio e tornerà a riposo allo scadere del tempo, ma anche se SW1 o SW2 verranno posti a massa (STOP o PAUSA).

Comandare un relé



di condensatori elettrolitici, diodi, integrati, dip-switch e display. A proposito, l'LCD va disposto in modo che il suo lato corto, riportante un rigonfiamento, si trovi dalla parte dei LED. Completato il montaggio, il timer prima di essere utilizzato deve essere tarato, nel senso che occorre regolare la posizione del cursore del trimmer per assicurarsi che l'NE555 generi 1 Hz; allo scopo dovreste avere un preciso frequenzimetro e collegarlo tra il piedino 3 dell'U4 e massa, ma la regolazione potete effettuarla impostando un tempo di 60 secondi e verificando con un cronometro la corrispondenza del tempo evidenziato dal display. Se il timer finisce i 60 secondi prima del cronometro R3 va ruotato in modo da aumentarne la resistenza (verso sinistra) mentre in caso contrario la rotazione deve essere nel verso opposto, in modo da ridurre la resistenza e accelerare il conteggio. Concludiamo con un'ultima nota: l'uscita del timer è sostanzialmente rappresentata dai LED; se dovete comandare un circuito digitale potete utilizzare direttamente lo stato logico presente sul piedino 13 dell'U9. Se invece intendete gestire un utilizzatore elettrico vi serve un

relé, di cui potete comandare la bobina di eccitazione mediante un transistor NPN collegato con l'emettitore a massa, il collettore su un capo e la base connessa al piedino 1 dell'U9 mediante una resistenza da 12 kohm. Riferitevi all'apposito disegno in queste pagine. Così facendo, quando il timer è in conteggio il relé viene eccitato e torna a riposo allo scadere del tempo; viene diseccitato anche se si sposta il deviatore SW1 per mettere in pausa il temporizzatore.



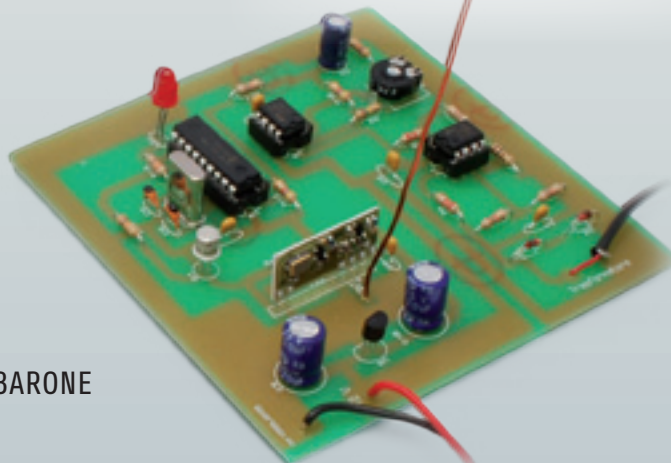
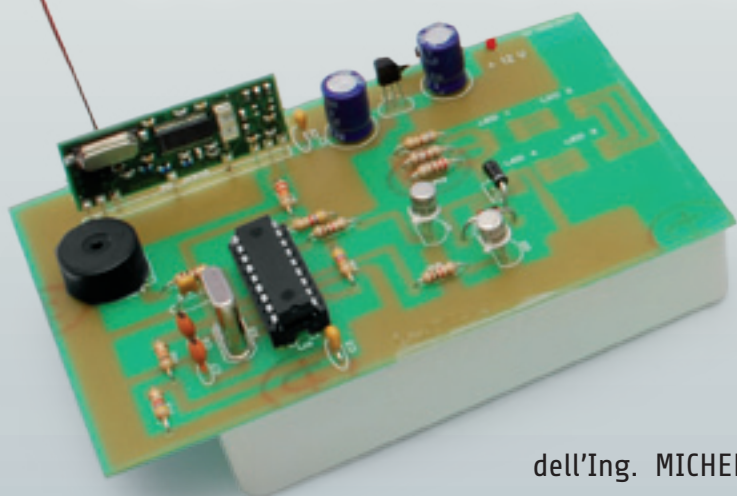
per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista (www.elettronicaain.it) ed utilizzato per realizzare la basetta. Il display LCD a 3 cifre e mezza (cod. 1450-LCD31-2) può essere acquistato presso Futura Elettronica al prezzo di 6,20 Euro.

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

CONSUMI ELETTRICI SOTTO CONTROLLO

Collegato all'impianto elettrico di casa, misura la corrente assorbita dall'insieme dei dispositivi collegati alla rete fornendo un'indicazione visiva proporzionale al consumo: il display cambia di colore, passando dal verde al rosso attraverso tutte le sfumature di giallo ed arancio, per un totale di 256 combinazioni di colore. Il visualizzatore a led è collegato al sensore di corrente in modalità wireless per poter essere collocato dove ritenuto più utile.



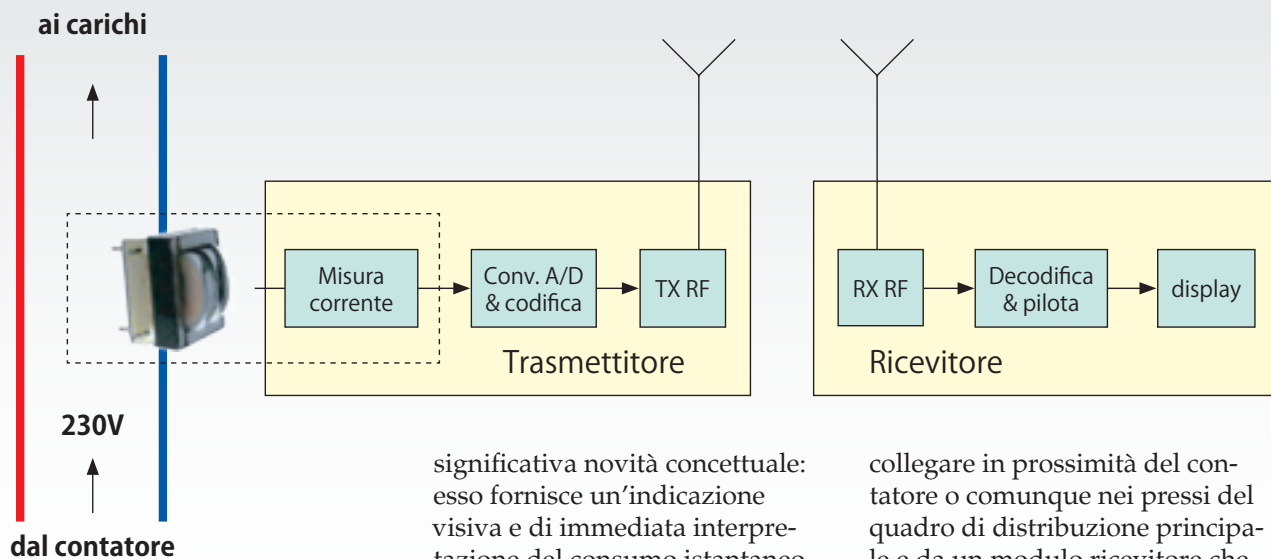
dell'Ing. MICHELE BARONE

Il contenimento dei consumi energetici, la riduzione degli sprechi, le fonti di energia alternativa: sono alcuni tra i fattori che sempre più diverranno parte del nostro vivere quotidiano. Negli ultimi anni, come diretta conseguenza della crescente domanda di energia e dell'assoluta necessità di ridurre gli impatti climatici, si è creata una convergenza di interessi scientifici, politici e industriali su come le tecnologie possano abilitare un processo di trasformazione di ogni fase del ciclo energetico, dalla generazione fino all'accumulo, al trasporto, alla distribuzione e soprattutto

all'uso intelligente dell'energia. Con riferimento alla *information and communication technology* (ICT), questo connubio viene indicato con il termine *Smart Grid*. Tra gli obiettivi dichiarati della Smart Grid è indicata la fornitura ai consumatori di strumenti utili ad ottimizzare il funzionamento del sistema globale. La Smart Grid è oggi una realtà molto più solida e diffusa di quanto si possa immaginare: basti pensare che Barack Obama, con lo *Stimulus Plan* lanciato a febbraio 2009 per risanare l'economia statunitense, ha stanziato ben 11



Schema a blocchi



miliardi di dollari "to jump start the transformation to a bigger, better, smarter grid".

Il progetto che proponiamo può essere annoverato tra quei prodotti utili al consumatore per consentirgli di diventare parte attiva e consapevole della fase del ciclo energetico che gli compete: l'utilizzo dell'energia.

Sulle pagine di questa rivista sono già stati presentati interessanti progetti utili al monitoraggio ed alla gestione dei consumi: circuiti che interrompono l'alimentazione dei carichi secondo predefinite priorità al superamento di una soglia di consumi, circuiti che misurano e visualizzano la potenza assorbita.

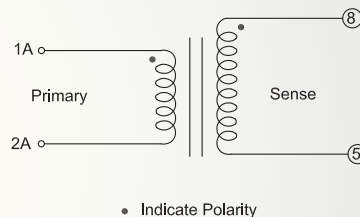
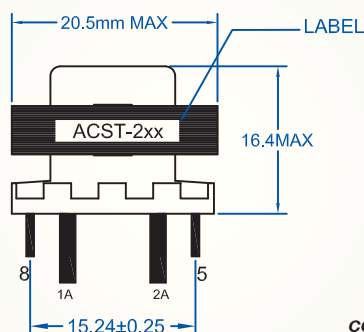
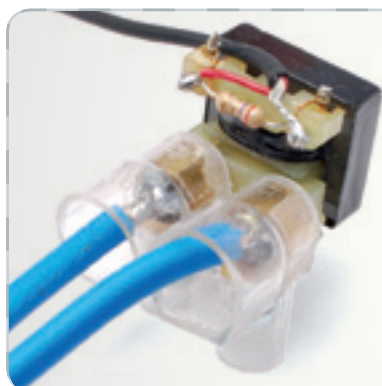
Il progetto che presentiamo in queste pagine introduce una

significativa novità concettuale: esso fornisce un'indicazione visiva e di immediata interpretazione del consumo istantaneo e mette il consumatore (e la sua intelligenza) al centro del processo di gestione dell'impianto: sarà il consumatore a decidere se in un dato momento gli converrà staccare un carico e quale carico staccare, ovvero se invece gli converrà approfittare di un momento di consumi bassi per attivare determinati apparecchi. In caso di superamento della potenza contrattuale sarà possibile ottenere una segnalazione acustica che permetterà all'utente di mettere rapidamente in atto le opportune azioni correttive. Dal punto di vista didattico, il progetto si segnala per la tecnica di misura (induttiva) della corrente assorbita e per la gestione PWM del display.

L'apparecchio è costituito da un modulo trasmettitore fisso, da

collegare in prossimità del contatore o comunque nei pressi del quadro di distribuzione principale e da un modulo ricevitore che potrà essere collocato nel punto più idoneo dell'appartamento e potrà essere spostato all'interno dell'abitazione, secondo le esigenze dell'utente. Il display del ricevitore è costituito da una matrice di led RGB che cambiano di colore, passando dal verde al rosso al crescere del consumo di energia.

Nello schema a blocchi dell'apparecchio, sono identificate le componenti logiche principali. Il blocco di misura acquisisce il valore istantaneo della corrente che fluisce lungo la linea. Pertanto, a rigore, la misura che si ottiene non è proporzionale alla potenza attiva bensì alla potenza apparente (sulla misura della potenza si veda l'interessante approfondimento su Elettronica In, ottobre 2008 n. 131, "Energy



Trasformatore utilizzato per rilevare la corrente assorbita dall'impianto di casa.

La misura della corrente

I trasduttori a shunt si basano sull'applicazione della legge di Ohm: $V = R \cdot I$. La corrente da misurare viene fatta scorrere attraverso un resistore di valore noto; misurando la caduta di tensione ai capi del resistore si ottiene un'indicazione proporzionale alla corrente. Uno dei principali svantaggi degli shunt è costituito dalla loro stessa natura: introducono una caduta di tensione nel circuito.

Per questo motivo è necessario che la caduta di tensione introdotta dal resistore sia la più bassa possibile e quindi il resistore R si sceglie di valore molto basso: da pochi microhm per la misura di correnti superiori ai 1000 A, fino ai decimi di ohm per correnti dell'ordine della decina di A. D'altra parte gli shunt sono trasduttori semplici, robusti e economici. Da segnalare, tra gli aspetti negativi, la dissipazione di potenza ($P = R \cdot I^2$), il conseguente innalzamento di temperatura del resistore e la variazione di resistività con la temperatura. Altro aspetto negativo è costituito dalla mancanza di isolamento.

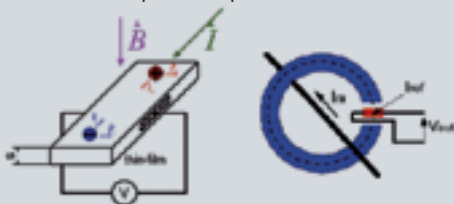
I trasduttori ad effetto Hall e magnetoresistivi si basano sulla forza di Lorentz: i portatori di carica immersi in un campo magnetico sono sottoposti ad una forza proporzionale



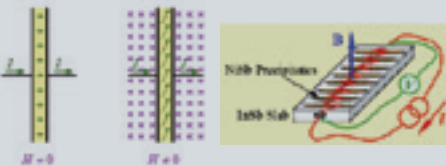
al campo.

Nei sensori ad effetto Hall la forza "sposta" i portatori di carica causandone l'accumulo in una parte del materiale sottoposto al passaggio delle cariche e di conseguenza si produce una differenza di potenziale nel materiale (la "barretta Hall"). Note le relazioni tra il campo, la corrente che la produce, la corrente nella barretta e la ddp prodotta, si può realizzare un circuito per la misura della corrente che produce il campo.

I sensori ad effetto Hall sono per lo più utilizzati nelle pinze amperometriche.



Nel caso dei sensori magnetoresistivi, la forza applicata ai portatori di carica ne allunga il percorso, producendo un effetto simile ad un aumento della resistenza del materiale sottoposto al campo magnetico. Applicando una tensione nota ai capi di una magnetoresistenza, la corrente misurata sarà fun-

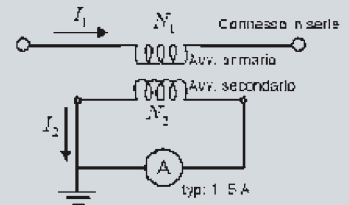


zione del campo magnetico. Note le relazioni tra il campo, la corrente che la produce e la resistenza della barretta si può ottenere una misura analogica della corrente da misurare (che produce il campo magnetico). I sensori magnetoresistivi sono caratterizzati da scarsa linearità e forte dipendenza dalla temperatura. Ben si adattano all'inserimento diretto nei circuiti integrati, date le piccole dimensioni e il basso costo.

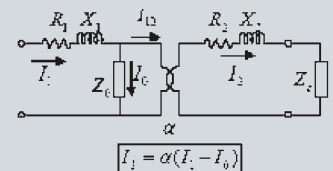
Il trasformatore di corrente (TA) trasforma una corrente elevata in una corrente più bassa, secondo un rapporto di trasformazione idealmente costante: $I_2 = K I_1$. Si caratterizza per precisione ed economicità. Inoltre garantisce l'isolamento galvanico, contrariamente allo shunt resistivo.

Se i trasformatori fossero componenti ideali, non avrebbe senso distinguere tra trasformatori di corrente e trasformatori di tensione: il trasformatore ideale trasferisce la potenza dal primario al secondario in rapporto 1:1, quindi un trasformatore di tensione con rapporto di trasformazione K potrebbe essere considerato come un trasformatore di corrente con rapporto di trasformazione $1/K$. Tuttavia i componenti reali sono caratterizzati da elementi parassiti (resistenze, perdite nel materiale magnetico) che portano alla definizione dello schema equivalente di un trasforma-

tore reale. Per comprendere gli effetti delle componenti parassite e le ragioni che portano alla distinzione fra trasformatori di corrente TA e di tensione TV (che si traducono in differenze costruttive), soffermiamoci su R_1 ed R_2 . In un TV l'effetto della R_1 può essere gestito



agendo sul numero delle spire mentre R_2 deve mantenersi più bassa possibile, in un TA è R_1 a dover essere bassa mentre il valore di R_2 non è critico. In sintesi, si può affermare che per misurare una corrente col metodo del trasformatore e volendo ad esempio ottenere una corrente secondaria

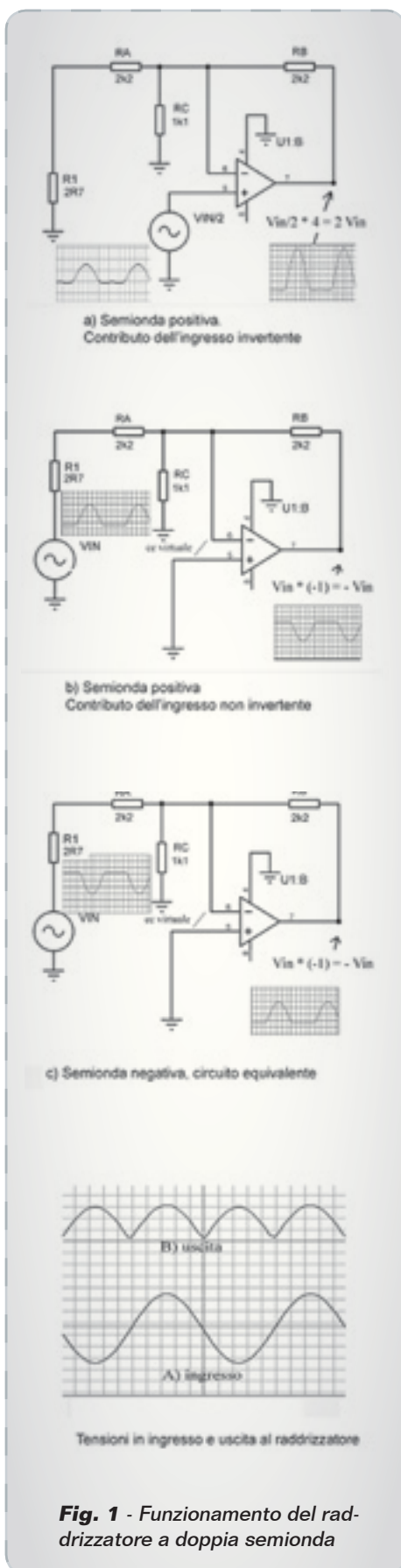


100 volte più piccola della primaria, non è utilizzabile un trasformatore di tensione in rapporto 100:1 montato "al contrario", col secondario a fare da primario: si deve impiegare un trasformatore pensato e progettato per realizzare la "trasformazione" della corrente.

meter per rete 220V"). Tuttavia considerando l'entità massima dello sfasamento corrente/tensione ammissibile in un impianto domestico, l'errore che si commette associando l'entità dei consumi alla corrente che fluisce sulla linea può considerarsi trascurabile. All'uscita del blocco di misura è presente una tensione variabile tra 0 e 5 Volt, linearmente proporzionale alla

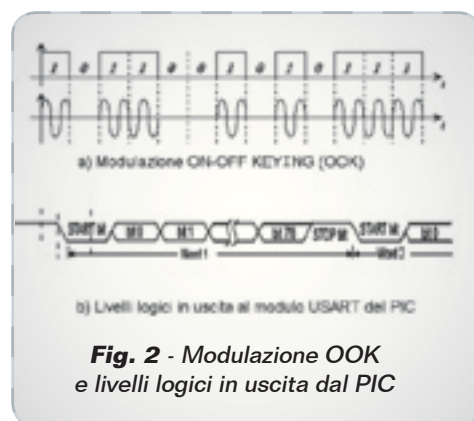
corrente che fluisce lungo la linea. In fase di taratura, come sarà descritto nel relativo paragrafo, si farà in modo che in uscita al modulo sia presente una tensione di 4,5 volt quando la corrente assorbita è pari alla corrente corrispondente alla massima potenza contrattuale. Ad esempio, per potenza contrattuale pari a 3 kW, il modulo sarà tarato per fornire 4,5 volt in corrispondenza

di una corrente sulla linea pari a $3000 \text{ W} / 230 \text{ V} = 13 \text{ A}$. Per una potenza contrattuale di 4,5 kW si effettuerà la taratura per ottenere 4,5 volt in corrispondenza di $4500 \text{ W} / 230 \text{ V} = 19,56 \text{ A}$ e così via. Per una potenza contrattuale di 3 kW e con la taratura indicata, la tensione in uscita al modulo è ricavabile dalla formula $V = 4,5 / 13 \cdot I_L$, dove I_L è la corrente che fluisce lungo la linea. Con

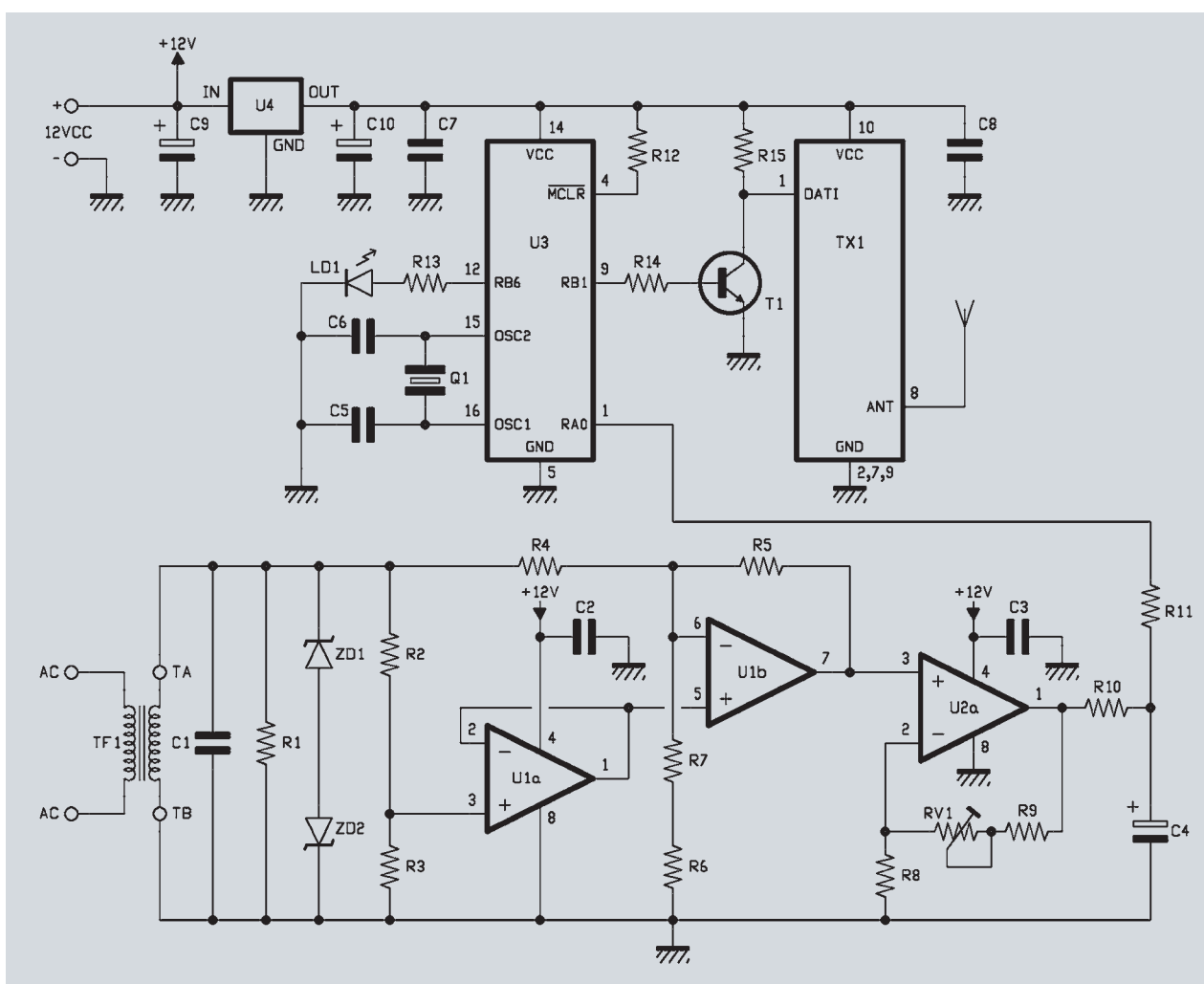


questa scelta, il modulo fornirà la massima tensione (5 V) se si impegnerà una potenza maggiore di circa il 10% rispetto alla potenza contrattuale ed il modulo ricevente sarà così in grado di gestire la segnalazione di sovraccarico. Il modulo successivo, basato sul PIC 18F1220 converte in digitale la tensione in uscita al modulo di misura. Il dato in formato binario è costituito da 1 byte (8 bit) che viene inviato al piedino di modulazione del successivo trasmettitore RF. La tensione in uscita dal modulo di conversione della corrente viene campionata ogni due secondi: in questo modo si introduce un ulteriore fattore di filtraggio di eventuali picchi transitori e si introduce una costante di tempo artificiale nel pilotaggio del display, per evitare variazioni di colore troppo repentine che potrebbero produrre un indesiderato effetto discoteca. Il trasmettitore è costituito da un modulo ibrido a RF a 433 MHz. Per ciascun campionamento della tensione di ingresso al convertitore A/D si inviano al trasmettitore tre byte consecutivi: il primo ed il secondo rappresentano un codice fisso (costituito perciò da complessivi 16 bit che consentono 65536 combinazioni), il secondo è rappresentativo della tensione misurata. All'interno del ricevitore si realizza un meccanismo di lettura che provvede ad interpretare il dato soltanto se la chiave trasmessa è quella attesa. Questo accorgimento consente di accoppiare ricevitore e trasmettitore, evitando interferenze provenienti da altri eventuali trasmettitori: il vicino di casa potrebbe disporre di un apparato analogo e in mancanza di un meccanismo di codifica, potremmo rilevare i suoi consumi anziché i nostri..... Inoltre questa tecnica consente di aumentare l'immunità ai disturbi

dell'intero sistema. Si consideri infatti che il modulo ricevente lavora su una frequenza utilizzata da un gran numero di apparati elettronici: si pensi ad esempio ai radiocomandi per apricancello e antifurto. I segnali emessi da questi apparati potrebbero essere interpretati come informazione emessa dal TX del dissuasore e conseguentemente potrebbero falsare l'indicazione del display sul modulo ricevente. Considerato che gran parte degli apparati interferenti lavorano per pochi



secondi (radiocomandi), con il sistema di codifica introdotto si realizza una sorta di "finestra temporale" nella quale il ricevitore è abilitato alla trascodifica, finestra che viene aperta in presenza di un preciso valore del byte ricevuto (mentre il segnale interferente è per sua natura variabile ed è bassa la probabilità che assuma ripetutamente lo stesso valore del codice trasmesso dal dissuasore). All'uscita del modulo RX RF del ricevitore sono presenti i byte inviati da trasmettitore: sequenze di triplette "codice-codice-dato". E' compito del decoder provvedere al riconoscimento del codice e al pilotaggio del display, in modo da determinare il cambiamento di colore dei led. Infine, il decodificatore comanda



l'emissione di un segnale acustico al superamento della potenza contrattuale; in caso di ripetute assenze di codice esatto (che fanno ipotizzare l'assenza di campo o l'accoppiamento con un TX dal codice differente) il decoder fa assumere al display il colore azzurro.

LO SCHEMA ELETTRICO DEL BLOCCO TRASMETTITORE MODULO DI MISURA DELLA CORRENTE

I trasduttori di corrente si dividono in quattro categorie:

- shunt
- sonda ad effetto hall
- sonda magnetoresistiva
- trasformatore di corrente

Rinviando al paragrafo di approfondimento per le conside-

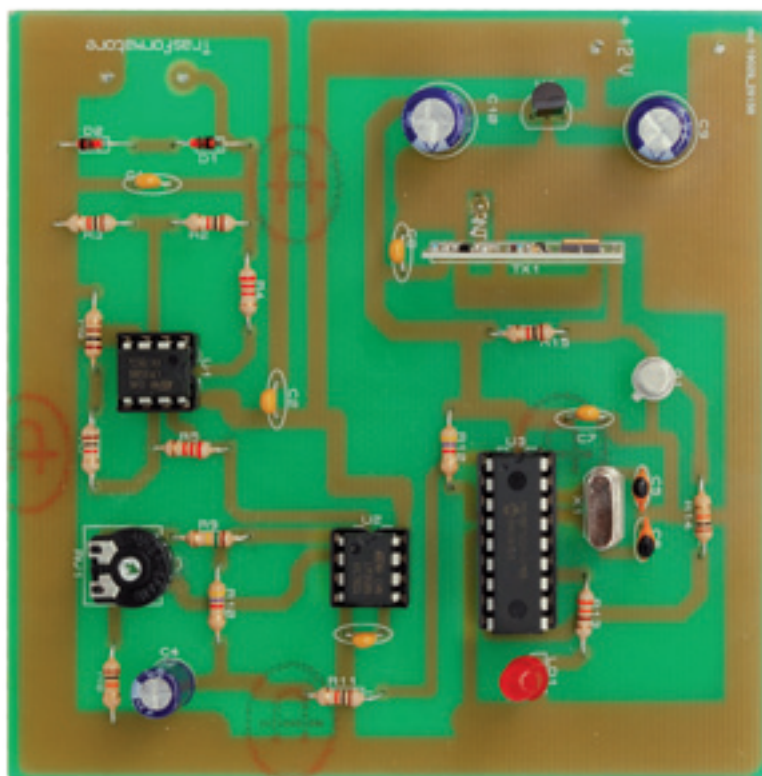
razioni teoriche e l'analisi delle differenti soluzioni, nel circuito (vedi schema elettrico) si è scelto di utilizzare un trasformatore di corrente in rapporto 1:100, prodotto dalla Zettler Magnetics (ACST-256-1), che sopporta una corrente primaria massima di 30 A. Il trasformatore di corrente deve essere chiuso su un carico più basso possibile, in teoria su un carico nullo. Il costruttore suggerisce di utilizzare un resistore di 60 ohm e in queste condizioni dichiara che relazione tensione di uscita / corrente di ingresso vale 64mV / A. Si osservi che in teoria un trasformatore di corrente con rapporto 1/100, chiuso su un resistore di 60 ohm dovrebbe fornire una tensione di 600mV/A, quasi dieci volte maggiore della tensione dichiarata dal costrut-

tore del TA. Questa differenza è essenzialmente dovuta al fatto che il trasformatore è chiuso su un carico non nullo, oltre che (in misura minore) alle inevitabili perdite nel materiale magnetico che costituisce il nucleo.

Durante le sperimentazioni sul modulo di misura si è osservato che col carico suggerito (60 ohm) il comportamento del TA è non lineare a partire da correnti di circa 4 A. Il nucleo va verso la saturazione, la forma d'onda sul secondario si distorce e non segue (in ampiezza) l'ampiezza della corrente al primario. Sperimentalmente si è osservato che caricando il secondario con un resistore di valore molto più basso (si ricordi che il TA vuole, in teoria, un carico nullo), si è ottenuto un significativo incre-

Elenco Componenti:

R1: 2,7 ohm	C8: 100 nF
R2: 1 kohm	multistrato
R3: 1 kohm	C9: 220 μ F 25 VL
R4: 2,2 kohm	elettrolitico
R5: 2,2 kohm	C10: 220 μ F 16 VL
R6: 100 ohm	elettrolitico
R7: 1 kohm	U1: LM358
R8: 10 kohm	U2: LM358
R9: 150 kohm	U3: PIC18F1220-I/P
R10: 4,7 kohm	(MF888TX)
R11: 1 kohm	U4: 78L05
R12: 4,7 kohm	T1: 2N2222
R13: 330 ohm	ZD1: 1N4370A
R14: 10 kohm	ZD2: 1N4370A
R15: 1 kohm	LD1: Led 3 mm rosso
RV1:	Q1: Quarzo 4 MHz
Trimmer 47 kohm MO	TX1: TX-SAW-MID-5V
C1: 100 nF multistrato	TF1: Trasformatore
C2: 100 nF multistrato	ACST-256-1
C3: 100 nF multistrato	(RS: 2509299883)
C4: 47 μ F 50 VL	
elettrolitico	Varie:
C5: 22 pF ceramico	- Zoccolo 4+4 (2 pz.)
C6: 22 pF ceramico	- Zoccolo 9+9
C7: 100 nF multistrato	- Circuito stampato



mento della linearità.

Si è scelto un resistore di 2,7 ohm; la tensione ai capi di R1 è di pochi mV/A, la successiva amplificazione porta il livello a valori adeguati per la successiva elaborazione.

Si è sperimentalmente rilevato che l'errore di misura è inferiore al 2%, nella gamma 0-3,1 kW, assolutamente trascurabile per un indicazione "qualitativa" della potenza assorbita.

I diodi zener D1 e D2 (2,4 Volt) provvedono a tagliare i picchi di sovratensione che si manifestano al secondario del trasformatore in corrispondenza di rapide variazioni di corrente al primario, dovute all'inserzione o al distacco di carichi di potenza significativa.

Si ricordi che la corrente al secondario è proporzionale alla velocità di variazione del flusso magnetico, a sua volta proporzionale alla velocità di variazione di

corrente al primario.

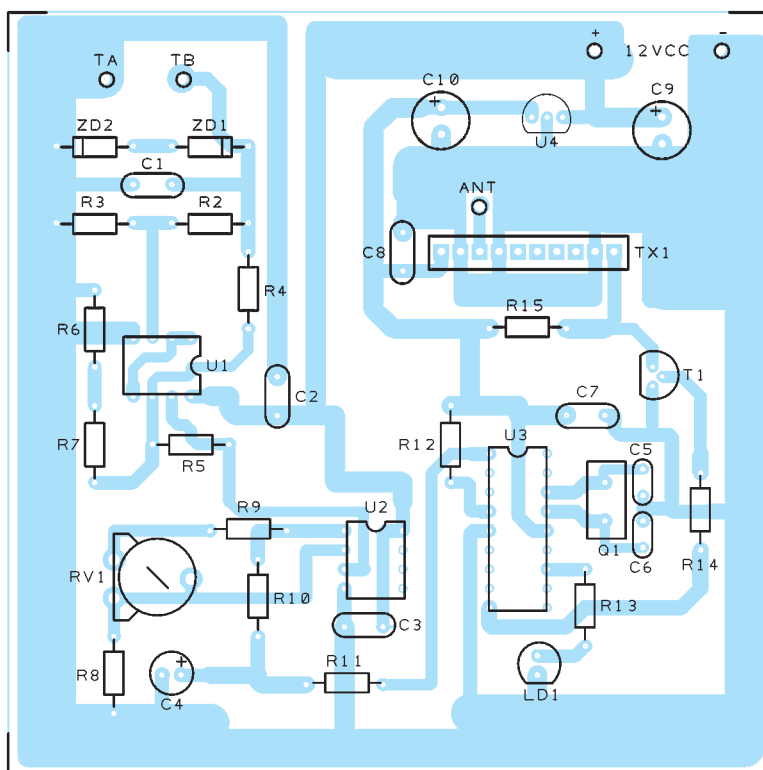
Lo stadio raddrizzatore è costituito dai due moduli contenuti nell'operazionale U1 (LM358). L'idea di utilizzare un diodo (raddrizzatore a mezza semionda) o un ponte di diodi (raddrizzatore a semionda intera) deve essere scartata a causa della caduta di tensione sui diodi: circa 0,6 – 0,7 volt per un singolo diodo e 1,2 – 1,4 volt per i due diodi di un ponte ad onda intera. Considerato che la tensione ai capi di R1 sarà inferiore al volt..., l'errore introdotto dal raddrizzatore a diodi è inaccettabile! Vediamo allora come funziona il raddrizzatore a doppia semionda realizzato con i due operazionali, alimentati con una tensione singola di 12 V.

Quando su R1 è presente la semionda positiva di ampiezza V_{in} , all'ingresso di U1A si misura una tensione pari a $V_{in}/2$ dovuta al partitore R2-R3 e all'uscita di

U1A, in configurazione a guadagno unitario, si ritrova $V_{in}/2$. Il secondo stadio può essere considerato un amplificatore differenziale, che analizziamo con il metodo della sovrapposizione degli effetti.

Per calcolare il contributo dovuto alla tensione sull'ingresso non invertente sostituiamo il generatore costituito dal TA e da R1 con la sua impedenza, pari a R1. Poiché R1 è molto minore di R4, il circuito equivalente risulta quello rappresentato in figura 1a: un amplificatore non invertente di guadagno pari a $1 + R_B/(R_A // R_C)$. Con i valori assegnati, il guadagno risulta pari a $1 + 2:2/3 = 4$.

Per calcolare il contributo dovuto alla tensione sull'ingresso invertente, sostituiamo il generatore costituito dal primo operazionale con la sua impedenza d'uscita, teoricamente nulla, ottenendo il circuito di figura 1b, che rappre-

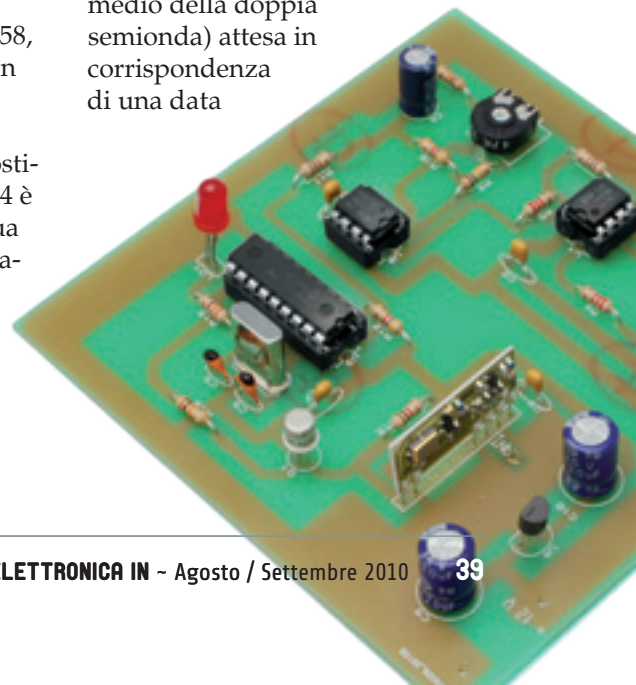


senta un amplificatore invertente a guadagno unitario al cui ingresso è applicata la tensione V_{in} . Si osservi che il corto circuito virtuale tra gli ingressi dell'operazionale rende non significativa la presenza del resistore R_C , che risulta con entrambi i morsetti collegati allo stesso potenziale e perciò non è attraversato da corrente. Pertanto, sommando i contributi dovuti ai due ingressi dell'amplificatore U1B, in corrispondenza della semionda positiva su R_1 , all'uscita si troverà una tensione pari a $V_{in}/2 * 4 - V_{in} = V_{in}$. Quando ai capi di R_1 è presente la semionda negativa, l'uscita del primo operazionale va al potenziale di riferimento (zero) e perciò il secondo operazionale risulta montato in configurazione invertente con guadagno pari a $R_B/R_A = 1$ (figura 1c). Per il contributo di R_C vale la considerazione precedente. Per-

ciò all'uscita di U1B ritroveremo la semionda negativa, invertita di fase. Complessivamente, considerando i contributi della semionda positiva e negativa, si ottengono nei punti A e B dello schema elettrico le forme d'onda riportate in figura 1d. Il segnale presente all'uscita del raddrizzatore viene inviato all'amplificatore costituito da una sezione del secondo LM358, montato in configurazione non invertente. All'uscita dell'amplificatore è disposto il gruppo filtrante costituito da R_{10} e C_4 ; ai capi di C_4 è presente una tensione continua che risulta quindi proporzionale alla corrente che scorre lungo la linea di distribu-

zione dell'energia elettrica. Per quanto precedentemente affermato, la tensione in uscita al raddrizzatore deve assumere, all'uscita dell'amplificatore, un valore variabile tra 0 e 5 Volt. Si osservi che il complesso raddrizzatore (U1a-U1b) può essere considerato un amplificatore a guadagno unitario e quindi fortemente retroazionato e conseguentemente caratterizzato da un'ampia banda passante. L'inserimento del condensatore C_1 riduce la banda passante complessiva del sistema, a vantaggio della stabilità e della immunità ai picchi transitori di corrente presenti sulla linea. Nei circuiti di misura si utilizzano solitamente componenti di alta precisione: nel nostro raddrizzatore, la simmetria della forma d'onda in uscita è assicurata se $R_A = R_B = 2R_C$ e $R_2 = R_3$. Nel caso in esame, utilizzando resistori al 5% di precisione, l'asimmetria nel caso peggiore (R_{Amin} , R_{Bmax} , R_{Cmin} , R_{2min} , R_{3max}) sarà del 6%, valore più che accettabile per una misura di tipo qualitativo. Inoltre la successiva amplificazione (a guadagno regolabile) e filtraggio permettono di compensare l'errore e recuperare la componente continua (valor medio della doppia semionda) attesa in corrispondenza di una data

Un particolare della basetta con il PIC ed il modulo RF dell'Aurel.





E in Francia stanno arrivando 35 milioni di contatori intelligenti "Linky"

Anche in Francia sta arrivando il contatore intelligente: si chiama Linky e dovrebbe aiutare i consumatori a risparmiare energia suggerendo l'orario più conveniente per mettere in funzione i carichi più energivori (lavatrici, boiler, ecc.) ma anche, in futuro, il momento più conveniente per ricaricare le batterie della propria vettura elettrica. Per il momento il nuovo contatore è in fase di test presso alcune migliaia di abitazioni nella regione di Tours e di Lione ma entro il 2016 dovrebbe essere installato in oltre 30 milioni di abitazioni. Secondo ErDF (l'ENEL francese) Linky dovrebbe consentire un risparmio dei consumi di elettricità compreso tra il 10 ed il 15%. La struttura tecnologica prevede un concentratore per ogni isolato con comunicazione bidirezionale tramite onde convogliate; a seguire troviamo dei concentratori di quartiere che comunicano con le centrali ErDF tramite rete wireless GSM/GPRS; gli utenti potranno conoscere i propri consumi in tempo reale tramite il display di Linky ma anche sul sito internet del gestore (<http://linky.erdfdistribution.fr>).

corrente in ingresso. Si può affermare che, una volta tarato il modulo di misura, quella che conta non è tanto la precisione assoluta dei componenti quanto la loro stabilità in funzione della temperatura.

Tuttavia per rispetto della teoria del circuito, pur potendo approssimare la RC con un resistore da 1000 o da 1200 ohm, si è scelto di utilizzare 2 resistori da 1000 e da 100 ohm, più facilmente reperibili (e più economici) rispetto ad un resistore da 1100 ohm.

Per garantire la linearità dell'uscita, il modulo deve essere alimentato con una tensione continua compresa tra 9 e 12 volt.

CONVERTITORE A/D, TRASMETTITORE RF E ALIMENTAZIONE

Il segnale in uscita dal modulo di misura della corrente viene inviato al convertitore A/D interno al PIC 18F1220, che dispone inoltre di un modulo USART per la gestione della comunicazione dei dati, vedi schema elettrico. La tensione in ingresso, variabile tra 0 e 5 Volt, viene convertita in binario ad 8 bit e pertanto in uscita al convertitore sarà disponibile un byte di contenuto variabile tra "00000000" e "11111111". All'interno del PIC, per ogni campionamento della tensione fornita dal modulo di misura (un campione ogni due secondi) si

costruisce la sequenza di byte da inviare al modulo ricevente. Per quanto detto nell'introduzione, la sequenza è costituita da due byte che identificano in modo univoco il trasmettitore e da un terzo byte che rappresenta la tensione misurata. Il codice del trasmettitore (i primi due byte) si definisce in fase di programmazione del PIC. Considerato che l'acquisizione del dato nel ricevitore avviene in tempi molto inferiori ai due secondi, non è necessario riempire l'intero intervallo di campionamento con l'invio della tripletta "codice-codice-tensione" ma è sufficiente inviare un numero ridotto di triplette. Facciamo qualche considerazione per calcolare il numero di triplette da inviare, ogni due secondi. La velocità di trasmissione è stata scelta pari a 2400 baud; non sono necessarie velocità più alte, un ridotto baud rate permette di poter utilizzare moduli di trasmissione a RF con limitata banda passante, così da mantenere bassi i costi. A 2400 baud, i 9 bit rappresentativi dei codici e del dato hanno una durata di $9/2400 = 3.75$ ms. E' necessario inviare 9 bit perché i primi 8 rappresentano il codice e/o il dato, il nono bit viene utilizzato per realizzare il meccanismo di "riconoscimento della chiave", sfruttando le potenzialità del modulo USART interno ai PIC (per ulteriori dettagli si rinvia al data sheet del PIC18F1220 e alla lettura del programma, listato 1). Distanziando di 2 ms i tre "byte estesi" per facilitare il riconoscimento del bit di start, si ricava che la durata di ciascuna tripletta è pari a $3.75 \times 3 + 6 = 17.25$ ms. Il successivo trasmettitore RF è costituito da un circuito ibrido prodotto dalla Aurel, il modello TX SAW MID 5V, capace di erogare circa 10 mW con 5 volt di alimentazione, alla frequenza di

Listato 1 - CICLO DEL PROCESSORE 18F1220

```
MAIN:
'conversione a/d
ADON = 1 'accendi convertitore a/d
led=1
DelayMS 100 'attesa stabilizzazione convertitore
GO_DONE=1 'avvia la conversione
in_conversione:
  If GO_DONE = 1 Then
    GoTo in_conversione 'conversione a/d in corso
  EndIf
  'fine conversione
ADON = 0 'spegni convertitore A/D
led=0
'2400 baud, ogni 9 bit 3.75 msec, x 3 = 11.25 msec., 6 msec ritardo.
'un ciclo di invio dura 17.25 msec. Volendo mantenere il tx in 'on' per 0.1 sec al secondo, considerando
' che si invia un burst di trasmissione ogni due secondi, si possono inviare, in un burst, 0.2/0.01725
' = 11 blocchi di dati
For i= 1 To 11
'trasmetti codice segnalazione invio dato
'verifica che lo shift register sia vuoto (trasmissione effettuata) prima di trasmettere
  trasmetti_codice1:
  If TRMT = 0 Then
    GoTo trasmetti_codice1 'shift register pieno, attendere svuotamento
  EndIf
  TX9D=1 'trasmissione indirizzo
  TXREG= %10101010 'chiave 1
  DelayMS 2 'ritarda l'inizio del blocco dati, per risincronizzare lo start bit
  trasmetti_codice2:
  If TRMT = 0 Then
    GoTo trasmetti_codice2 'shift register pieno, attendere svuotamento
  EndIf
  TX9D=1 'trasmissione indirizzo
  TXREG= %10101010 'chiave 2
  DelayMS 2 'ritarda l'inizio del blocco dati, per risincronizzare lo start bit
'trasmetti dato convertito
  trasmetti_dato:
  If TRMT = 0 Then
    GoTo trasmetti_dato 'shift register pieno, attendere svuotamento
  EndIf
  DelayMS 2
  TX9D=0 'trasmissione dato
  TXREG = ADRESH
Next i
DelayMS 1800 'effettua una misura di corrente circa ogni 2 secondi
GoTo MAIN
```

433.92 MHz. La raccomandazione 70-03 emessa dall'European Conference of Postal and telecommunications administration (CEPT) indica che la banda di frequenza 433.05 – 434.79 MHz è destinata alla trasmissione dati (no audio e voce), la massima potenza irradiabile è 10dbm e il duty cycle di trasmissione deve essere inferiore al 10%. Quindi per ottemperare alla CEPT 70-30 il nostro trasmettitore dovrà lavorare per non più di 0,2 secondi per ciascuno dei due secondi di intervallo tra un campionamento e il successivo. Di conseguenza, volendo occupare interamente questi 0.2 secondi, possiamo inviare $0.2 / 0.01725 = 11$ triplette conse-

cutive, ogni due secondi. Tuttavia non abbiamo ancora individuato il meccanismo che "spegne" il trasmettitore quando non ci sono triplette in uscita dal PIC. Ci viene in aiuto la tecnica di modulazione adottata dal modulo Aurel, che è una modulazione in ampiezza di tipo "OOK" (on-off-keying). E' una tecnica di modulazione adatta alla trasmissione di dati digitali che assumono solo due valori: in sostanza la portante RF viene trasmessa in corrispondenza del livello logico alto mentre non c'è trasmissione in corrispondenza del livello logico basso, figura 2a. La modulazione OOK è tuttavia caratterizzata da un alto tasso di errore in trasmissione, tanto mag-

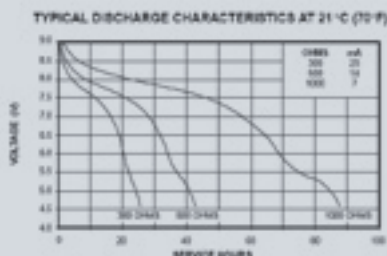
giore quanto più basso è il rapporto S/N. Pertanto per l'affidabilità della trasmissione è importante trasmettere un segnale quanto più forte possibile (nei limiti dei 10dbm imposti) ed utilizzare un ricevitore ad alta sensibilità. Quando analizzeremo il circuito del ricevitore, vedremo un accorgimento software che permette di raggiungere tassi di errore estremamente ridotti. Resta un'ultima considerazione da fare, per poter dire di aver ottemperato alla CEPT 70-03. L'uscita della porta USART del PIC è riportata in figura 2b: il livello logico "base" è il livello 1, la caduta al livello 0 individua il bit di start (e i successivi bit) fino al bit di stop del dato inviato,

Alimentazione a batteria

Il PIC necessita di 5 volt, il rad-drizzatore/amplificatore necessita di almeno 9 Volt per garantire la linearità dell'uscita fino ai valori massimi di escursione, il trasmettitore può essere alimentato con una tensione variabile tra 3 e 12 Volt. Perciò ci si è soffermati sull'ipotesi di impiego di una pila alcalina da 9V; la figura sottostante riporta la curva di scarica tipica di una Duracell da 9 V.

Il componente che consuma la corrente maggiore è il trasmettitore RF (circa 6 mA, da datasheet del costruttore) e complessivamente, sommando la corrente assorbita da U1, U2 e U3 si può approssimare il consumo intorno ai 10 mA, corrispondenti ad un carico di circa 1000 ohm. In queste condizioni, la batteria fornisce una tensione di 7.5 volt (valore limite per garantire la linearità dell'amplificatore) dopo circa 50 ore di funzionamento continuo. In effetti sarebbe possibile cambiare il modulo trasmettitore RF, impiegando un modello che dispone di piedino di abilitazione. Considerando che la tensione fornita dal modulo di misura della corrente viene campionata ogni 2 secondi, si potrebbe alimentare il TX solo per brevi periodi nell'intorno dell'istan-

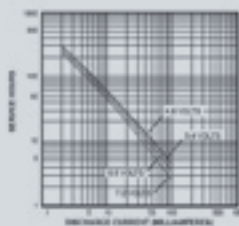
te di campionamento. Il tempo di stabilizzazione del trasmettitore, da datasheet, è di circa 1 ms. La trasmissione del burst di trasmissione, per quanto detto nell'analisi del trasmettitore, ha un duty cycle



di 0,1. Perciò, approssimando per eccesso (ipotizzando che tutto il circuito segua il duty cycle 0.1 e non solo il TX) si può ritenere che l'utilizzo effettivo della batteria

è pari a 1/10 rispetto all'utilizzo continuativo (trasmettitore sempre acceso). Le 50 ore di funzionamento per raggiungere la tensione minima di 7.5 volt diventano perciò $50 \times 10 = 500$ ore, corrispondenti a $500/24 = 20$ giorni: troppo frequente il cambio di batteria.

La situazione migliorerebbe aumentando il baud rate, che però è fissato dalle caratteristiche del sistema ricetrasmittente a RF. In conclusione, l'alimentazione a batteria risulta poco economica.



dopodichè l'uscita torna al livello 1. Quindi il segnale in uscita dal PIC è al livello 1 per gran parte del tempo: con queste caratteristiche di segnale modulante, il duty cycle del trasmettitore è molto prossimo al 100%! E' necessario perciò invertire la fase del segnale modulante e a questo provvede il transistor Q1. In ricezione sarà necessario provvedere alla re-inversione di fase del segnale, per ricostruire correttamente i burst trasmessi dal TX. Si osservi che il duty cycle che si ottiene, adottando la modulazione OOK e inviando 11 triplette da circa 18 millisecondi sarà comunque inferiore a 0.1 in quanto anche nei 18 millisecondi

di presenza dei dati, ci saranno istanti in cui il trasmettitore sarà spento (quando si trasmettono gli "1"). Il firmware che dovrà essere caricato sul PIC è estremamente semplice: tralasciando la parte di inizializzazione dei registri e delle variabili, nel riquadro di listato 1 si riporta il ciclo elaborativo del processore. Il led collegato alla porta RB6 si accende per il tempo in cui è acceso il convertitore A/D, segnalando visivamente gli istanti di acquisizione del dato.

Il circuito può essere alimentato con una tensione continua variabile tra 9 e 12 Volt e considerato l'assorbimento di corrente molto basso (poco più di 10 mA, può

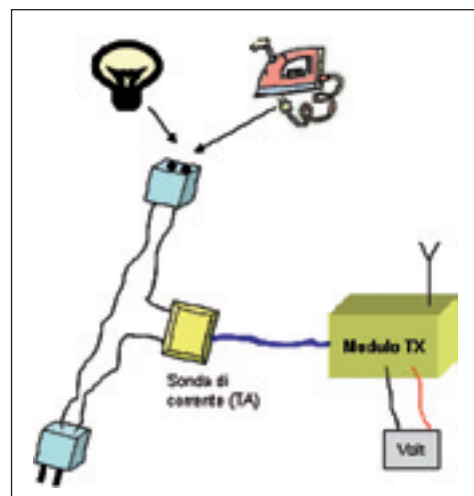
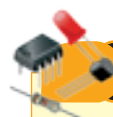


Fig. 7 - Taratura del misuratore di corrente

andar bene uno dei tanti piccoli alimentatori che tipicamente abbondano nelle nostre abitazioni (telefonini, piccoli apparati e giochi elettronici). La tensione fornita dall'alimentatore è applicata direttamente a U1-U2. L'alimentazione di U3 e del trasmettitore si ottiene tramite un regolatore fisso a bassa potenza 78L05. In fase di progetto si è valutata l'opportunità di alimentare il modulo trasmettitore con una batteria; si veda il riquadro di approfondimento.

Il prossimo mese presenteremo la seconda parte del progetto, ovvero la sezione ricevente con i LED.



per il MATERIALE

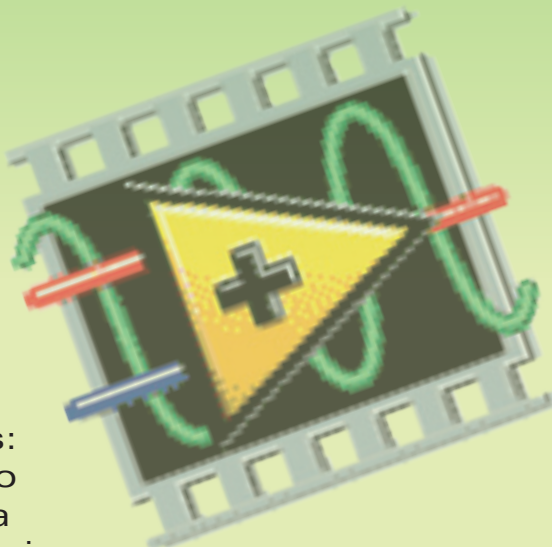
Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista così come il firmware utilizzato per programmare il PIC. Il trasmettitore a radiofrequenza TX SAW MID-5 V costa 6,00 Euro (IVA compresa) e può essere acquistato presso Futura Elettronica.

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

Conoscere e usare LabVIEW



Completiamo il programma di esempio iniziato il mese scorso, riguardante l'acquisizione di segnali mediante la scheda USB-6008 di National Instruments: stavolta aggiungiamo un Tab Control per la gestione degli ingressi analogici e delle uscite digitali. Sesta Puntata.



di
FRANCESCO
FICILI

LabVIEW

Nella precedente puntata ci siamo lasciati dopo aver realizzato insieme un primo esempio di software di acquisizione dati in LabVIEW facente uso di una scheda NI USB-6008. In questa sesta puntata completeremo la scrittura di tale software, in modo da avere una visione globale sulla stesura di un programma di acquisizione dati completo, che sfrutti appieno le potenzialità offerte dall'hardware NI. La NI USB-6008 è uno dei prodotti entry-level; indubbiamente National Instruments propone prodotti e dispositivi molto più performanti, tuttavia i concetti applicati alla realizzazione del nostro

software applicativo possono essere facilmente estesi anche ad altre schede, sfruttando la portabilità del DAQ-Assistant. Quindi, con poche modifiche, sarà possibile adattare il programma scritto per la USB-6008 ad unità DAQ più potenti e operanti su bus differenti dall'USB. Dunque, mettiamoci al lavoro riprendendo da dove eravamo rimasti: riapriamo il programma esempio *6.vi* sul quale avevamo iniziato a lavorare nella scorsa puntata. Il front panel del VI è riportato nella **Fig. 1**. Ad esso vogliamo effettuare alcune modifiche, in modo da rendere più funzionale il programma; ma prima di procedere salviamo nuovamen-





Fig. 1 - Pannello frontale del programma esempio 6.

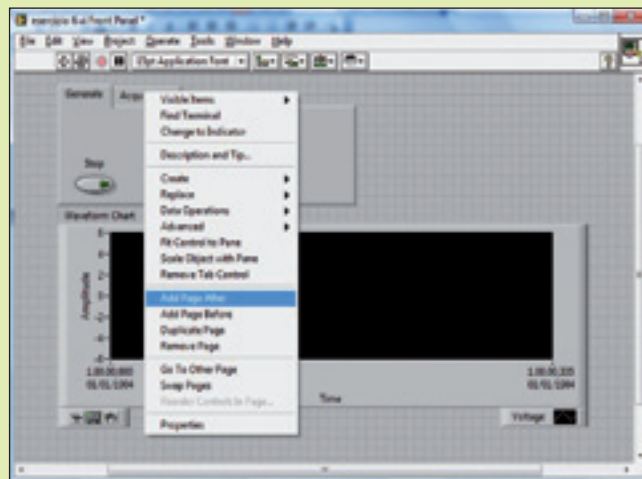


Fig. 2 - Inserimento del Tab Control.

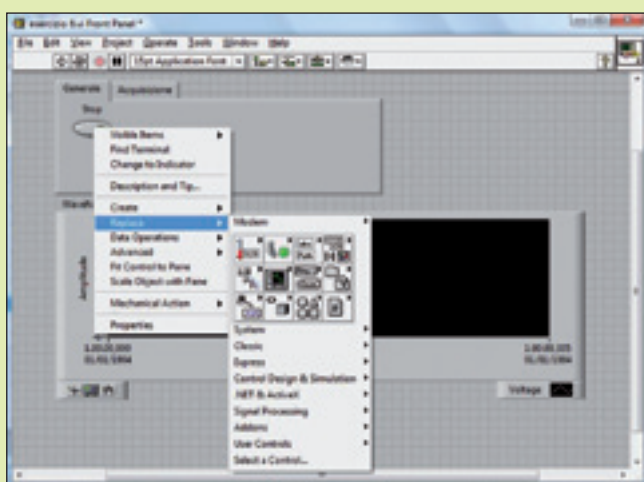


Fig. 3 - Replace del comando stop.

te il VI e cambiamone il nome in *esempio 7*. Partiamo naturalmente dal front panel, per poi passare al foglio di lavoro, come di consueto. Per prima cosa inseriamo nella zona relativa ai comandi un Tab Control (*Containers/Tab Control*) che sarà destinato a contenere i vari controlli, creando inoltre una separazione ideale tra le varie macrofunzioni. Il Tab Control permette di risparmiare spazio sul pannello frontale, consentendo di racchiudere pagine (Tab) all'interno della stessa area, in modo che ogni pagina sia accessibile premendo la relativa linguetta posta nella parte superiore del Tab. Rinominiamo le due label presenti per impostazione predefinita, rispettivamente come "Generale" e "Acquisizione", ed aggiungiamone una terza (basta fare clic con il tasto destro del mouse su una delle label e poi selezionare il comando *add page after*, come illustrato nella Fig. 2) nominandola come "I/O

Digitali". A questo punto modifichiamo il pulsante di stop e sostituiamo il pulsante dotato di LED con uno generico (Boolean/Stop Button). Per rimpiazzare un comando (o indicatore) esistente con uno di altro tipo, senza perdere tutte le proprietà associate a quel comando o indicatore (label, proprietà meccaniche, estremi, ecc.) è sufficiente cliccare con il tasto destro del mouse sopra l'oggetto da sostituire e selezionare l'opzione *Replace*, come illustrato nella Fig. 3. Ora si aprirà una nuova controlpalette che permetterà di selezionare il nuovo oggetto grafico da inserire sul pannello. Adesso inseriamo un altro pulsante generico (Boolean/OK Button) nominandolo "START".

L'avvio del programma verrà vincolato al tasto start tramite un ciclo while, in modo da creare un loop di attesa all'avvio.

A questo punto vogliamo creare altre due pagine sul Tab Control, che permettano di accedere ad una serie di funzioni della scheda; per l'esattezza, intendiamo creare una pagina per il controllo del canale di ingresso analogico ed una per il controllo delle uscite digitali. Sulla pagina relativa all'ingresso analogico posizioniamo due controlli numerici, nominandoli, rispettivamente, *Frequenza di campionamento* e *Numero di campioni per ciclo*. Questi controlli numerici permetteranno di controllare tali parametri attraverso il DAQ-Assistant. Sulla pagina di controllo delle linee digitali, invece, posizioniamo otto pulsanti generici ed altrettanti LED; nelle figure 4, 5 e 6 abbiamo riportato le pagine del Tab Control, in modo da illustrare la disposizione degli elementi grafici. Vediamo adesso come è stato modificato il codice relativo all'esempio in questione. Per prima cosa abbiamo inserito, collegato al tasto

START, un ciclo di attesa software che arresta l'esecuzione del programma fino alla pressione del tasto; il codice grafico che implementa quanto detto è visibile nella Fig. 7.

CONTROLLO DEGLI I/O DIGITALI

La scheda NI USB-6008 dispone di due porte di I/O digitali, denominate rispettivamente port0 e port1, la cui piedinatura è riportata nella Fig. 8. Tutte le porte, come i contatori e gli ingressi analogici, sono accessibili tramite dei blocchi con morsetti a vite forniti in dotazione con la scheda.

Il DAQ-Assistant, che abbiamo già utilizzato per la generazione del task di acquisizione, può essere utilizzato anche per creare il task deputato al controllo degli I/O digitali. Esattamente come era stato fatto per il task di acquisizione, inseriamo all'interno del foglio di lavoro un *express task* di tipo DAQ Assistant (*Express/ Input/DAQAssist*). Comparirà quindi il wizard che già conosciamo, visibile nella Fig. 9. Stavolta, anziché scegliere la prima opzione (*Acquire Signal*), optiamo per *Generate Signal*. Come si vede è possibile generare diversi tipi di segnali in uscita, sia analogici (sfruttando i DAC) che digitali (sfruttando le porte di uscita).

Scegliamo l'opzione *Digital Output* e all'interno di essa *Line Output*. L'opzione *line output* permette di controllare una o più linee di uscita di una data porta.

Una volta selezionata l'opzione, comparirà la schermata visibile nella Fig. 10. Come detto in precedenza, la NI USB-6008 dispone di due porte digitali, utilizzabili come porte GPIO (*General Purpose Input Output*), la port0 (8-bit wide) e la port1 (4-bit wide). Per il nostro

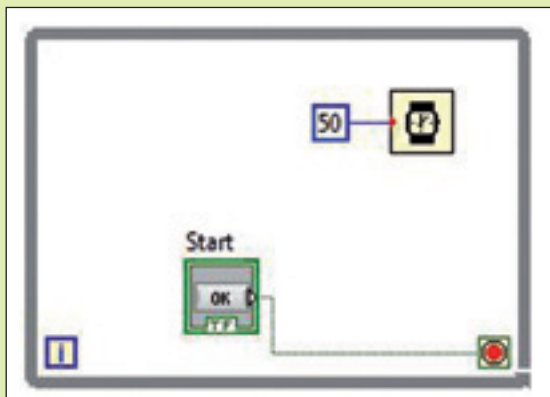


Fig. 7
Ciclo di attesa sul tasto Start.

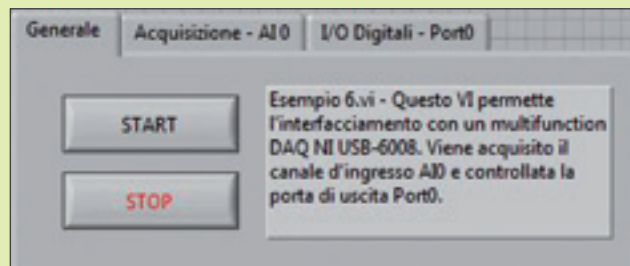


Fig. 4 - Tab Control Generale.

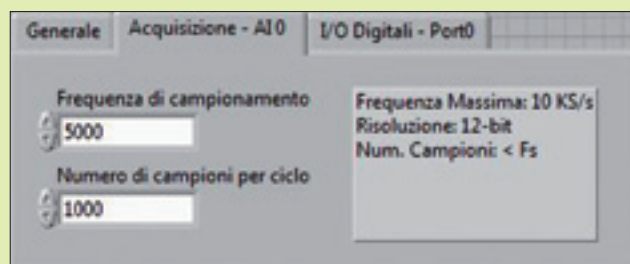


Fig. 5 - Tab Control Acquisizione.

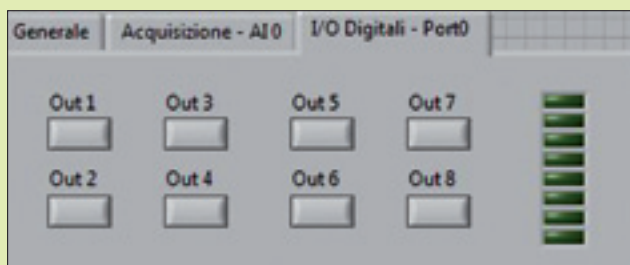


Fig. 6 - Tab Control I/O Digitali.

Fig. 8 - Pin-out della scheda NI USB-6008.



Terminal	Signal
17	P0.0
18	P0.1
19	P0.2
20	P0.3
21	P0.4
22	P0.5
23	P0.6
24	P0.7
25	P1.0
26	P1.1
27	P1.2
28	P1.3
29	PFI0
30	+2.5V
31	+5V
32	GND

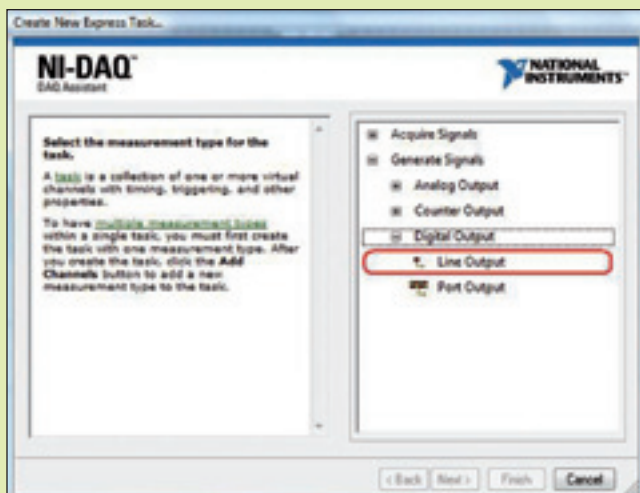


Fig. 9 - Nel wizard di configurazione del task scegliamo l'opzione Line Output, che si trova all'interno del percorso Generate Signal / Digital Output.

esempio utilizzeremo per intero la port0. Per selezionare tutti gli otto canali di quest'ultima, possiamo utilizzare il mouse ed il tasto shift per effettuare una selezione multipla, come visibile nella Fig. 11. Premendo il tasto finish apparirà la schermata illustrata nella Fig. 12. Dato che la generazione dei segnali di output avverrà on-demand, non dobbiamo cambiare nulla relativamente alle opzioni di time settings (*Generation mode: 1 sample (on demand)*). Possiamo verificare il funzionamento del task utilizzando l'apposito strumento di debug posto nella parte alta della schermata, come visibile nella Fig. 13.

Sono poi presenti alcune opzioni, come la possibilità di invertire le linee ed altro ancora. Per visualizzare l'output delle linee digitali potete servirvi di una bread board e di 8 LED, collegati alla scheda sfruttando i connettori a morsettiera forniti in dotazione. Una volta verificato che il task funziona è sufficiente fare clic sul tasto OK: il task verrà generato ed il VI sarà posizionato sul foglio di lavoro. Ora il task per la gestione degli I/O digitali è pronto; ciò che bisogna fare è inserirlo all'interno del ciclo principale, in modo che lavori insieme al task di acquisizione. Abbiamo due possibilità: i due task possono lavorare in parallelo (si tratta di un multitasking puro) oppure in sequenza. Le due situazioni sono schematizzate nella Fig. 14.

Nel programma di esempio si è scelta un'esecuzione di tipo sequenziale, ma solo per una questione di leggibilità del programma. Naturalmente il fatto di disporre in maniera sequenziale i due task penalizza leggermente la velocità di attuazione delle uscite digitali (in quanto bisogna attendere la fine del task di acquisizione). Per programmi con tempistiche non troppo stringenti, questa soluzione può anche andar bene, mentre in situazioni *time-critical* si dovrà necessariamente ricorrere ad un'esecuzione di tipo parallelo.

Vediamo a questo punto come è stato modificato il codice grafico del software per implementare quanto illustrato logicamente prima: l'ultima volta avevamo scritto il codice grafico appena sufficiente a far girare il task di acquisizione, che riportiamo, per comodità, in Fig. 15. Aggiungiamo a tale codice due strutture di tipo stacked sequence, la prima (che chiameremo *sequence 2*) immediatamente

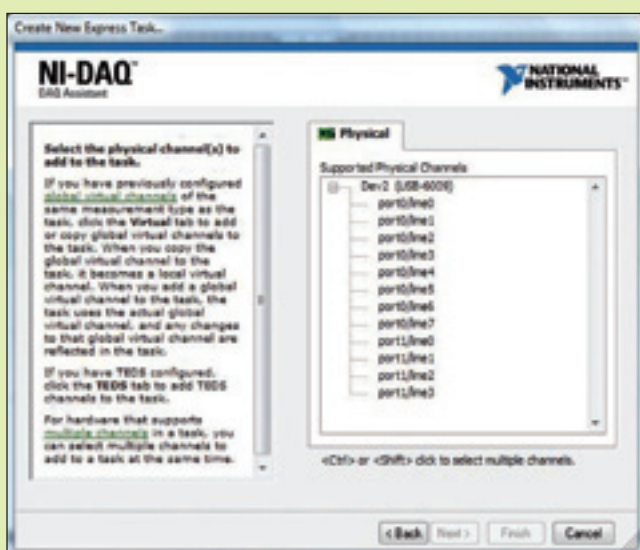


Fig. 10 - Elenco delle linee disponibili.

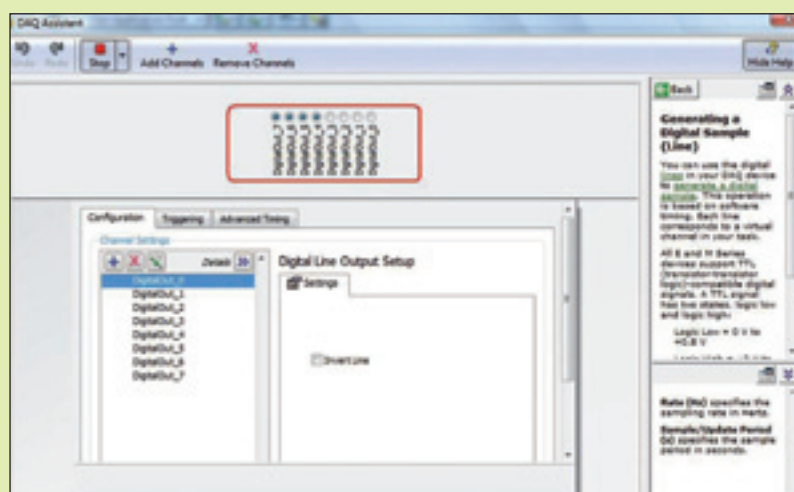
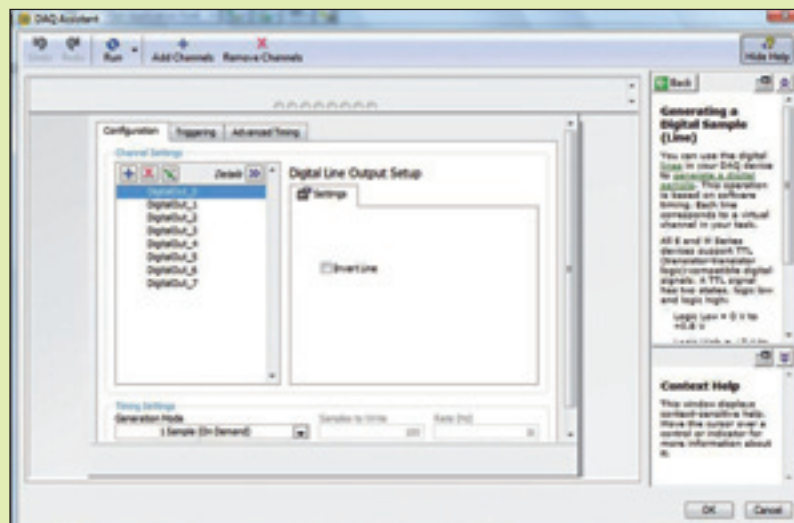


Fig. 11 - Selezione delle 8 linee di port0.

Fig. 12 - Settaggio dei parametri del task.

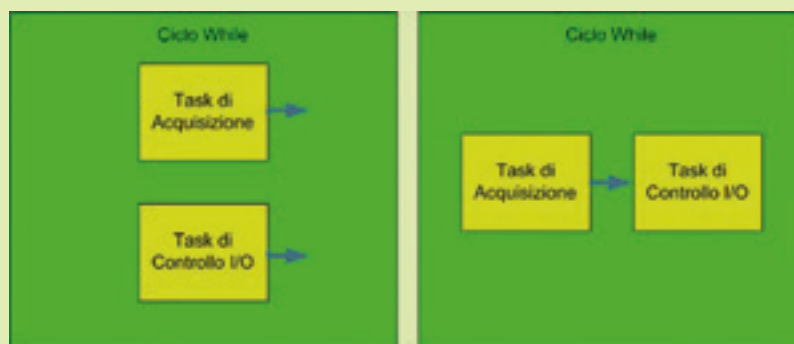
all'interno del while e l'altra (che chiameremo *sequence 1*) a parte. Entrambe le strutture devono avere due frame. Ora inseriamo nel primo frame della struttura esterna il codice visibile nella **Fig. 7**, che vincola l'esecuzione del programma alla pressione del tasto START sul pannello frontale. Nel secondo frame trasportiamo il vecchio codice, con la nuova struttura *sequence* all'interno. La struttura del programma sarà quindi la seguente: la struttura *sequence 1* contiene nel primo frame il ciclo while bloccante, vincolato al tasto start, e nel secondo frame ha il ciclo while contenente il resto del programma.

Passiamo ora al secondo frame della struttura *sequence 1*: il codice all'interno del ciclo while deve essere modificato come illustrato nella **Fig. 16**, dove si vede che il task di acquisizione è stato espanso (per farlo basta posizionarsi con il puntatore del mouse sul bordo inferiore del VI, cliccare e tenere premuto il tasto sinistro e trascinare il puntatore verso il basso) per mostrare tutti gli input e gli output del blocco. Colleghiamo i due controlli numerici inseriti precedentemente nel front panel (Numero di campioni per ciclo e Frequenza di campionamento) rispettivamente agli ingressi *number of sample* e *rate* del VI. Ciò ci consentirà di variare a piacimento, ad ogni avvio del programma, questi parametri. A riguardo, si tengano presenti i seguenti aspetti: il parametro *rate* (controllato dal controllo numerico frequenza di campionamento) per la 6008 non può essere superiore alla massima frequenza di campionamento (10 kS/s), ed il numero di campioni per ciclo non può essere mai superiore al valore

Fig. 14 - Esecuzione parallela o sequenziale dei due task.**Fig. 13** - Uso dello strumento di debug del task.

impostato del parametro *rate*. Il numero di campioni per ciclo incide esclusivamente sulla velocità con cui i dati vengono presentati sul grafico. Ad esempio, la seguente impostazione:

$$\text{Numero di campioni per ciclo} = \text{Frequenza di campionamento}$$



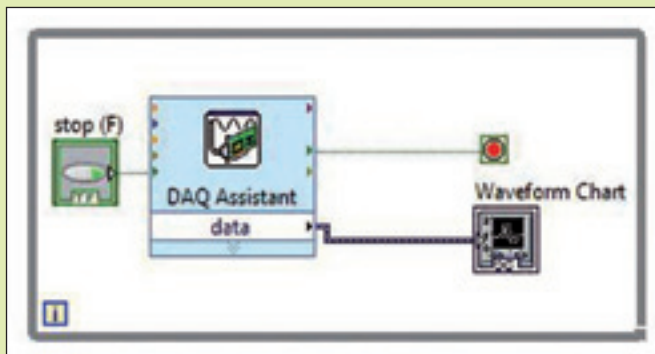


Fig. 15 - Codice grafico relativo al programma esempio 6, scritto nella puntata precedente.

Genera un aggiornamento dei dati per ciclo, quindi un aggiornamento al secondo. Valori inferiori di questo parametro, relativamente al parametro *rate*, determinano aggiornamenti più veloci, ma caricano di lavoro maggiormente il processore grafico. Con i valori massimi di frequenza di campionamento garantiti dalla 6008 è difficile notare qualche differenza anche con macchine poco performanti, ma con valori più elevati e con programmi più complessi, bisogna iniziare a tenere in considerazione il numero di campioni per ciclo.

Come si può vedere sempre nella **Fig. 16**, la linea di tipo boolean che agisce sull'ingresso di stop del ciclo while è stata trasportata fuori dalla sequenza tramite un tunnel; quest'operazione è stata resa necessaria dal fatto che non è possibile portare l'ingresso di stop del while all'interno di un'altra struttura. Si noti che è stato usato un tunnel e non un sequence local, dato che il sequence local serve a spostare segnali da un frame all'altro di una struttura stacked sequence, e non al di fuori di essa.

Passiamo ora al secondo frame della struttura sequence 2, che contiene il codice grafico relativo alla gestione delle linee di I/O digitale. Il codice utilizzato è stato riportato nella **Fig. 17**. In questo frame sono stati spostati, oltre al task di I/O, gli otto pulsanti e gli otto LED generati sul front panel, per la gestione di questo task. I LED non sono strettamente necessari in questo caso, dato che i pulsanti cambiano colore in base al loro stato e, volendo, possono anche esplicitarlo in maniera più chiara tramite un testo associato al loro stato booleano. Tuttavia l'inserimento dei LED permette di evidenziare una caratteristica dei controlli che non è stata ancora presentata all'interno del corso, e che vedremo successivamente.

Iniziamo a vedere come vengono utilizzati gli otto pulsanti per la gestione delle linee. Il task di I/O accetta una singola linea di ingresso, di tipo array di boolean. Se si provasse, quindi, a collegare un semplice ingresso di tipo booleano (anche ad un task che gestisce una singola linea di I/O) si genererebbe un errore. Ciò che bisogna fare è costruire un array a partire dalle singole variabili booleane associate ai tasti. Per farlo ci occorre un blocco specifico, in particolare il blocco build array (*array/build array*). Una volta posizionato il blocco sul foglio di lavoro occorre dimensionarlo, dato che, per impostazione predefinita, si presenta con un singolo ingresso. Per dimensionarlo in modo

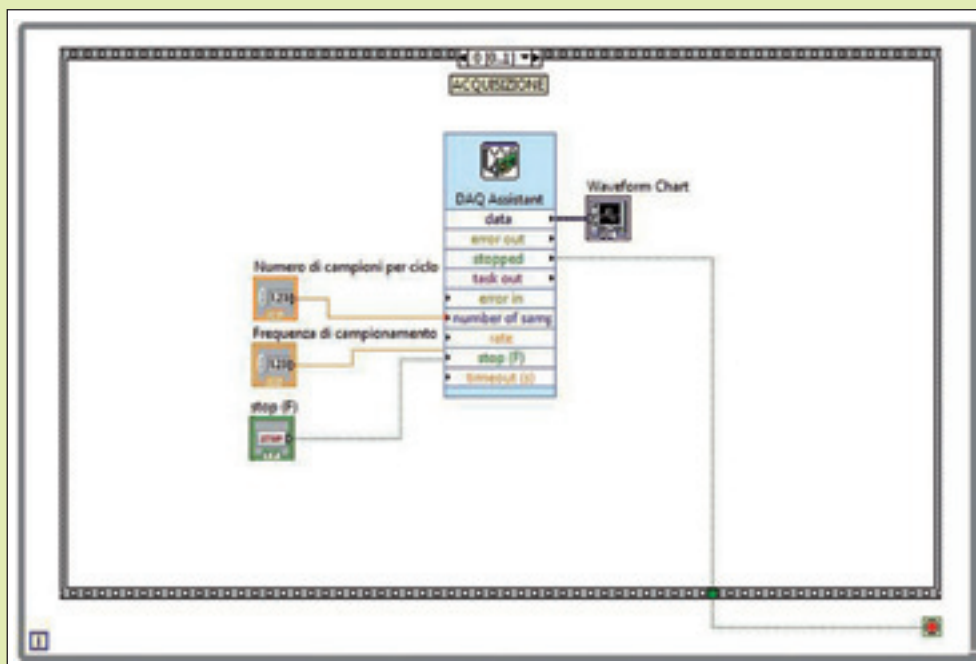


Fig. 16
Codice grafico
relativo al task di
acquisizione.

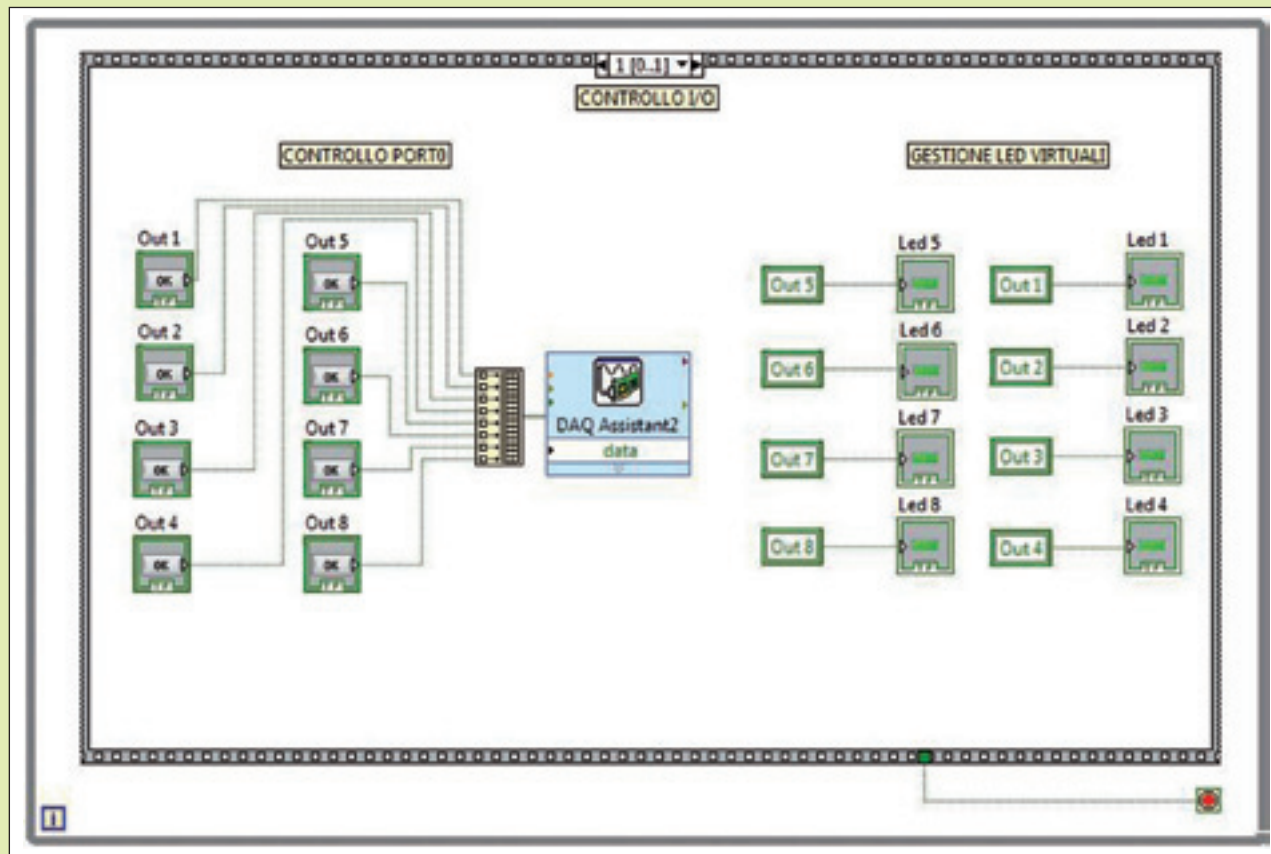


Fig. 17 - Codice grafico relativo al task di Output Digitale.

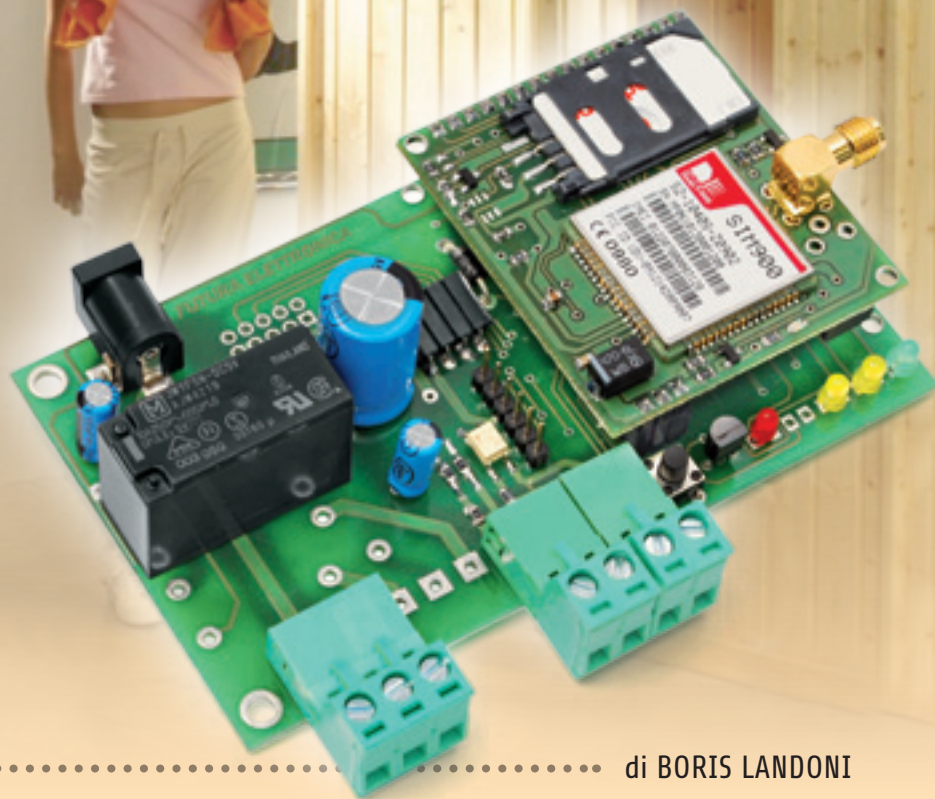
che ne possa accogliere otto, basta posizionarsi sul bordo inferiore del blocco con la freccia del mouse, premere e mantenere premuto il tasto sinistro e trascinare verso il basso, fino a quando gli ingressi del blocco non diventano otto. Ora colleghiamo una ad una le uscite booleane di tutti gli otto controlli di tipo pulsante che sono stati inseriti. Perché l'associazione tasto/linea di I/O sia corretta, il collegamento va fatto in base al nome del controllo, quindi il controllo Out1 andrà sul primo ingresso del blocco build array (quello più in alto), Out2 sul secondo e via discorrendo. Terminata quest'operazione bisogna collegare l'uscita del blocco build array all'ingresso data del task. Una volta fatto, le linee gestite dal task passano sotto il diretto controllo dei pulsanti virtuali del front panel. In ultimo vediamo cosa è stato fatto per la gestione dei LED. Naturalmente, volendo associare direttamente lo stato di un pulsante ad un LED virtuale basta collegare la linea booleana del tasto direttamente al LED. Questo va bene con programmi piccoli, ma con grossi schemi a blocchi, con centinaia di controlli ed indicatori può diventare un problema e la quantità di linee può rendere poco leggibile il programma. Inoltre, nel caso in cui si desideri collegare

un controllo ad un indicatore in programmi composti da molte strutture annidate, bisognerebbe usare una grossa quantità di sequence local e tunnel. Anche questo tipo di operazione può rendere poco leggibile un programma. Per ovviare a tale problema, LabVIEW permette di generare delle variabili locali associate ad un qualsivoglia controllo o indicatore; allo scopo, basta cliccare con il tasto destro del mouse sul controllo/indicatore e selezionare l'opzione *Create/Local Variable*. Il blocchetto generato è un riferimento alla variabile di partenza che può essere trasportato in un punto qualsiasi del foglio di lavoro del VI. In questo modo, nel nostro programma sono state generate otto variabili locali, riferite agli otto tasti e collegate agli otto LED. L'effetto è lo stesso che si ottiene collegando le linee logiche dei pulsanti ai LED, a beneficio, però, di una migliore leggibilità del programma.

Bene, a questo punto la versione 2 del nostro programma di esempio è completa. Nella prossima puntata vedremo come sfruttare le conoscenze acquisite finora per salvare su file i risultati dell'acquisizione e processare i dati, tramite i potenti blocchi di post-processing forniti da LabVIEW. ■

TERMOSTATO AMBIENTE CON GSM

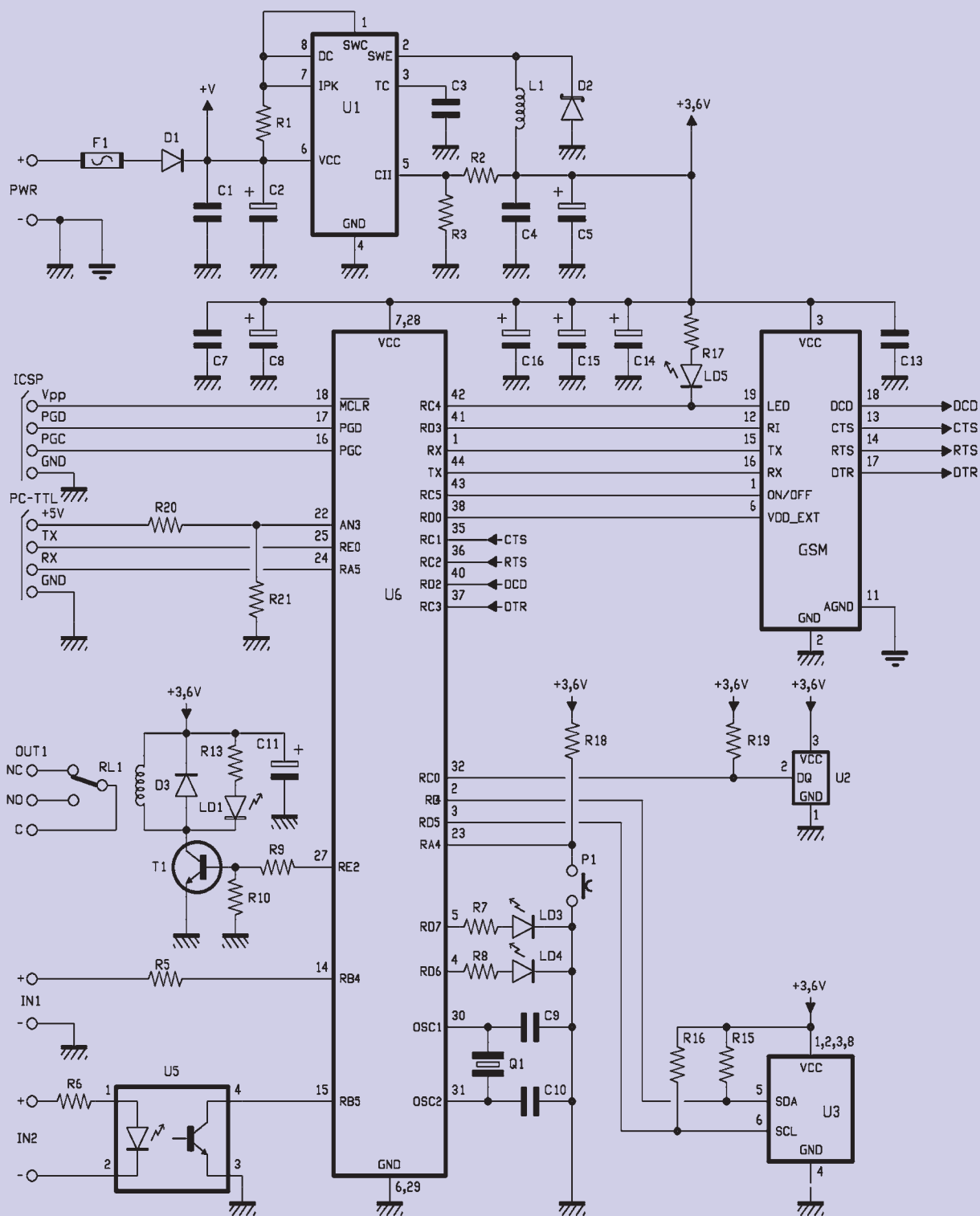
Permette di gestire a distanza il riscaldamento o il condizionamento in un locale. Può inoltre avvisare, tramite SMS o chiamate, quando la temperatura rilevata si discosta dalla finestra impostata o l'ingresso di allarme è attivo.



di BORIS LANDONI

In estate come in inverno, può essere comodo installare nella propria abitazione abituale o nella casa di vacanza, un dispositivo di controllo a distanza in grado di attivare l'impianto di condizionamento d'aria o di riscaldamento prima di arrivare, così da trovare il clima ottimale. Una soluzione è certamente il progetto descritto in queste pagine,

che fa anche qualcosa di più; infatti è finalizzato specificatamente al monitoraggio ed all'impostazione della temperatura di un locale da remoto: opportunamente collegato alla caldaia, permette di controllarla al fine di mantenere la temperatura impostata dall'utente, ma anche di comunicare a distanza, in automatico o su richiesta, il valore attuale.



Lo stesso può fare con un impianto di condizionamento d'aria. In pratica si tratta di un termostato in grado di agire localmente, tramite un relé con cui comandare il condizionatore o la caldaia,

in modo autonomo allo scopo di conservare una temperatura impostata da remoto; il progetto usa come base il telecomando modulare già descritto nel fascicolo n° 146, rispetto al quale monta un

solo relé ed un solo fotoaccoppiatore, ma richiede il sensore termico ed il pulsante. Tutti i comandi relativi al controllo ed alle impostazioni del dispositivo si inviano mediante

SMS. Le modalità di funzionamento previste sono 3: forzata (M), automatica (A) e asservita (T); possono essere impostate sia da SMS, sia con un pulsante locale. Nel primo caso (M) l'impianto di riscaldamento viene forzato a funzionare in continuazione, a prescindere dalle impostazioni di temperatura, e ciò fino a quando non si passa ad una differente modalità di funzionamento. Nel secondo caso (A), l'impostazione della temperatura avviene tramite un SMS ed il circuito utilizza il sensore di cui è dotato per verificare la temperatura ambiente e fare sì che questa venga mantenuta al valore definito dall'utente con l'SMS. Nel terzo caso (T), il dispositivo comanda il proprio relé di uscita per fargli seguire un termostato esistente nell'ambiente da controllare, in modo da far funzionare l'impianto in maniera tradizionale; ciò permette di integrare il telecontrollo in un impianto tradizionale, dando all'utente l'opportunità di scegliere di affidare il controllo ad esso o al termostato normale. Per rendere più completa l'integrazione con l'impianto esistente, il nostro sistema realizza anche la funzione di teleallarme: dispone di un ingresso optoisolato a livello di tensione che può essere collegato alla spia di malfunzionamento della caldaia e configurato per adattarlo alle diverse condizioni di funzionamento (ciò per dare allarme se riceve tensione oltre la soglia o se è a zero volt); così facendo, nel caso l'impianto vada in blocco (per un'ostruzione della canna fumaria, l'eccessivo abbassamento della pressione del liquido nei caloriferi, la mancanza di gas e via di seguito) il sistema comunica a distanza l'anomalia con un SMS di allarme o una telefonata. L'allarme sull'ingresso optoisolato

vale solo quando il circuito lavora in modalità automatica (A). Ma non è tutto qui: è previsto anche un allarme termico, che scatta quando la temperatura rilevata nell'ambiente esce da una finestra impostata dall'utente, ovvero eccede la soglia massima o scende al di sotto di quella minima; come per quello collegato alla spia della caldaia, l'allarme termico può determinare l'invio di SMS o telefonate. Il circuito è in grado di memorizzare al massimo otto numeri telefonici cui inviare gli allarmi sotto forma di messaggi di testo o semplici chiamate.

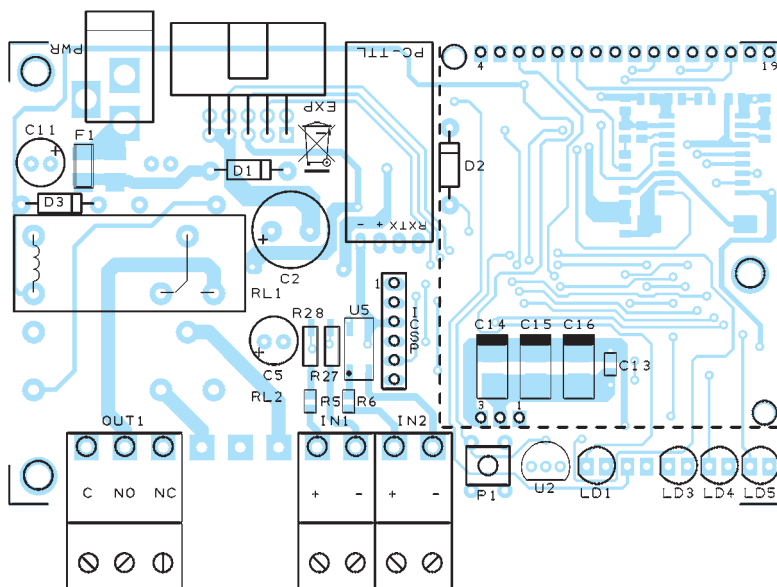
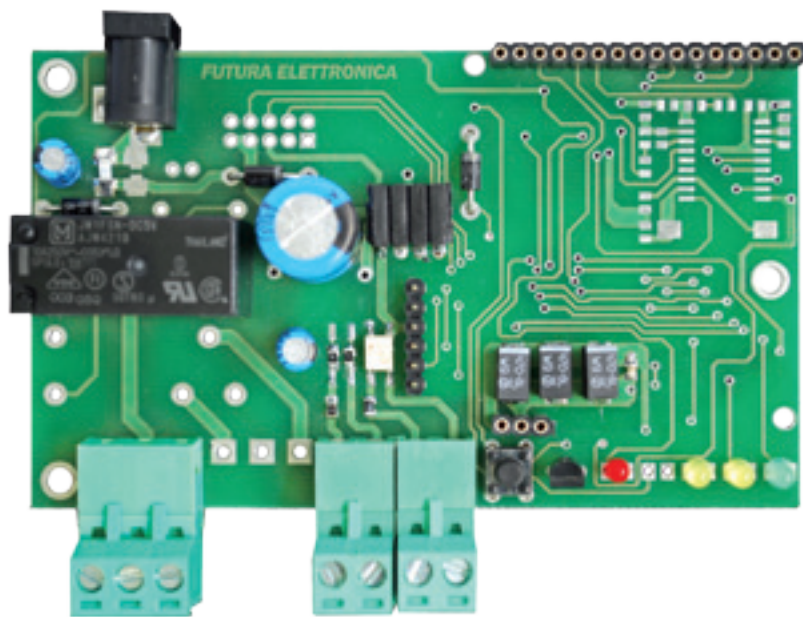
SCHEMA ELETTRICO

L'intero termostato GSM viene gestito da un microcontrollore Microchip PIC18F46K20-I/PT, programmato con un firmware che svolge il controllo della temperatura e si occupa del dialogo con il modulo GSM, che è il componente siglato GSM; questo è in realtà la basetta che ospita il cellulare, quindi i numeri dei pin si riferiscono ai contatti dei pin-strip utilizzati. Il firmware legge la temperatura ambiente mediante una sonda intelligente Dallas DS1820, controlla la condizione logica dell'ingresso optoisolato e comanda il relé all'occorrenza. Detto ciò, diamo un'occhiata più approfondita allo schema elettrico, partendo dal blocco di alimentazione, che ricava dalla tensione d'ingresso (l'alimentazione, applicata ai punti + e - PWR, deve essere una tensione continua di valore compreso tra 5 e 32 V) una componente stabilizzata di 3,6 V di ampiezza, avvalendosi di un regolatore switching. La tensione d'ingresso viene filtrata a valle del diodo di protezione contro l'inversione di polarità (D1) mediante i condensatori C1 e C2 ed il fusibile F1 proteg-

ge efficacemente il circuito e la fonte di alimentazione in caso di cortocircuito nel regolatore integrato che segue. Il regolatore è U1 (un MC34063) e funziona da DC/DC converter step-down; viene utilizzato nella classica configurazione di regolatore PWM buck a carica d'induttanza, la cui tensione di uscita dipende dall'energia immagazzinata in L1. La tensione che esso fornisce viene stabilizzata retroazionando il DC/DC mediante la retroazione di una porzione della componente livellata ai capi di C4 e C5, operata dal partitore resistivo R2/R3. I 3,6 volt presenti ai capi di questi condensatori vengono abbondantemente filtrati da altri condensatori collocati sulle linee di alimentazione del microcontrollore e del modulo GSM, il quale in trasmissione presenta picchi di assorbimento cui suppliscono C7, C8, C13, C14, C15 e C16, evitando che la richiesta impulsiva di corrente causi impulsi di disturbo al micro.

Il microcontrollore viene utilizzato nella configurazione con oscillatore di clock interno; nello schema e nel circuito stampato appare comunque il quarzo esterno, che abbiamo previsto per futuri sviluppi. Il micro, una volta inizializzate le linee di I/O, verifica la condizione logica dell'ingresso optoisolato a livello di tensione facente capo alla linea RB4 e quella delle linee RC4, RC5, RD0, RD3, RX, che servono a ricevere dal modulo cellulare le principali segnalazioni; precisamente, RD3 viene usata per rilevare l'arrivo delle chiamate entranti (si interfaccia con RI del modulo cellulare) mentre RC4 controlla il "LED di campo" del GSM, leggendone lo stato dell'uscita corrispondente. Dalla frequenza e dalla durata degli impulsi, il PIC comprende le con-

[piano di **MONTAGGIO**



Elenco Componenti:

- R1: 0,1 ohm 1W (1206)
- R2: 2,2 kohm (0805)
- R3: 1,2 kohm (0805)
- R4: -
- R5: 4,7 kohm (0805)
- R6: 4,7 kohm (0805)
- R7: 330 ohm (0805)
- R8: 330 ohm (0805)
- R9: 4,7 kohm (0805)
- R10: 10 kohm (0805)
- R11: -
- R12: -
- R13: 330 ohm (0805)
- R14: -
- R15: 1,5 kohm (0805)
- R16: 1,5 kohm (0805)
- R17: 330 ohm (0805)
- R18: 4,7 kohm (0805)
- R19: 4,7 kohm (0805)
- R20: 1 kohm (0805)
- R21: 2,2 kohm (0805)
- R22: -
- R23: -
- R24: -
- R25: -
- R26: -
- R27: -
- R28: 0 ohm
- R29: 0 ohm
- C1: 100 nF multistrato (0805)
- C2: 1000 μ F 35 VL elettrolitico
- C3: 100 pF ceramico (0805)
- C4: 100 nF multistrato (0805)
- C5: 100 μ F 16 VL elettrolitico
- C6: -
- C7: 100 nF multistrato (0805)
- C8: 470 μ F 6,3 VL tantalio (CASE-D)
- C9: -
- C10: -
- C11: 100 μ F 16 VL elettrolitico
- C12: -
- C13: 100 nF multistrato (0805)

dizioni della copertura radiomobile e si regola di conseguenza. Il micro contiene un UART accessibile dai piedini 44 (trasmissione) ed 1 (ricezione) che usa per dialogare con il cellulare; per l'esattezza, mediante il primo pin (TX) interroga ciclicamente il modulo allo scopo di verificare l'arrivo di SMS, mentre TX ed RX insieme servono, durante le procedure

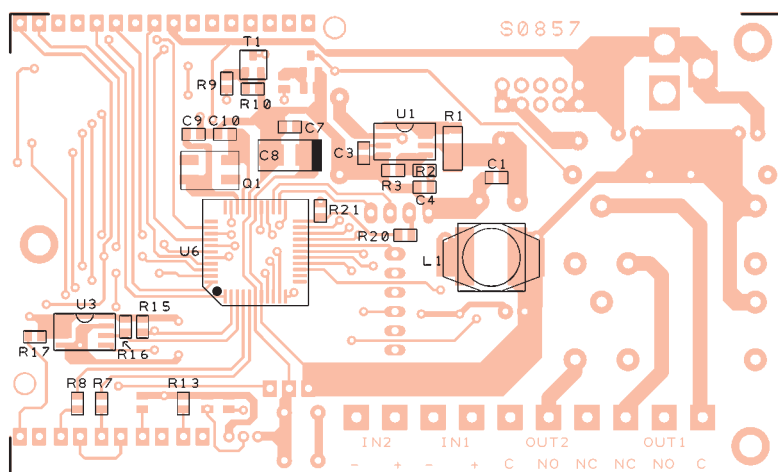
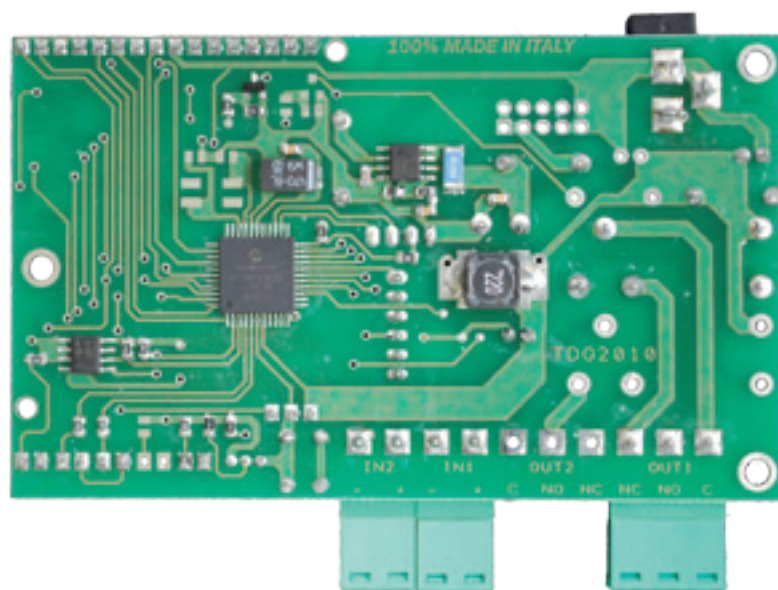
di chiamata, ricezione e trasmissione messaggi, per la comunicazione tra il microcontrollore e il GSM. Sempre dell'UART, vengono usati i segnali di controllo CTS (Clear To Send) RTS (Request To Send) e DCD (Data Carrier Detect) che corrispondono a quelli del modulo cellulare usato. Completano il set di I/O destinati al cellulare, le linee RC5 ed RD0:

la prima comanda l'accensione e lo spegnimento del GSM (tramite un transistor sistemato nella basetta del cellulare) mentre la seconda linea provvede al reset del cellulare. Il pulsante per la gestione locale della modalità di funzionamento del dispositivo viene letto mediante la linea RA3, configurata come ingresso e dotata di resistore di pull-up esterno

C14: 470 μ F 6,3 VL tantalio (CASE-D)
 C15: 470 μ F 6,3 VL tantalio (CASE-D)
 C16: 470 μ F 6,3 VL tantalio (CASE-D)
 C17: -
 C18: -
 C19: -
 C20: -
 Q1: -
 Q2: -
 U1: MC34063AD
 U2: DS18B20
 U3: 24FC256-SN
 U4: -
 U5: TLP181
 U6: PIC18F46K20-I/PT (MF857)
 U7: -
 D1: 1N4007
 D2: 1N5819
 D3: 1N4007
 D4: -
 T1: BC817
 T2: -
 LD1: led 3 mm rosso
 LD2: -
 LD3: led 3 mm giallo
 LD4: led 3 mm giallo
 LD5: led 3 mm verde
 L1: Bobina 20 μ H
 RL1: Relè 5V 1 scambio
 RL2: -
 P1: Microswitch
 F1: Fusibile rapido 2 A (1206)

Varie:

- Morsettiera 2 poli (2 pz.)
- Morsettiera 3 poli
- plug alimentazione
- Strip maschio 6 poli
- Strip femmina 3 poli
- Strip femmina 16 poli
- Strip femmina 4 poli 90°
- Circuito stampato



(R11) e quindi attiva a livello basso. Particolare attenzione la merita la gestione degli ingressi, in quanto qui, rispetto allo schema generale del telecomando modulare, è stata prevista una variante: per leggere direttamente la condizione del contatto del termostato ambiente dell'impianto cui il circuito si applica, bisogna bypassare il fotoaccoppiatore U4,

che perciò non viene usato; allo scopo immaginiamo di cortocircuitare il pin 1 col 4 e il 2 col 3. La lettura dell'ingresso IN1 avviene mediante la linea RB4, configurata come input e provvista di pull-up interno: a contatto aperto, resta a livello alto, mentre quando il termostato chiude il proprio contatto assume lo zero logico. All'IN1 deve essere collegato il

contatto normalmente aperto dell'eventuale termostato già esistente nell'impianto, ma solo se si intende attivare la modalità asservita (T): infatti in questa il relè RL1 segue le vicende del termostato. Quanto all'IN2, si legge tramite RB5, la quale acquisisce lo stato del transistor di uscita del fotoaccoppiatore U5; l'input è attivo quando sottoposto a una



Arance per riscaldare

L'ultima novità in tema di biocombustibili si chiama Refolo: è un sistema ideato dai ricercatori dell'Istituto per lo studio dei materiali nanostrutturati (Ismn) del Cnr, per trasformare gli scarti degli agrumi in combustibile da riscaldamento. Consentirà alle aziende del settore di reimpiegare gli scarti, il cui smaltimento comporta grandi spese, poiché questi "rifiuti speciali non pericolosi" non trovano posto nella catena del compostaggio a causa di problemi chimici dipendenti dalla loro acidità. L'Italia ogni anno produce un milione di tonnellate di arance, di cui 300 mila (anche per le tanto discusse quote stabilite dall'UE) finiscono in discarica. Ciò comporta un aumento dei costi della filiera e una riduzione dei margini di profitto per il comparto agricolo e manifatturiero. Ed è qui che entra in gioco Refolo: attraverso un sistema di disidratazione spinta, gli scarti vengono ridotti in polvere e, tramite compattazione a freddo, trasformati in combustibile sotto forma di pellet di 3 centimetri, che possono essere combusti nelle stufe e negli impianti. I biopellett ottenuti hanno un potere calorifero dai 16 ai 20 Mega Joule per chilogrammo, uguale o superiore a quello del legno, a differenza del quale però non contengono metalli, cloro o zolfo e hanno una percentuale di cenere minore. Una curiosità è che il pellet mantiene il profumo degli agrumi.

tensione di valore compreso fra 3 e 30 V. Quando all'ingresso IN2 è applicata una tensione di almeno 3 volt, il fotoaccoppiatore ha il LED acceso e il fototransistor d'uscita in conduzione, cosicché il collettore (piedino 4) è a circa zero volt, per effetto della caduta sul resistore di pull-up interno configurato, durante l'inizializzazione degli I/O. Se l'ingresso non è polarizzato, l'optoisolatore è interdettato e il suo pin 4 risulta a livello alto. L'ingresso IN2 può essere usato per monitorare lo stato di una linea elettrica, quale, ad esempio, quella corrispondente alla spia di allarme della calda-

ia. Va notato che IN1 determina solo azioni locali ed esclusivamente nella modalità asservita al termostato esistente, mentre IN2 avvia l'esecuzione della sequenza di allarme, consistente nell'invio, in base all'impostazione fatta dall'utente, di SMS o chiamate dirette ai numeri memorizzati in lista. La sequenza parte quando l'ingresso viene sottoposto a una differenza di potenziale superiore a quella di soglia (circa 3 volt). Passando alle uscite, qui ce n'è solo una, corrispondente a RL1. Quando deve attivare la caldaia o il condizionatore, il microcontrollore pone a livello logico alto

la propria linea RE2, facendo andare in saturazione il transistor T1 e comandando così l'eccitazione del relé RL1; ogni attivazione viene segnalata da LD1, alimentato in parallelo alla bobina del relé. Per proteggere la giunzione di collettore dei transistor nei passaggi dalla saturazione all'interdizione, quando l'induttanza della bobina del relé genera picchi di tensione inversa, abbiamo collegato un diodo in parallelo alla bobina, che elimina, appunto, tali impulsi. Del RL1 è disponibile l'intero scambio, così da poter gestire circuiti che richiedono un contatto normalmente chiuso o uno normalmente aperto. La programmazione del PIC avviene in-circuit, mediante il connettore ICSP, che fa capo alle linee /MCLR, PGU e PGC; all'ICSP giungono anche l'alimentazione e la massa del micro. Non contenti, abbiamo anche previsto un'interfaccia di comunicazione seriale finalizzata a consentire la programmazione dei dati di funzionamento (ad esempio la lista dei numeri telefonici, la gestione dei livelli di ingresso, il testo degli SMS inviati dal circuito in risposta ai comandi ecc.) tramite un PC: ciò consente di programmare un telecomando al banco, prima di metterlo in funzione e risparmiare l'invio degli SMS di configurazione, che possono comunque essere usati ma cui conviene ricorrere solo una volta che il sistema è installato sul posto. Avendo già impegnato l'UART per il dialogo con il modulo cellulare, la comunica-

zione seriale avviene mediante le linee RE0 ed RA5, sfruttate rispettivamente come TX ed RX; la AN3, assegnata all'A/D converter interno, serve per rilevare la presenza dei 5 volt e quindi l'inserzione del connettore. L'interfaccia seriale è a livello TTL e può essere facilmente connessa a un converter USB quale ad esempio l'FT232 della FTDI, per interfacciare il microcontrollore con un PC fornito di USB. Per l'interfaccia potete usare il piccolo modulo FT782M della Futura Elettronica (www.futurashop.it) che è già provvisto di un connettore pin-strip a passo 2,54 mm per l'inserzione diretta nel nostro circuito, sul connettore TTL, che è una femmina SIL a passo 2,54 mm. Per la programmazione da PC abbiamo previsto un'apposita utility progettata per l'ambiente Windows: un semplice programma scaricabile gratuitamente dal nostro sito www.elettronica.in. it e che, installato e lanciato, dà accesso a un pannello di controllo dal quale l'utente può, facendo clic nelle apposite caselle, introdurre o modificare dati. Restando in tema di configurazione dell'unità, dovete notare che i dati corrispondenti non vengono salvati nella EEPROM del microcontrollore, ma in un chip di memoria esterno, siglato 24FC256-SN; si tratta di una EEPROM CMOS da 256 kbit ad accesso seriale, con interfaccia I²C-Bus. Per dialogare con essa il micro inizializza le linee di I/O RD4 ed RD5, usate rispettivamente come SDA (linea dati) ed SCL (linea di clock). L'aver spostato i dati di configurazione del telecontrollo in una memoria esterna ci permette di sfruttare tutta la EEPROM interna al micro per aumentare le funzioni disponibili. Veniamo adesso al rilevamento della temperatura

ambiente, che serve nella modalità automatica per decidere quando accendere la caldaia e in tutti gli altri casi per comunicare, su richiesta remota, il valore attuale; il rilevamento della temperatura serve anche quando l'utente desidera sfruttare la funzione di teleallarme termico, che gli consente di essere avvisato del superamento della soglia massima o dell'abbassamento oltre quella minima impostata. Il rilevamento della temperatura ambientale viene effettuato dal microcontrollore con l'ausilio di una precisissima sonda DS18B20 della Dallas Semiconductor, capace di misurare temperature comprese nel range -5÷150 °C con un'accuratezza di appena ± 0,5 °C (nel campo -10÷85 °C), di digitalizzarle e di renderle disponibili su una linea seriale unidirezionale, collegata al pin RC0 del micro. La DS18B20 esprime la temperatura rilevata con una risoluzione di 11 bit; per interrogarla, il micro utilizza un comando "Convert T" (codice 44h) corrispondente all'avvio dell'operazione di conversione della misura. La sonda risponde con uno 0 mentre esegue la conversione e con un 1 ad operazione terminata. A questo punto nei registri del componente sono disponibili due byte corrispondenti al valore della temperatura rilevata in gradi Celsius. Quindi, non ci resta che leggerli con un codice comando "Read ScratchPad (codice BEh)". Nella modalità da noi scelta, dei 16 bit restituiti dal DS18B20 ogni volta che il microcontrollore lo interroga, i primi 11 (tutto il byte meno significativo e primi tre bit di quello più significativo) esprimono il valore della temperatura e gli ultimi cinque il segno, che viene indicato con tutti 0 se la temperatura rilevata è positiva e tutti 1 se, invece, la misura si

riferisce a una temperatura negativa (sotto lo zero). La particolarità del componente Dallas è che può esprimere la temperatura con i decimali: i primi quattro bit meno significativi si riferiscono a valori minori di 1, dato che il primo vale 2 alla -4, ossia 1/16, il secondo vale 0,125, il terzo 0,25 e il quarto 0,5. Così il sensore Dallas riesce ad esprimere praticamente tutti i valori di temperatura che può misurare. Per fare un esempio chiarificatore, immaginiamo di misurare 64,25 °C positivi; all'uscita dell'integrato DS18B20 avremo due byte, il primo dei quali sarà (partendo dal bit più significativo) 00000100 e il secondo (sempre tenendo a sinistra il bit maggiormente significativo) 00000100. Analizzando i dati, il byte LSB dà il valore di 0,25 e il secondo byte dà +64 gradi. Volendo considerare il dato nell'insieme, ossia il valore complessivo formato dai due byte, possiamo dire che i possibili valori esprimibili dall'uscita del DS18B20 sono compresi tra 11111101101110000 (-55 °C) e 00000111110110000 (+125 °C). Trasferendo questi valori in formato esadecimale, possiamo dire che l'escursione è compresa fra AB70h e 07D0h. Sempre ragionando in esadecimale, vediamo che una temperatura di 10,125 °C corrisponde a 00A2h e che una di -20 °C si esprime con BCC0h. La sonda dialoga usando il protocollo One-wire, che permette il colloquio bidirezionale su un unico filo; allo scopo viene richiesta la resistenza di pull-up R19. In questa applicazione utilizziamo linee distinte per alimentazione e dati, ma la DS18B20 potrebbe funzionare anche ponendo in comune l'alimentazione e i dati, ovvero utilizzando a pieno il protocollo one-wire.

IL FUNZIONAMENTO

Descritto lo schema elettrico, vediamo adesso come funziona il termostato GSM: all'accensione il PIC lancia l'Easy Setup, che è una modalità grazie alla quale si può configurare rapidamente il funzionamento del dispositivo; si attiva se, nei primi tre minuti di esercizio, riceve una chiamata da un telefono che chiami senza nascondere l'ID. In tal caso, il telecomando memorizza nella prima posizione della propria lista il numero del telefono che ha effettuato la chiamata.

Se non riceve la chiamata, trascorsi tre minuti il circuito abbandona l'Easy Setup ed entra nel normale funzionamento, nel quale comanda la caldaia nella modalità richiesta dall'utente, verifica l'eventuale allarme in arrivo all'ingresso optoisolato e gestisce il relé. Alla prima accensione, a meno che l'unità non sia stata collegata ad un PC e da esso configurata, per impostazione predefinita funziona in automatico (A) ossia attiva il relé quando la temperatura rilevata nell'ambiente è minore di quella impostata dall'utente, ovvero, se questi non l'ha mai modificata, di quella predefinita (22 °C). La condizione è evidenziata dall'accensione del LED giallo (LD2).

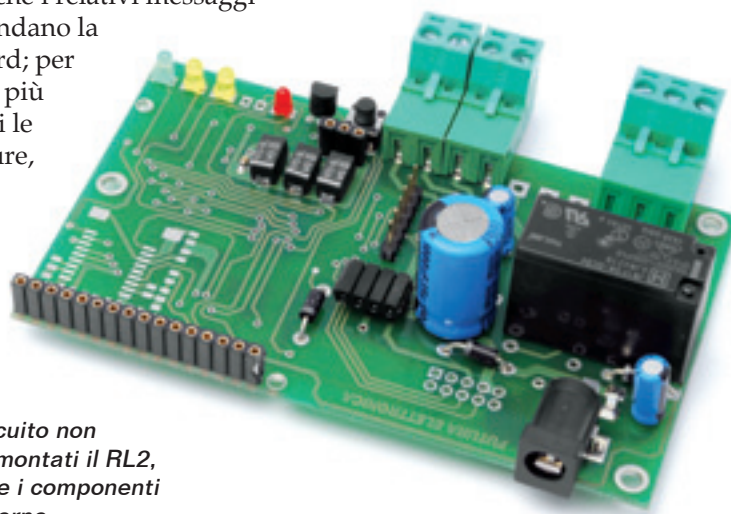
Agendo sul pulsante P1 si può definire sequenzialmente una diversa modalità: premendolo si passa dall'automatica alla forzata (M) nella quale RL1 è sempre attivo fin quando non lo si disattiva da SMS o non si cambia modalità; la condizione è visualizzata dall'accensione dei LED LD1 (verde) e LD2 (giallo). Premendo il pulsante un'altra volta, si passa nel modo asservito (T) che viene segnalato dall'accensione del solo LD1: qui, il circuito segue le vicende del termostato eventualmente collegato all'ingresso

IN1, nel senso che il relé scatta (chiude C su NO) se l'input viene cortocircuitato e rimane a riposo nel caso contrario (ingresso aperto). Un'ulteriore pressione di P1 riporta l'unità nella modalità automatica. Resta inteso che se tramite SMS si ordina al circuito di impostare una modalità differente da quella decisa localmente con P1, l'ordine è prioritario, nel senso che il comando da telefono (ma anche quello da computer) annulla quello da pulsante.

IMPOSTAZIONI E COMANDI

Ci sono due modi per impartire i comandi: il primo consiste nell'inviare gli opportuni SMS secondo la sintassi richiesta e corredata (quando occorre) dalla password, indispensabile ad evitare che la configurazione possa essere operata da estranei; il secondo consta nell'introdurlo, mediante Easy Setup, il numero del telefono cellulare abilitato a fare le successive impostazioni tramite SMS. In tal caso buona parte dei messaggi potrà essere inviata senza la password, perché il circuito ammette i comandi provenienti da un cellulare il cui numero sia già in lista. Gli SMS di impostazione possono essere inviati da qualsiasi telefonino, a patto che i relativi messaggi comprendano la password; per rendere più semplici le procedure,

alcuni comandi, se inviati da numeri già in lista, possono fare a meno della password. I numeri contenuti in tale lista sono gli stessi ai quali (se abilitati) verranno inviati i messaggi in caso di allarme. Il telecomando accetta SMS multipli, ossia contenenti più di un comando o comandi riguardanti uno o più numeri telefonici; i comandi devono essere separati ognuno da quello successivo con una virgola. Ogni comando di impostazione prevede una risposta che il dispositivo invia tramite SMS; di conseguenza un comando multiplo produrrà più messaggi di risposta; per evitarli, bisogna disattivare la risposta, cosa che si ottiene terminando l'SMS multicomando con il testo RISP. Dopo questa necessaria premessa, possiamo iniziare a descrivere i comandi, riepilogati in **Tabella 1**. Iniziamo con il comando per la modifica della password, che consiste in un SMS del tipo `PWDxxxxx;pwd`, dove al posto di `xxxxx` si deve scrivere la nuova password (numerica, di cinque cifre...) mentre `pwd` indica la password attuale; la password predefinita è 12345. La memorizzazione di uno degli otto numeri si effettua inviando un SMS contenente il testo



Nel circuito non vanno montati il RL2, l'8870 e i componenti di contorno.

I comandi del termostato GSM

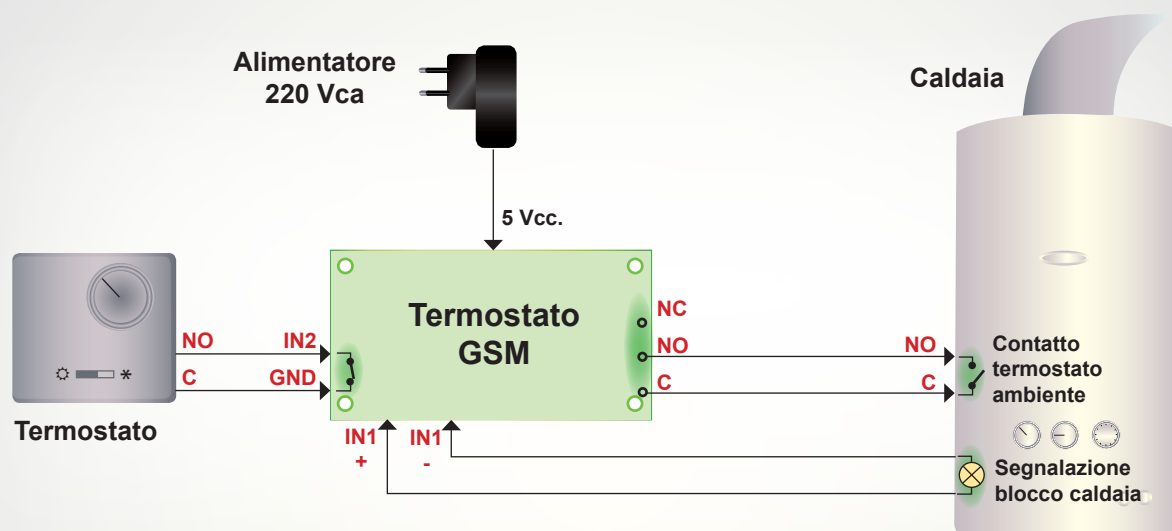
FUNZIONE	COMANDO SMS	VALORE PREDEFINITO	PWD OBBLIGATORIA
CAMBIARE LA PASSWORD	PWDxxxx; 12345	12345	SEMPRE
MEMORIZZARE UN NUMERO IN POSIZIONE "x" (fino ad 8 numeri, al massimo di 19 caratteri ognuno)	NUMx+392229876543;12345	-	SE LA POSIZIONE x NON È VUOTA
CANCELLARE UN NUMERO	NUMx;12345	-	SEMPRE
VERIFICARE I NUMERI MEMORIZZATI	NUM?;12345	-	SEMPRE
EFFETTUARE IL RESET COMPLETO DEL TELECONTROLLO	RES	-	SEMPRE
ATTIVARE IL RELÉ IN MODALITÀ BISTABILE	OUT:ON	Relé a riposo	
DISATTIVARE IL RELÉ IN MODALITÀ BISTABILE	OUT:OFF	Relé a riposo	
RIPRISTINARE LO STATO RELÉ DOPO UN BLACK_OUT (x=1: ripristino attivo. x=0: relé disattivato dopo il black-out)	RIPx	0	
RICHIEDERE LO STATO DELLA FUNZIONE "RIPRISTINO"	RIP?	-	
IMPOSTARE LA MODALITÀ DI FUNZIONAMENTO (T=Termostato, A=Automatico, M=Manuale)	FUN:X	A	
RICHIEDERE LA MODALITÀ DI FUNZIONAMENTO	FUN:?		
RICHIEDERE LA TEMPERATURA	TEMP		
AZZERARE LE TEMPERATURE MINIMA E MASSIMA	TRES		
IMPOSTARE LA TEMPERATURA DA MANTENERE (00÷99)	NOR:XX	22	
IMPOSTARE L'ALLARME PER TEMPERATURA MINIMA (y = "+" o "-"; xx = 00 ÷ 99)	ALLMIN:yxx		
IMPOSTARE L'ALLARME PER TEMPERATURA MASSIMA (y = "+" o "-"; xx = 00 ÷ 99)	ALLMAX:yxx		
RICHIEDERE LE TEMPERATURE DI ALLARME	ALL?		
TEMPO INIBIZIONE ALLARME TEMPERATURA (da 00 a 59 min.)	INT:mm	5	
TEMPO INIBIZIONE INGRESSO CALDAIA (da 00 a 59 min.)	INC:mm	5	
IMPOSTARE I NUMERI CUI VENGONO INVIATI GLI SMS	SMSxxxxxxx: ON/OFF	Tutti attivi	SEMPRE
IMPOSTARE I NUMERI CUI VIENE FATTO UNO SQUILLO	VOCxxxxxxx: ON/OFF	Tutti attivi	SEMPRE
INVIARE AUTOMATICAMENTE UN SMS ALL'ACCENSIONE (x=1 per l'invio automatico; x=0 per disattivare l'invio)	AVVx	0	
MODIFICARE IL TESTO DI: ALLARME TEMPERATURA SUPERIORE A SOGLIA ALLMAX (massimo 100 caratteri)	THI:xxxxxxxxxxx	ALLARME!! SUPERATA SOGLIA TEMPERATURA	
MODIFICARE IL TESTO DI: ALLARME TEMPERATURA INFERIORE A SOGLIA ALLMIN (massimo 100 caratteri)	TLO:xxxxxxxxxxx	ALLARME!! SUPERATA SOGLIA TEMPERATURA	
MODIFICARE IL TESTO DI: ALLARME INGRESSO CALDAIA QUANDO È PRESENTE TENSIONE (massimo 100 caratteri)	TIN:xxxxxxxxxxx	ALLARME INGRESSO!!	
MODIFICARE IL TESTO DI: MESSAGGIO ACCENSIONE (massimo 100 caratteri; invio solo al primo numero in memoria)	TSU:xxxxxxxxxxx	SYSTEM STARTUP	
RICHIEDERE LA TEMPERATURA (comando TEMP)	Chiamata telefonica al numero del telecontrollo		
DISABILITARE LA RISPOSTA PER UN MULTIMESSAGGIO	RISP (in coda al multimessaggio)		
EASY SET UP (memorizza in posizione 1 il primo numero che chiama entro 3 minuti dall'accensione)	Chiamata telefonica con identificativo chiamante visibile		

NUMx+nnnnnnnnnnnnnn;pwd, dove la *x* definisce la posizione in rubrica (1÷8) del numero che si sta memorizzando, *nnnnnnnn* il numero e *pwd* l'attuale password. Sono ammessi numeri di 19 cifre. Per esempio, l'inserimento del numero 0039111222111 in ottava posizione si opera con un comando del tipo: NUM8+111222111;pwd. Per memorizzare un numero inviando il comando da un cellulare già in lista è necessario introdurre la password solamente quando si tenta di sovrascrivere una posizione già

occupata; se la posizione è vuota, basta inviare un SMS contenente NUMx+nnnnnnnnnnnnnn. Resta inteso che se si invia il comando da un telefono estraneo, la password è sempre necessaria. La rimozione di un numero si effettua con il comando NUMx;pwd dove al posto della *x* va scritto il numero della posizione che contiene il numero da cancellare; *pwd* è la solita password, necessaria anche quando il telefono è in lista. Per richiedere la lista dei numeri memorizzati si usa l'SMS NUM?;pwd: questo comando vale con qualsiasi telefono.

Vediamo adesso i comandi per dire al dispositivo quali numeri della lista devono, in caso di allarme (presenza di tensione) all'ingresso optoisolato, ricevere gli SMS o le chiamate (semplici squilli). Per definire i numeri cui inviare i messaggi, il comando è SMSxxxxxxx:ON;pwd, dove le *x* indicano le posizioni dei numeri telefonici memorizzati (da 1 a 8): ad esempio, SMS1367:ON;pwd significa che in caso di allarme il telecontrollo invierà messaggi ai numeri memorizzati nelle posizioni 1, 3, 6 e 7. Se al posto di ON fosse stato scritto "OFF", il

Il collegamento a una Caldaia



Tipico impiego del termostato GSM in un impianto di riscaldamento esistente: l'ingresso IN2 va collegato in parallelo alla spia di malfunzionamento della caldaia (se presente); nel fare il cablaggio accertatevi che la spia sia in bassa tensione e che il positivo sia rivolto sul + dell'ingresso. IN2 si può usare anche per monitorare altri parametri o verificare la presenza della tensione di rete. Quanto al relé, il suo contatto normalmente aperto dev'essere usato per chiudere il comando remoto della caldaia; se questa risulta collegata a un termostato già esistente, interrompetene il collegamento ed effettuate le connessioni illustrate nel disegno. In pratica dovete

connettere i due fili del termostato all'ingresso IN2 e le uscite NO e C del relé al comando della caldaia, ovvero ai punti su cui andavano i fili del termostato che avete sconnesso. Se il termostato che scollegate è del tipo estate/inverno, bisogna considerare il solo contatto corrispondente al comando della caldaia (posizione "inverno"); insomma, non usate il contatto riservato al condizionamento. Se il termostato ha un solo contatto ed è elettronico o comunque dispone del selettore estate/inverno, deve essere disposto su inverno (simbolo col fiocco di neve). Una volta fatti tutti i cablaggi, alimentate l'apparecchio con un alimentatore capace di fornire almeno 5 V/600 mA.

comando avrebbe avuto l'effetto contrario. Per impostazione predefinita, tutti i numeri nelle otto posizioni della lista riceveranno SMS in caso di allarme.

Il comando per configurare i numeri che ricevono chiamate vocali è `VOCxxxxxx:ON;pwd`, per il quale valgono le regole viste per il comando relativo agli SMS.

Il termostato GSM ci permette di stabilire, impostandolo con un apposito comando SMS, un eventuale tempo di inibizione (un intervallo durante il quale non viene considerato lo stato di IN1) per discriminare situazioni di disturbo da segnali veri e propri. L'intervallo si può scegliere tra 0 e 59 minuti primi (quello predefinito è 5 minuti). L'impostazione si effettua con il messaggio `INC:mm`, dove *mm* identifica

il tempo in minuti. L'allarme può essere generato, oltre che dalla presenza di tensione all'ingresso optoisolato, anche quando la temperatura ambiente esce da una finestra preventivamente impostata tramite i comandi `ALLMIN:xxx` (soglia minima) e `ALLMAX:xxx` (soglia massima): in entrambi, al posto di *y* si deve mettere il segno + o - a seconda che la temperatura sia positiva o negativa e in *xx* si scrive il valore assoluto nel formato a due cifre, tenendo conto che è possibile impostare valori compresi tra -10 e 99 °C. Ad esempio, i comandi `ALLMIN:+2` e `ALLMAX:+35` definiscono una finestra compresa tra 2 e 35 °C. Per impostazione predefinita o dopo un reset totale, non sono inseriti valori e quindi l'allarme termico è disattivato;

è possibile impostare anche un solo valore, ma in questo caso l'allarme scatta solo per il superamento della relativa soglia. L'allarme termico può essere disinserito con il comando `ALL:OFF` ed eventualmente reinserito con `ALL:ON`. È possibile richiedere l'invio di un SMS contenente i valori delle temperature di allarme tramite il comando `ALL?`. Anche per l'allarme termico è stata prevista la possibilità di impostare un intervallo di inibizione (il valore predefinito è 5 minuti) che serve, ad esempio, quando l'utente, ricevendo l'allarme indicante che l'ambiente si è raffreddato eccessivamente, attiva forzatamente da remoto il riscaldamento. In tal caso, se non si imposta un intervallo di inibizione sufficiente a dare al

riscaldamento il tempo di alzare la temperatura oltre la soglia di allarme, il telecomando continua a inviare SMS e telefonate.

L'impostazione si effettua inviando il comando *INT:mm*, nel quale al posto di mm va scritto il numero di minuti (tra 00 e 59) di inibizione dal rilevamento dell'allarme.

Il termostato GSM può tenere memoria delle temperature più bassa e più alta registrate dall'ultimo reset o dalla prima accensione e comunicarle; per richiederle, gli si deve inviare l'SMS *TEMP*. Il circuito risponde con un messaggio, diretto al numero del telefono che ha impartito il comando, contenente i valori minimo e massimo. È anche possibile ottenere le temperature senza inviare alcun SMS: basta chiamare, con un telefono il cui numero sia in lista, quindi riagganciare; l'unità invierà il solito messaggio. In ogni momento è possibile azzerare la memoria della temperatura con il comando *TRES*. Il comando per definire la temperatura da mantenere in modalità automatica, è *NOR:xx*, dove al posto delle *x* va inserito il valore desiderato (ad esempio 24) espresso in gradi centigradi.

Visto che abbiamo parlato di modalità di funzionamento, vediamo il comando da utilizzare per impostarla da remoto, via cellulare: è un SMS contenente il testo *FUN:x*, in cui *x* può assumere i seguenti valori:

- T, modalità asservita (RL1 segue il termostato esterno)
- A, modo automatico
- M, il relé è forzato attivo.

Se non ricordiamo in quale modalità sta lavorando il circuito, possiamo chiederglielo inviando un SMS contenente il testo *FUN?* a seguito del quale si ottiene un SMS di risposta contenente la

modalità attiva. Notate che la modalità viene memorizzata e quindi eventuali black-out non la alterano.

Passiamo adesso ai comandi riguardanti l'attività del relé, il quale può essere gestito indipendentemente dalla modalità di funzionamento scelta; in particolare, in ogni momento è possibile attivarlo (se a riposo) o disattivarlo (se è attivo) con i comandi *OUT:ON* e *OUT:OFF*. È possibile richiedere in qualsiasi momento lo stato del relé tramite il comando *OUT?*, al quale il telecomando risponde inviando un SMS. Il telecomando può memorizzare lo stato del relé e ripristinarlo dopo un eventuale black-out. Tale funzione (disabilitata di default) può essere attivata con i comandi *RIP1* e *RIP0*. Al comando *RIP?*, il telecomando restituisce un SMS indicante l'attuale condizione del ripristino espressa in forma numerica, ossia 1 se il ripristino è attivo e 0 se, invece, non lo è. Bene, siamo giunti ai comandi che permettono di definire i messaggi di notifica inviati dal telecomando. Il sistema prevede l'invio di un SMS personalizzabile ogni volta che viene alimentato; per impostazione predefinita (e dopo ogni reset totale) la funzione è disattivata; può essere impostata tramite il comando *AVVx*, in cui *x* vale 1 se la si vuole attivare e 0 se la si vuole disattivare.

Nel caso la funzione sia attiva, il testo del messaggio inviato all'accensione (*SYSTEM STARTUP*) è il messaggio predefinito) si imposta con il comando *TSU:xxxxxxxxxx*, dove al posto delle *x* va scritto il testo, restando nel limite di 100 caratteri (compresi gli spazi). Il messaggio di notifica dell'allarme sull'ingresso optoisolato viene modificato dal comando *TIN1A:xxxxxxxxxx*,

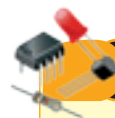
che segue le stesse regole viste per il comando *TSU*.

Quanto ai messaggi inviati in caso di allarme termico, è possibile definire singolarmente quelli derivanti dal superamento della soglia superiore e dall'abbassamento della temperatura oltre la soglia inferiore, con i comandi *THI:xxxxxxxx* e *TLO:xxxxxxxx* per quello dovuto alla soglia minima.

In ogni momento, in possesso della password possiamo ripristinare le impostazioni predefinite del sistema e cancellare tutti i numeri telefonici memorizzati tramite il comando *RES:pwd*, nel quale *pwd* è la password corrente.

REALIZZAZIONE PRATICA

Per il termostato GSM abbiamo previsto un circuito stampato a doppia faccia a fori metallizzati, realizzabile per fotoincisione (le tracce sono nel nostro sito www.elettronica.in.it); una volta procurata la basetta, dovete iniziare il montaggio con i componenti



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Questo controllo remoto (cod. TDG139) è anche disponibile già montato e collaudato al prezzo di 96,00 Euro. Il dispositivo comprende anche l'antenna a stilo e la sonda DS18B20. Sono in ogni caso disponibili separatamente il GSM SIM900 (Euro 48,00), il modulo con GSM TDGSM_900 (Euro 60,00), il DS18B20 (Euro 7,00) e l'antenna ANTSMAGSM (Euro 8,00). Tutti i prezzi si intendono 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>

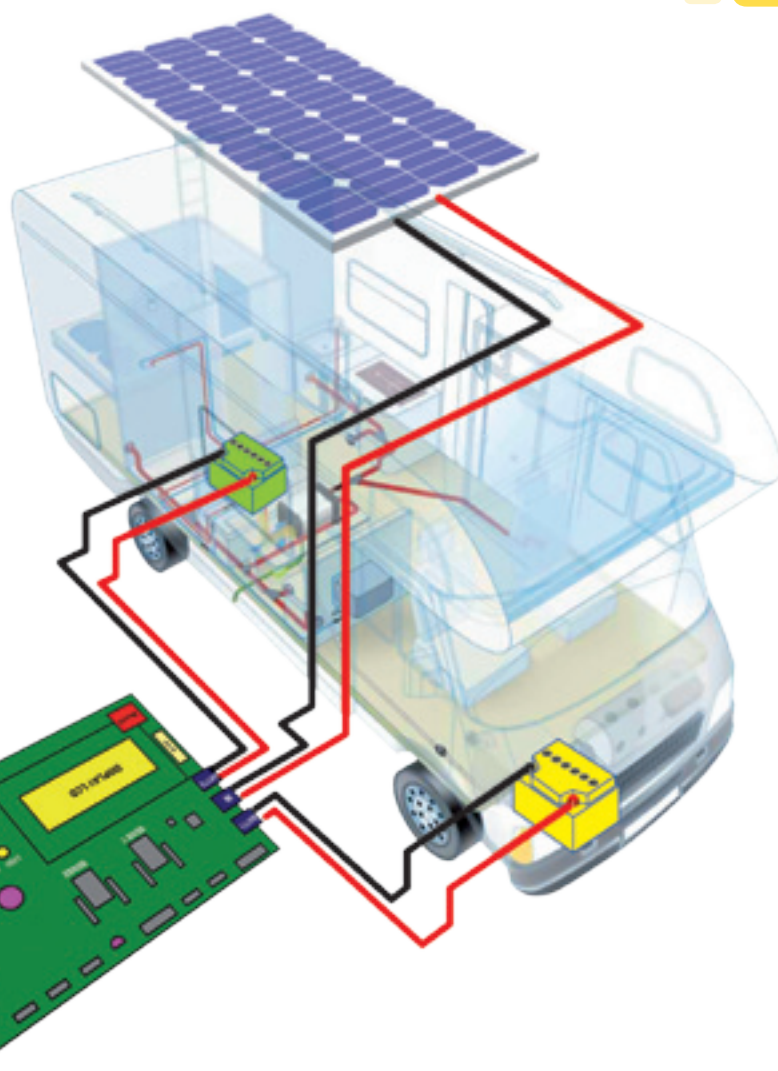
REGOLATORE DI CARICA...



... PER PANNELLI FOTOVOLTAICI

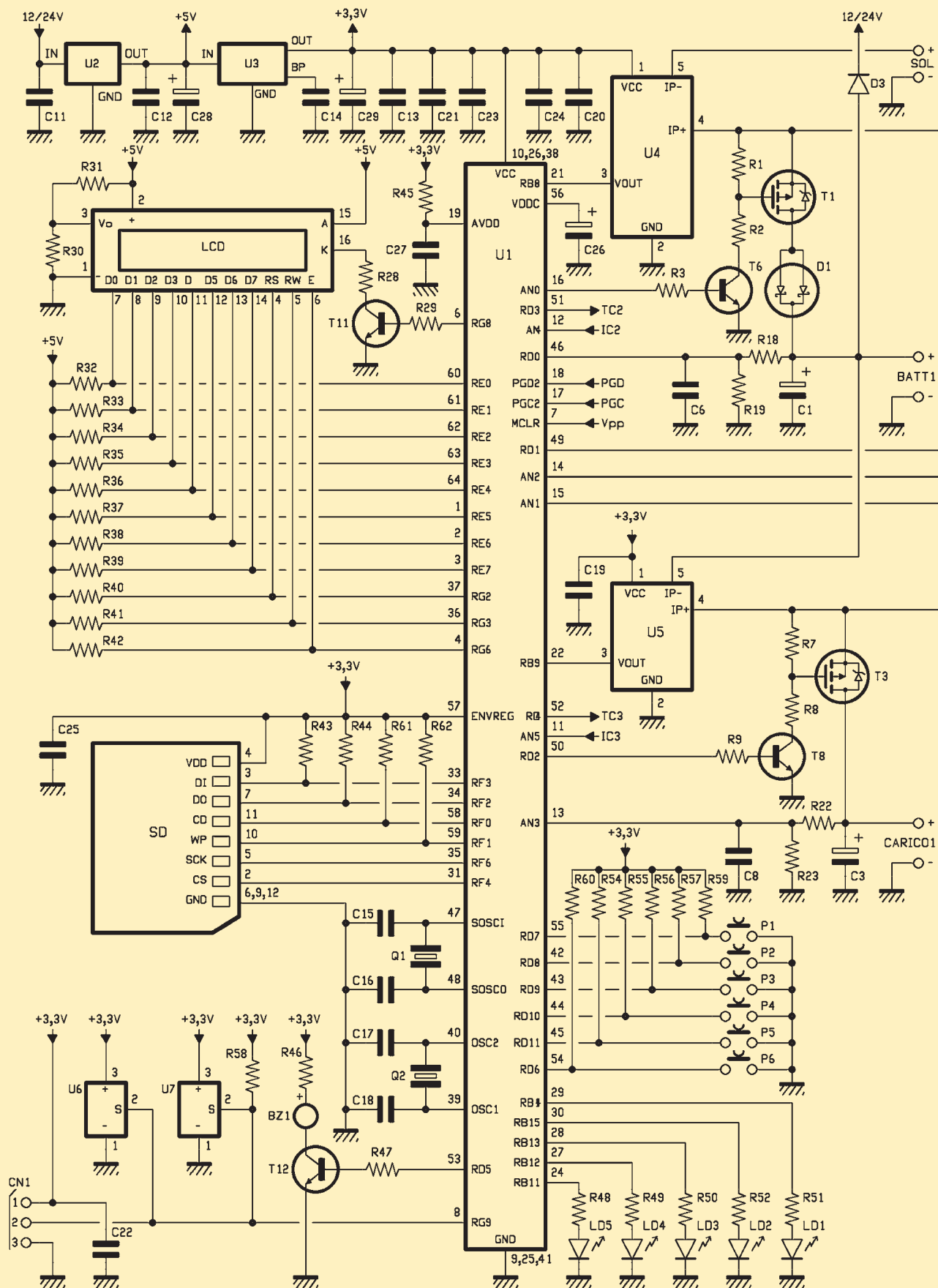
di LUCA PASCARELLA

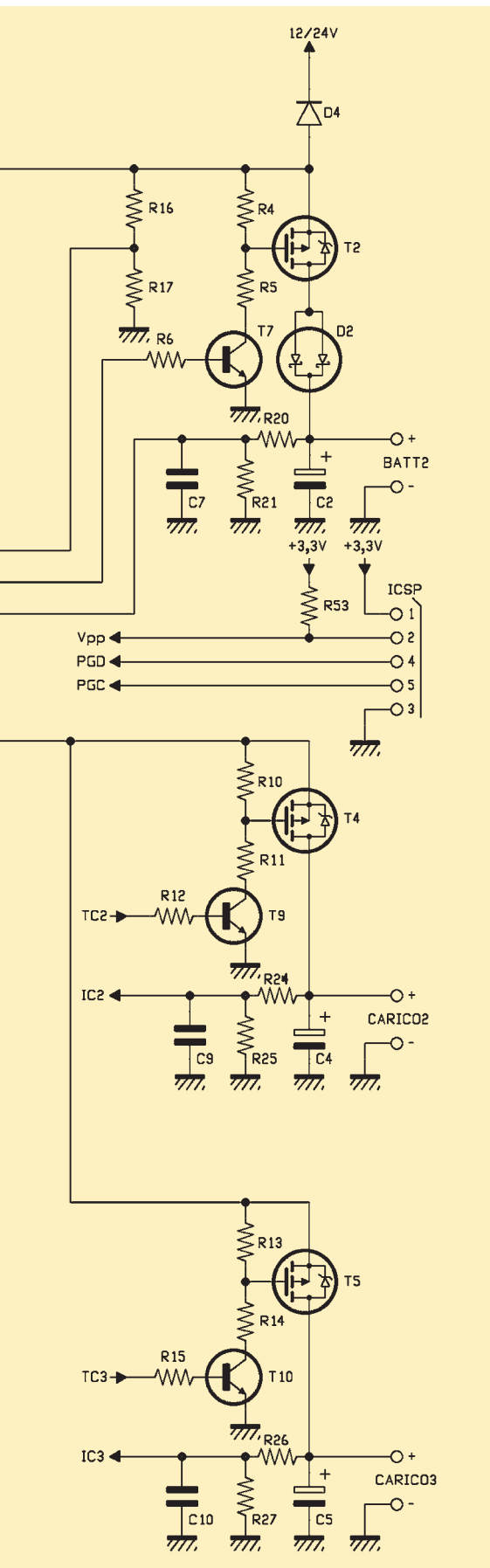
Unità programmabile per la carica ad energia solare di due batterie al piombo. Adatto per prolungare la vita delle batterie, ad esempio di un camper. Prima puntata.



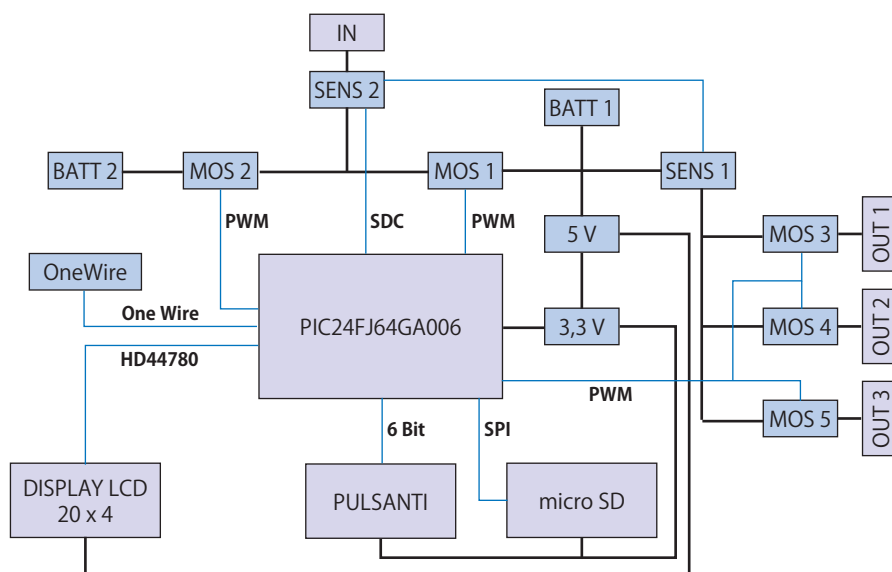
Quando si parla delle energie alternative, quali ad esempio fotovoltaico, eolico, idroelettrico, biomasse, le aspettative sembrano buone ed interessanti, sebbene ancora i costi correlati al loro utilizzo non siano competitivi con quelli delle fonti fossili; dalla loro, però, le rinnovabili hanno l'impatto ambientale, che è quasi nullo in termini di inquinamento, oltre che la possibilità di produrre l'energia localmente senza far ricorso a grandi linee elettriche

per il trasporto. Almeno per il solare fotovoltaico, la progressiva riduzione dei costi d'esercizio oggi sta aprendo nuovi spiragli e consente la nascita e la diffusione di impianti sia domestici (fino a 5 kW) sia destinati a servire interi stabili o piccole aree urbane. Il principale vincolo del fotovoltaico nei piccoli impianti da casa è nel fatto che per risparmiare sui pannelli solari conviene scegliere di potenza minore di quella che occorre ed accumulare l'elettricità





Schema a blocchi



prodotta in batterie, capaci poi di fornire tutta la corrente che serve; una simile configurazione si dimostra utile anche perché permette l'accumulo di energia prodotta in eccesso durante il giorno e il riutilizzo di notte, quando non è possibile ricavare elettricità dal sole. Nel settore del fotovoltaico, la ricerca e la grande domanda di mercato stanno portando alla nascita di soluzioni economiche ed affidabili con tempi di vita media che superano abbondantemente i 10 anni, consentendo all'utente di recuperare nel corso del tempo la spesa iniziale.

Il progetto di cui ci occupiamo in queste pagine è quello di un regolatore di carica intelligente, in grado di mantenere in carica ottimale due batterie al piombo mediante un impianto di pannelli fotovoltaici; il tutto può trovare applicazione ad esempio in camper, autocaravan o in tutti quei mezzi in cui può essere utile caricare e gestire due batterie separatamente.

L'esempio del camper è l'applicazione tipica del dispositivo qui proposto, perché siccome nei mezzi a motore lo spunto iniziale

richiesto dall'avviamento può essere consistente, tipicamente si usa montare due accumulatori: uno per l'avviamento e l'impianto del veicolo ed uno che fornisce elettricità ai servizi del caravan vero e proprio, ossia luce, pompa dell'acqua, inverter per la TV, ecc. Dato che l'energia elettrica è un bene prezioso, spesso nelle aree di sosta riservate ai camper viene fatta pagare a parte, quindi la possibilità di acquistare pannelli fotovoltaici di media potenza ad un costo contenuto spinge oggi molti amanti della vacanza in caravan a valutare l'idea di realizzare un impianto fotovoltaico mobile. Ciò si dimostra particolarmente utile nelle giornate soleggiate d'estate, quando l'energia fornita dal sole è abbondante ed un pannello da 200 W fornisce quanto è richiesto per ricaricare la batteria dei servizi e consentire l'utilizzo di PC, TV, telefoni e quant'altro di tecnologico. Così facendo, le batterie dei servizi sono soggette a molti cicli di carica e scarica, quindi abbinare al generatore fotovoltaico un buon caricabatteria diviene un "must". Il problema della scarica, anche se risolto per la batteria dei

Tabella 1 - Segnalazioni dei LED.

# LED	Tipologia	Spento	Acceso	Lampeggiante
LD1	Batteria 1	Batteria 1 scollegata.	Batteria 1 in carica.	Batteria 1 in mantenimento. Avvio carica Batteria 2.
LD2	Batteria 2	Batteria 2 scollegata.	Batteria 2 in carica.	Lampeggio veloce batteria 2 in mantenimento. Lampeggio lento batteria 2 in attesa di ricarica.
LD3	Out 1	Carica 1 scollegato.	Carica 1 attivo.	Carico 1 programmato o temporizzato.
LD4	Out 2	Carica 2 scollegato.	Carica 2 attivo.	Carico 2 programmato o temporizzato.
LD5	Out 3	Carica 3 scollegato.	Carica 3 attivo.	Carico 3 programmato o temporizzato.

servizi, rimane per la batteria ordinaria, la quale nelle prolungate soste è soggetta ai fenomeni tipici dell'autoscarica, che il forte caldo estivo e il freddo intenso dei periodi invernali accelerano. Per ovviare a ciò, in commercio esistono anche dei deviatori di carica che consentono di riservare parte della corrente usata per caricare la batteria dei servizi alla ricarica alla batteria ordinaria; tali "aggeggi" spesso non sono null'altro che diodi a bassa caduta di tensione, che consentono la carica delle batterie ma non permettono il ritorno dell'energia immagazzinata nella batteria ordinaria verso quella dei servizi. Con questa configurazione il regolatore di carica vedrà solo un parallelo tra due batterie, probabilmente di capacità e tipologia diverse; ma caricare due batterie differenti in parallelo è quanto di peggio si possa immaginare, perché inevitabilmente le due hanno curve di carica diverse, perciò quando quella di minor capacità si è caricata, l'altra ancora non è carica. Il nostro regolatore di carica non è limitato solo all'uso nei camper, perché si potrebbe sfruttare la possibilità della doppia batteria anche in ambiente domestico: ad esempio, se volessimo costruire un lampione solare per il nostro giardino, al circuito più semplice (composto da un modulo fotovoltaico, una lampada a basso consumo ed un regolatore di carica per la ricarica della batteria) potremmo aggiungere un secondo accumulatore per disporre di un surplus di energia nelle giornate meno soleggiate o in condizioni d'emergenza.

IL CIRCUITO

Il nostro regolatore di carica è composto da una scheda a microcontrollore e consta di un hardware schematizzato nello schema a blocchi nella **Fig. 1** e meglio descritto nello schema elettrico che trovate in queste pagine. Lo schema elettrico può essere diviso in due sezioni principali: quella di controllo e quella di potenza. Iniziamo la descrizione proprio da quest'ultimo punto. Come già accennato, il circuito è in grado di caricare separatamente due batterie al piombo: quella primaria o dei servizi, e la batteria secondaria (ordinaria). Rispetto a quest'ultima, quella per i servizi è connessa tramite MOSFET di potenza anche a tre uscite che possono funzionare indipendentemente da interruttori controllati, in serie ai quali si possono collegare dei carichi generici con la certezza che il firmware interrompa l'erogazione di corrente prima di raggiun-

gere il limite di scarica completa dell'accumulatore.

La batteria ordinaria, invece, è completamente separata dal circuito e la sua energia non viene utilizzata né per fornire corrente in uscita ai carichi, né per alimentare il circuito nelle ore notturne (a ciò provvedono la batteria primaria o il modulo fotovoltaico). I moduli fotovoltaici possono essere collegati in serie o in parallelo, in modo da fornire in ingresso al connettore siglato SOL una tensione di 12 V o 24 V; la corrente massima tollerabile dal circuito è 100 A, valore fortemente dipendente dal tipo di MOSFET impiegato e dalle dimensioni del dissipatore associato. La corrente disponibile all'ingresso attraversa direttamente il primo dei due sensori di corrente, capace di misurare intensità di oltre 150 A, con una sensibilità di appena 13 mV/A. Il segnale di tensione in uscita dal sensore U4 viene campionato, insieme al segnale fornito dal partitore di tensione formato da R16 ed R17 (necessario per individuare



Il microcontrollore prende posto sotto il display LCD, quest'ultimo montato mediante un pin-strip.

Caratteristiche del PIC24FJ64GA006

Il microcontrollore scelto per questo progetto, è il "fratello maggiore" del PIC impiegato nel lettore MP3 indirizzabile comparso sulla rivista numero 146; si tratta del PIC24FJ64GA006, cioè un componente appartenente alla famiglia dei PIC 24F della Microchip (è un micro basato su word a 16 bit). Il suo set di istruzioni,

come per tutti i PIC24F, è ottimizzato per il linguaggio C di alto livello; il compilatore fornito dalla Microchip è il C30 ed è adatto ai PIC delle famiglie: 24F, 24H, 30F e 33F. Le caratteristiche principali di questo PIC rispecchiano quelle del PIC24FJ64GA002 e sono visibili nella **Tabella 2**. La scelta di tale microcontrollore nasce

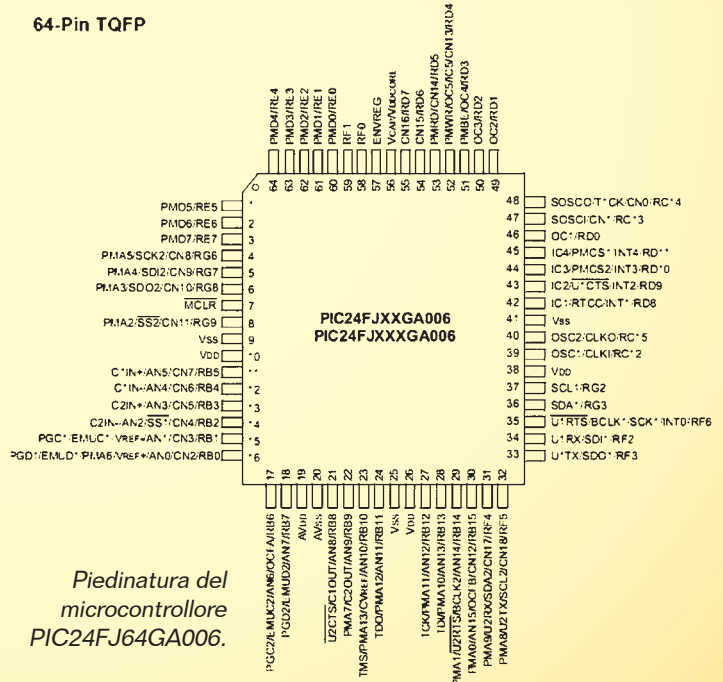
dalla possibilità di usare gli ingressi analogici con l'apposito convertitore ADC SAR (SAR sta per *Successive Approximation*) ed i 5 PWM (*Pulse Width Modulation*) in uscita per pilotare i MOSFET del circuito. Un'altra caratteristica ben apprezzata di questo microcontrollore è il modulo RTCC (*Real-Time Clock and Calendar*) che ha consentito di implementare l'orologio

e la gestione degli eventi temporizzati senza l'impiego di RTC esterni. Per quanto riguarda le prestazioni dinamiche, il nostro PIC, come molti altri della categoria, è capace di funzionare a 32 MHz con un MIPS pari a 16, semplicemente fornendo un'onda quadra in ingresso pari ad 8 MHz e lasciando l'amplificazione al PLL interno.

Tabella 2 - Caratteristiche tecniche del PIC24FJ64GA006.

Architettura	16 bit
Velocità di calcolo CPU (MIPS)	16
Tipo di memoria	Flash
Program Memory (kB)	64
RAM (byte)	8192
Temperatura di funzionamento °C	da -40 a 125
Tensione di alimentazione (V)	da 2 a 3,6
Pin di I/O	53
Numero di Pin	64
System Management Features	BOR
Oscillatori interni	8 MHz, 32 kHz
Periferiche di comunicazione Digitale	2-UART, 2-SPI, 2-I2C
Periferiche Analogiche	1-A/D 16x10-bit @ 500(kcps)
Comparatori	2
Capture/Compare/PWM	5/5
Timer	5 x 16 bit
Porta parallela	PMP
Hardware RTCC	Yes

64-Pin TQFP



la presenza di tensione elettrica dal modulo fotovoltaico e la modalità di collegamento 12 V o 24 V) per determinare energia e potenza prodotte. Notate il collegamento dei diodi D3 (connesso al positivo della batteria dei servizi) e D4 (collegato all'uscita di U4) che permettono di alimentare il circuito con il pannello fotovoltaico o la batteria dei servizi, mantenendo queste due fonti separate l'una dall'altra. Infine, la corrente in uscita da U4 è resa disponibile ai MOSFET di potenza T1 e T2, impiegati rispettivamente per alimentare e controllare le due batterie. I circuiti a valle ed a monte di T1 e T2 sono simmetrici, quindi ci limiteremo a descrivere solo

il primo. I transistor impiegati come interruttori controllati sono MOSFET a canale P (SUP75P03) con una corrente nominale di 75 A ed una Vgs di +/-20 V. La Vgs (differenza di potenziale tra Gate e Source) è la tensione di attivazione che consente di portare il transistor in conduzione. È evidente che se la Vgs massima è +/-20 V ed il circuito può funzionare con tensioni che superano, in condizioni di funzionamento normale, 24 V (la tensione dei pannelli solari da 24 V a circuito aperto può arrivare a 40 V) bisogna limitare la tensione di Gate con un partitore di tensione, nel nostro caso formato da R1, R2. Il MOSFET è comandato dal transistor T6 (alimentato tramite

la resistenza R3). Per evitare che la tensione della batteria ritorni indietro, nel circuito è stato aggiunto il diodo (D1 per T1 e D2 per T2); siccome lavoriamo a bassa tensione, abbiamo preferito uno Schottky, dato che la sua caduta diretta è dell'ordine di 0,3 V, contro gli almeno 0,7 V di un tradizionale diodo in silicio. Facciamo notare che in realtà sia D1 che D2 sono doppi diodi a catodo comune, di cui ognuno ha connessi i diodi in parallelo; ciò ci consente di far transitare correnti molto elevate, mantenendo una minima perdita di potenza. A valle del diodo D1, quindi direttamente connesso alla batteria, troviamo un ulteriore partitore di tensione formato da R18, R19 e

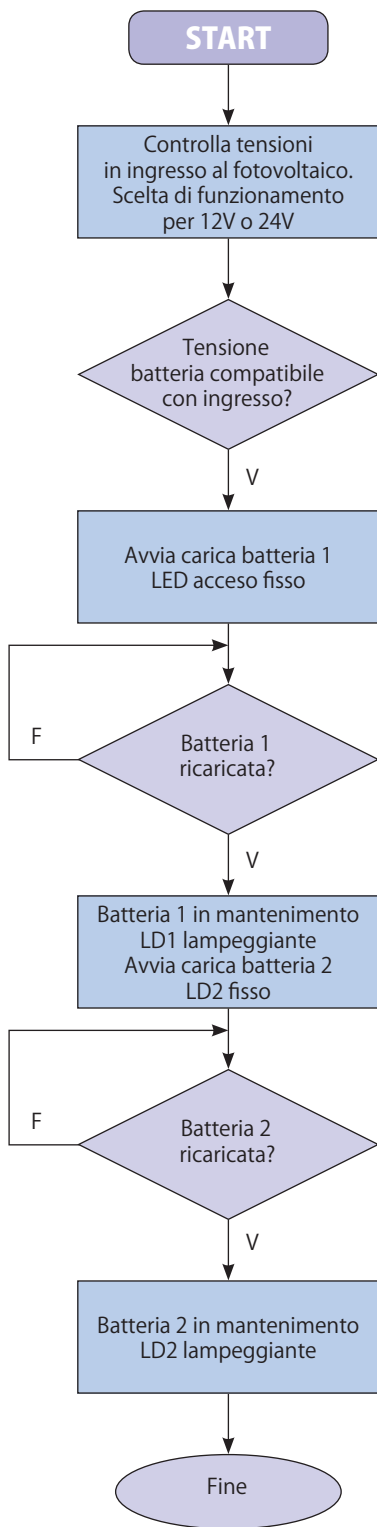


Fig. 2 - Diagramma di flusso implementato per la carica separata delle due batterie.

attenuare i disturbi. Per quanto riguarda i tre stadi d'uscita, il discorso rimane molto simile al precedente: la corrente in uscita attraversa il sensore U5 e raggiunge i tre MOSFET di potenza; qui, però, non abbiamo previsto alcun diodo di blocco della corrente, in quanto si ritiene che il carico, quando i MOSFET sono interdetti, non generi tensione che possa ritornare ad essi. Con il circuito così costruito, il microcontrollore ha tutti i dati necessari per un corretto controllo del funzionamento; per consentire l'interazione e la programmazione da parte dell'utente, abbiamo preferito aggiungere un display LCD (si tratta del tipico 4 righe per 20 caratteri dotato di controller compatibile HD44780, collegato in configurazione ad 8 bit) e 6 pulsanti che funzionano da navigatore per scorrere nel menu. Inoltre al circuito sono collegati due sensori di temperatura OneWire: si tratta dei DS18B20, da noi già impiegati in molti altri progetti. Completa l'insieme una microSD per memorizzare il log degli eventi. L'ultima nota sullo schema elettrico riguarda gli stabilizzatori di tensione impiegati per alimentare rispettivamente il display LCD a 5 V (U2) ed il resto della logica di controllo a 3,3 V (U3). U2 è uno stabilizzatore ad alta tensione di ingresso (fino a 48

V) capace di fornire in uscita 70 mA di corrente a 5 V. U3, invece, è un modello già ampiamente impiegato in altri progetti ed è in grado di fornire 3,3 V con una corrente massima di 100 mA. I condensatori C11, C12, C13, C14, C28 e C29 servono per fornire la giusta energia nei picchi di corrente. I condensatori C21, C22, C23, C24 e C25 invece sono impiegati per filtrare i disturbi di commutazione dei componenti di potenza, evitando che si propaghino sulle linee digitali di comunicazione che attraversano tutto il circuito.

COME FUNZIONA

Il principio di funzionamento del circuito è molto semplice: il circuito riceve in ingresso l'energia elettrica prodotta dai moduli fotovoltaici, che possono essere da 12 o 24 V. Il microcontrollore di bordo rileva questa tensione ed adatta le soglie di carica automaticamente. Anche le batterie vengono preventivamente controllate e nel caso in cui la tensione del pannello risulti essere troppo grande rispetto a quella delle batterie, queste ultime, per sicurezza, non vengono caricate. Tale situazione può verificarsi per due motivi differenti: il primo è un'errata configurazione (ad esempio i pannelli hanno un potenziale di 24 V mentre invece le batterie sono a 12 V); il secondo motivo potrebbe essere che le batterie sono troppo scariche o potenzialmente in corto e quindi è sempre opportuno tenere i circuiti separati ed evitare di far circolare grosse correnti in maniera imprevedibile ed indesiderata. Il funzionamento tipico prevede entrambe le batterie collegate ai morsetti *Batt 1* e *Batt 2* ed il pannello solare collegato all'ingresso *IN*. Quando entrambe le batterie sono collegate correttamente

C6; il ruolo del partitore è ottenere un riferimento di tensione che consenta di monitorare lo stato della batteria cui è connesso. Il condensatore da 100 nF funziona da filtro passa-basso, utile ad

ai due ingressi *Batt1* e *Batt2*, il sistema carica alla massima potenza la batteria primaria, quindi il MOSFET di controllo è in saturazione (LD1 acceso fisso) e tutta l'energia del pannello può essere trasferita alla batteria fino al raggiungimento della prima soglia di tensione, oltre la quale la modalità di carica passa allo stato di mantenimento; LD1 lampeggia velocemente ed il PWM viene regolato in modo che il valore di retroazione letto dall'ADC faccia sempre in modo di tenere la batteria alla tensione nominale.

Quando la batteria primaria passa alla modalità di mantenimento, vuol dire che la potenza prodotta dal pannello è in eccesso e si può avviare la carica della batteria secondaria che rispetta lo stesso meccanismo di carica, ma con le seguenti funzioni associate alla spia luminosa LD2: acceso fisso vuol dire batteria in carica, mentre lampeggiante velocemente indica batteria in mantenimento; infine, lampeggiante lentamente significa batteria presente ma non in carica. Quando i LED 1 e 2 sono spenti, vuol dire che il sistema non ha rilevato la presenza delle batterie e quindi non è stato in grado di riconoscere la loro tensione di esercizio oppure il pannello solare non sta fornendo sufficiente energia per ricaricare le batterie.

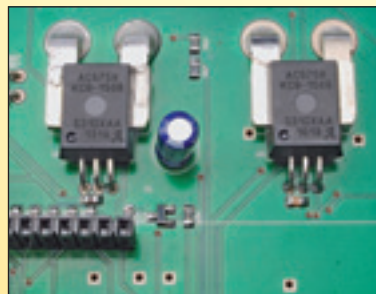
I tre MOSFET di uscita (T3, T4, T5) possono funzionare come i transistor di ingresso, ovvero, sia in saturazione quindi da interruttori che controllati con tecnica PWM. Inoltre l'attivazione dei carichi è abilitata solo quando la batteria non raggiunge il limite minimo di tensione oltre il quale si rischiano danni permanenti alle piastre di piombo.

I tre carichi possono essere abilitati separatamente e da 3 eventi:

I sensori di corrente dell'Allegro Semiconductors sono ad effetto Hall. L'effetto Hall in Fisica è un fenomeno che consente la creazione di una differenza di potenziale (definita come *potenziale di Hall*, dal fisico che per primo ne studiò gli effetti) sulle facce opposte di un conduttore elettrico attraversato da un campo magnetico perpendicolare alla corrente elettrica che scorre all'interno del sensore stesso. L'effetto Hall è stato scoperto nel 1879, ma solo recentemente l'abbiamo visto impiegato nell'elettronica consumer. Le sue applicazioni sono svariate: si va dai sensori di corrente a quelli di prossimità magnetici, fino alle bussole digitali in grado di percepire la presenza e l'orientamento del campo magnetico terrestre. L'Allegro Semiconductors produce una vasta gamma di sensori ad effetto Hall, tra cui l'ACS758KCB (Fig. 8) da noi impiegato; questo componente è stato un'ottima scelta rispetto al solito sistema a shunt per misurare le correnti elettriche, in virtù della bassissima resistenza elettrica mostrata ai suoi capi. In genere il metodo più semplice per misurare una corrente elettrica consiste nello sfruttare la legge di Ohm ($V = R \times i$): rilevando

la caduta di tensione (V) ai capi di un resistore percorso dalla corrente che ci interessa misurare, di intensità i , nota la sua resistenza R , otteniamo un'informazione sull'intensità. È evidente che per ottenere buone misure bisogna disporre di tensioni rilevanti, ma ciò implica di far crescere R ; tuttavia una resistenza troppo grande comporta una perdita energetica notevole soprattutto per circuiti, come i regolatori di carica per fotovoltaico, dove le tensioni in gioco sono molto basse. Ecco perché nel nostro caso ci siamo orientati verso la soluzione ad effetto di Hall.

Tra le doti del ACS758KCB, emerge la possibilità di misurare correnti sia in CC che in CA, purché con una frequenza inferiore ai 120 kHz. Inoltre l'impiego di componenti esterni è limitato ai soli condensatori di disaccoppiamento e dell'apposito ADC. L'alimentazione del sensore può essere a 3,3 V o a 5 V, quindi adattabile in base alle soglie del convertitore ADC. A riposo, la tensione presente sul pin *Vout* è pari alla metà di quella di alimentazione e può aumentare o diminuire in proporzione alla corrente che fluisce nel sensore in uno o nell'altro verso.



I sensori di corrente montati; notate i robusti piedini della linea di potenza.

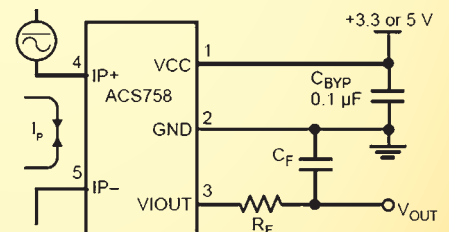


Fig. 8 - Diagramma dei pin del sensore di corrente ad effetto Hall ACS758KCB.

il primo è il funzionamento manuale (tramite display è possibile accendere o spegnere il MOS) il secondo è l'attivazione programmata in Ore e Minuti ed infine c'è l'attivazione temporizzata, che consente di accendere e spegnere il carico per un intervallo di tempo prestabilito.

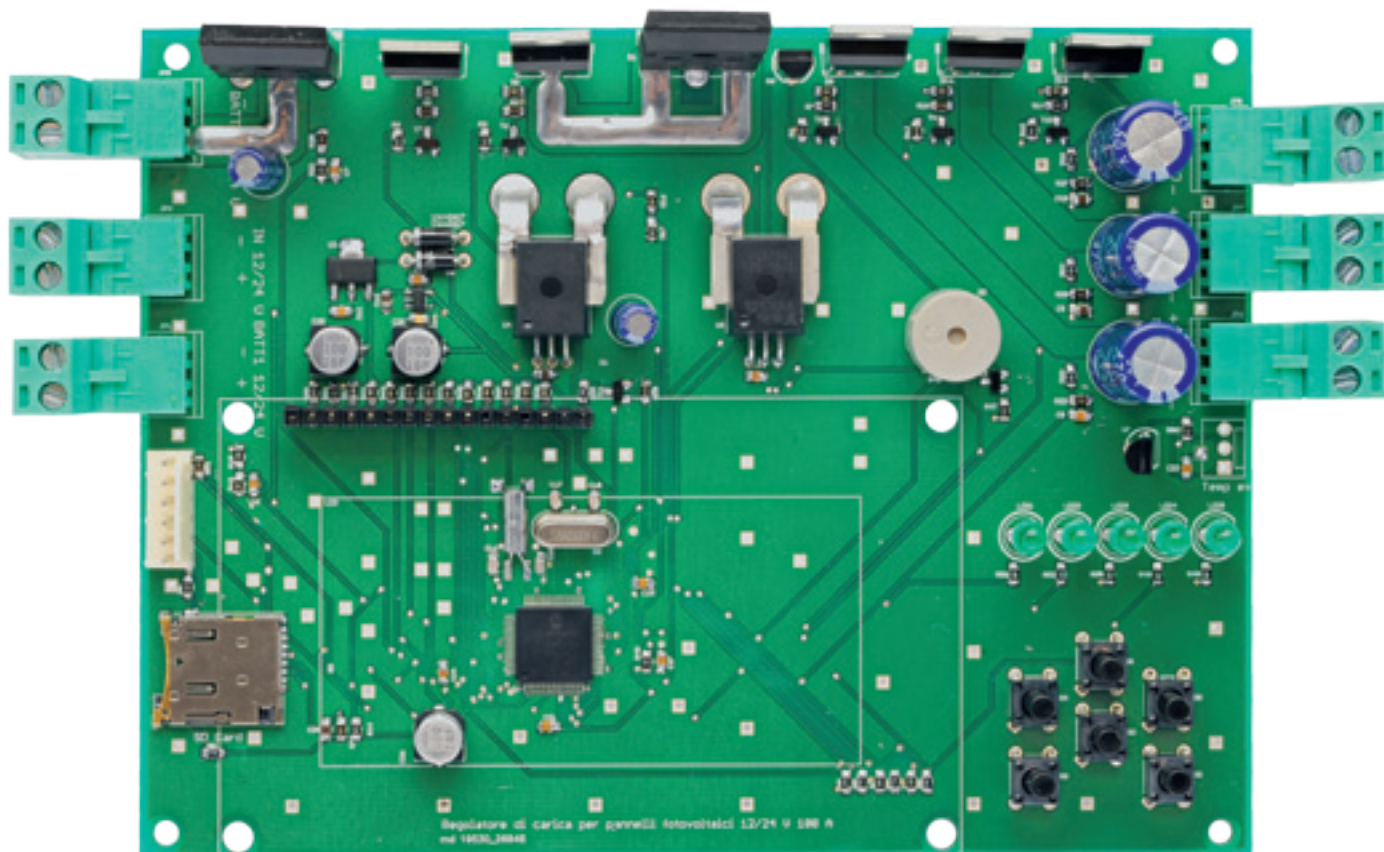
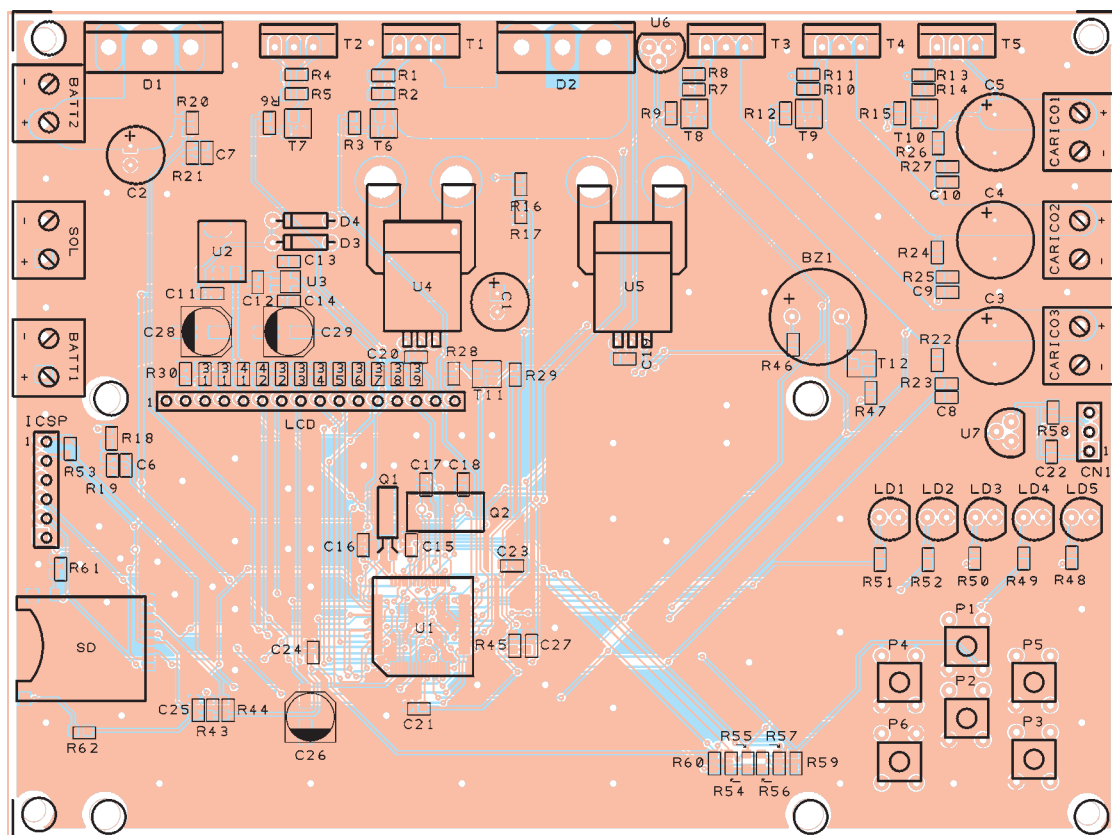
Le informazioni fornite dai cinque LED presenti nel dispositivo sono riassunte nella **Tabella 1**. Nella Fig. 2 potete vedere, tramite il flow-chart semplificato,

l'approccio logico alla gestione della carica delle due batterie.

REALIZZAZIONE PRATICA

Nel disegnare lo schema elettrico ed il circuito stampato di questo regolatore di carica è stata posta attenzione alla disposizione ottimale dei componenti; in particolare, i componenti di potenza sono stati disposti in modo da dissipare bene il calore da essi prodotto (MOSFET e diodi Schottky) mentre invece una dispo-

[piano di **MONTAGGIO**]



Elenco Componenti:

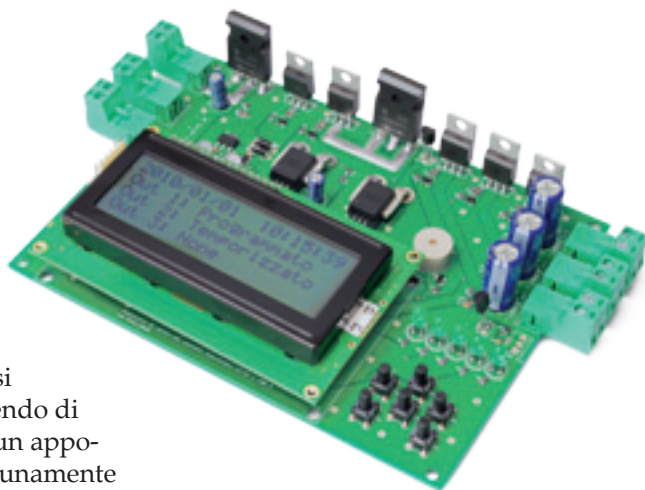
R1, R4, R7, R10, R13: 10 kohm (0805)
R2, R3, R5, R6, R8, R9, R11, R12, R14, R15, R29, R31, R47, R58: 4,7 kohm (0805)
R16: 560 kohm 1% (0805)
R17: 47 kohm 1% (0805)
R18, R20: 1 Mohm 1% (0805)
R19, R21: 120 kohm 1% (0805)
R22, R24, R26: 100 kohm 1% (0805)
R23, R25, R27: 12 kohm 1% (0805)
R28: 22 ohm (0805)
R30: 470 ohm (0805)
R32, R33, R34, R35, R36, R37, R38, R39, R40, R41, R42: 10 kohm (0805)
R43, R44, R53: 47 kohm (0805)
R45, R46: 10 ohm (0805)
R48, R49, R50, R51, R52: 1 kohm (0805)
R54, R55, R56, R57, R59, R60, R61, R62: 470 kohm (0805)
C1, C2: 100 μ F 63 VL elettrolitico
C3, C4, C5: 1000 μ F 63 VL elettrolitico
C6, C7, C8, C9, C10, C11, C12, C13: 100 nF multistrato (0805)
C14: 470 pF ceramico (0805)
C15, C16, C17, C18: 22 pF ceramico (0805)
C19, C20, C21, C22, C23, C24, C25, C27: 100 nF multistrato (0805)
C26: 10 μ F 63 VL elettrolitico (D)
C28, C29: 10 μ F 63 VL elettrolitico (D)
D1, D2: MBR6060PT
D3, D4: 1N4007
BZ1: Buzzer con elettronica
P1, P2, P3, P4, P5, P6: Microswitch
T1, T2, T3, T4, T5: SUP75P03
T6, T7, T8, T9, T10, T11, T12: BC847
U1: PIC24FJ64GA006-I/PT (MF903)
U2: MCP1790-5002E
U3: TC1015-3.3VCT713
U4, U5: ACS758KCB-150B-PFF-T
U6, U7: DS18B20
Q1: Quarzo 32.768 kHz
Q2: Quarzo 8 MHz
LCD: Display LCD 16x2
LD1, LD2, LD3, LD4, LD5: LED 5 mm verde

Varie:

- Morsettiera 2 poli (6 pz.)
- Strip maschio 3 poli
- Strip maschio 6 poli
- Strip maschio 16 poli
- Strip femmina 16 poli
- Connettore micro sd-card
- Circuito stampato

sizione ottimale del display e dei pulsanti di comando consente una più facile interazione con l'utente. Le dimensioni, anche se non minime, consentono comunque l'installazione nei pressi della batteria, permettendo di abbinare il circuito ad un apposito contenitore opportunamente arieggiato. Anche in questo progetto abbiamo fatto un discreto uso di componentistica per montaggio superficiale, almeno per la parte di piccola potenza. Passiamo allora alla descrizione del montaggio del circuito stampato, cercando, come sempre, di fornire utili consigli per rendere semplice la realizzazione di circuiti in SMT.

Fortunatamente tale tecnologia prende sempre più piede anche nel settore hobbistico, il che ha incentivato la diffusione di strumentazione low-cost che consente una facile maneggevolezza anche con componenti che spesso sono di dimensioni inferiori al millimetro. Usando una buona stazione di saldatura con saldatore a punta fine e della lega saldante da 0,5 mm di diametro, l'assemblaggio di questo circuito, non essendoci parti estremamente critiche, è abbastanza semplice e si conduce in breve tempo. Andando per ordine, consigliamo di saldare il microcontrollore U1 dopo averlo centrato rispetto alle piazzole e fermato con una goccia di stagno, magari facilitando le operazioni con l'aiuto di una buona dose di fluxante, il quale ha il duplice pregio di favorire la fusione dello stagno e guidare il metallo fuso verso le piazzole e i terminali, impedendo che vada a cortocircuitare pin attigui. Fatto ciò, procedete con tutti gli altri componenti SMT, quali conden-



satori e resistori in formato 0805, gli stabilizzatori di tensione U2 ed U3 e i transistor T6, T7, T8, T9, T10, T11 e T12; quest'ultimi potete facilmente maneggiarli con delle opportune pinzette. Particolare attenzione deve essere tenuta quando si monta il connettore della micro SD, in quanto presenta i pin di saldatura rientranti nella sagoma; in questo caso è opportuno fissare il componente con un punto di saldatura e, solo dopo essersi assicurati dell'allineamento rispetto al PCB, procedere alla saldatura stando attenti ad evitare di tenere il saldatore troppo tempo sui connettori, in quanto quest'ultimi stando a stretto contatto con il materiale plastico con cui è costruito il connettore trasmettono facilmente il calore e rischiate di deformare l'alloggio per la card. Dopo aver montato questi elementi, che richiedono tanta calma e pazienza, si può passare alla saldatura dei componenti discreti in formato THT, i quali anche se non richiedono le cure e le attenzioni dell'SMT è opportuno maneggiare con cura.

Per completare l'insieme, occorre fissare MOSFET e diodi di potenza a un dissipatore da 1,5 °C/W, interponendo tra le loro parti metalliche e il radiatore dei foglietti di teflon grigio. Bene, per ora abbiamo terminato; nella prossima puntata concluderemo il discorso spiegando il firmware. ■



Raccontiamo l'esperienza di un gruppo di studenti dell'ITIS Facchinetti di Busto Arsizio, che ha realizzato un interessante progetto.

LA SERRA DOMOTICA

..... a cura della REDAZIONE

Una delle attività che ci sta più a cuore è mantenere un filo diretto con le scuole ad indirizzo tecnico per conoscere l'attività di sviluppo di nuove iniziative e dare spazio a progetti interessanti sotto il profilo tecnico e dell'attualità. Di recente siamo andati a curiosare in quel che fanno all'ITIS Facchinetti di Busto, e in queste pagine vogliamo parlare del lavoro svolto da alcuni studenti di questo istituto, che hanno realizzato una serra a controllo domotico gestita mediante iPhone; descrivere questo progetto è insieme l'oc-

casione per segnalare l'impegno di quanti si sono cimentati nell'opera, divulgarne la conoscenza e mostrare un utilizzo pratico di un nostro progetto: l'interfaccia a relé K8090, da noi pubblicata nel fascicolo n. 146. La cosa funziona molto semplicemente: nella serra sono stati inseriti dei dispositivi come motori per comandare l'apertura di sportelli e un illuminatore a LED, che possono venire gestiti a distanza tramite un iPhone, il quale si collega tramite LAN ad un PC sulla cui porta USB è collegata la scheda K8090; i relé di questa ultima

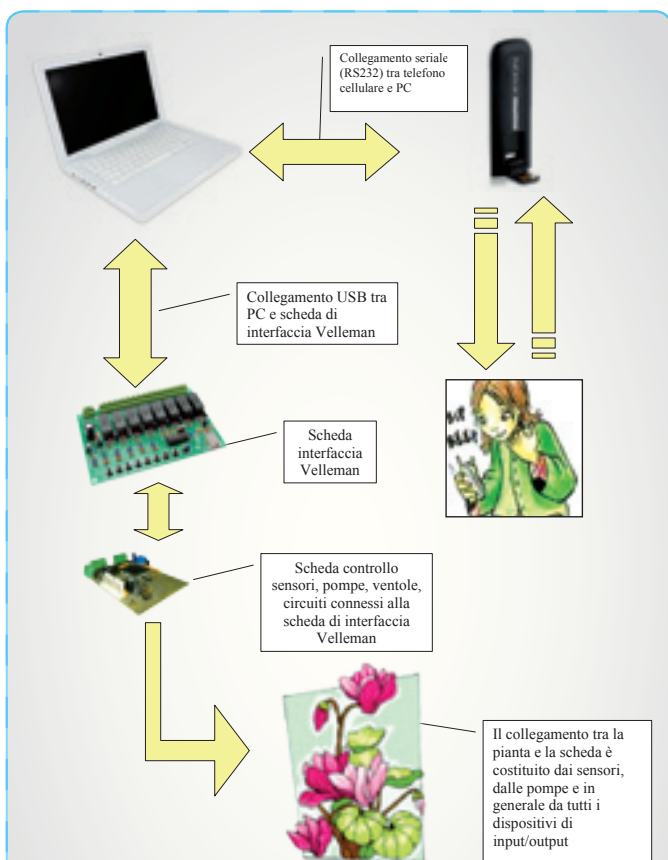
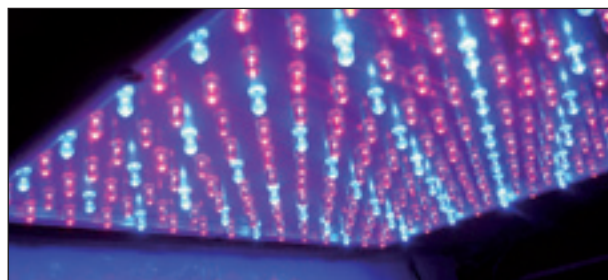


Fig. 1 - Il sistema consta di un PC che rileva i dati dai sensori della serra mediante una scheda di governo e li acquisisce tramite la K8090; questa stessa unità provvede a comandare gli attuatori locali. Lo stesso PC si interfaccia con la rete cellulare mediante una Internet Key.

attivano e disattivano i dispositivi della serra. Ma non è tutto qui: infatti siccome l'interfaccia K8090 è bidirezionale, gli studenti hanno usato il connettore di input per trasmettere al computer e da esso all'iPhone le condizioni climatiche rilevate da appositi sensori posti nella serra.



Il soffitto della serra è un tappeto di LED di vari colori.

IL PROGETTO

È finalizzato alla cura di una pianta d'appartamento, ossia a mantenere, nutrire e gestire a distanza una pianta, attraverso il servizio SMS offerto dalla rete GSM per mezzo di una normale Internet Key: in pratica possiamo interrogare via SMS il dispositivo domotico e questo ci risponderà inviandoci i parametri dell'ambiente in cui la pianta si trova; se è richiesto il nostro intervento, con un messaggio è possibile variare la luce, l'umidità ecc.

Il progetto si prefigge di offrire all'utente uno strumento completo per poter controllare lo stato delle sue piante a distanza e permette non soltanto una semplice visione passiva della situazione, ma, in caso di necessità, consente di intervenire attivamente, ad esempio per dare acqua.

Per soddisfare tutte le esigenze sono stati parametrizzati tutti i caratteri della pianta ed il software che gestisce la serra ne tiene conto allo scopo di adattare le condizioni a qualsiasi tipo di pianta. L'interfaccia del PC con la scheda per gestire ingressi e uscite di tutti i parametri controllati avviene via USB.

Questo progetto è una tipica applicazione della domotica, una tecnologia che si pone l'obiettivo di garantire una maggiore sicurezza, comfort, risparmio energetico e soprattutto di gestire a distanza elettrodomestici e altri dispositivi, avvicinandosi al concetto di "casa intelligente".

Il progetto in pratica si suddivide in due parti fondamentali:

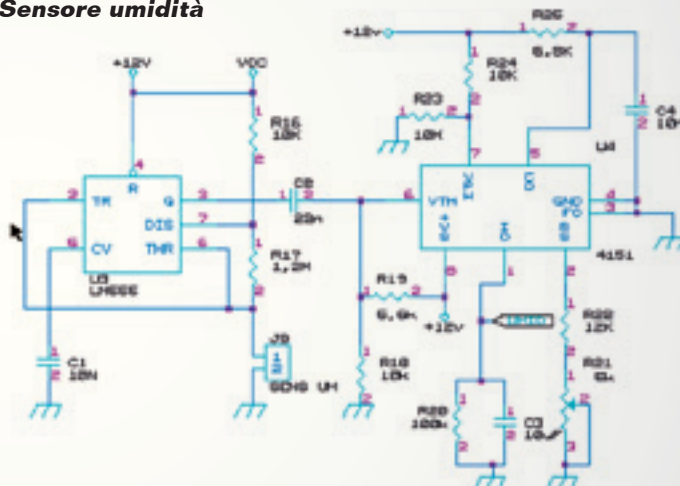
- hardware, cioè tutto lo sviluppo della parte elettronica, di tutti i circuiti e i sensori atti a controllare e a curare la pianta e l'ambiente ad essa circostante;
- software per gestire la parte hardware, l'invio e la ricezione dei dati riguardanti lo stato della serra e l'interfaccia con l'utente.

Se l'utente ha un minimo di conoscenze agrarie riguardanti gli effettivi bisogni della pianta, potrà cambiare tutti i parametri relativi all'umidità, all'illuminazione, al calore ecc. (setup personalizzabile). Per consentire queste operazioni, il software mette a disposizione un'interfaccia semplice e chiara, fa-

Gestione relé



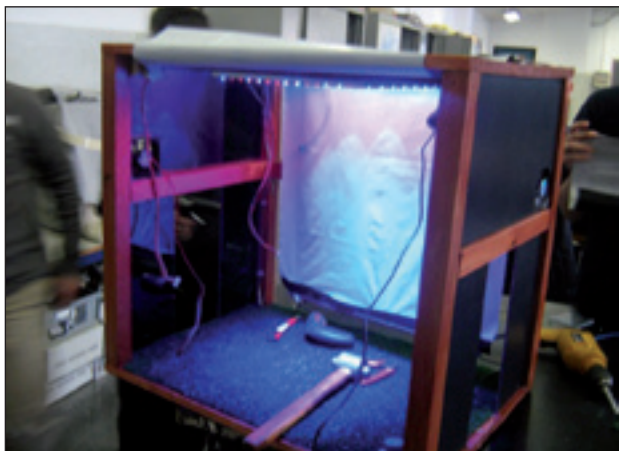
Sensore umidità



cilmente comprensibile da ogni tipologia di utente. In ogni caso sono presenti dei profili di parametri già impostati di default nel programma: *setup umido*, *setup secco*, *setup mite*.

L'HARDWARE

Il sistema proposto dagli studenti dell'ITIS è composto da due unità: una locale, situata dove si trova la serra, composta, oltre che dalla stessa serra, da un Personal Computer, una scheda K8090 Velleman,



una scheda di acquisizione e gestione dei sensori e due moduli di controllo dei motori usati per muovere gli sportelli; una remota, che sostanzialmente è il telefono cellulare. Il computer dell'unità locale coordina l'attività della K8090 mediante un software (applicazione Server) appositamente realizzato dagli studenti dell'ITIS Facchinetti e provvede all'invio degli SMS di notifica, ovvero alla ricezione dei messaggi contenenti i comandi in arrivo dall'utente, mediante una Internet Key (con all'interno SIM Card) collegata tramite porta USB. La scheda K8090

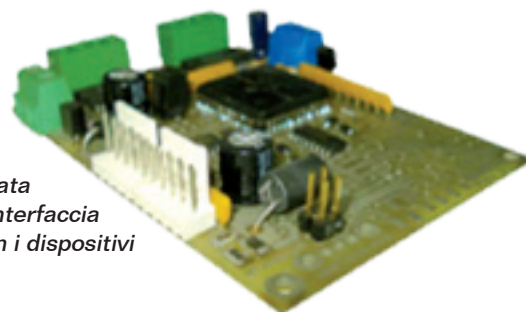
sarà in grado di acquisire diversi segnali riguardanti le condizioni della pianta e dell'ambiente ad essa circostante, attraverso ingressi digitali e analogici collegati alla scheda di governo creata dai ragazzi dell'Istituto Tecnico Facchinetti, dotata di sensori elettronici per temperatura, umidità, luce, vento ecc. La Fig. 1 schematizza il sistema realizzato.

Il controllo elettronico è attuato attraverso l'utilizzo della K8090, che, mediante connessione USB, realizza l'interfaccia tra il software residente nel PC e gli apparati presenti nella serra. Questi ultimi sono gestiti dalla scheda di governo con la quale è possibile mantenere sotto controllo i parametri ambientali (temperatura, umidità e luminosità) il cui equilibrio, indispensabile per la vita delle piante che vi verranno alloggiate, è garantito dall'attivazione di tutti quei dispositivi (ventole, sportelli di aerazione, riscaldatore, tende oscuranti, pompa per irrigazione e lampada idroponica) di cui la serra è stata dotata.

Il monitoraggio dei dati ambientali viene effettuato mediante i sensori descritti qui di seguito.

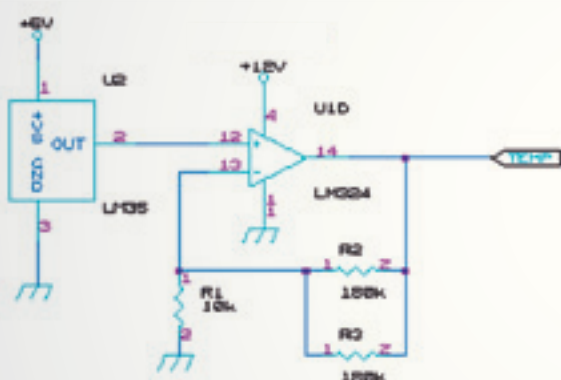
SENSORE LUMINOSITÀ

Il controllo della luminosità viene gestito senza passare attraverso la scheda K8090. Si utilizza un semplice comparatore con isteresi realizzato con

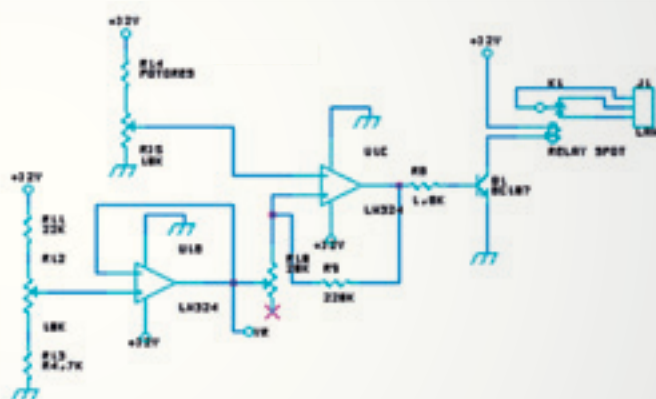


La scheda di controllo realizzata dagli studenti; si interfaccia con la K8090 e con i dispositivi della serra.

Sensore temperatura



Sensore luminosità



un LM324, che pilota direttamente il relé di comando di una lampada idroponica. Con tale circuito, attraverso la taratura dei trimmer in esso presenti, è possibile stabilire le condizioni di intervento per garantire la corretta illuminazione. La condizione di buio, altrettanto indispensabile, sarà ottenuta disattivando la lampada attraverso un secondo relé, collegato all'uscita digitale n° 5 della scheda Velleman.

al primo dei due ingressi analogici presenti sulla scheda Velleman.

AERAZIONE E LUMINOSITÀ

Alle uscite digitali 3 e 4 della K8090 sono collegati i due relé che gestiscono le ventole, di cui il primo comanda l'attivazione, mentre il secondo, con il collegamento a due tensioni diverse (5 V e 12 V), decide se farle funzionare a pieno regime oppure a velocità ridotta.

Il riscaldatore e la pompa per l'irrigazione sono comandati da ulteriori relé facenti capo alle uscite digitali 1 e 2.

L'apertura degli sportelli di areazione è stata realizzata mediante l'utilizzo di servomotori che, seppure apparentemente collegati alla scheda, sono di fatto pilotati direttamente dalle uscite PWM presenti sulla scheda Velleman. Con tali uscite si riesce a gestire in modo adeguato sia la velocità che il verso di rotazione, garantendo l'apertura e la chiusura degli sportelli quando se ne ravvisa l'esigenza. Una coppia di motori passo-passo viene utilizzata, infine, per comandare con estrema precisione il movimento delle tende oscuranti e, di conseguenza, dosare in modo preciso il grado di luminosità desiderato. Per la gestione di tali motori sono impiegate altre due schede, che utilizzano gli integrati L297 e L298 e ricevono in ingresso i comandi del clock e della direzione, generati dal computer e forniti dalle uscite digitali n° 6, 7 e 8 della scheda K8090.

SENSORI UMIDITÀ

Al secondo ingresso analogico della K8090 viene, invece, inviata la tensione proveniente dal circuito di rilevamento dell'umidità. Quest'ultimo è costituito da un astabile realizzato con l'integrato LM555, che rende in uscita un segnale di frequenza proporzionale al valore di capacità assunto dal sensore di umidità nelle diverse condizioni. La frequenza di tale segnale sarà convertita in una tensione variabile (sempre nel range 0÷5 V) dall'integrato 4151. In definitiva si otterrà, anche in questo caso, un valore di tensione proporzionale alla umidità ambientale.



SENSORE TEMPERATURA

La temperatura viene rilevata attraverso l'utilizzo di un sensore LM35 della National Semiconductors, collegato ad un circuito amplificatore: l'operazionale LM324; collegato in configurazione non invertente, questo fornisce in uscita un valore di tensione pari ad un decimo della temperatura ambientale (0,1 V per °C). Tale tensione viene inviata



PHOTO GEOTAGGING

di DAVIDE SCULLINO

Utilizziamo un minuscolo data-logger GPS, una fotocamera digitale ed un apposito software, per realizzare un'applicazione che ci permette di georeferenziare le foto del tragitto compiuto dal data-logger e creare un album fotografico cronologico di un viaggio o far apparire su una mappa foto di luoghi già visitati.

Molti di voi certo conoscono Street View, quel servizio di Google che permette, accedendo alle mappe (Google Maps) di vedere la strada così com'è stata ripresa dalle auto del famoso provider, che hanno percorso

migliaia di km in tutto il mondo e che tuttora sono in giro ad effettuare riprese per consentirci di vedere, oltre alla mappa della zona che cerchiamo, anche com'è fatta dal vivo, aiutandoci perciò quando dobbiamo trovare un indirizzo.

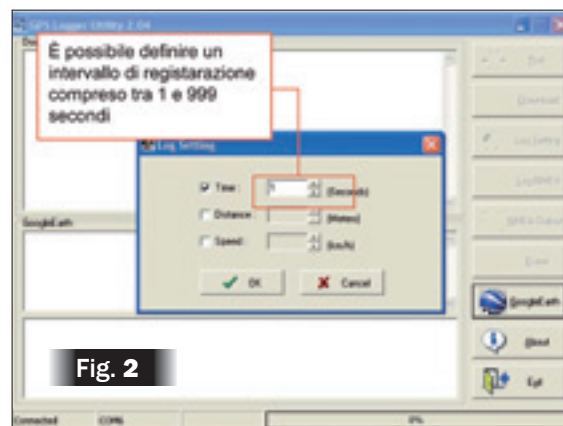
Caratteristiche del localizzatore con data-logger ML7.

Descrizione generale	Parametri
Frequenza	L1 frequency: 1575,42 MHz supporta 51 canali indipendenti
Sensibilità	-146 dBm
Frequenza di aggiornamento	1Hz
Acquisizione	Cold Start: 36 sec. Warm Start: 33 sec. Hot Start: 1 sec.
Riacquisizione	<1 sec.
Dati dinamici	Altitudine: 18.000 m (max.) Velocità: 515 m / sec. (max.) Vibrazioni: 4G (max.)
Tensione di alimentazione	5 Vdc (batteria interna ricaricabile tramite porta USB)
Consumo	Data Record Mode <60 mA Power Saving Mode <0,5 mA Tracking: 30 mA
Formato dei messaggi	NMEA0183, default 115200 Uscita: GGA,GSA,GSV,RMC Log: RMC
Formato dati georeferenziazione	WGS-84 (predefinito)
Interfaccia seriale	TTL@2,8 V
Peso	12,4 g
Dimensioni	46,9 mm (L) x 18,1mm (W) x 14,9 mm (H)
Batteria	Include 170 mA
Durata della batteria	5-6 ore
Temperatura di funzionamento	0~45°C
Temperatura di stoccaggio	-5~35°C
Umidità di funzionamento	<= 95%, senza condensa

Il principio di Street View si può applicare per vedere, disponendo di un localizzatore GPS provvisto di data-logger e di una macchina fotografica digitale, una rassegna fotografica cronologica delle zone attraversate da una persona o un veicolo che porta con sé il localizzatore, ovvero per creare foto che vengono automaticamente visualizzate facendo clic su dei punti di una mappa di Google. Questa applicazione si chiama *Photo Geotagging* ed è resa possibile dal fatto che normalmente le fotocamere digitali

memorizzano i file contenenti le fotografie che scattano associando loro la data di creazione dei file; un semplice software collega i dati scaricati dal localizzatore con data-logger alle foto, semplicemente verificando l'orario dei record relativi alle posizioni e abbinando le fotografie che risultano scattate in corrispondenza. Ad esempio se il data-logger registra una posizione con coordinate 30°, 10' e 20" di latitudine e 35°, 20' e 0" di longitudine, alle 10.05 e a detta ora è stata scattata una foto, il software collega i due dati

e georeferenzia la foto, nel senso che attribuisce al fotogramma le coordinate anzidette. Disponendo di un programma in grado di tracciare percorsi, è possibile abbinare al tracciato sulla mappa i fotogrammi, ottenendo qualcosa di simile a Street View. Lo stesso sistema vi consente, portando con voi un localizzatore GPS con data-logger ed una macchina fotografica, di documentare un viaggio creando una sequenza di foto ordinate cronologicamente. Il localizzatore può essere l'ML7 commercializzato dalla ditta Futura Elettronica di Gallarate (www.futurashop.it): si tratta di un dispositivo miniaturizzato (è il più piccolo data-logger GPS presente in commercio e misura appena 46,9 x 18,1 x 14,9 mm) che dispone di un ricevitore GPS, un microcontrollore che ne acquisisce i dati e li salva in una memoria Flash. Il ricevitore ha 51 canali indipendenti e una sensibilità di -146 dBm; la memoria Flash è da 2 MB, che consentono all'apparecchio di memorizzare tracciati con circa 60.000 punti. Grazie alle sue ridottissime dimensioni, l'ML7 può essere utilizzato per controllare gli spostamenti di persone, animali, automobili, mezzi a noleggio, furgoni, ed in qualsiasi applicazione dove è richiesta una documentazione dettagliata circa



la posizione GPS, la velocità e la distanza percorsa dal veicolo. Il percorso desunto dalle coordinate viene trasferito via USB ad un Personal Computer, mediante un apposito software per ambiente Windows che, tra le altre cose, permette di scegliere la frequenza (ogni quanto tempo effettuare la registrazione di un punto) di registrazione delle posizioni e quali dati memorizzare della stringa standard NMEA0183. Il localizzatore dispone di una batteria al litio ricaricabile interna, che viene caricata tramite l'USB quando il dispositivo è collegato al computer. Il software permette di estrarre i dati e inviarli a Google Earth, il programma di Google per visualizzare le viste da satellite e le mappe stradali. Vediamo dunque come utilizzarlo in un'applicazione di Photo Geotagging: per prima cosa bisogna procurarsi l'apparecchio e installarne il software ed i driver forniti a corredo. Effettuata la connessione, occorre installare il software che trovate nel CD, seguendo il percorso *Data Logger Series\Documents & Utility\GL24\Utility Software\ML-7\Utility\GPS Logger Utility V2.06*; nella cartella *GPS Logger Utility V2.06* fate clic sull'icona *Setup* ed installate il programma. In base alla data di acquisto del localizzatore ML7, potete trovarvi di fronte a una versione più recente del software, ma non preoccupatevi: salvo diversa indicazione nel manuale d'uso, installate il programma che trovate nella cartella chiamata *GPS Logger Utility*, indipendentemente dalla versione. Quando vi viene richiesta la password, occorre digitare *demover*. Fatto ciò, si può connetterlo all'USB mediante un cavo USB A/mini-USB. Poi bisogna

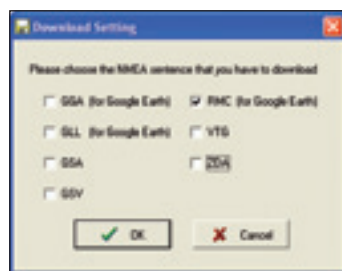


Fig. 3

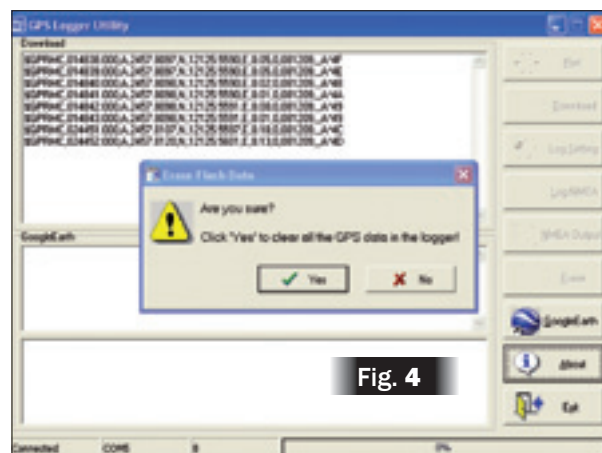


Fig. 4

aprire il programma per scaricare i dati del logger: la finestra di dialogo principale è quella mostrata nella Fig. 1; adesso si deve aprire la porta di comunicazione, facendo clic sul pulsante *Port*. Fatto ciò, appare la finestra di dialogo in cui dovete selezionare la porta in uso; se non sapete qual è, aprite *Gestione Risorse* di Windows e fate clic su *Porte COM e LPT*, quindi, dalla finestra corrispondente, visualizzate la COM virtuale assegnata all'USB per la comunicazione con l'ML7. Dal menu a tendina della finestra di dialogo del programma del logger, scegliete il numero di COM che avete desunto da *Gestione Risorse*. Aperta la COM, il programma propone la finestra di dialogo principale; la prima cosa che dovete fare per preparare il localizzatore all'uso è configurarne i parametri di lavoro, quindi fate clic sul pulsante *Log Setting* ed accedete alla relativa finestra di dialogo. In questa, spuntate l'opzione *Time* e nella casella corrispondente inserite l'intervallo che volete vi sia tra una registrazione di coordinate e la successiva (Fig. 2); potete anche imporre al localizzatore di effettuare una registrazione ogni volta che si sposta di un certo numero di metri: ciò lo fate spuntando *Distance* e specificando, nella casella accanto, i metri di spostamento percorsi

i quali l'ML7 deve registrare. Infine, potete anche impostare la velocità, raggiunta la quale viene registrata la coordinata; allo scopo spuntate *Speed* e inserite nella relativa casella il valore in km/h desiderate. Per il momento impostate solo l'intervallo, imponendo 5 secondi. Fatte le impostazioni, con il pulsante OK le memorizzate e potete procedere oltre. Adesso, dopo aver lasciato il dispositivo connesso al computer quanto basta a caricare bene la batteria (un'ora dovrebbe bastare); prendetelo con voi insieme alla macchina fotografica (della quale dovete preventivamente aver impostato l'ora in modo che combaci con quella del localizzatore, ovvero che sia l'ora esatta, altrimenti le fotografie verranno riferite in modo non corretto) e andate a fare un giro per la città o per i dintorni. Prima di iniziare a scattare le



Fig. 5



Fig. 6



Fig. 7

fotografie, spostate l'interruttore dell'ML7 in posizione ON e attendete che la spia rossa (indicante il funzionamento del localizzatore) inizi a lampeggiare: ciò segnala che il fix è stato effettuato e il dispositivo è pronto per rilevare le proprie coordinate. Da adesso potete fare le vostre foto. Se, durante il viaggio, nell'ML7 si accende il LED arancione, vuol dire che la batteria è scarica e bisogna al più presto sospendere il lavoro e portare l'interruttore in posizione OFF. Al ritorno scaricate le fotografie in una cartella del disco rigido del computer e poi collegate il localizzatore all'USB mediante il cavo in dotazione (un cavo USB qualunque non funzionerebbe perché, per proteggere i dati del logger, nel cavo originale

è inserito un chip che permette di far funzionare i driver) quindi preparatevi a scaricare le coordinate registrate. Allo scopo aprite il programma e, nella finestra principale, fate clic sul pulsante *Download*, aprendo così la finestra *Downloading Setting*, nella quale dovete scegliere quali porzioni (sentenze) delle stringhe NMEA volete scaricare: per la nostra applicazione ci bastano le RMC, perché contengono coordinate geografiche e ora, quindi de-selezionate tutto tranne RMC (lasciate il segno di spunta solo su quest'ultima voce, come mostra la Fig. 2). Fate clic su OK e attendete che avvenga il download, al termine del quale verrete avvertiti da un'apposita finestra di avviso, che indica anche la cartella in cui

è stato salvato il file contenente i dati scaricati dall'ML7; fate clic sul pulsante OK di quest'ultima e vedrete apparire una finestra di dialogo in cui vi viene chiesto se cancellare o meno la Flash del localizzatore (Fig. 3). Decidete cosa fare, fermo restando che non rimuovendo i dati, la prossima volta che accendete il localizzatore la memorizzazione proseguirà dall'ultimo record, quindi la prossima volta che farete il download vi ritroverete i vecchi dati.

LA CORRELAZIONE TRA FOTO E COORDINATE

Ora viene il momento di collegare i dati, ovvero di assegnare a ciascuna fotografia le coordinate in cui è stata scattata. La referenziazione delle foto si ottiene con un programma di libero utilizzo che può essere scaricato senza condizioni dal sito www.locr.com. Dalla sezione *Download* di questo sito, scaricate *LocrGPSPhoto124*, quindi, una volta ottenuto il file. exe, fatevi doppio clic e lanciate

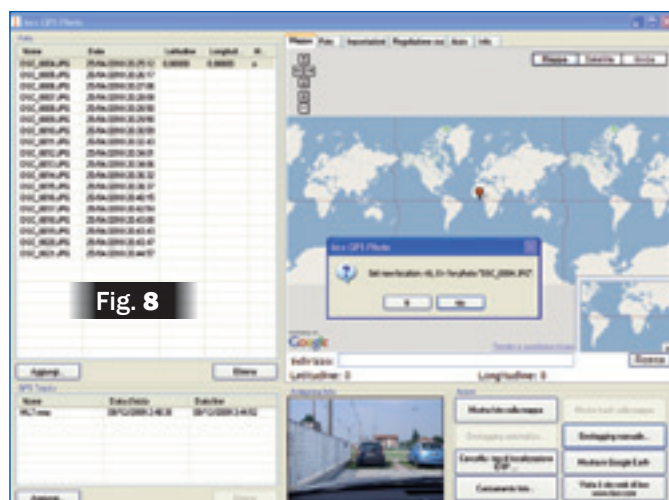


Fig. 8



Fig. 9

l'installazione; notate che è richiesto nel computer Microsoft .NET Framework 2.0 per architettura x86: se non l'avete vi verrà chiesta l'autorizzazione per andarlo a scaricare automaticamente dal sito Microsoft ed installarlo. Fate, dunque, clic su *Next* e, quando vi appare la finestra di dialogo illustrata nella **Fig. 5**, scegliete il percorso dove installare il programma e fate clic su *Next*. Completata l'installazione, chiudete la finestra facendo clic su *Close*. Ora andate in *Start>Tutti i Programmi* e fate clic su *Locr>Locr GPS Photo*; dopo qualche istante vi si presenterà la finestra di dialogo visibile nella **Fig. 6**, nella quale per prima cosa dovete caricare il file con le coordinate: fate clic sul pulsante *Aggiungi* che trovate nel riquadro in basso a sinistra (*GPS Tracks*) e, nella finestra di dialogo che si apre scegliete il percorso che contiene i file (è lo stesso che *GPS Logger Utility* vi aveva comunicato al momento del download dall'ML7). Il file è nella cartella `\nme` del disco rigido ed è del tipo *ML7.nme*; se avete memorizzato più file nella cartella, i file verranno nominati come *ML7_x.nme*, dove al posti di *x* c'è il numero progressivo di registrazione. Ora, fate clic sul pulsante *Aggiungi* del riquadro in alto a sinistra (*Foto*) e, nella finestra che si apre, selezionate il percorso dei file delle foto; se questi sono più d'uno, selezionateli trascinando il mouse col pulsante sinistro premuto e fate clic sul pulsante *Apri*. Notate che il programma funziona a condizione che le fotografie siano salvate in formato JPEG.

Ora che avete caricato le foto nel programma *Locr GPS Photo* potete procedere (**Fig. 7**). Per collegarle ai dati di localizzazione avete due possibilità, una delle quali è abbinare una foto alla volta, facendo clic sul nome del file corrispondente e quindi sul pulsante *Geotagging Manuale* che trovate nel riquadro in basso a destra nella solita finestra di dialogo (**Fig. 8**). Se volete, con il pulsante *Mostra foto sulla Mappa* potete vedere la posizione in cui vi trovavate quando avete scattato la fotografia selezionata al momento. Potete anche effettuare il Geotagging automatico; allo scopo notate che se ci sono rilevanti discrepanze tra l'orario del ricevitore GPS e quello dei file JPEG, la correlazione non riesce. Anche in questo caso il programma offre una soluzione: aprendo la scheda *Regolazione Ora*, potete impostare manualmente l'orario per adattare quello delle foto a quello delle coordinate rilevate dal GPS (**Fig. 9**); più esattamente, vi è possibile impostare la differenza tra i due orari, in modo che il programma sappia a che ora delle registrazioni del GPS corrispondono le fotografie. In pratica, perché tutto funzioni, facendo clic sui pulsanti a freccia accanto



Fig. 10

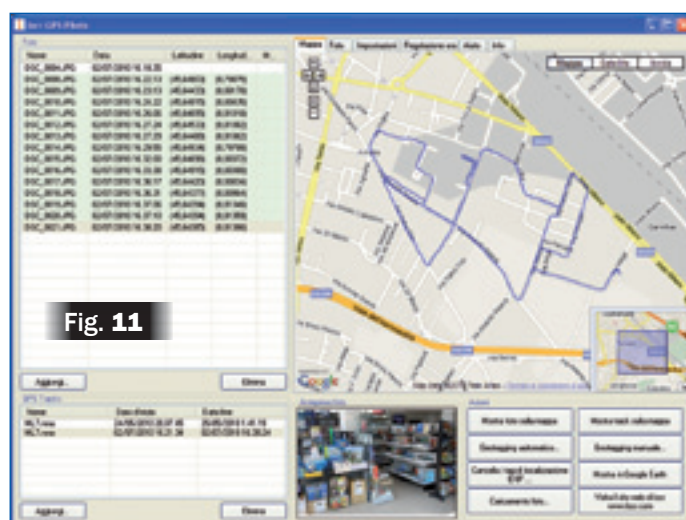


Fig. 11

alle singole caselle, dovete aggiungere o sottrarre giorni fin quando data e orario della fotocamera non siano compatibili con quelli di creazione del file log dell'ML7. Una volta effettuato il collegamento, facendo clic sul pulsante *Mostra track sulla mappa* potete visualizzare sulla mappa anche il percorso compiuto (**Fig. 10**); facendo clic sul nome di ciascun fotogramma nel riquadro in alto a sinistra, vedrete nel riquadro Photo Preview la fotografia corrispondente e sulla mappa verrà evidenziato il punto in cui eravate al momento dello scatto. In alternativa alla visione della mappa potete vedere percorso e punti nella vista da satellite. Facendo clic sui vari punti mostrati nel percorso sulla mappa vi sarà possibile vedere le foto che avete scattato quando vi trovavate lì (**Fig. 11**) direttamente nella mappa. Con il programma potete anche collegarvi al sito www.locr.com e inviare ad esso le foto georeferenziate, in modo da vedere in esso la sequenza; allo scopo, dovete preventivamente registrarvi ed ottenere un nome utente e una password, che vi saranno richieste nella scheda *Impostazioni*.

Progettiamo un sistema VoIP
che ci permetterà di instaurare
comunicazioni
audio e testuali,
liberi dall'uso del PC.
Ultima puntata.



REALIZZIAMO IL VOIP SENZA PC

..... di LUCA PASCARELLA

Nella precedente puntata abbiamo presentato il progetto di un sistema VoIP, descrivendo il principio di funzionamento e le caratteristiche generali del circuito che lo realizza dal punto di vista hardware. In questa seconda ed ultima puntata vogliamo concludere il discorso iniziato nel fasci-

colo n° 148 focalizzando l'attenzione sul firmware, che abbiamo scritto appositamente per il microcontrollore PIC, oltre che sulle modalità d'impiego delle unità per far funzionare il sistema di comunicazione utilizzando Internet. Termineremo con una breve guida all'uso delle funzioni disponibili.



Fig. 1 - I sotto-menu contraddistinti ognuno dal numero che hanno nel menu principale.

IL FIRMWARE

Il firmware per le unità de VoIP è stato scritto in linguaggio C, scegliendo come ambiente di sviluppo MPLAB IDE e per compilatore il C18 della Microchip. Il firmware è molto corposo, infatti più del 90 % della memoria Flash del PIC viene impiegato per memorizzare tutte le routine; minore è, invece, il quantitativo di RAM richiesta, che ammonta a circa i due terzi del totale. Il file principale che contiene anche il *main* del programma è *MainDemo.c*; come da prassi, le prime operazioni eseguite dal PIC all'accensione corrisponderanno all'inizializzazione delle linee di I/O e delle periferiche connesse. In ordine, nel *main* si incontrano: *InitializeBoard()*, *InitLcd()*, *PrintInit()*, *TickInit()*, *MPFSInit()*, *InitAp-*

pConfig() e *StackInit()*. Ognuna delle inizializzazioni precedenti ha uno scopo specifico e la maggior parte di esse deve essere eseguita esattamente nell'ordine indicato. Al termine della fase di inizializzazione, il controllo entra in un ciclo *while*, indefinito in quanto dichiarato su condizione sempre vera. I diversi compiti del PIC sono stati suddivisi in task, ognuno dei quali riveste un proprio ruolo; l'invocazione periodica delle diverse attività consente di creare un meccanismo che ricorda quello del multiprocesso tipico dei Personal Computer. In questo caso, però, i processi chiamati task rilasciano la CPU solo volontariamente al termine della loro esecuzione e per questo motivo è necessario porre molta attenzione a non sbilanciare troppo il carico di lavoro tra un task e l'altro,

onde evitare lo stallo della CPU. Le funzioni *StackTask()*, *StackApplications()*, *CallTCPClient()*, *ChatTCPClient()*, *TCP-Server()* e *ReadIO()* coincidono con le attività da svolgere. La prima nell'elenco è la funzione base fornita nello *Stack TCP/IP* della Microchip, che al proprio interno implementa tutti i meccanismi per consentire al PIC di scambiare dati con la rete LAN. Le successive tre funzioni sono invece il cuore dell'attività di codifica e decodifica del flusso audio, nonché dello scambio di messaggi testuali (chat). Infine *ReadIO()* si preoccupa di gestire i pulsanti previsti sul circuito stampato e, in corrispondenza delle azioni richieste, aggiornare il contenuto del display. È proprio su quest'ultimo punto che vogliamo inizialmente focalizzare l'attenzione: gestire un display a colori a matrice di punti si è rivelato per alcuni aspetti implementativi molto complesso; per semplificare l'uso e per ridurre al minimo le risorse hardware richieste al PIC, abbiamo dimezzato le sue potenzialità. Non dovendo creare delle finestre interattive con una grafica molto complessa, ci è stato possibile assegnare

un valore numerico alle diverse possibili schermate. Dal punto di vista implementativo, il valore della schermata corrente viene conservato in una variabile globale: per l'esattezza, nella variabile *Menu-Display*. La sequenza delle schermate è quella visibile nella Fig. 1. La navigazione tra le diverse voci del menu avviene mediante gli switch montati sul circuito stampato. Per semplicità, ad SW1 è stato associato il movimento di un ipotetico cursore (segnalato con l'asterisco di colore rosso sul display). Anche quest'informazione dinamica viene conservata nell'apposita variabile *RowDisplay*. In base alla combinazione delle due variabili *MenuDisplay* e *RowDisplay*, il PIC è in grado di far spostare l'utente da una pagina all'altra navigando tra i diversi menu; in particolare, ogni qual volta uno dei due pulsanti *UP_SX* e *DOWN_SX* viene premuto, la variabile *RowDisplay* incrementa o decrementa di un'unità. Inoltre, la pressione del pulsante *OK* in corrispondenza della voce al momento selezionata consente all'utente di entrare in detta voce. Questa operazione viene gestita



dal microcontrollore copiando il contenuto della variabile *RowDisplay* in *MenuDisplay*. Dalla Fig. 1 è chiaramente visibile che i diversi sotto-menu sono accessibili tramite un numero progressivo che corrisponde alla loro posizione nel menu principale.

L'idea di usare valori numerici consente di adottare come selettore del menu l'istruzione *switch()*, la quale ha un duplice vantaggio: in primo risulta molto efficiente in termini di prestazioni (se confrontata con gli *if()* in cascata, che rappresentano l'alternativa; al tempo stesso, gli *switch()* facilitano la scrittura del codice, che può così avvenire per step progressivi, aggiungendo di volta in volta il *case* interessato. La traduzione del "numero di menu" in nome di funzione avviene nel file *MenuLCD.c* ed in particolare nella funzione *Print(num_menu)*. Quest'ultima, come visibile nel Listato 1, si occupa in prima analisi di ridisegnare il frame tramite la funzione *PrintFrame()*; poi, una volta ripulito lo schermo, invoca, tramite l'istruzione *switch*, il menu desiderato. Diversa è, invece, la gestione del display per quanto riguarda la funzione di chat. In questo caso le funzioni designate sono: *PrintChat()*, *PrintChat-*

Send(), *PrintChatReceive()* ed *InitChat()*. Nel Listato 2 sono riportate le funzionalità

principali di questo processo: prima fra tutte c'è l'inizializzazione *InitChat()*, che avviene a

tempo d'avvio del PIC e consente di svuotare i buffer che poi conterranno le scritte da

Listato 1

```
void Print(char ind){
    //Disegna il frame principale
    PrintFrame();

    //Seleziona la funzione richiesta corrispondente al valore contenuto in
    ind
    switch(ind){
        case 0: PrintMenu();
                break;
        case 1: PrintContactList();
                break;
        case 2: PrintSetStatus();
                break;
        case 3: PrintSetContrast();
                break;
        case 4: PrintSetLocal();
                break;
        case 5: PrintSetVolume();
                break;
        case 6: PrintSetGain();
                break;
        case 7: PrintMicLine();
                break;
        case 8: PrintCloseCall();
                break;
        case 9: PrintAnswer();
                break;
        case 11: PrintCallChat();
                break;
        case 12: PrintFrameChat();
                break;
        case 13: PrintCalling();
                break;

        default: PrintMenu();
    }
}

void PrintMenu(void){
    switch(MyStatus){
        case MY_STATUS_FREE: LCDPutRomStr((rom char*)"Menu", 6, 45, MEDIUM, BLACK, GREEN); break;
        case MY_STATUS_BUSY: LCDPutRomStr((rom char*)"Menu", 6, 45, MEDIUM, BLACK, ORANGE); break;
        case MY_STATUS_CHAT: LCDPutRomStr((rom char*)"Menu", 6, 45, MEDIUM, BLACK, MAGENTA); break;
        case MY_STATUS_CALL: LCDPutRomStr((rom char*)"Menu", 6, 45, MEDIUM, BLACK, RED); break;
    }

    if(CallContact != 10)
        PrintStar(8);
    else
        PrintStar(7);

    LCDPutRomStr((rom char*)"Contact List", 25, 14, MEDIUM, BLACK, YELLOW);
    LCDPutRomStr((rom char*)"Set Status", 35, 14, MEDIUM, BLACK, YELLOW);
    LCDPutRomStr((rom char*)"Set Contrast", 45, 14, MEDIUM, BLACK, YELLOW);
    LCDPutRomStr((rom char*)"Local Info", 55, 14, MEDIUM, BLACK, YELLOW);
    LCDPutRomStr((rom char*)"Set Volume", 65, 14, MEDIUM, BLACK, YELLOW);
    LCDPutRomStr((rom char*)"Set Gain", 75, 14, MEDIUM, BLACK, YELLOW);
    LCDPutRomStr((rom char*)"Mic Line", 85, 14, MEDIUM, BLACK, YELLOW);
    if(CallContact != 10)
        LCDPutRomStr((rom char*)"Close Call", 95, 14, MEDIUM, BLACK, YELLOW);
}
}
```

```
void PrintChat(char ch){
char a=0, i, j;

// Controlla il valore dell'ultimo byte ricevuto
if(ch == '\n')
{
    // Un byte di new line è stato ricevuto
    if(CountBufferChatPrew == 0)
        return;

    // Cancella il contenuto della sezione di Preview
    LCDSetRect(104, 4, 127, 127, FILL, YELLOW); //Clear BufferChat

    while(ColumnBufferChatPrew < 15){
        BufferChatPrew[RowBufferChatPrew][ColumnBufferChatPrew++] = ' ';
        CountBufferChatPrew++;
    }

    for(i=0; i<=RowBufferChatPrew; i++)
        for(j=0; j<15; j++)
            Buffer[50+a++] = BufferChatPrew[i][j];
    SendChat = CountBufferChatPrew;

    PrintChatSend(RowBufferChatPrew+1);

    RowBufferChatPrew = 0;
    ColumnBufferChatPrew = 0;
    CountBufferChatPrew = 0;
}
else if(ch == 8)
{
    // Un byte di backspace è stato ricevuto
    if(ColumnBufferChatPrew == 0 && RowBufferChatPrew == 0)
        return;
    // Cancella l'ultimo carattere inserito se il cursore di trova sulla riga giusta
    // altrimenti decrementa una riga
    if(ColumnBufferChatPrew > 0)
    {
        ColumnBufferChatPrew--;
        LCDPutChar(' ', (104+RowBufferChatPrew*8), (5+ColumnBufferChatPrew*8), MEDIUM, BLACK,
YELLOW);
        BufferChatPrew[RowBufferChatPrew][ColumnBufferChatPrew] = ' ';
        CountBufferChatPrew--;
    }
    else
    {
        ColumnBufferChatPrew = 14;
        RowBufferChatPrew--;
        LCDPutChar(' ', (104+RowBufferChatPrew*8), (5+ColumnBufferChatPrew*8), MEDIUM, BLACK,
YELLOW);
        BufferChatPrew[RowBufferChatPrew][ColumnBufferChatPrew] = ' ';
        CountBufferChatPrew--;
    }
}
else if(ColumnBufferChatPrew < 15)
{
    // Inserisce l'ultimo carattere stampabile nel buffer se la riga non è piena
    LCDPutChar(ch, (104+RowBufferChatPrew*8), (5+ColumnBufferChatPrew*8), MEDIUM, BLACK, YELLOW);
    BufferChatPrew[RowBufferChatPrew][ColumnBufferChatPrew] = ch;
    ColumnBufferChatPrew++;
    CountBufferChatPrew++;
}
else if(RowBufferChatPrew == 0)
{
    // Inserisce l'ultimo carattere stampabile nella riga successiva del buffer
    RowBufferChatPrew++;
    ColumnBufferChatPrew = 0;
    LCDPutChar(ch, (104+RowBufferChatPrew*8), (5+ColumnBufferChatPrew*8), MEDIUM, BLACK, YELLOW);
    BufferChatPrew[RowBufferChatPrew][ColumnBufferChatPrew] = ch;
    ColumnBufferChatPrew++;
    CountBufferChatPrew++;
}
}
```

visualizzare sul display. Come abbiamo visto nella puntata precedente, quando viene in-

staurata una connessione di chat la pressione di un tasto sulla tastiera provoca l'esecuzione della

funzione *ReceiveChar()*. Questa, oltre ad ottenere il corrispondente codice ASCII, inserisce il

```

else if(RowBufferChatPrew == 1)
{
    // Inserisce l'ultimo carattere stampabile nella riga successiva del buffer
    RowBufferChatPrew++;
    ColumnBufferChatPrew = 0;
    LCDPutChar(ch, (104+RowBufferChatPrew*8), (5+ColumnBufferChatPrew*8), MEDIUM, BLACK, YELLOW);
    BufferChatPrew[RowBufferChatPrew][ColumnBufferChatPrew] = ch;
    ColumnBufferChatPrew++;
    CountBufferChatPrew++;
}
}

void PrintChatSend(char RowPrew){
char i, j, g, h, dec = 0;

// Copia il contenuto dell'array bidimensionale BufferChatPrew
// nell'array bidimensionale BufferChat
for(i=0; i<RowPrew; i++){
    for(j=0; j<15; j++){
        BufferChat[i+RowChat-dec][j] = BufferChatPrew[i][j];
        dec++;
        RowChat++;
        // Se il display è pieno cancella la prima riga e sposta tutti
        // i caratteri di una riga in modo da liberare una nuova riga
        if(RowChat > 10){
            RowChat--;
            for(g=0; g<10; g++){
                for(h=0; h<15; h++){
                    BufferChat[g][h] = BufferChat[g+1][h];
                }
            }
        }

        // Aggiorna il contenuto del display con i cambiamenti apportati a BufferChat
        for(g=0; g<10; g++){
            for(h=0; h<15; h++){
                LCDPutChar(BufferChat[g][h], (20+g*8), (5+h*8), MEDIUM, BLACK, YELLOW);
            }
        }
    }
}

void PrintChatReceive(char* pStr, char len){
char a=0, b, i, j, g, h, dec = 0;

// Calcola il numero di righe
b = len/15;

// Copia il contenuto dell'array lineare puntato da pStr
// nell'array bidimensionale BufferChat

// Loop esterno sul numero di righe
for(i=0; i<b; i++){
    // Loop interno sul numero di colonne
    for(j=0; j<15; j++){
        BufferChat[i+RowChat-dec][j] = pStr[a++];
        dec++;
        RowChat++;
        // Se il display è pieno cancella la prima riga e sposta tutti
        // i caratteri di una riga in modo da liberare una nuova riga
        if(RowChat > 10){
            RowChat--;
            for(g=0; g<10; g++){
                for(h=0; h<15; h++){
                    BufferChat[g][h] = BufferChat[g+1][h];
                }
            }
        }

        // Aggiorna il contenuto del display con i cambiamenti apportati a BufferChat
        for(g=0; g<10; g++){
            for(h=0; h<15; h++){
                LCDPutChar(BufferChat[g][h], (20+g*8), (5+h*8), MEDIUM, BLACK, YELLOW);
            }
        }
    }
}

```

valore corrispondente in un buffer, che è *kb_buffer*. Il firmware è strutturato in modo da rendere

trasparente l'utilizzo della tastiera; nel frattempo l'utente digita i messaggi e il *main* controlla

periodicamente la presenza di byte nel buffer *kb_buffer*, tramite la funzione *getchar()*. Se quest'ul-

Le memorie S-RAM ed EEPROM

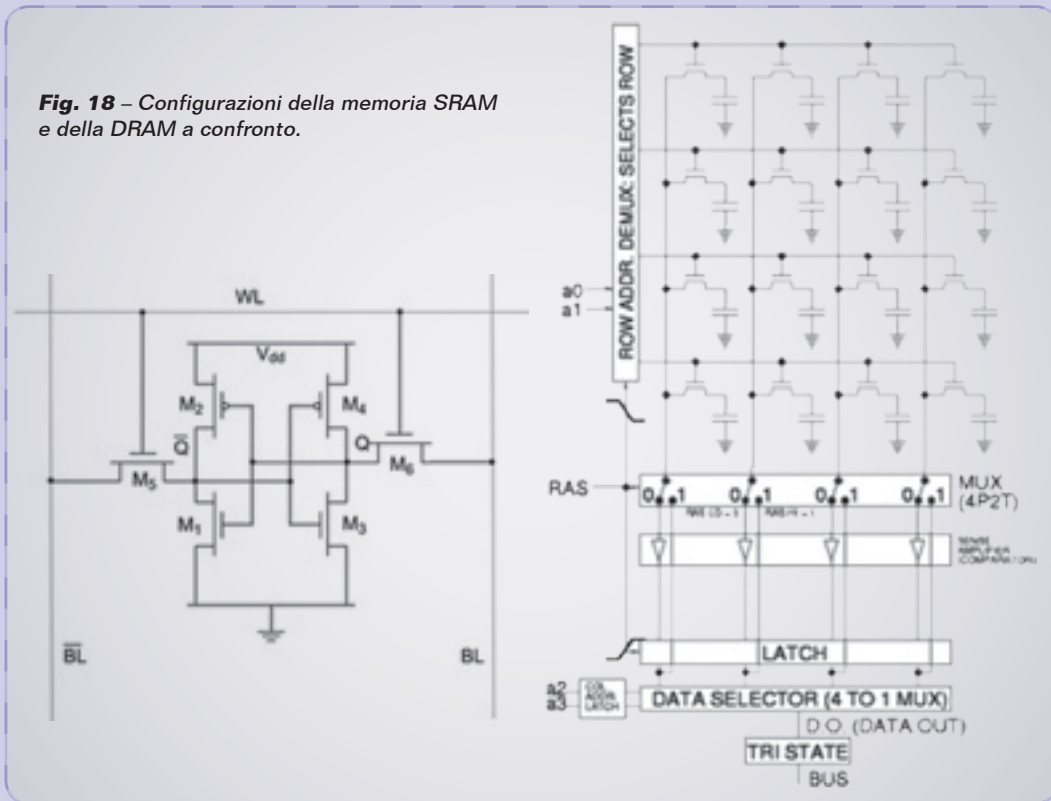
Il componente 23k256 è una memoria RAM statica realizzata dalla Microchip. Le RAM statiche (SRAM) si differenziano dai modelli dinamici (DRAM) in quanto sono più veloci ed immuni ad eventuali disturbi elettromagnetici. La cella elementare della RAM statica viene realizzata creando una rete di sei transistor in modo da formare un bistabile, mentre quella delle RAM dinamiche può essere realizzata a più basso costo perché consta di un condensatore ed un unico transistor (**Fig. 18**). La presenza della capacità però implica che i dati devono essere aggiornati periodicamente, altrimenti vengono persi; l'aggiornamento si chiama *refresh* della memoria. Le DRAM hanno lo svantaggio di essere notoriamente più lente, ma realizzabili con un costo per cella che è di circa 100 volte minore di una cella SRAM, per cui è possibile

raggiungere con costi contenuti grandi capacità, anche dell'ordine dei giga byte; viceversa, le SRAM possono

essere circa 10 volte più veloci, ma la loro capacità, per costi e dimensioni, è limitata all'ordine di qualche

kbyte. Le SRAM vengono impiegate per lo più all'interno dei microprocessori o dei microcontrollori, laddove sono

Fig. 18 – Configurazioni della memoria SRAM e della DRAM a confronto.



tima ritorna un valore positivo, il buffer contiene dei caratteri pronti per essere visualizzati sul display (solo modalità chat), quindi il controllo passa alla funzione *PrintChat()* visibile nel **Listato 2**. *PrintChat()* a sua volta effettua una serie di controlli per distinguere i caratteri stampabili dai codici ASCII di controllo, che nel nostro caso sono solo '\n' ovvero *new line* e *backspace*. A questo punto è chiaro il ruolo primario assegnato a *PrintChat()* la quale, quando riconosce che il byte ricevuto è un carattere stampabile, ne fa stampare il valore sul display e lo carica in un ulteriore buffer chiamato *BufferChatPrew*. Quest'ultimo è un array bidimensionale. Dall'analisi della struttura dei font scelti è emerso che tutti i caratteri riproducibili dalla mappa

dei caratteri inserita in questo display sono grandi 8x8 pixel, di conseguenza si può conoscere a priori la massima capacità del buffer in funzione delle dimensioni del display; *BufferChatPrew* sarà grande 3 righe e 15 colonne. Se lo spazio si esaurisce, gli ultimi caratteri inseriti vengono semplicemente scartati fin quando sulla tastiera non viene premuto *enter* per inviare il messaggio di chat all'altro dispositivo. L'invio del contenuto testuale della chat avviene per opera della funzione *PrintChatSend()* che si occupa in primo luogo di copiare il contenuto del buffer *BufferChatPrew* nel buffer *BufferChat*. Quest'ultimo è un ulteriore buffer bidimensionale grande 11 righe per 15 colonne, dove verranno inseriti i messaggi di chat in uscita ed in entrata, proprio come nella chat

reale. L'informazione contenuta nel buffer verrà copiata sul display ed inviata tramite socket TCP all'altro dispositivo. L'uso del buffer *BufferChat* è reso necessario dal fatto che vogliamo far scorrere i messaggi sul display. Il PIC crea la piccola cronologia corrispondente aggiornando di volta in volta il contenuto del display con i nuovi messaggi in ingresso ed uscita. Un'ulteriore limitazione imposta a favore del risparmio di risorse nel microcontrollore riguarda l'impossibilità di rileggere i messaggi che non entrano nello schermo del display, in quanto quest'ultimi, per questioni di spazio (memoria RAM) vengono persi. Per poter creare una cronologia intera di tutti i messaggi si potrebbe usare della RAM esterna appositamente aggiunta sul circuito stampato;

sufficienti poche quantità di dati ma l'accesso in lettura e scrittura deve essere paragonabile ai tempi di lavoro della CPU. Per rendere la comunicazione estremamente veloce vengono costruite sullo stesso wafer di silicio con l'impiego di aree molto vaste. La 23k256 è una RAM esterna da abbinare alle MCU nel caso in cui sia necessaria una quantità di spazio dinamico maggiore della memoria di bordo; anche se accedere all'esterno implica scambiare i dati su una periferica lenta (SPI in questo caso) la soluzione è ugualmente vantaggiosa e consente di usare microcontrollori con poca dotazione di RAM, ampliandone le possibilità laddove è strettamente necessario. Il chip in questione lavora a frequenze massime di 20 MHz, quindi è compatibile con molte periferiche e garantisce uno spazio extra di 32 kbyte. La sua realizzazione, in tecnologia CMOS, permette di consumare pochissima energia (il consumo in stand-by è di soli 4 μ A). Le modalità di funzionamento sono varie e permettono la lettura sequenziale, per pagina o per singolo byte. Il componente è fornito

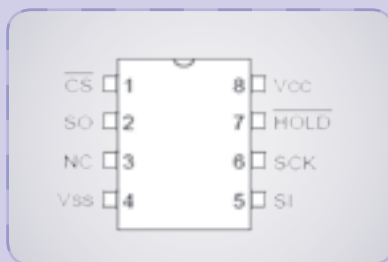


Fig. 19 – Pinout della memoria S-RAM 23k256.

in tre diversi package: PDIP, SOIC e TSSOP; la pin-out è visibile nella **Fig. 19**. Nel nostro circuito, oltre alla SRAM usiamo anche una EEPROM ad accesso seriale (SPI, nel nostro caso) che ci serve a memorizzare in maniera permanente tutte le informazioni di configurazione dell'unità VoIP; la memoria è una 24LC1024 ed ha una capacità di 1 Mbit, ovvero 128 kbyte, strettamente sufficiente a contenere le pagine web, il plugin per l'encoder, le informazioni di sistema e la lista di contatti. Come per la memoria SRAM anche la EEPROM consente l'accesso in lettura in maniera random, e sequenziale, distinguendo i due casi con un opportuno opcode. Il clock massimo è di 1 MHz e la durata di vita è dell'ordine di un milione di cicli di scrittura.

mente a quanto descritto nel progetto "trasmettiamo l'audio su rete ethernet" descritto nel fascicolo n° 144, con la differenza che la comunicazione in questo caso è bidirezionale. Nei file *GenericTCPClient.c* e *GenericTCPServer.c* sono riportate le funzioni che implementano l'architettura Client/Server. Come è intuitivo, il socket in ascolto (modalità Server) viene creato con l'istruzione *MySocket = TCPOpen(0, TCP_OPEN_SERVER, AppConfig.LocalPort, TCP_POSE_GENERIC_TCP_SERVER)* non appena la board dispone di un indirizzo IP. L'esecuzione di questa funzione consentirà di ricevere una connessione in ingresso sul numero di porta prestabilito. Dopo l'esecuzione di *TCPOpen()* il PIC controlla periodicamente l'arrivo di nuove connessioni, mediante l'esecuzione di *TCPIsConnected(MySocket)*; se questo comando ritorna un valore positivo, vuol dire che il

in questo caso la capienza di un ipotetico buffer potrebbe estendersi ben oltre le capacità della memoria interna del PIC. Bene, il funzionamento della chat è quasi concluso: manca solo da descrivere il comportamento nel caso in cui un messaggio di chat viene ricevuto dal socket TCP. Il discorso è del tutto simile a quanto detto per *PrintChat()* e *PrintChatSend()*, salvo che per delle piccole semplificazioni. In questo caso, infatti, per stampare i messaggi ricevuti non bisogna attendere la pressione del tasto *enter*, bensì è sufficiente invocare la funzione *PrintChatReceive()* in corrispondenza di byte provenienti dal socket TCP. Passiamo ora ad analizzare il resto del firmware. La gestione dell'encoder e del decoder VS1053 avviene molto simil-

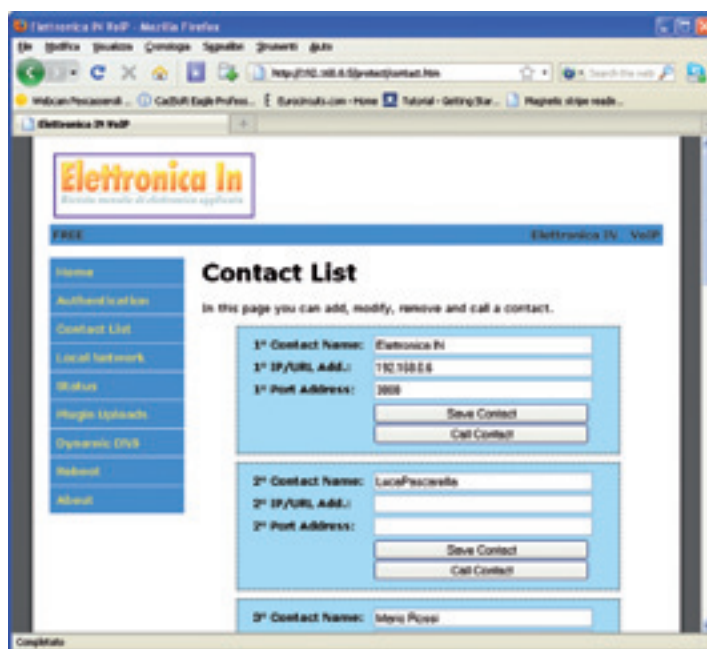


Fig. 2 - Pagina Web di impostazione della lista dei contatti. L'accesso è protetto da username e password, che, per impostazione predefinita, sono admin e password; è consigliabile cambiarli accedendo all'apposita pagina Authentication.



La P.A. adotta il VoIP

Il Ministero per la Pubblica Amministrazione e l'Innovazione ha deciso di adottare il VoIP open-source sviluppato all'Università di Ferrara (VoIP-Fe) dove funziona già da tre anni; ciò consentirà un drastico taglio dei costi telefonici, in quanto il VoIP si appoggia sulla rete a larga banda fornita da Lepida. Il servizio già permette chiamate tra numeri fissi e in mobilità, sfruttando tecnologie wireless/mobile. L'adozione dell'architettura open-source consentirà di adattare il sistema alle esigenze della P.A. grazie alle estensioni fornite da comunità di sviluppatori. L'Università di Ferrara è attualmente impegnata nel semplificare il sistema e creare una community VoIP open-source delle università italiane e della Pubblica Amministrazione.

socket è stato occupato da una nuova connessione.

I primi byte provenienti dal flusso di stream corrisponderanno a delle informazioni di controllo che consentiranno di individuare il tipo di connessione, distinguendo tra *Chat* e *Call*; se il dispositivo è libero e quindi il suo stato personale non è stato impostato su *Busy*, il Server notificherà al Client la disponibilità ad instaurare la comunicazione ed avvierà le procedure per prelevare il flusso di byte dal VS1053 (modalità encoder) ed inviarlo sulla rete. Allo stesso tempo i byte provenienti dalla comunicazione TCP verranno inviati al VS1053 (modalità decoder). Il discorso lato Client non può estendersi per entrambe le modalità di funzionamento, nel senso che nel caso *Chat* e in quello *Call* si adottano due funzioni diverse.

In particolare, le funzioni designate sono *ChatTCPClient()* e

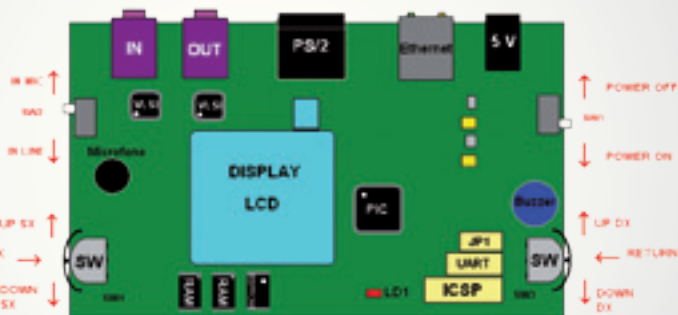
CallTCPClient(). Queste due procedure hanno molti punti in comune, tra cui la struttura generale, mentre ciò che cambia è l'accesso ai buffer in lettura ed in scrittura. Nel caso di *ChatTCPClient()* il buffer interrogato corrisponde a quello precedentemente nominato, ovvero *BufferChat*; viceversa, nel caso di *CallTCPClient()* il buffer interrogato corrisponde a quello gestito dalle funzioni *VLSIPutArray(Buffer, ByteRead)* e *VLSIGetArray(Buffer, 32)*. Queste

ultime due non sono nuove; infatti sono già state impiegate nel progetto audio su ethernet e pertanto evitiamo di spiegarle nuovamente, rimandando, chi fosse interessato, alla seconda puntata di tale articolo, pubblicata nel fascicolo n° 145 di *Elettronica In*. L'ultimo punto che accomuna la gestione del meccanismo di *Chat* e di *Call* riguarda la gestione della connessione partendo dalla lista dei contatti.

I contatti memorizzabili all'interno della EEPROM possono essere al massimo 10 e la loro posizione è statica e va da 0 a 9. La restrizione ad una lista statica consente di utilizzare un valore numerico compreso tra 0 e 9 per individuare un singolo contatto.

Questo stratagemma, inoltre, consente di impiegare due variabili globali: *CallContact* e *ChatContact*, che permettono di controllare istante per istante lo stato della connessione. Le variabili possono essere impiegate non solo per conservare traccia della connessione attuale, ma anche per notificare uno stato di "ready" ed uno di "busy". In particolare modo, si è scelto di assegnare un valore pari a 10 per indicare l'assenza di comunicazione, ovvero nessun utente contattato ed un valore pari ad 11 per indicare che momentaneamente la board è impegnata con una comunicazione in ingresso.

Fig. 3
Funzionalità associate alla scheda per il nostro sistema VoIP.



GESTIONE WEB DELLA LISTA DEI CONTATTI

La lista dei contatti può essere consultata in due modalità: la prima è tramite browser; la seconda modalità prevede la diretta interazione con il display LCD. Entrambe hanno delle restrizioni, ossia dal Web non è possibile avviare la funzionalità di chat, tuttavia è possibile compiere delle connessioni vocali o aggiungere/modificare/eliminare un contatto; viceversa, dal menu sul display è possibile solo compiere telefonare o avviare sessioni di chat con un contatto già presente in elenco. Il file *contact.htm* contiene il codice html per consentire le operazioni sopra descritte; in particolare, come visibile dal **Listato 3**, per ogni contatto vengono impiegati due *FORM* che consentono di invocare in modalità POST, rispettivamente, la pagina *contact.htm* o la *call.htm*. L'invocazione di queste due pagine da parte dell'utente comporta l'avvio dei meccanismi già descritti nell'articolo sulla trasmissione dell'audio via ethernet; in particolare, il metodo POST viene gestito con delle funzioni appositamente scritte nel file *CustomHTTPApp.c*. La funzione su cui vogliamo soffermare l'attenzione è *HTTPPostCall()*, visibile nel **Listato 3**, la cui routine ha una doppia funzionalità e dipende dalla pagina dalla quale viene invocata. Vediamo i dettagli caso per caso: se *HTTPPostCall()* viene chiamata come da **Listato 3**, il valore della variabile nascosta `<input type="hidden" value="0" name="ind"></input>` è compreso tra 0 e 9 e quindi assegnando questo valore alla variabile *CallContact* si può instaurare una connessione audio con il contatto prescelto. Inoltre, se *CallContact* ha un valore diverso da 10, come avviene in questo caso, l'utente

Listato 3

```

<!--Primo-->
<fieldset>
<form method="post" action="/protect/contact.htm" name="config">
<div><label>1° Contact Name:</label><input type="text" name="N11"
value="~N11~" maxlength="14" /></div>
<div><label>1° IP/URL Add.:</label><input type="text" name="N12" value="~N12~"
maxlength="100" /></div>
<div><label>1° Port Address:</label><input type="text" name="N13"
value="~N13~" maxlength="5" /></div>
<div><input type="submit" value="Save Contact" />
</form>
<form method="post" action="/protect/call.htm" name="config">
<input type="hidden" value="0" name="ind"></input>
<input type="submit" value="Call Contact" /></div>
</form>
</fieldset>

static HTTP_IO_RESULT HTTPPostCall(void)
{
    BYTE *ptr;
    BYTE i;
    extern char CallContact;
    extern char MyStatus;
    extern char PrintRefresh;

    if(curHTTP.byteCount > TCPIsGetReady(sktHTTP) +
TCPGetRxFIFOFree(sktHTTP))
        goto ConfigFailure;

    // Ensure that all data is waiting to be parsed. If not, keep wait-
    ing for
    // all of it to arrive.
    if(TCPIsGetReady(sktHTTP) < curHTTP.byteCount)
        return HTTP_IO_NEED_DATA;

    // Read all browser POST data
    while(curHTTP.byteCount)
    {
        // Read a form field name
        if(HTTPReadPostName(curHTTP.data, 6) != HTTP_READ_OK)
            goto ConfigFailure;

        // Read a form field value
        if(HTTPReadPostValue(curHTTP.data + 6, sizeof(curHTTP.
data)-6-2) != HTTP_READ_OK)
            goto ConfigFailure;

        // Parse the value that was read

        if(!strncpy2ram((char*)curHTTP.data, (ROM char*)"ind"))
            CallContact = ASCIItoNumeric(curHTTP.data+6);
    }

    if(CallContact == 10){
        strncpy2ram((char*)curHTTP.data, (ROM char*)" /protect/
contact.htm");
    }else{
        MyStatus = MY_STATUS_CALL;
        PrintRefresh = 1;
        strncpy2ram((char*)curHTTP.data, (ROM char*)" /protect/
call.htm");
    }

    curHTTP.data[20+16] = 0x00; // Force null termination
    for(i = 20; i < 20u+16u; i++)
    {
        if(curHTTP.data[i] == ' ')
            curHTTP.data[i] = 0x00;
    }
    curHTTP.httpStatus = HTTP_REDIRECT;

    return HTTP_IO_DONE;

ConfigFailure:
    lastFailure = TRUE;
    strncpy2ram((char*)curHTTP.data, (ROM char*)" /protect/contact.
htm");
    curHTTP.httpStatus = HTTP_REDIRECT;

    return HTTP_IO_DONE;
}

```

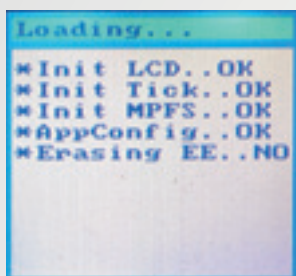


Fig. 4 - Immagine mostrata dal display al termine della fase di inizializzazione.



Fig. 5 - Menu principale e sottovoci di menu accessibili tramite il pulsante SW4.

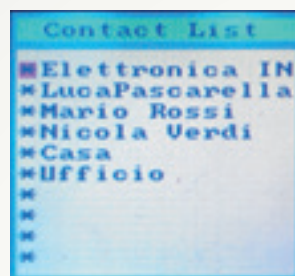


Fig. 6 - Lista dei contatti: con SW4 UP, DOWN ed OK si seleziona un destinatario e con SW3 (Return) è possibile tornare al menu principale.

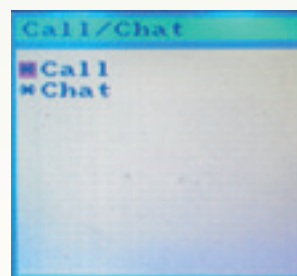


Fig. 7 - Scelta della modalità con cui contattare il destinatario selezionato: SW4 consente la normale navigazione, mentre SW3 consente di tornare al menu principale.



Fig. 11 - Funzionalità di Chat: il frame ha aggiunto una nuova area in basso, nel quale vengono riportati i messaggi ancora da inviare. Lo spazio centrale contiene la cronologia dei messaggi inviati e ricevuti. Con il pulsante return di SW3 si termina la sessione di chat.

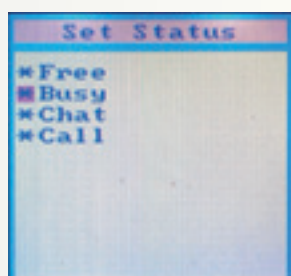


Fig. 12 - Menu "Set Status": consente di modificare il proprio stato. Quando lo stato è in Free le connessioni di chat e call verranno accettate automaticamente; in tutti gli altri stati verranno scartate.



Fig. 13 - Per adeguare il contrasto dei diversi display, da questa voce di menu è possibile selezionare il valore preferito. Il valore selezionato rimane anche in caso di spegnimento del circuito.

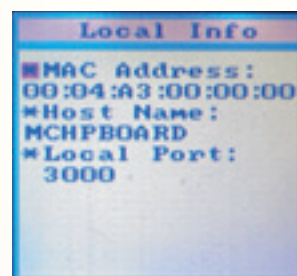


Fig. 14 - Da questo menu è possibile conoscere le informazioni preliminari della board, tra cui il MAC Address, che è utile conoscere per evitare conflitti con unità che hanno lo stesso valore.

viene reindirizzato verso la pagina *call.htm*.

Nella pagina *call.htm* tra il codice HTML prende posto un secondo FORM con metodo POST, simile a quello del **Listato 3**, ma questa volta con l'obiettivo di terminare la comunicazione; perciò se il parametro passato alla funzione *HTTPPostCall()* sarà pari a 10 (`<input type="hidden" value="10" name="ind"></input>`) la comunicazione verrà terminata. Sullo schermo, l'utente verrà reindirizzato alla pagina della lista contatti.

FUNZIONAMENTO DEI DISPOSITIVI

Siamo finalmente giunti alla descrizione delle modalità di impiego e d'uso dell'unità per VoIP. Come anticipato in diverse occasioni, il firmware, per quanto possibile, è stato scritto in maniera da rendere il tutto di facile approccio ed intuitivo. Partiamo con l'accensione della board: sullo stampato, come visibile dalla **Fig. 3**, è stato previsto uno switch (SW1) per accendere e spegnere la board; muovendolo verso l'alto stacca l'alimentazione, mentre

verso il basso alimenta il circuito. All'accensione, il display mostra una serie di messaggi di inizializzazione (come visibile nella **Fig. 4**) dai quali l'utente può apprendere il corretto funzionamento. Dopo la fase di inizializzazione, la scheda è in funzione e pronta a ricevere telefonate o messaggi di chat.

Per interagire con essa e con il display è sufficiente navigare con SW3 ed SW4 (**Fig. 3**); le funzioni assegnate ai pulsanti cambiano da menu a menu e per questo saranno descritte ognuna distinta-

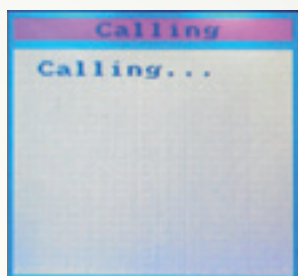


Fig. 8 - Immagine mostrata nell'attesa che l'utente contattato risponda.

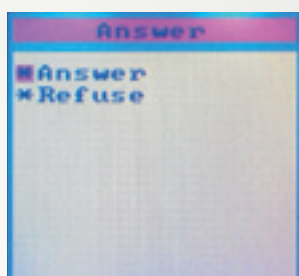


Fig. 9 - Menu di scelta tra risposta (Answer) o rifiuto (Refuse) visualizzato nel momento in cui arriva una chiamata. Il menu resta attivo per 45 secondi, durante i quali il buzzer emetterà un suono per avvisare della chiamata in ingresso (suoneria).

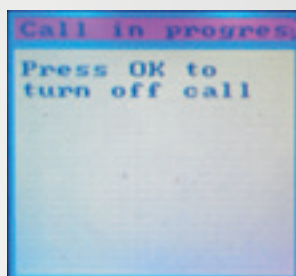


Fig. 10 - Menu di chiamata in corso: premendo il pulsante OK di SW4 si termina la telefonata, mentre premendo il Return di SW3 si torna al menu principale. A questo sarà aggiunta la voce "Close Call", che consentirà di tornarvi.



Fig. 15 - Schermata di controllo del livello del volume audio. Funzioni UP e DOWN associate al pulsante SW3.

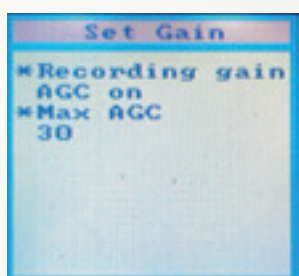


Fig. 16 - Controllo del livello del guadagno all'ingresso audio e dell'AGC corrispondente.



Fig. 17 - Configurazione dell'encoder per ricevere un segnale dall'ingresso audio RCA o dal microfono interno. In corrispondenza della selezione software è importate spostare manualmente lo switch SW2 come mostrato nella Fig. 3.

re di rispondere oppure rifiutare la chiamata (Fig. 9). La Fig. 10 mostra lo stato di conversazione vocale attiva. In questa fase si può notare che lo sfondo del menu è passato da verde a rosso. Nel caso in cui dal menu Call/Chat (Fig. 7) si selezioni la voce Chat, il display passa alla modalità Chat, visibile nella Fig. 11, ed il colore di sfondo del titolo di menu apparirà viola. La modalità Chat, come nei normali applicativi di *instant messaging*, non necessita dell'approvazione da parte del ricevente e la sessione viene automaticamente avviata.

Tornando alle voci del menu principale, diciamo che la seconda consente di modificare il proprio stato, come illustrato nella Fig. 12. Se si seleziona lo stato *Busy* (occupato) il colore del titolo cambia da verde ad arancione e le telefonate in arrivo o le chat verranno ignorate. Dal menu principale di Fig. 5 è possibile accedere a *Set contrast* per modifi-

care il margine di contrasto, come visibile nella Fig. 13. Partendo dal menu principale è inoltre possibile conoscere delle informazioni preliminari sulla board, tra cui il MAC Address e il numero di porta (Fig. 14) semplicemente seguendo la voce di menu *Local info*. Infine, le ultime due possibili operazioni consentono di modificare il volume ed il guadagno per l'ingresso audio, come visibile, rispettivamente, nelle figure 15 e 16.

È importante sottolineare che il volume può essere modificato

mente per ciascuna voce di menu. Partendo dal menu principale (Fig. 5) troviamo le funzionalità base: si può accedere a tutti i sotto-menu utilizzando il pulsante SW4, che consente di spostare il cursore in alto o in basso e di selezionare la voce di menu evidenziata dall'asterisco rosso. Nel menu principale non è assegnata alcuna funzione al pulsante SW3. La prima voce di menu, visibile nella Fig. 5, consente di accedere alla lista di contatti (Fig. 6); anche in questa pagina le funzioni di SW4 sono quelle standard, alle

quali si aggiunge la funzione "Return" per SW3 (Fig. 3) che serve a tornare al menu principale. Se in corrispondenza di un contatto viene premuto il pulsante OK, si passa al menu di Fig. 7, dove è possibile selezionare il tipo di connessione desiderata scegliendo tra le modalità *Call* e *Chat*. Premendo ulteriormente il pulsante OK in corrispondenza della voce *Call*, si passa alla schermata di Fig. 8; qui è chiaramente visibile lo stato di attesa durante il quale l'utente contattato può sentire "squillare" il dispositivo e decide-

solo con le funzioni *UP_DX* e *DOWN_DX*, mentre il guadagno dell'ingresso audio viene modificato da *UP* e *DOWN*, con la particolarità che le funzioni di sinistra controllano il *recording Gain* e quelle di destra impostano il massimo valore di *AGC (Automatic Gain Control)*.

All'appello manca solo la funzione associata allo switch *SW2*; quest'ultimo riveste un ruolo fondamentale nella comunicazione audio, perché può essere usato per selezionare come sorgente sonora l'ingresso interno alla board (la capsula microfonica) o l'ingresso esterno associato al connettore *IN*. È importante selezionare, in corrispondenza della posizione *SW2*, la giusta configurazione software, che potrà avvenire in due modi: seguendo l'ultima voce di menu, come visibile nella **Fig. 17**, oppure interagendo con la pagina *Web Status*.

COLLEGAMENTO

Come è chiaramente visibile dalle immagini e dalla descrizione, la scheda è dotata di connettore femmina *RJ45* che consente l'inserimento di un plug per reti LAN. Ogni unità può funzionare

sia all'interno della LAN, senza riscontrare particolare problemi, sia affacciandosi in Internet; in quest'ultimo caso la presenza di un router diviene fondamentale. È inoltre importante configurare il *NAT (Network-Area-Translation)* del router affinché le connessioni provenienti da Internet possano essere indirizzate alla scheda VoIP desiderata. Per questo motivo riportiamo anche una breve panoramica sul principio di funzionamento del *NAT*.

I dispositivi da noi usati per i test hanno dato buoni risultati quando erano impiegati in LAN anche di grandi dimensioni; viceversa abbiamo riscontrato forti ritardi nella comunicazione, dovuti alla latenza, nel caso in cui i dispositivi venivano impiegati in Internet con linee *ADSL* non troppo veloci.

COS' È IL NAT

Si tratta di una tecnica impiegata nella traduzione degli indirizzi IP, che consente di modificare l'indirizzo IP in transito in un router pacchetto per pacchetto. Spesso il *NAT* è anche conosciuto come *Port-Forwarding*, termine che rende meglio l'idea dell'azione svolta. Tale tecnica è stata, per alcuni aspetti, implementata per sopperire alla crescente carenza di indirizzi *IPv4* (protocollo tutt'ora in uso ma ben presto destinato a far posto all'*IPv6*). Per facilitare la comprensione dell'argomento, proviamo a semplificare: sappiamo che digitando un indirizzo Web nella barra degli indirizzi di un browser, ad esempio *http://www.elettronica.in.it/*, l'azione finale che si sta compiendo è l'instaurazione di una connessione *TCP* con una macchina remota, ovvero un server, ed in particolar modo verso un processo in esecuzione sul dato server, che nel caso della naviga-

zione Web coincide con il Web server (ad esempio Apache) in ascolto sulla porta 80. Il processo così contattato prende in carico la richiesta e risponde con l'invio della pagina desiderata.

A questo punto possiamo semplificare il discorso immaginando che l'indirizzo IP serve solo per individuare una macchina collegata in rete, mentre il numero di porta consente di individuare un particolare processo in esecuzione sulla macchina contattata. Nel caso dei sistemi VoIP, le schedine sono le "macchine" cui viene assegnato un indirizzo IP e i processi in esecuzione sono le applicazioni Server che girano in back-ground e consentono di ricevere le chiamate in ingresso. Ciò che accade quando colleghiamo una di queste board alla nostra rete domestica (LAN) è l'acquisizione di un indirizzo IP (per mezzo del *DHCP*) appartenente ad un gruppo di indirizzi privati: solo chi è all'interno della rete può conoscere e contattare gli Host presenti, ossia tutti gli utenti esterni alla rete. Il mondo Internet, ad esempio, vede un unico dispositivo, ossia il router, cui il provider dei servizi ha assegnato un indirizzo IP univoco (spesso dinamico) e tutte le richieste saranno rivolte ad esso. Di conseguenza, dalle considerazioni fatte è evidente come dall'esterno della rete sarebbe impossibile vedere gli Host interni alla LAN, perché "mascherati" dal router.

Quando, come nel caso precedente del browser, è un processo interno alla rete LAN ad effettuare il collegamento verso Internet, il lavoro svolto dai router è molto semplice e, grazie ad un piccolo artificio, è possibile consentire a tutte le macchine della LAN di affacciarsi ad Internet ed instaurare delle connessioni. Senza le regole di *NAT* sarebbe impossi-



le gestire le richieste provenienti dall'esterno.

Analizziamo in dettaglio le soluzioni dei due casi evidenziati: nel primo il router conosce esattamente l'indirizzo IP e la porta dell'Host da cui è partita la richiesta (sorgente) e può reindirizzare esattamente alla stessa macchina tutti i pacchetti di ritorno (risposta). È importante ricordare che tutte le richieste delle macchine all'interno della LAN vengono instradate in Internet sempre con l'unico indirizzo IP assegnato dal provider dei servizi al router e con un numero di porta generato casualmente dal browser al momento della connessione; di conseguenza, i pacchetti di ritorno che fanno tutti riferimento al nostro router possono essere discriminati in base al numero di porta, che si presuppone essere univoco.

L'azione del router a questo punto consiste nel modificare il pacchetto di risposta con l'indirizzo della macchina (all'interno della LAN) da cui è partita la richiesta ed inviarlo solo a quella macchina.

Questa operazione, che avviene in maniera automatica tutte le volte che si instaura una connessione da una LAN all'esterno, si chiama di SNAT o source NAT e viene gestita dai router con la creazione di tabelle nelle quali la chiave è il numero della porta sorgente.

Il discorso si fa un po' più complesso nel secondo caso, ovvero, quando è un Host esterno alla rete LAN a voler contattare un particolare processo residente all'interno della LAN: non avendo conoscenza degli indirizzi IP assegnati all'interno (perché privati) l'unico riferimento rimane l'indirizzo del router. Quest'ultimo a sua volta, quando riceve la richiesta non conosce a quale

degli Host interni inoltrarla; però con la tecnica del *destination NAT* o DNAT, può compiere questa scelta analizzando il numero di porta.

Senza il DNAT è subito evidente l'inefficienza di una rete privata ed è per questo motivo che è nata la soluzione DNAT, la quale, anche se non risulta efficiente in tutti i casi in quanto implica la creazione manuale di regole all'interno del router, risolve un problema molto importante.

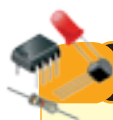
La tecnica DNAT (che consiste nel creare delle regole di forwarding, ovvero di reindirizzamento) opera interagendo con il router (le modalità sono fortemente sensibili alla marca ed al modello); la sua azione è creare delle corrispondenze sfruttando i numeri di porta. Quindi tutte le richieste provenienti dall'esterno con dato numero di porta saranno instradate solo verso l'host da noi selezionato. Questa operazione, in gergo, spesso viene definita con "apertura delle porte", perché dà la sensazione di rendere visibile all'esterno un indirizzo privato; in realtà ciò che accade è solo il reindirizzamento per un dato numero di porta, in quanto per tutti i restanti numeri l'indirizzo designato risulterà ancora privato e nascosto dal router.

ESPANDIBILITÀ

Le potenzialità della scheda VoIP da noi proposta sono numerose e molte possono essere le modifiche da apportare al firmware per migliorare prestazioni e funzionalità. Per ovvi motivi si è scelto di ridurre al minimo il numero di controlli nel caso in cui l'utente erroneamente introducesse dei comandi non validi; ciò non costituisce un bug di sistema, perché in realtà si tratta di limitazioni introdotte appositamente per non appesantire eccessivamente il

software e non renderlo illeggibile. Chi vorrà, potrà migliorare a suo gusto il firmware, dato che sulla board ci sono ancora molte potenzialità non sfruttate. Ad esempio, il buzzer, le due RAM e la porta UART non vengono impiegati. Un loro impiego potrebbe estendere di molto le potenzialità: si pensi alla chat ed al discorso fatto sulla possibilità di rileggere i messaggi inviati.

Bene, con questa puntata possiamo ritenere concluso il discorso iniziato il mese scorso. Anche stavolta riteniamo di avervi presentato un progetto innovativo, utile sia per scopi puramente didattici, sia per risolvere problemi quotidiani. Probabilmente molti di voi hanno già potuto testare il funzionamento del circuito, mentre altri avranno preferito aspettare la conclusione del progetto con la descrizione del firmware. A tutti auguriamo buon divertimento. ■



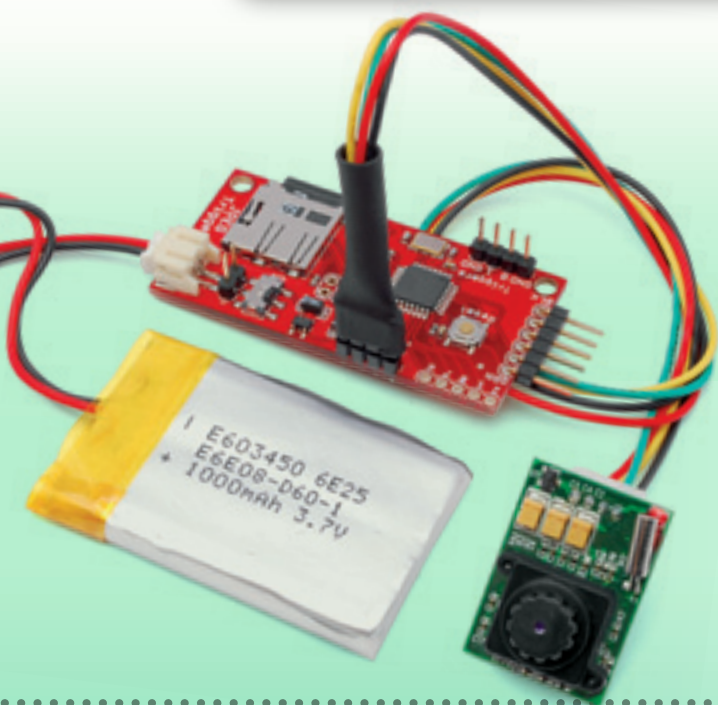
per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista così come il firmware utilizzato per programmare il microcontrollore. Dei componenti un po' "strani", presso Futura Elettronica sono disponibili il PIC18F67J60-I/PT (costo 10,00 Euro), il VS1053B (costo 22,00 Euro) e il display LCD-128NOKIA (NOKIA 6100) che costa 18,50 Euro. Tutti i prezzi si intendono 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>



Realizziamo una fotocamera digitale usando una telecamera CMOS e un modulo open-source, che attivato da un semplice livello logico cattura le immagini, salvandole in formato JPEG in una SD-Card.



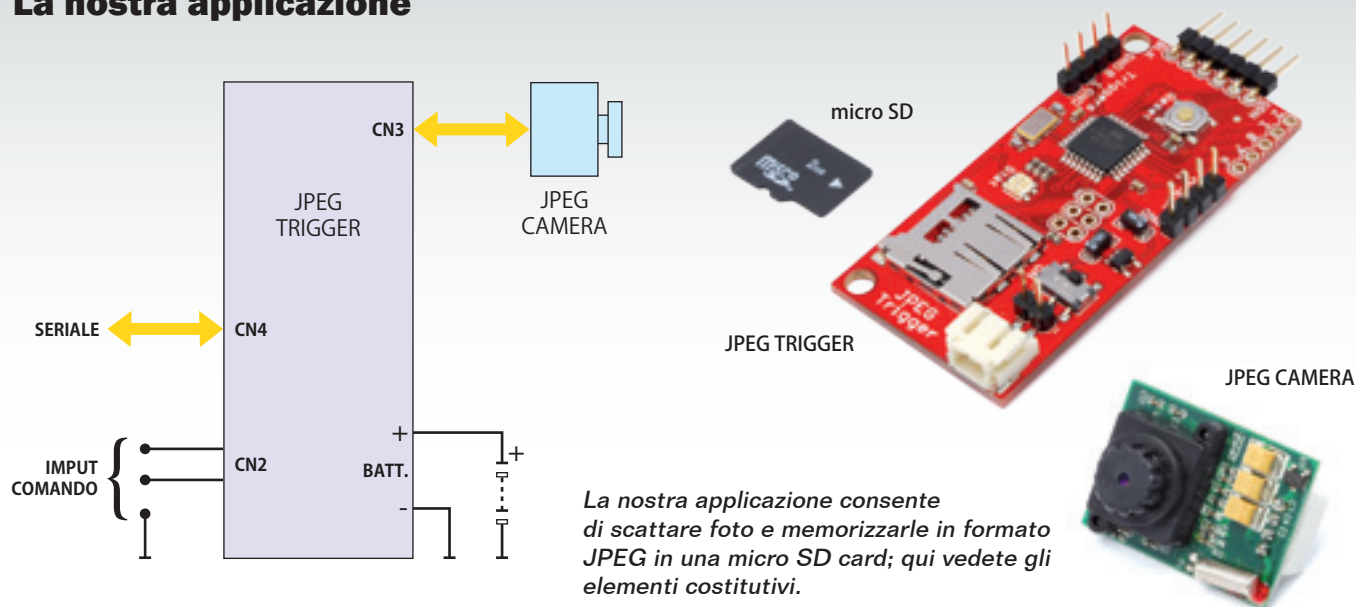
FOTOCAMERA TRIGGERABILE

di DAVIDE SCULLINO

In diversi ambiti, in special modo nella video-sorveglianza locale e da remoto, è necessario registrare le riprese di una o più telecamere anche non in tempo reale, ma programmando l'acquisizione periodica di fotogrammi ad intervalli definibili a piacimento; questo compito lo svolgono i DVR (Digital Video Recorder) ma può anche essere affidato, all'occorrenza, a dispositivi più piccoli personaliz-

zati per impieghi di nicchia. Uno di questi è certamente il progetto qui descritto, che consta di un modulo open-source chiamato JPEG Trigger, dotato di due ingressi digitali attivando uno dei quali (o anche entrambi) si forza la cattura dell'immagine al momento ripresa da una piccola telecamera digitale ad interfaccia seriale e la scrittura in una micro SD-Card (della capienza massima di 4 GB) dei dati

La nostra applicazione



corrispondenti; la telecamera utilizzata è la JPEG CAMERA, prodotta dalla ComMedia e commercializzata dalla ditta Futura Elettronica (www.futurashop.it) che contiene, oltre a una microcamera a colori (alimentata a 3,3 V e che assorbe soli 60 mA) capace di una risoluzione VGA (640x480 punti) un'UART operante a una velocità massima di 115,2 kbps ed un'unità elettronica di controllo che digitalizza l'immagine ripresa e la rende disponibile su richiesta inviata lungo la linea seriale. Più esattamente, la telecamera contiene un microcontrollore in cui gira un firmware (contenuto in una EEPROM a parte)

che, ricevuto un apposito comando seriale, acquisisce e trasforma in formato JPEG il fotogramma ripreso al momento; invia poi, i dati corrispondenti, lungo l'interfaccia seriale. Un simile progetto si presta a svariate applicazioni, che vanno dalla memorizzazione periodica di immagini riprese anche (viste le piccole dimensioni dell'insieme) senza dare troppo nell'occhio, alla registrazione a seguito del verificarsi di particolari eventi, che possono essere il passaggio di una persona o l'accesso in una zona, l'arrivo di un'automobile, l'accensione della luce in un locale buio, l'attivazione di un impianto di allarme

ed altro ancora; estremizzando, il dispositivo potrebbe anche essere abbinato ad un rilevatore di velocità per scattare fotografie digitali e memorizzarle in micro SD-Card al superamento di un certo limite, ovvero collegato ad un sensore di passaggio per fotografare un veicolo che passa un punto d'accesso (ad esempio per tenere memoria degli automezzi che entrano in un'azienda o in un parcheggio). Ancora, abbinato a una barriera ad infrarossi il dispositivo può servire a scattare immagini tipo "fotofinish" in competizioni di atletica o ciclistiche. Interfacciato con un microcontrollore che ge-

Mostra Mercato **Mondo Elettronica.net**

FIERA DELL'ELETTRONICA
INFORMATICA E RADIANTISMO

FIERA DI FERRARA 9-10 OTTOBRE

Via della Fiera, 11 - Autostrada A13 uscita "Ferrara sud"

VALIDO COME RIDOTTO (ANCHE IN FOTOCOPIA)

ORARIO 9-18,30

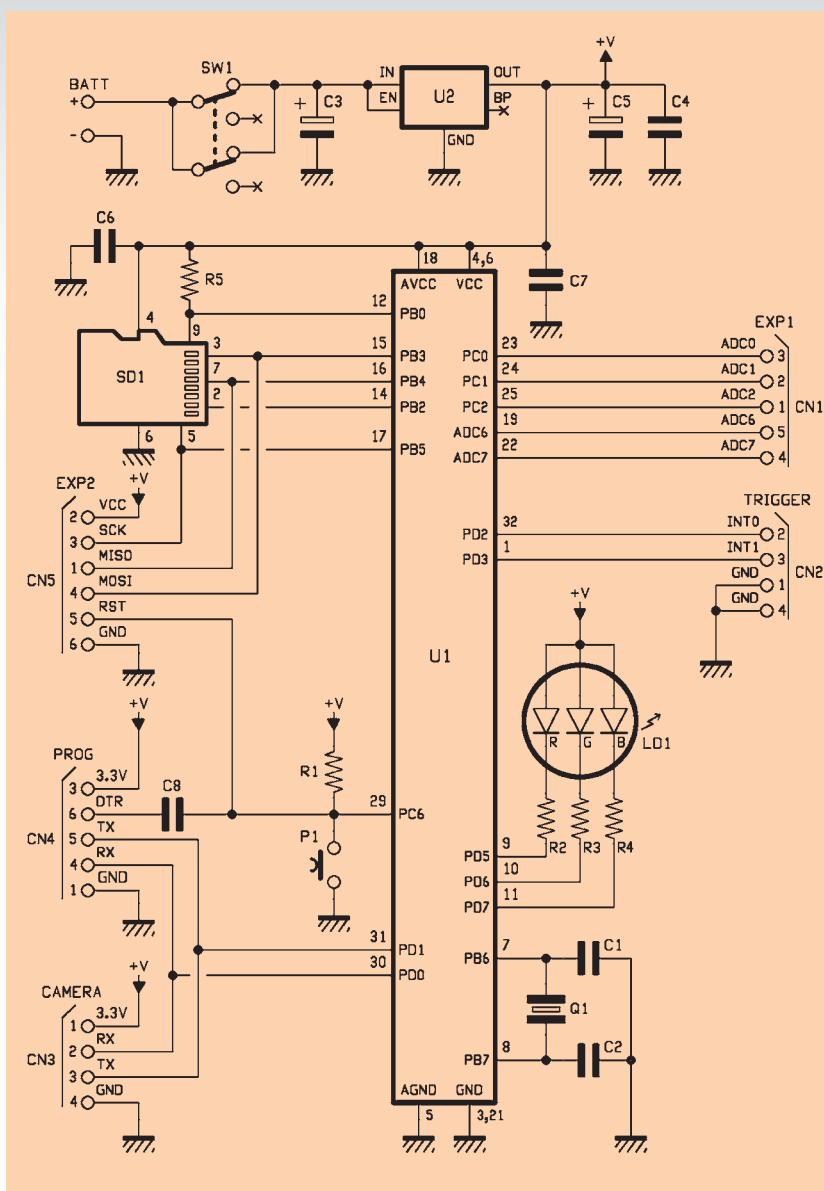
Cell. 348 9030832 - www.mondoelettronica.net

[schema **ELETTRICO**]

Delle due unità che compongono il nostro dispositivo, diamo dunque uno sguardo al modulo JPEG Trigger, analizzandone lo schema elettrico illustrato in queste pagine: il circuito è basato su un microcontrollore ATmega328 dell'Atmel, siglato U1 e funzionante con un clock scandito dall'oscillatore interno che si avvale di un quarzo da 14,7456 MHz; il micro gestisce mediante le proprie linee PB0, PB2, PB3, PB4 e PB5, la micro SD-Card, implementando un bus SPI. Per l'esattezza, il piedino 15 si comporta da MOSI (*Master Output and Slave Input*, ovvero è l'uscita dei dati diretti alla card) il 16 è il MISO (*Master Input and Slave Output*, cioè l'uscita dei dati dalla SD) mentre il 17 è il clock della comunicazione. Le stesse linee vengono ripetute sul connettore CN5, a disposizione di eventuali circuiti che necessitano di acquisire i dati sulla cattura dell'immagine in tempo reale.

del micro, corrispondenti ai canali, rispettivamente, di ricezione e trasmissione.

Le stesse linee sono in comune con il connettore CN4, che è stato montato per interfacciare il dispositivo con un Personal Computer da cui caricare il firmware o eventuali aggiornamenti; il tutto nella piena osservanza della filosofia open-source in cui il sistema si colloca. La programmazione avviene in-circuit e mediante bootloader precaricato nella flash-memory del microcontrollore: dopo l'accensione, per effettuare l'upload del firmware basta inviare un impulso di



Una videocamera da taschino

In vacanza e nella vita quotidiana, un po' tutti ci sentiamo registi e giriamo, grazie anche all'integrazione delle fotocamere nei cellulari, filmati ad ogni cosa che stimola la nostra curiosità. Per chi vuol riprendere tutto senza i limiti del cellulare e non vuol portare con sé le tradizionali videocamere, Muvi ha prodotto una camcorder che, con i suoi 5,5x2x1,6 cm, è la più piccola al mondo. Il sensore d'immagine ha una risoluzione di 2 megapixel e l'ottica presenta un'apertura angolare di ben 72 gradi; i filmati vengono registrati in formato AVI su una micro SD da 2 GB (la massima capacità utilizzabile è 8 GB) ad una risoluzione di 640x480 pixel

(la frequenza di refresh è 30 frame al secondo). La Muvi micro camcorder prevede anche il comando vocale, per attivarla quando si hanno le mani occupate. La batteria in dotazione si ricarica tramite USB, mentre i filmati possono essere ritoccati e montati con il software incluso. L'involucro è in robusto alluminio, a prova di riprese in condizioni estreme.



reset, ottenibile semplicemente facendo commutare per un istante o a permanenza la linea DTR dell'interfaccia seriale; ciò, tramite la rete C8/R1, applica un impulso a zero logico alla linea PC6 del micro, che il bootloader riconosce come richiesta di inizio della procedura di colloquio per l'acquisizione del firmware. Per la programmazione si possono percorrere due strade: la prima prevede il collegamento del CN4, ovvero delle sue linee TX, RX e DTR ad un convertitore seriale TTL/RS232 bidirezionale (come il noto MAX232 della Maxim); questa soluzione è adatta al collegamento con PC provvisti di porte COM.

La seconda via è l'interfacciamento al computer via USB, realizzabile connettendo il circuito ad uno speciale modulo d'interfaccia TTL/USB che sul lato TTL riporta anche il DTR: si tratta dell'FT232R Breakout della Sparkfun (commercializzato dalla ditta Futura Elettronica con il codice 7300-BOB0718).

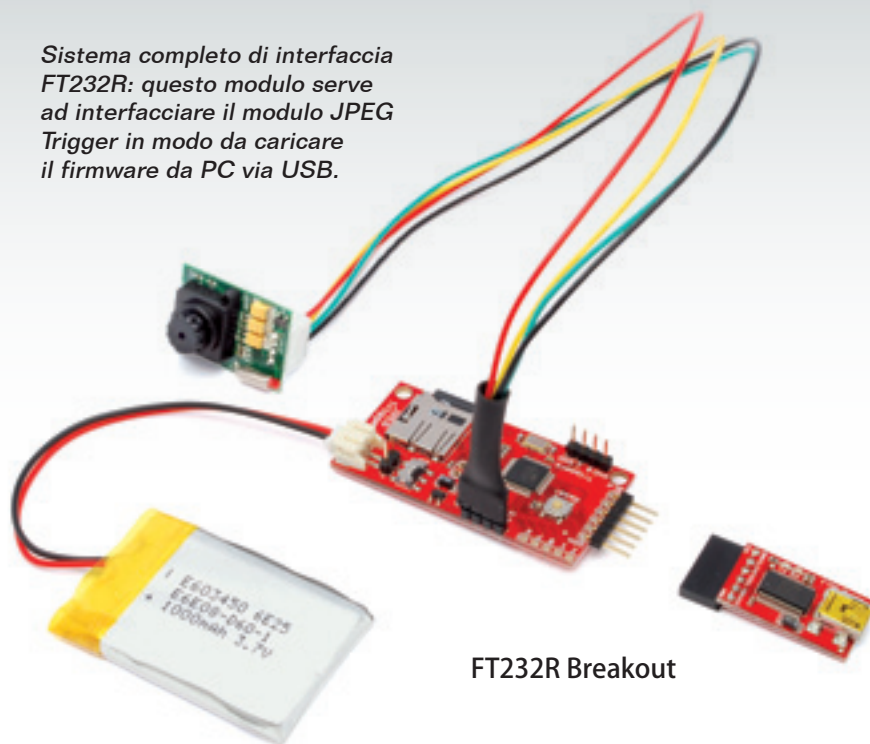
Proseguendo la carrellata sui connettori troviamo CN1, riservato ad espansioni ed utilizzabile da chi se la sente di aggiungere nuove funzioni alla scheda previo ampliamento delle capacità del firmware. Più interessante è CN2, vitale perché è il connettore che fisicamente permette di eccitare il circuito, ovvero di impartire il comando di scatto della fotografia alla telecamera JPEG e di memorizzazione in SD-Card del file corrispondente; in esso si trovano due linee, siglate INTO e INT e collegate agli I/O rispettivamente PD2 e PD3 dell'U1 (ciascuna dotata di resistore di pull-up attivato all'interno del microcontrollore

dal firmware) che permettono, portate a livello logico basso, di attivare lo scatto della foto. Proprio la possibilità di comandare il sistema con un semplice stato logico, ci consente di scattare fotogrammi e memorizzarli al verificarsi di vari eventi e semplicemente comandando il dispositivo con un'uscita a relé o a transistor NPN open-collector (o MOSFET a canale N montato ad open-drain): è sufficiente portare a zero anche per un breve istante una delle due linee o entrambe per dare il comando.

Naturalmente se una linea è già a zero logico, il comando non sortisce alcun effetto, nel senso che se i due ingressi vengono portati a livello basso uno dopo l'altro, lo scatto avviene in corrispondenza dell'arrivo a zero logico della prima in ordine cronologico. Le linee libere del microcontrollore gestiscono un LED RGB ad anodo comune usato per le segnalazioni; il triplo diodo luminoso ha il seguente comportamento:

- all'avvio, la luce rossa fissa segnala che il file-system della SD-Card non è stato inizializzato, ovvero che la card manca o è guasta;
- la luce rossa lampeggiante indica che non c'è comunicazione seriale con la telecamera JPEG o che i relativi collegamenti sono interrotti;
- la luce blu lampeggiante indica che tutto è ok, l'inizializzazione è andata a buon fine e il circuito attende il segnale di trigger sulle linee INT0/INT1;
- una luce verde lampeggiante significa che la scheda è in fase di interrogazione della telecamera JPEG e che questa sta procedendo all'acquisizione dell'immagine ed all'invio all'ATMega per la scrittura nella micro SD-Card.

Sistema completo di interfaccia FT232R: questo modulo serve ad interfacciare il modulo JPEG Trigger in modo da caricare il firmware da PC via USB.

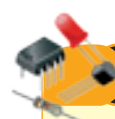


FT232R Breakout

In quest'ultimo caso, completata l'acquisizione, il led torna a lampeggiare di blu. L'intera scheda JPEG Trigger funziona con una tensione di valore compreso tra 3,7 e 9 Vcc, applicata ai punti + e - BATT; può anche essere alimentata da una semplice batteria ricaricabile al litio (basta un elemento da 3,7 V) o da una pila a secco.

La tensione d'ingresso viene regolata a 3,3 volt dal regolatore U2, che è un MIC5205-3.3 della Micrel: il componente lavora nella sua tipica configurazione e l'EN (piedino di Enable usato per attivare o mettere in standby il regolatore) è collegato fisso a 1 logico (a zero disattiverrebbe l'integrato). La tensione uscente dall'U2 viene filtrata dai condensatori C4 e C5 e va ad alimentare sia la micro SD-Card che le due linee AVCC (ossia linea di alimentazione della sezione analogica) e VCC (alimentazione della sezione digitale) del microcontrollore ATMega. Una volta alimentato, il modulo JPEG Trigger richiede che gli si carichi

il firmware; il file del caso potete scaricarlo dalla pagina web http://www.sparkfun.com/commerce/product_info.php?products_id=9459, dove trovate anche il file sorgente che, chi di voi se la sente, può modificare a piacimento per ottenere dal sistema funzioni personalizzate per specifiche applicazioni. ■



per il MATERIALE

Il sistema descritto in questo articolo utilizza una microcamera a colori jpeg con interfaccia UART (cod. JPEGCAMERA) e una scheda Jpeg Trigger (cod. 7300-DEV09459) reperibili presso Futura Elettronica. I due elementi costano rispettivamente 59,00 e 25,00 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>

PicoBlaze



Andiamo alla scoperta del “picocontrollore” Xilinx realizzabile all’interno di una FPGA e programmabile direttamente in Assembler. Prima puntata.

TUTORIAL PICOBLAZE

di MARIANO SEVERI

Le macchine a stati finiti (FSM) o automi a stati finiti, sono astrazioni matematiche che descrivono il comportamento di un sistema in termini di un numero finito di stati del sistema stesso e transizioni tra questi. Le transizioni sono dettate dai segnali in ingresso; la macchina a stati genera quindi controlli in uscita che possono dipendere dal solo stato (macchina di Moore) o anche dal valore degli ingressi (macchina di Mealy). I linguaggi di descrizione hardware (quali ad esempio il VHDL) supportano in maniera nativa la

descrizione di tali macchine. Il **Listato 1** (a pagina seguente) ne mostra un semplice esempio, che riguarda il diagramma a bolle visibile nella **Fig. 1**; in questo è stata adottata la classica notazione in cui ogni ramo riporta un indice i/j , dove il campo i rappresenta la condizione degli ingressi che detta la transizione, mentre j riporta il valore delle uscite in corrispondenza di quest’ultima.

Sebbene la sintassi sia piuttosto semplice, quando la complessità logica, in termini ad esempio di numeri di stati o di controlli d’ingresso, supera un certo li-

Listato 1

```

1. library ieee;
2. use ieee.std_logic_1164.all;
3.
4. entity example_1 is
5. port
6. (
7.   arst_n : in std_logic;
8.   clk    : in std_logic;
9.   a      : in std_logic;
10.  q       : out std_logic
11. );
12. end entity example_1;
13.
14. architecture RTL of Example_1 is
15. type t_state is (s0, s1, s2);
16. signal state_r : t_state;
17. signal state_c : t_state;
18.
19. begin
20.
21. SMReg : process (arst_n, clk)
22. begin
23.   if arst_n='0' then
24.     state_r <= s0;
25.   elsif clk'event and clk='1' then
26.     state_r <= state_c;
27.   end if;
28. end process SMReg;
29.
30. SMComb : process (a, state_r)
31. begin
32.   case state_r is
33.
34.     when s0 =>
35.       if a='1' then
36.         state_c <= s1;
37.       else
38.         state_c <= s2;
39.       end if;
40.     q <= '0';
41.
42.     when s1 =>
43.       if a='0' then
44.         state_c <= s2;
45.       else
46.         state_c <= s1;
47.       end if;
48.     q <= '1';
49.
50.     when s2 =>
51.       if a='1' then
52.         state_c <= s1;
53.       else
54.         state_c <= s2;
55.       end if;
56.     q <= '0';
57.
58.     when others =>
59.       state_c <= s0;
60.     q <= '0';
61.   end case;
62.
63. end process SMComb;
64.
65. end architecture RTL;

```

mite, la manutenzione del codice può diventare difficile, come più complessa diviene, soprattutto, la tracciabilità di tutti gli stati e delle corrette transizioni tra questi. In effetti, una macchina a

stati finiti è di per sé un circuito sequenziale e quindi, nei casi più complessi, si presta meglio ad essere realizzata con un micro-controllore.

Per contro, la natura sequenziale dei microcontrollori è il motivo principale del degrado delle prestazioni che questi dispositivi evidenziano tipicamente nella gestione del data path; in questo caso, il supporto delle logiche programmabili per la realizzazione di strutture concorrenti operanti in parallelo promette certamente migliori risultati.

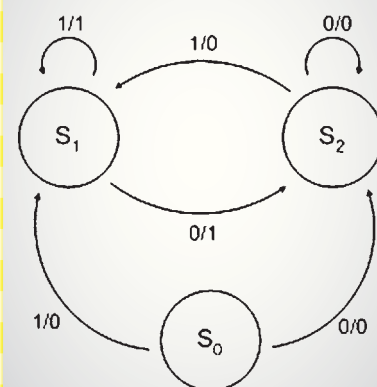
La soluzione ideale, come sempre, è di compromesso e si può ricercare nella realizzazione di architetture sequenziali micro-controllate all'interno delle FPGA, programmabili direttamente mediante assembler o linguaggio di più alto livello.

Da questo punto di vista, non stupisce il crescente interesse che sta maturando in Internet, dove infatti è piuttosto facile trovare, disponibili anche sotto licenza open-source, modelli sintetizzabili, ad esempio, di alcune delle più diffuse architetture, come PIC, 8051, AVR o Z80.

La scelta dell'architettura più adatta alla particolare applicazione dipende ovviamente da una serie di parametri non facilmente classificabili, che includono, ad esempio, l'esperienza del progettista, l'eventuale disponibilità di parte del software applicativo, eventuali costi di licenza o manutenzione, le prestazioni del sistema.

Tuttavia, se soffermiamo la nostra attenzione sulla considerazione da cui siamo partiti (ovvero, l'impiego di un micro-controllore per la gestione dei control path complessi in logiche programmabili) è indubbio che i fattori chiave sono la semplicità dell'architettura, la sua flessibi-

Fig. 1 - Diagramma a bolle (VHDL) di una semplice macchina a stati finiti.



lità e l'economicità nell'occupazione di risorse all'interno della FPGA. Questi, infatti, consentono di adattare la piattaforma alla propria applicazione, ottimizzando prestazioni e consumi, permettendo allo stesso tempo di integrare in un unico dispositivo più controllori operanti in parallelo.

Di soluzioni di questo tipo non ce ne sono molte, probabilmente anche a causa di una certa riluttanza (del tutto "filosofica") dei progettisti hardware (abituati alle descrizioni HDL) ad adottare un approccio del genere.

Tra quelle disponibili, una delle più interessanti è certamente il **PicoBlaze**, sviluppato da Xilinx per i propri dispositivi logici programmabili. Di seguito ne sono presentate le caratteristiche principali con il supporto di semplici esempi di riferimento; nel prossimo fascicolo vedremo invece come il nostro "picocontrollore" può essere usato per realizzare un'applicazione più complessa: il clone del gioco *Virtual Orchestra Studio*.

Tutti i progetti sono sviluppati per la scheda di valutazione per FPGA Spartan-3AN (nei riferimenti [3] è riportato il link alla

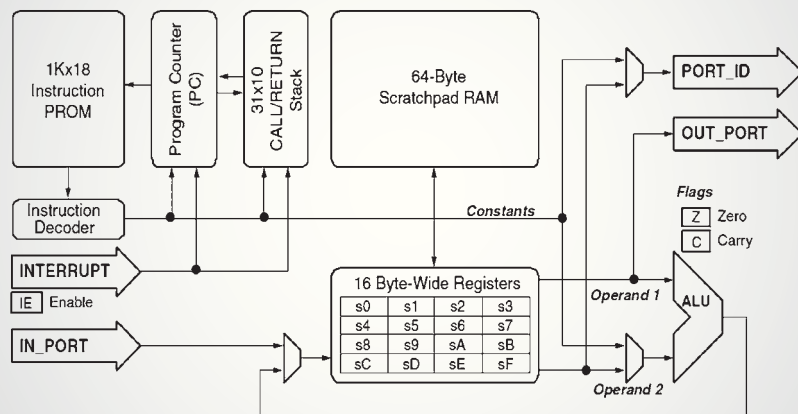
relativa documentazione) che abbiamo presentato nel corso introduttivo alle logiche programmabili pubblicato nei fascicoli dal 140 al 144 di Elettronica In, ma possono facilmente essere riadattati per altre piattaforme con FPGA Xilinx. Per quanti non avessero dimestichezza sull'utilizzo della scheda di sviluppo, la documentazione fornita a corredo è disponibile sul sito web di Xilinx (vedi [3]) chiarisce il setup da utilizzare per le prove.

UN RISC DI RIDOTTE DIMENSIONI

PicoBlaze è un microcontrollore RISC ad 8-bit sviluppato da Ken Chapman, Senior Staff Engineer presso Xilinx. Il codice sorgente contenente la descrizione del microcontrollore in linguaggio VHDL o Verilog, può essere scaricato direttamente dal sito web dell'azienda, previa registrazione, all'indirizzo [1] riportato nei riferimenti; le cartelle `<install_dir>/vhdl` e `<install_dir>/verilog` contengono dette descrizioni nei linguaggi VHDL e Verilog rispettivamente. Il file `kcpsm3.vhd` (o `kcpsm3.v` per gli utenti Verilog) contiene l'entità di più alto livello.

La Fig. 2 mostra uno schema di principio dell'architettura del PicoBlaze, desunta dalla fonte di cui al punto [2] dei riferimenti. Il picocontrollore supporta uno spazio d'indirizzamento per le istruzioni di fino a 1.024 locazioni, una memoria scratch-pad da 64 byte e fino 256 porte di I/O. Dispone di un register file che consta di 16 registri di tipo general-purpose ed una ALU ad 8-bit che supporta operazioni aritmetiche di base, quali addizione e sottrazione tra interi unsigned, operazioni logiche bit-a-bit di tipo *and*, *or* e *xor*, operazioni di confronto e comparazione bit-a-

Fig. 2 - Schema a blocchi dell'architettura del PicoBlaze).



bit, operazioni di *shift* e *rotate*. L'ALU rende, inoltre, accessibili all'utente i flag di *carry* (il cui significato dipende dall'ultima istruzione eseguita) e *zero* (che indica, invece, che tale ultima istruzione eseguita ha ritornato zero come valore). Le operazioni di moltiplicazione e divisione non sono direttamente supportate dall'ALU, ma possono essere implementate mediante connessione ad una delle porte di I/O del picocontrollore di una periferica custom (alcuni dispositivi FPGA dispongono, in particolare, di moltiplicatori embedded) o mediante routine software dedicate. Allo stesso modo possono essere emulate da software operazioni di somma e sottrazione tra operandi a più di 8 bit. Nella guida utente di cui al punto [2] dei riferimenti, sono riportati esempi propedeutici a tali applicazioni. Le istruzioni sono di lunghezza fissa, pari a 18-bit; ciascuna, in tutte le condizioni nominali, viene eseguita in due cicli di clock. Tale costanza favorisce ovviamente la determinazione delle prestazioni del sistema e la predicibilità del comportamento di quest'ultimo. Le istruzioni a singolo operando operano su operando di tipo *registro*, utilizzato sia come sor-

gente, sia come destinazione. Nel caso di istruzioni a due operandi, invece, il primo è ancora di tipo *registro* (ed anche in questo caso lo stesso registro è usato come destinazione) mentre il secondo è di tipo *registro* o *immediato* (costante). L'istruzione *load* consente di inizializzare un registro con una costante o con il contenuto di un secondo registro. Le istruzioni di *fetch* e *store* consentono invece l'accesso in lettura e scrittura alla RAM scratch-pad, mentre quelle di *input* ed *output* riguardano le porte di I/O del picocontrollore. In entrambi i casi l'indirizzamento della memoria o della porta è di tipo diretto o indiretto. Nel caso di indirizzamento indiretto, il puntatore alla locazione di memoria (o alla porta di ingresso/uscita) cui accedere è memorizzato nel register file; l'operando destinazione (nel caso di accessi in lettura) o sorgente (se si tratta di accessi in scrittura) è ancora una volta di tipo *registro*. Il PicoBlaze dispone di un registro Program Counter per l'accesso alla memoria di programma allo scopo di effettuare il fetch delle istruzioni; tale registro non è direttamente accessibile dall'utente. Il controllo del flusso di programma può essere gestito, quindi, mediante le istruzioni di salto

La capacità di calcolo varia tra 44 MIPS e 100 MIPS, in funzione della frequenza di clock e della FPGA in cui è implementato. Tali prestazioni consentono di realizzare diversi tipi di applicazioni tra le quali, ad esempio, processori DSP audio, dialer telefonici per toni DTMF, sistemi di controllo temperatura e ventole, controllo PID, controllo motori

The screenshot shows the gnuProlog 2.0.0 IDE. The main window displays a Prolog program for a 2D point. The program defines a point as a list of two numbers and includes a main predicate 'main' that prints the point. The output window shows the execution of the program, displaying the point (1, 2).

```

%gprolog 2.0.0 -f /usr/local/share/gnuProlog2/bin/gnuProlog2 -w2d_point.pl

File Edit View Settings Windows Help

GNU Prolog 2.0.0
[Icons: Run, Stop, Step Over, Step Into, Step Out, Break, etc.]

Status
[ ] Demo
[ ] Core
[ ] Break

Interrupt
[ ] Break
[ ] Error
[ ] Error

Registers
0000 00000000
0001 00000000
0002 00000000
0003 00000000
0004 00000000
0005 00000000
0006 00000000
0007 00000000
0008 00000000
0009 00000000
000A 00000000
000B 00000000
000C 00000000
000D 00000000
000E 00000000
000F 00000000
0010 00000000
0011 00000000
0012 00000000
0013 00000000
0014 00000000
0015 00000000
0016 00000000
0017 00000000
0018 00000000
0019 00000000
001A 00000000
001B 00000000
001C 00000000
001D 00000000
001E 00000000
001F 00000000
0020 00000000
0021 00000000
0022 00000000
0023 00000000
0024 00000000
0025 00000000
0026 00000000
0027 00000000
0028 00000000
0029 00000000
002A 00000000
002B 00000000
002C 00000000
002D 00000000
002E 00000000
002F 00000000
0030 00000000
0031 00000000
0032 00000000
0033 00000000
0034 00000000
0035 00000000
0036 00000000
0037 00000000
0038 00000000
0039 00000000
003A 00000000
003B 00000000
003C 00000000
003D 00000000
003E 00000000
003F 00000000
0040 00000000
0041 00000000
0042 00000000
0043 00000000
0044 00000000
0045 00000000
0046 00000000
0047 00000000
0048 00000000
0049 00000000
004A 00000000
004B 00000000
004C 00000000
004D 00000000
004E 00000000
004F 00000000
0050 00000000
0051 00000000
0052 00000000
0053 00000000
0054 00000000
0055 00000000
0056 00000000
0057 00000000
0058 00000000
0059 00000000
005A 00000000
005B 00000000
005C 00000000
005D 00000000
005E 00000000
005F 00000000
0060 00000000
0061 00000000
0062 00000000
0063 00000000
0064 00000000
0065 00000000
0066 00000000
0067 00000000
0068 00000000
0069 00000000
006A 00000000
006B 00000000
006C 00000000
006D 00000000
006E 00000000
006F 00000000
0070 00000000
0071 00000000
0072 00000000
0073 00000000
0074 00000000
0075 00000000
0076 00000000
0077 00000000
0078 00000000
0079 00000000
007A 00000000
007B 00000000
007C 00000000
007D 00000000
007E 00000000
007F 00000000
0080 00000000
0081 00000000
0082 00000000
0083 00000000
0084 00000000
0085 00000000
0086 00000000
0087 00000000
0088 00000000
0089 00000000
008A 00000000
008B 00000000
008C 00000000
008D 00000000
008E 00000000
008F 00000000
0090 00000000
0091 00000000
0092 00000000
0093 00000000
0094 00000000
0095 00000000
0096 00000000
0097 00000000
0098 00000000
0099 00000000
009A 00000000
009B 00000000
009C 00000000
009D 00000000
009E 00000000
009F 00000000
00A0 00000000
00A1 00000000
00A2 00000000
00A3 00000000
00A4 00000000
00A5 00000000
00A6 00000000
00A7 00000000
00A8 00000000
00A9 00000000
00AA 00000000
00AB 00000000
00AC 00000000
00AD 00000000
00AE 00000000
00AF 00000000
00B0 00000000
00B1 00000000
00B2 00000000
00B3 00000000
00B4 00000000
00B5 00000000
00B6 00000000
00B7 00000000
00B8 00000000
00B9 00000000
00BA 00000000
00BB 00000000
00BC 00000000
00BD 00000000
00BE 00000000
00BF 00000000
00C0 00000000
00C1 00000000
00C2 00000000
00C3 00000000
00C4 00000000
00C5 00000000
00C6 00000000
00C7 00000000
00C8 00000000
00C9 00000000
00CA 00000000
00CB 00000000
00CC 00000000
00CD 00000000
00CE 00000000
00CF 00000000
00D0 00000000
00D1 00000000
00D2 00000000
00D3 00000000
00D4 00000000
00D5 00000000
00D6 00000000
00D7 00000000
00D8 00000000
00D9 00000000
00DA 00000000
00DB 00000000
00DC 00000000
00DD 00000000
00DE 00000000
00DF 00000000
00E0 00000000
00E1 00000000
00E2 00000000
00E3 00000000
00E4 00000000
00E5 00000000
00E6 00000000
00E7 00000000
00E8 00000000
00E9 00000000
00EA 00000000
00EB 00000000
00EC 00000000
00ED 00000000
00EE 00000000
00EF 00000000
00F0 00000000
00F1 00000000
00F2 00000000
00F3 00000000
00F4 00000000
00F5 00000000
00F6 00000000
00F7 00000000
00F8 00000000
00F9 00000000
00FA 00000000
00FB 00000000
00FC 00000000
00FD 00000000
00FE 00000000
00FF 00000000
0100 00000000
0101 00000000
0102 00000000
0103 00000000
0104 00000000
0105 00000000
0106 00000000
0107 00000000
0108 00000000
0109 00000000
010A 00000000
010B 00000000
010C 00000000
010D 00000000
010E 00000000
010F 00000000
0110 00000000
0111 00000000
0112 00000000
0113 00000000
0114 000000
```

Il PicoBlaze è programmabile direttamente in linguaggio assembler. Nella directory `<install_dir>/assembler` è incluso il compilatore *kcpasm3*, eseguibile direttamente da riga di comando, che consente di generare una descrizione HDL della memoria ROM contenente l'applicativo in formato binario nativo. Il file *embedded_kcpasm3.vhd* nella directory `<install_dir>/vhdl` (o, allo stesso modo, per quanti preferiscano usare il Verilog, il file *embedded_kcpasm3.v* nella directory `<install_dir>/verilog`) riporta la descrizione di alto livello con le connessioni tra il core PicoBlaze e la memoria che contiene, per l'appunto, l'eseguibile (e che sarà anch'essa sintetizzata all'interno della FPGA utilizzando i banchi di RAM embedded disponibili

Nella pratica, considerati i ridotti tempi necessari per ricompilare il progetto nel caso si apportino

modifiche all'architettura della FPGA, la possibilità di scaricare direttamente sul dispositivo target un nuovo applicativo (come vedremo in seguito) mediante la porta JTAG, e anche la disponibilità dei tool di debug in circuit come Chipscope (resi disponibili direttamente da Xilinx), è in genere preferibile effettuare il test del sistema direttamente sul campo. In alternativa, essendo il PicoBlaze distribuito in formato sorgente, si può sempre pensare di utilizzare classici tool di simulazione HDL; ISE Webpack, l'ambiente di sviluppo per FPGA Xilinx disponibile gratuitamente per il download dal sito web dell'azienda, include un simulatore proprietario che può utilmente venire impiegato in questo caso.

Nei prossimi paragrafi vedremo, con alcuni semplici esempi, come programmare il PicoBlaze e realizzare applicazioni custom. Come accennato in precedenza, faremo riferimento alla scheda di sviluppo per Spartan-3AN (vedere punto [3] in riferimenti) come piattaforma di lavoro.

UNA CONFIGURAZIONE MINIMALE

La Fig. 4 mostra lo schema di principio della configurazione minimale di un sistema basato su PicoBlaze; per semplicità, le linee di clock e reset sono state omesse. L'architettura include la CPU e la memoria embedded per il codice applicativo e due porte, di cui una d'ingresso ed una di uscita. Quella d'ingresso è connessa a degli switch da SW0 a SW3 della scheda di sviluppo, mentre la porta di uscita ai led da LED0 a LED3. La corrispondente descrizione HDL si trova nel file *simple_architecture.vhd* incluso nella directory *example/src* del progetto di esempio in allegato a questo

Fig. 4 - Una configurazione minimale per il PicoBlaze.

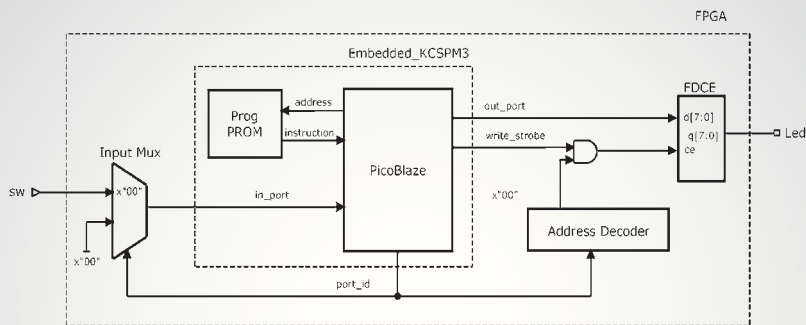
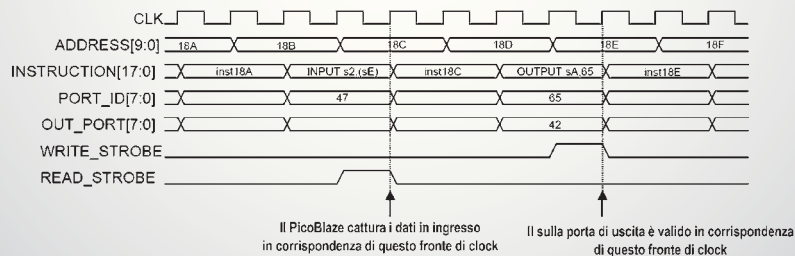


Fig. 5 - Timing di accesso alle porte di ingresso ed uscita del PicoBlaze.



articolo e disponibile per il download dal sito Internet della nostra rivista (www.elettronica.in).

Come accennato in precedenza, il componente *embedded_KCSPM3* è descritto nell'omonimo file (con estensione .vhd) incluso nella directory *<install_dir>/vhdl* della distribuzione di cui al punto [1] dei riferimenti e scaricabile dal sito web Xilinx.

Abbiamo quindi aggiunto, nella descrizione di più alto livello, la logica necessaria per poter gestire le porte di ingresso e di uscita.

La Fig. 5 mostra, in linea generale, il timing che il PicoBlaze

utilizza per tali accessi. Come si vede nella figura, l'operazione dura due cicli di clock; PORT_ID segnala l'indirizzo della porta cui s'intende accedere. Nel caso di accessi in lettura, quindi, la linea STROBE segnala l'acquisizione del dato presente sulla linea IN_PORT.

Il Listato 2 riporta il codice necessario per gestire le porte di ingresso. Come si vede in esso, l'istruzione "case" alla riga 7 crea un multiplexer per selezionare la porta da acquisire e correntemente indirizzata.

È il caso di osservare che in realtà,

Listato 2 - Codice HDL per la gestione delle porte di ingresso del PicoBlaze

```

1. InPort : process (arst_n, clk)
2. begin
3.   if arst_n = '0' then
4.     in_port<=x"00";
5.   elsif clk'event and clk='1' then
6.     case port_id is
7.       when x"00" =>
8.         in_port<=x"0" & sw;
9.       when others =>
10.        in_port<=x"00";
11.     end case;
12.   end if;
13. end process InPort;
```

Listato 3 – Codice HDL per la gestione delle porte di uscita del PicoBlaze

```
1. OutLed : process (arst_n, clk)
2.   variable v : integer;
3.   Begin
4.     if arst_n = '0' then
5.       led_reg<=x"00";
6.     elsif clk'event and clk='1' then
7.       if (write_strobe='1' and port_id=x"00") then
8.         led_reg <= out_port;
9.       end if;
10.    end if;
11. end process OutLed;
```

nel nostro esempio, è presente una sola porta; avremmo quindi potuto rimuovere il multiplexer e connettere la stessa direttamente ad IN_PORT. Abbiamo scelto la strada che vedete, semplicemente per dare evidenza del modo corretto di procedere in generale; in questo caso, infatti, dovrà essere aggiunta una serie di istruzioni del tipo riportato alle righe 7-8 per ogni porta presente nel progetto. Nell'esempio abbiamo inoltre ommesso di spiegare, per semplicità ed in quanto non strettamente inerenti all'argomento, l'implementazione della logica di sincronizzazione delle porte di ingresso che non siano sincrone con il clock di sistema e di anti-rimbalzo, nel caso siano da acquisire pulsanti e switch. Ora vediamo, invece, come gestire gli accessi in scrittura. Come mostrato nella **Fig. 5**, la linea OUT_PORT del PicoBlaze riporta il valore da scrivere sulla porta indirizzata; la linea WRITE_STROBE è attiva nel secondo ciclo di clock per confermare l'accesso alla porta.

Il **Listato 3** riporta il codice HDL necessario per creare la porta in accordo con tale protocollo di accesso; come si vede alla riga 8, è presente un controllo sull'indirizzo della porta per abilitare la scrittura.

Nel caso, più generale, in cui siano necessarie più di una porta di uscita, per ognuna di esse dovrà essere istanziato nel codice sorgente un processo del tipo

mostrato nel listato.

Nella guida utente (punto [2] dei riferimenti) sono spiegate alcune soluzioni per ottimizzare la decodifica d'indirizzo nell'accesso alle porte di uscita qualora quella semplicistica proposta nel **Listato 2** non consenta di soddisfare i requisiti di massima frequenza di funzionamento del dispositivo. Una volta creata l'architettura hardware ed istanziate le risorse necessarie, supponiamo ora di voler sviluppare un semplice applicativo che accenda o spenga i LED in funzione del valore degli switch; tale applicativo consiste di un ciclo infinito che, in sequenza, accede alla porta d'ingresso per leggere lo stato degli switch e quindi alla porta di uscita per aggiornare corrispondentemente lo stato dei LED.

Il codice (il sorgente *simple_led.psm* si può trovare nella directory *example/code* del nostro progetto di esempio) è riportato nel **Listato 4** ed è scritto per il compilatore incluso in pBlazeIDE; per compilare l'applicativo con *kcpsm3* sono necessarie alcune modifiche (documentate nella guida utente di cui al punto [2] dei riferimenti).

L'istruzione alla riga 1 istruisce il compilatore sul file (*ROM_blank.vhd*) da utilizzare come "template" per la generazione del listato (*simple_led_rom.vhd*) contenente la descrizione della ROM con l'applicativo compilato; il campo *prog_rom* definisce il nome dell'entità che sarà creata

all'interno del file. Le successive istruzioni definiscono mediante le direttive DSIN e DSOUT, le due porte di ingresso e di uscita, Led ed Sw, entrambe mappate ad indirizzo 0x00; le direttive in oggetto sono supportate esclusivamente dal simulatore ISS incluso in pBlazeIDE. L'istruzione ORG \$0 istruisce quindi il compilatore sulla locazione di partenza in memoria ROM del codice compilato che segue; il PicoBlaze inizia ad eseguire dopo il reset dalla locazione 0x00.

L'istruzione alla riga 7 definisce l'entry point BEGIN; seguono quindi le due istruzioni di lettura della porta di ingresso (corrispondente allo stato degli switch) nel registro s0 e di copia del contenuto di questo sulla porta di uscita (per l'aggiornamento del valore dei LED). L'istruzione di salto finale riporta il PC all'entry point BEGIN così da realizzare un ciclo infinito.

Per compilare l'applicativo di esempio presentato è sufficiente eseguire *simulate->simulate* da pBlazeIDE. Se non vengono riscontrati errori, nella directory di lavoro viene creato il file *simple_led_rom.vhd*. Possiamo quindi verificare con il simulatore che il progetto funzioni correttamente; eseguiamo *simulate->run*. A destra dell'editor di test compaiono rappresentazioni grafiche degli switch e dei LED che abbiamo dichiarato; abilitando uno switch vedremo illuminarsi il LED corrispondente.

Il passo successivo è ovviamente implementare sull'hardware reale il nostro esempio. Nella cartella *example/xproj*, all'interno del package scaricato dal sito web della nostra rivista come allegato al presente articolo, è presente il progetto per la versione 12 dell'ambiente di sviluppo ISE WebPack.

Tale ambiente può essere scaricato gratuitamente, previa registrazione, alla pagina indicata nei riferimenti al punto [5].

Unitamente al progetto di riferimento, è possibile scaricare dal nostro sito anche una guida passo-passo che mostra come ricreare l'esempio partendo da zero.

Dopo aver aperto il progetto in ISE, dalla finestra che si trova nel riquadro a sinistra, dove è presente la lista dei file di progetto, selezioniamo *simple_design.vhd* con il tasto destro del mouse e quindi la voce "Set as top module"; quindi dallo stesso menu di popup, eseguiamo il comando "Implement top module".

Per evitare errori, può essere opportuno verificare che il file contenente la descrizione della ROM da implementare, corrisponda a quello in precedenza generato dal compilatore pBlazeIDE.

Al termine della fase di place&route, se non sono stati segnalati errori, sarà possibile programmare l'FPGA sulla scheda di sviluppo, eseguendo il comando "Configure Target Device" dalla finestra *Processes*, disposta sulla sinistra dell'ambiente di lavoro. Ricordate che, per consentire la programmazione della FPGA, la scheda di sviluppo deve essere connessa al PC mediante il cavo USB in dotazione.

Se la programmazione si è conclusa senza errori, dovremmo finalmente avere il nostro sistema attivo e funzionante: commutando gli switch, i LED dovrebbero accendersi e spegnersi.

UPLOAD

DEL SOFTWARE APPLICATIVO

Nella configurazione minimale che abbiamo presentato, il codice applicativo è memorizzato nella memoria ROM che è inizializzata alla configurazione della FPGA.

Listato 4 - Codice assembler *simple_led.psm*

```
1. VHDL "ROM_blank.vhd", "simple_led_rom.vhd", "prog_rom"
2. Sw DSIN $0 ; Create input port Sw linked to port address 0
3. Led DSOUT $0 ; Create output port Led linked to port address 0
4.
5. ORG $0 ; Programs always start at reset vector 0
6.
7. BEGIN:
8. IN s0,Sw ; read Sw input port value into s0 register
9. OUT s0,Led ; write the content of s0 into out port Led
10. JUMP BEGIN ; infinite loop
```

Modificare l'applicativo richiede di modificare il contenuto della ROM.

Ci sono diversi modi per fare questo: il più semplice è ovviamente compilare il programma e sintetizzare il progetto utilizzando la nuova descrizione della memoria ROM correttamente aggiornata. Sebbene nei progetti più semplici questo non richieda troppo tempo, può apparire piuttosto noioso nel caso siano necessarie diverse modifiche durante la fase di debug.

Un'alternativa è quella di utilizzare l'utilità JTAG_Loader inclusa nella directory *<install_dir>\JTAG_loader*; qui è mostrato come dotare la memoria istruzione di una porta di scrittura ed accederevi mediante la porta JTAG della FPGA. In questo modo è possibile scaricare direttamente nella memoria on-chip il nuovo applicativo senza dover, appunto, sintetizzare l'intero progetto o riprogrammare l'FPGA.

L'utilità si basa sull'uso dell'assembler *kcpsm3* fornito da Xilinx, ma nella pagina web riportata nei riferimenti al punto [6] è mostrato come utilizzarla con pBlazeIDE. Supporta un solo picocontrollore (quindi non è

applicabile nel caso di sistemi multi-core) ed usa una primitiva BSCAN per la gestione della porta JTAG, perciò non è applicabile nel caso il dispositivo FPGA abbia disponibile una sola risorsa del genere e questa sia già impiegata (come, ad esempio, qualora si utilizzi ChipScope per il debug in-circuit del circuito).

Nel caso più generale, la soluzione preferibile è l'aggiornamento del contenuto della memoria programma direttamente all'interno del file di programmazione del dispositivo (mediante l'utilità *data2mem* inclusa nell'ambiente di sviluppo ISE Webpack) e la successiva riprogrammazione dello stesso.

Nella directory *<install_dir>/DATA2MEM_assistance* è mostrato come utilizzare tale soluzione con l'assembler *kcpsm3*. Per utilizzarla, invece, con pBlazeIDE, è sufficiente aggiungere nel sorgente alcune direttive, come mostrato nel **Listato 5** alle linee da 2 a 4.

La prima direttiva (linea 2) istruisce il compilatore allo scopo di generare l'immagine della ROM programma in formato MEM compatibile con il tool DATA-2MEM.

Listato 5 - Direttive per il pBlazeIDE per l'upload dell'applicativo

```
1. VHDL "ROM_blank.vhd", "simple_led_rom.vhd", "prog_rom"
2. MEM "pblaze.mem"
3. EXEC "update_bitstream.bat"
4. EXEC "impact -batch download.cmd"
```

Listato 6 - Descrizione HDL di un timer

```
1. Timer : process (arst_n, clk)
2.   variable v : integer;
3.   begin
4.     if arst_n = '0' then
5.       v:=0; timer_Irq<='0';
6.     elsif clk'event and clk='1' then
7.       if v=500000000 then
8.         timer_Irq<='1';
9.       else
10.        Timer_Irq<='0';
11.      end if;
12.      v:=v+1;
13.    end if;
14.  end process Timer;
```

La seconda (linea 3) istruisce il tool pBlazeIDE ad eseguire, al termine della compilazione, il file batch *update_bitstream.bat* per l'aggiornamento del bitstream di configurazione; l'ultima direttiva (linea 4) esegue la programmazione della FPGA e può essere rimossa qualora non si voglia eseguire in automatico tale operazione. Il file di comando *update_bitstream.bat* è fornito unitamente ai progetti di esempio che stiamo presentando ed è incluso nella cartella *example/code*; è praticamente identico al file di comando fornito da Xilinx per l'uso di DATA2MEM con kcp-sm3, tranne che per la mancanza della chiamata a tale assembler. Il file *download.cmd*, anch'esso presente in *example/code*, definisce invece le opzioni di programmazione dell'FPGA ed è scritto per la scheda di sviluppo per Spartan-3AN cui ci stiamo riferendo.

Qualora si intenda usarlo su altre piattaforme, sarà necessario modificare opportunamente i

parametri di configurazione che vi sono dichiarati.

Un'interessante variante dell'aggiornamento del bitstream di programmazione che vi abbiamo appena descritto, è invece quella che si può trovare all'indirizzo web indicato, nei riferimenti, al punto [7]. In tal caso, in testa alla ROM viene memorizzato un bootloader che consente il caricamento del software applicativo nella parte bassa della memoria mediante porta di comunicazione seriale. In questo modo non è necessario modificare il bitstream di programmazione, né riprogrammare il dispositivo per aggiornare l'applicativo che si intende eseguire.

Tale soluzione può anche essere utile in quelle applicazioni che prevedano l'esecuzione da parte del picocontrollore di diversi applicativi in funzione delle particolari condizioni operative.

LA GESTIONE DELLE INTERRUZIONI

Il PicoBlaze dispone, come accennato inizialmente, di una linea d'interruzione; se abilitata, in risposta all'asserzione di quest'ultima, il picocontrollore esegue un salto all'indirizzo 0x3ff.

Il riconoscimento dell'interruzione è quindi segnalato dall'attivazione della linea *interrupt_ack*.

Dopo il salto, l'interrupt viene disabilitato per evitare un ciclo infinito; è responsabilità del soft-

ware applicativo abilitare nuovamente l'interrupt al termine della routine di servizio conseguente all'interruzione.

Qui di seguito viene spiegato come gestire le interruzioni; allo scopo faremo riferimento ad un semplice esempio che istanzia un timer nell'architettura di sistema. Il suddetto timer genera un'interruzione ogni secondo, in risposta alla quale il PicoBlaze commuta lo stato di un LED.

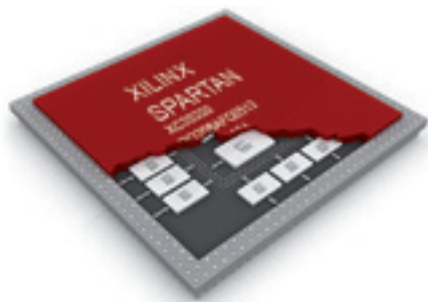
Il file sorgente necessario per tale configurazione, *irq_example.vhd*, è, come per gli altri visti in precedenza, disponibile per il download dal nostro sito web; il file *irq_example.psm* contiene il corrispondente applicativo, mentre in *irq_example_rom.vhd* è presente la descrizione (creata da pBlazeIDE) della memoria istruzioni conseguentemente inizializzata.

Il Listato 6 estrae dal file sorgente la sezione di codice per la descrizione del timer; si tratta di un semplice contatore il cui fondo scala è fissato a 500 milioni, così da derivare un impulso di frequenza pari ad 1 secondo dal clock locale della scheda (che ha una frequenza di 50 MHz).

Come tutte le macchine sequenziali, il PicoBlaze non presenta una risposta immediata alle interruzioni, ragion per cui l'evento deve essere memorizzato. Il Listato 7 descrive la logica per tale scopo.

Come si vede, la linea di interruzione viene asserita (riga 7) in corrispondenza dell'evento del timer e rilasciata (linea 4) quando il picocontrollore ha riconosciuta l'interrupt. Il codice per l'applicativo di esempio è, infine, mostrato nel Listato 8.

Il sistema esegue un ciclo infinito (linee 9-10) in cui non compie alcuna azione. Alla locazione 0x3ff a cui il PC viene fatto puntare in



Listato 7 – Descrizione HDL di un timer

```
1. PicoIrq : process (arst_n, clk)
2. begin
3.   if arst_n = '0' then
4.     Interrupt<='0';
5.   elsif clk'event and clk='1' then
6.     if InterruptAck='1' then
7.       Interrupt<='0';
8.     elsif Timer_Irq='1' then
9.       Interrupt<='1';
10.    else
11.      Interrupt<=Interrupt;
12.    end if;
13.  end process PicoIrq;
```

corrispondenza di una interruzione è memorizzata (linee 17-18) l'istruzione di salto alla routine ISR di gestione dell'interrupt; quest'ultima inverte il valore del registro s0 e ne scrive il valore sulla porta di uscita all'indirizzo 0x00. La routine ritorna, quindi, al controllo del programma, riabilitando poi il riconoscimento dell'interrupt.

Per provare questo esempio, è necessario compilare l'applicativo, quindi selezionare dal progetto ISE di riferimento il corrispondente file *'top module'* e procedere come descritto nel paragrafo precedente.

LA GESTIONE DELLA MEMORIA SCRATCH-PAD

Come accennato in precedenza, il PicoBlaze dispone di una memo-

ria scratch-pad della capacità di 64 byte accessibile, con indirizzamento diretto o indiretto, sia in lettura che in scrittura, mediante le istruzioni *fetch* e *store*. Nella guida utente del PicoBlaze (punto [2] in riferimenti) è mostrato come usare la memoria per generare una look-up table o implementare uno stack per il register file nel caso di chiamate a subroutine.

Lo stack implementato in maniera nativa dal PicoBlaze memorizza, infatti, soltanto il Program Counter in corrispondenza delle chiamate; non viene salvato, invece, lo stato del register file che è quindi comune a tutte le funzioni.

Nell'esempio riportato di seguito, vedremo invece come utilizzare la memoria scratch-pad per

creare una coda per la gestione di dati mediante un semplice schema producer-consumer. In questo esempio, la funzione *producer* riempie costantemente la coda con un contatore one-hot. Il consumer viene chiamato, quindi, in risposta ad una interruzione generata da un timer ad 1 secondo; legge il dato dalla coda e lo invia su una porta di uscita associata ai led della scheda di sviluppo.

Nella pratica, dovrebbero vedersi i LED lampeggiare in sequenza ad una frequenza di 1 secondo esatto.

Il file con la descrizione dell'architettura hardware di tale progetto è *ScratchpadRAM_example.vhd*. Nel Listato 9 è riportato invece l'estratto del codice applicativo (si veda il file *ScratchpadRAM_example.psm*)

Come si vede la coda è gestita mediante due variabili, Pntr e Cnt, che rappresentano, rispettivamente, la testa della coda e la sua lunghezza. La coda ha una dimensione massima di 16 locazioni.

Il produttore verifica inizialmente che la coda non sia piena (linea 22). Se non lo è, calcola il valore del puntatore di scrittura (linea

Listato 8 – Codice assembler irq_example.psm

```
1. VHDL "ROM_blank.vhd", "irq_example_rom.vhd", "prog_rom"
2.
3. Led DSOUT 0; Create output port Led linked to port address 0
4.
5. ORG 0 ; Programs always start at reset vector 0
6.
7. EINT
8.
9. BEGIN:
10. JUMP BEGIN
11.
12. ISR: ; Interrupts are automatically disabled when an interrupt is recognized
13. XOR s0, 1
14. OUT s0, 0
15. RETI ENABLE ; Return from interrupt service routine
16.
17. ORG $3FF ; Interrupt vector is located at highest instruction address
18. JUMP ISR ; Jump to interrupt service routine, ISR
```

Listato 9 – Codice assembler scratchpadram_example.psm

```
1. VHDL "ROM_blank.vhd", "scratchpadram_example_code.vhd", "prog_rom"
2.
3. Led DSOUT 0           ; Create output port Led linked to port address 0
4. Pntr EQU s0           ; define name for register s0
5. Cnt EQU s1            ; define name for register s1
6. WrPntr EQU s2         ; define name for register s2
7. WrData EQU s3         ; define name for register s3
8. RdData EQU s4         ; define name for register s4
9.
10. ORG 0                ; Programs always start at reset vector 0
11.
12. EINT ; If using interrupts, be sure to enable the INTERRUPT input
13.
14. INIT:
15. LOAD Pntr, 0
16. LOAD Cnt, 0
17. LOAD WrPntr, 0
18. LOAD WrData, 1
19. JUMP PRODUCER
20.
21. PRODUCER:
22. COMP Cnt, $F; check queue for full
23. JUMP Z, PRODUCER
24. LOAD WrPntr, Pntr
25. ADD WrPntr, Cnt
26. AND WrPntr, $F
27. STORE WrData, WrPntr
28. ADD Cnt, 1
29. AND Cnt, $F
30. RL WrData
31. JUMP PRODUCER
32.
33. CONSUMER:
34. COMP Cnt, 0; check queue for empty
35. JUMP Z, ISR_RETURN
36. FETCH RdData, Pntr
37. ADD Pntr, 1
38. AND Pntr, $F
39. SUB Cnt, 1
40. AND Cnt, $F
41. OUT RdData, Led
42. JUMP ISR_RETURN
43. ISR_RETURN:
44. RETI ENABLE ; Return from interrupt service routine
45.
46. ORG $3FF           ; Interrupt vector is located at highest instruction address
47. JUMP CONSUMER      ; Jump to interrupt service routine, ISR
```

24-26) e quindi scrive il dato in memoria scratchpad (linea 27); infine aggiorna il valore della lunghezza della coda (linea 28-29) e del prossimo dato da scrivere (30).

Il consumer, invece, viene chiamato (come si diceva poc'anzi) in risposta ad una interruzione (si vedano le linee 46-47), verifica che la coda non sia vuota (linea 34) e, se in quest'ultima sono invece presenti dati, procede a leggere il valore correntemente in testa ad essa (linea 36); quindi aggiorna il puntatore alla testa della coda (linea 37-38) e la sua lunghezza (linee 38-39).

Infine, scrive il valore letto dalla coda sulla porta di uscita.

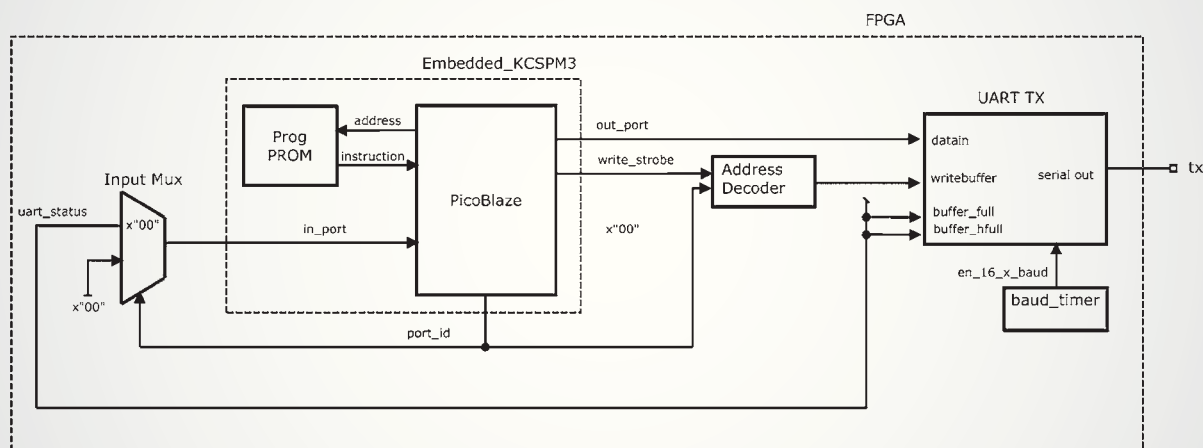
PERIFERICHE CUSTOM

Uno dei vantaggi principali del PicoBlaze è la flessibilità dell'architettura. Per ridurre l'occupazione di risorse il picocontrollore non dispone di periferiche standard. Periferiche custom possono però essere aggiunte al sistema sfruttando le porte di I/O. Di seguito vedremo in particolare come dotare il sistema di una porta di comunicazione UART. Questo ci permetterà di eseguire un classico programma del tipo "Hello World!".

Nella directory <install_dir>/vhdl sono presenti i file *uart_rx.vhd* e *uart_tx.vhd* che implementano, rispettivamente, ricevitore e trasmettitore UART (con protocollo 8-N-1). I moduli integrano buffer dati di 16 byte.

Per una descrizione dettagliata delle interfacce di tali moduli e del loro funzionamento si può consultare la guida *Uart_Manual.pdf* presente all'interno della distribuzione (scaricata dall'indirizzo web al punto [1] nei riferimenti) contenente tutti gli altri file di progetto per il PicoBlaze. La Fig. 6 mostra uno schema a blocchi di un sistema basato su

Fig. 6 - Un sistema basato su PicoBlaze con trasmettitore UART.



PicoBlaze, che integra il pico-controllore, con la memoria di programma, e il modulo Uart_Tx. La corrispondente descrizione HDL si può trovare nel file *hel-*

lo_world.vhd incluso negli esempi di riferimento allegati al presente articolo e disponibili per il download dal sito della nostra rivista. Come si vede dalla figura, un

divisore di clock chiamato *baud_timer* genera un opportuno segnale di sincronizzazione, ad una frequenza pari a 16 volte il baud-rate della comunicazione

Listato 10 - Codice assembler *hello_world.psm*

```

1.  VHDL "ROM_blank.vhd", "hello_world_rom.vhd", "prog_rom"
2.  UART_status_port EQU $00          ; UART status input
3.  tx_half_full EQU $01              ; UART Tx fifo half full flah mapped to bit 0 in UART_status_port
4.  tx_full EQU $02                   ; UART Tx fifo half full flah mapped to bit 1 in UART_status_port
5.  UART_write_port EQU $01           ; UART Tx data output
6.  UART_data EQU sf                  ; used to pass data to and from the UART
7.
8.  Main:
9.  LOAD UART_data, character_h
10. CALL send_to_UART
11. LOAD UART_data, character_e
12. CALL send_to_UART
13. LOAD UART_data, character_l
14. CALL send_to_UART
15. LOAD UART_data, character_l
16. CALL send_to_UART
17. LOAD UART_data, character_o
18. CALL send_to_UART
19. LOAD UART_data, character_BS
20. CALL send_to_UART
21. LOAD UART_data, character_w
22. CALL send_to_UART
23. LOAD UART_data, character_o
24. CALL send_to_UART
25. LOAD UART_data, character_r
26. CALL send_to_UART
27. LOAD UART_data, character_l
28. CALL send_to_UART
29. LOAD UART_data, character_d
30. CALL send_to_UART
31. LOAD UART_data, character_CR
32.
33. send_to_UART: IN s0, UART_status_port          ;test Tx_FIFO buffer
34. TEST s0, tx_full
35. JUMP Z, UART_write
36. JUMP send_to_UART
37.
38. UART_write:
39. OUT UART_data, UART_write_port
40. RET

```

Riferimenti

- [1] <http://www.xilinx.com/products/ipcenter/picoblaze-S3-V2-Pro.htm>
- [2] Xilinx "PicoBlaze 8-bit Embedded Microcontroller User Guide" – ug129.pdf
- [3] <http://www.xilinx.com/products/devkits/HW-SPAR3AN-SK-UNI-G.htm>
- [4] <http://www.mediatronix.com/pBlazeIDE.htm>
- [5] <http://www.xilinx.com/tools/webpack.htm>
- [6] http://forums.xilinx.com/xlnx/board/crawl_message?board.id=PicoBlaze&message.id=218
- [7] <http://sites.google.com/a/systemnicola.org/www/bootloadprog>

da implementare sulla linea seriale.

Una porta di uscita del PicoBlaze è utilizzata per passare al trasmettitore i dati da serializzare, mentre i segnali di stato del buffer dati di questo sono accessibili mediante una porta di ingresso del picocontrollore.

La gestione delle porte d'ingresso e di uscita segue lo schema tracciato in precedenza; per la porta di uscita connessa all'ingresso dal trasmettitore UART è creata solo la logica per la decodifica di indirizzo in quanto il registro per la memorizzazione del dato è presente nel modulo stesso.

Il **Listato 10** riporta il codice applicativo scritto per eseguire il classico programma "Hello World!".

Il programma principale consiste, come si può notare, in una serie di scritture sulla porta di uscita dei caratteri che compongono il messaggio di benvenuto.

Ogni scrittura consiste in un caricamento del carattere nel registro s0 e quindi di invio di quest'ultimo sulla linea seriale. Tale invio è gestito mediante la routine `sent_to_UART` (linea 33) che dapprima verifica che la FIFO di trasmissione non sia piena (linea 34), quindi richiama la funzione `UART_Write` (linea 35).

Tale funzione copia il contenuto del registro di trasmissione temporaneo sulla porta di uscita (linea 39).

Un esempio più complesso di quanto presentato finora è incluso nella directory `<install_dir>/vhdl`.

Il file `uart_clock.vhd` descrive l'architettura per realizzare un real-time clock in grado di ricevere ed interpretare da porta seriale alcuni comandi; tra questi vi sono, ad esempio, un comando per richiedere l'invio, sempre su porta seriale, del tempo corrente, uno per impostare tale tempo ad

un valore specificato, un altro per definire un tempo di allarme in corrispondenza del quale inviare un messaggio di notifica.

Il file `<install_dir>/assembler/uclock.psm` contiene il codice corrispondente all'applicazione.

CONCLUSIONI

In questo articolo abbiamo presentato il PicoBlaze, un controllore ad 8 bit di ridotte dimensioni implementabile in FPGA, programmabile direttamente in assembler, flessibile ed espandibile in funzione della particolare applicazione.

Ne abbiamo presentato in maniera sparsa le diverse risorse e mostrato come usarle, avvalendoci di semplici esempi.

Nella prossima puntata vedremo come mettere insieme tutto quello che abbiamo appreso in queste pagine per realizzare un divertente clone del gioco *Virtual Orchestra Studio*. ■



Mostra Mercato **Mondo Elettronica.net** 6ª EDIZIONE

FIERA DELL'ELETTRONICA
INFORMATICA E RADIANTISMO
ORARIO 9,00/18,30 - PARCHEGGIO GRATUITO

BOLOGNA **PALANORD - PARCO NORD**
Uscita Tangenziale n. 7 bis

23-24 OTTOBRE 2010

Tel. 0545 27548 - Cell. 348 9030832 - www.mondoelettronica.net

VALE COME RIDOTTO
ANCHE IN FOTOCOPIA

TOUCH SENSING

Scopriamo le tecnologie che quotidianamente ci permettono di usare apparecchi di vario genere senza toccarli fisicamente o semplicemente sfiorando uno schermo interattivo.

..... di MASSIMILIANO LEGGENDA

I display di tipo touch-screen hanno ormai invaso il mercato dei componenti elettronici: sono impiegati in un gran numero di dispositivi che utilizziamo comunemente, dalle macchine che distribuiscono i biglietti dei treni in tutte le stazioni

ferroviarie agli smartphone più avanzati. Tanti costruttori utilizzano i sensori tattili al posto di comuni interruttori, potenziometri o slider, per rendere i propri apparecchi più accattivanti da un lato, ma anche più affidabili dall'altro; non biso-

Il proiettore da oggi è anche touch-screen



Come si fa ad allestire un grande touch-screen senza uno schermo? È possibile avere un grande display interattivo senza spendere un patrimonio e senza doverselo portare dietro? A queste domande risponde Light Blue Optics, che al recente CES di Las Vegas ha presentato Light Touch, l'innovativo proiettore la cui proiezione permette di usare qualsiasi superficie piana come uno schermo tattile; può quindi proiettare, ad esempio, sulla scrivania dell'ufficio un documento da consultare coi colleghi o una finestra di dialogo sulla quale impartire comandi, la tastiera dell'antifurto o del telecomando domotico sul muro di casa, il menu e il modulo per ordinare sul tavolo di un ristorante, la pulsantiera del telecomando della TV o del telefono sulla coperta del letto o la schermata di un videogioco sul pavimento della camera dei ragazzi. Una volta scelta la superficie desiderata e proiettata l'immagine, è sufficiente semplicemente toccare con le dita le zone di questo display improvvisato, per gestire funzionalità e contenuti accessibili dai pulsanti interattivi. Light Touch nasce grazie all'Holographic Laser Projection, un sistema integrato nel proiettore che ha il compito principale di creare immagini e video con risoluzione WVGA, ed a gruppi di sensori di movimento in grado di rilevare il cambiamento dell'immagine proiettata, ovvero l'interazione tra la persona e l'immagine. Quando un soggetto sfiora la superficie, i sensori rilevano il gesto e trasmettono i dati corrispondenti ad un touch-screen virtuale che interpreta il comando.

gna dimenticare, infatti, che un comune interruttore meccanico è soggetto all'usura e, sebbene il costruttore garantisca la funzionalità del proprio prodotto fino ad un numero estremamente alto di pressioni, questo numero non è sicuramente infinito. Invece la

durata di un sensore tattile si può considerare illimitata, ammettendo un uso consono, naturalmente. Utilizzando tale tecnologia, ad esempio, per rimpiazzare il potenziometro del volume di un apparecchio audio, si risolve alla radice il problema del fastidioso

rumore che il potenziometro tradizionale introduce quando comincia a deteriorarsi o ad ossidarsi. Ancora, dotando un dispositivo di comandi "al tatto" è possibile dar sfogo alla fantasia creativa del designer, che potrà disegnare pulsanti di ogni forma e colore e dare un tocco di classe agli apparecchi.

Insomma, questi motivi, uniti al fatto che il costo della tecnologia touch-sensing è ormai molto basso, hanno fatto sì che l'impiego di sensori tattili e touch-screen sia ormai all'ordine del giorno.

Mentre ognuno di noi può facilmente elencare, tra i dispositivi che montano sensori tattili, quelli che usa comunemente, spesso non sappiamo di utilizzare oltre ad essi anche dei sensori di prossimità. Questi sensori non sono molto diversi dai primi: sfruttano lo stesso principio di funzionamento ma si differenziano da essi per il fatto che, per essere azionati, non devono essere toccati ma basta avvicinarsi loro.

Pensate ai cellulari: tanti di voi hanno uno smartphone, un telefonino di ultima generazione con un bel touch-screen che permette di comandarlo completamente toccandolo con un dito; per fare una telefonata tocchiamo l'icona relativa alla funzione telefono, selezioniamo dalla rubrica la persona che vogliamo chiamare o digitiamo sul tastierino numerico (sempre sul touch-screen) il suo numero e poi tocchiamo il tasto "chiama". Al termine di queste operazioni portiamo il telefonino all'orecchio e aspettiamo che il nostro interlocutore risponda. Ma quando avviciniamo il telefono all'orecchio potremmo toccare il touch-screen con la guancia e quindi involontariamente andremmo ad impartire al telefono dei comandi sbagliati e, magari, interromperemmo la

conversazione senza neppure avvedercene.

In questi casi ci viene in soccorso proprio un "proximity sensor", ossia un sensore che capisce che vicino al touch-screen c'è qualcosa di ben più grosso di un dito (la guancia, appunto...) e dice al telefono di inibirne momentaneamente il funzionamento. La stessa informazione viene utilizzata per spegnere il display, riducendo considerevolmente il consumo di corrente del telefonino e aumentando così la durata della batteria.

Ci sono poi altri casi più particolari in cui l'utilizzo dei sensori di prossimità è d'obbligo: si pensi ad esempio ad una macchina che deve essere azionata da un'operatore che indossa dei guanti o che ha, per il lavoro che svolge mentre la usa, le mani sporche o bagnate; in questo caso si può realizzare un interruttore per azionare il quale è necessario semplicemente avvicinare la mano alla macchina stessa.

Un altro campo di applicazione dei sensori di prossimità è il controllo degli accessi: si possono facilmente contare le persone che passano attraverso una porta (ad esempio, montando un sensore di prossimità sugli stipiti) senza

arrecare loro disturbo, né fargli indossare un badge o chiedergli di premere un pulsante al loro passaggio.

COME FUNZIONANO I SENSORI TATTILI

Ma vediamo cosa sono e come funzionano questi dispositivi. I sensori tattili sono perlopiù di tipo "capacitivo", nel senso che risultano collegati ad un circuito che ne misura costantemente la capacità: se vengono toccati da un corpo, alla capacità intrinseca del sensore si aggiunge quella del dito e il circuito si accorge di questa variazione.

La Microchip ha introdotto su alcuni dei più diffusi microcontrollori PIC una periferica che serve proprio a questo, lasciando al progettista il solo compito di scrivere il firmware adatto alla sua applicazione. Tale periferica si chiama CSM, acronimo di *Capacitive Sensing Module* ed è presente, ad esempio, nella famiglia del PIC16F727.

Una periferica più avanzata di questa e che richiede una maggior capacità di calcolo ed un hardware più complicato (serve un ADC...) è stata introdotta in alcuni microcontrollori ad 8 bit (ad esempio il PIC18F46J11) e in altri a 16 bit come il PIC24FJ256GB110; questa periferica si chiama CTMU, acronimo di *Charge Time Measurement Unit*. Il grande vantaggio della tecnologia touch in questione è che il sensore vero e proprio può essere realizzato tracciando opportunamente due piste sullo stesso stampato che ospiterà i componenti del circuito di rilevamento contenente il microcontrollore; si sfrutta il fatto che due conduttori vicini tra di loro e separati da un materiale dielettrico costituiscono un condensatore, la cui capacità viene costantemente

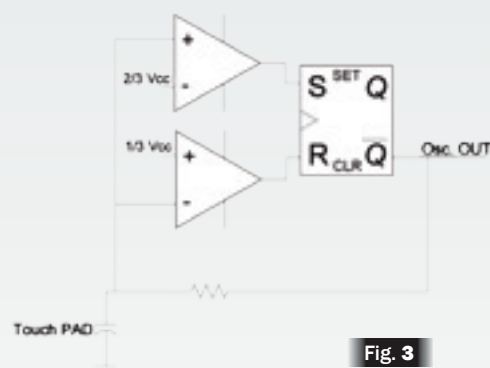


Fig. 3

monitorata dal circuito.

I disegni qui pubblicati mostrano chiaramente il principio su cui si basa la tecnologia touch-sensing: nella Fig. 1 si evidenzia come le due piste di rame sui due lati del circuito stampato separate dal materiale di cui è costituito il PCB stesso (ad esempio l'FR4) costituiscano un vero e proprio condensatore.

Nella Fig. 2 si vede come cambiano le cose nel momento in cui un dito tocca il sensore: i metalli presenti (in forma di elettroliti) nel sangue che circola all'interno del dito sono a tutti gli effetti un elemento conduttore che, accoppiato al sensore sottostante, costituisce un vero e proprio condensatore; quest'ultimo si trova in parallelo a quello del sensore stesso, con il risultato di farne aumentare la capacità.

Per dare un'idea delle quantità in gioco, basti sapere che la capacità introdotta nel circuito dal dito che tocca il sensore è dell'ordine dei 10 pF.

Bisogna quindi costruire un circuito in grado di rilevare variazioni di capacità dell'ordine di pochi pF; le due periferiche presenti nei microcontrollori Microchip assolvono, in maniera differente, a questo compito. Vediamo come. Il Capacitive Sensing Module è composto da due macroblocchi, come mostrato nella Fig. 3: si tratta di un

Fig. 1

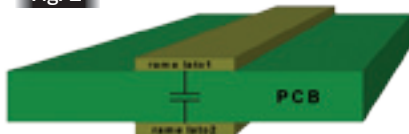
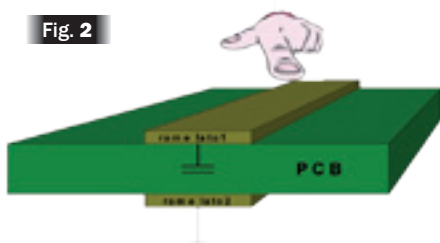


Fig. 2





Andamento della frequenza di oscillazione del Capacitive Sensing Module al variare della capacità.

oscillatore a rilassamento seguito da un contatore.

L'oscillatore a rilassamento sfrutta la capacità del sensore per creare, insieme ad una resistenza, la costante di tempo (e quindi la frequenza) dell'oscillatore stesso. Il primo stadio dell'oscillatore è un semplice comparatore a finestra: la sua uscita è a livello logico 1 quando l'ingresso supera la soglia alta (tipicamente $2/3 V_{cc}$) e a livello logico 0 quando è inferiore alla soglia bassa (tipicamente $1/3 V_{cc}$). L'uscita di questo comparatore viene usata per caricare/scaricare la capacità del sensore tramite una resistenza.

È evidente che se la capacità cambia perché qualcuno ha toccato il sensore, anche la frequenza di oscillazione varia (Fig. 4).

Il blocco che segue è un contatore utilizzato per misurare la frequenza di oscillazione; in esso un timer interno permette di generare la base dei tempi e un contatore conta quanti cicli dell'oscillatore avvengono prima che il timer vada in overflow. Da queste informazioni ricava la frequenza. A questo punto, utilizzare l'informazione che il CSM ha prodotto per rilevare il tocco sul sensore, è una mera questione di software.

Vediamo adesso l'altra periferica presente nei controllori Microchip, ossia quella chiamata CTMU. Il suo principio di funzio-

namento è il seguente: un generatore di corrente costante viene utilizzato per caricare la capacità del sensore per un tempo prefissato; al termine di questo intervallo, la tensione sviluppata sul condensatore viene convertita da un ADC e utilizzata per ottenere la misura della capacità del condensatore.

Nel momento in cui il tocco di un dito fa aumentare la capacità, il valore convertito dall'ADC al termine del tempo prefissato sarà inferiore e quindi il software dichiarerà l'avvenuta pressione del tasto. È possibile sfruttare tutti i canali dell'ADC presenti nel microcontrollore per realizzare altrettanti sensori; considerando poi che la lettura di un singolo canale può avvenire in soli $4 \mu s$, è evidente che questa soluzione si presta laddove si voglia realizzare con un solo integrato il controllo di numerosi sensori. Non a caso vi sono ADC con 16 ingressi, mediante i quali è possibile realizzare altrettanti interruttori tattili. Un'altra azienda di semiconduttori che produce circuiti integrati capaci di controllare i sensori tattili è la Atmel; i sensori che gli integrati di questa casa possono utilizzare sono anch'essi di tipo capacitivo, ma il principio di funzionamento su cui si basano è totalmente diverso da quello sfruttato da Microchip. Infatti in essi viene sfruttato il principio

del trasferimento di carica: la capacità del sensore viene prima di tutto caricata applicandole una determinata tensione. La carica immagazzinata nella capacità C vale:

$$Q = C \times V$$

La carica così accumulata viene poi trasferita in un condensatore posto sulla board, che chiamiamo *capacità di storage* o C_s ; una volta che la carica Q viene immagazzinata, il condensatore mostrerà ai suoi capi una differenza di potenziale pari a:

$$V = Q / C_s$$

Il ciclo viene poi ripetuto periodicamente. Dopo un numero " n " di cicli, la tensione ai capi di C_s viene misurata (convertita in digitale tramite un ADC interno al chip e poi elaborata); nel caso in cui qualcuno tocchi il sensore la sua capacità aumenta, come già spiegato. Così facendo, ad ogni ciclo accumulerà più carica e dunque la tensione su C_s al termine degli " n " cicli sarà più alta. Il chip si accorgerà di questa variazione e dichiarerà l'avvenuto tocco del sensore.

Questa tecnica è adatta anche alla realizzazione di sensori di prossimità, visto che può rilevare cambiamenti della capacità del sensore dell'ordine dei femtofarad (millesimi di pF), variazione che si ottiene solo avvicinando la mano al sensore, senza bisogno di toccarlo.

La tecnica appena descritta (che Atmel ha chiamato *QTouch*) ben si presta alla realizzazione di semplici sensori tattili (touch sensors) usando pochi componenti esterni e con un layout del PCB estremamente semplice e poco critico. Nel caso in cui si vogliano realizzare molti interruttori o uno

slider (o entrambi) si può utilizzare un'altra soluzione che Atmel ha chiamato *QMatrix*. In questo caso si usa una coppia di elettrodi per ogni tasto (a dire il vero uno di essi può essere condiviso fra molti tasti, posti in una matrice): un elettrodo è collegato al chip e viene pilotato con un treno di impulsi; l'altro elettrodo si accoppia al primo tramite la capacità del sensore e riceve dunque una parte dell'energia iniettata nel primo. Nel momento in cui qualcuno tocca il sensore, l'accoppiamento viene turbato dal dito e, di conseguenza, varia l'energia trasmessa; il chip, misurando la variazione, rileva il tocco.

Nella **Fig. 5** viene schematizzato il principio di funzionamento del *QMatrix*. Entrambe le soluzioni

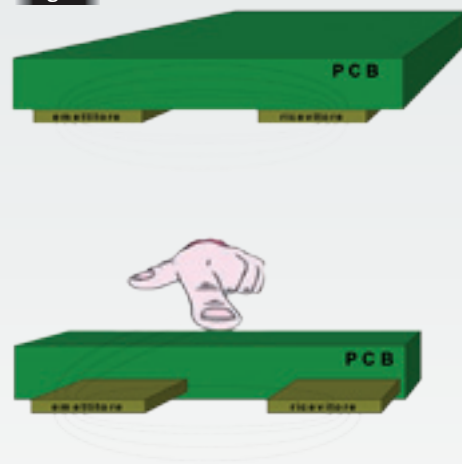
appena descritte risultano molto sensibili ai disturbi e sono causa di emissioni elettromagnetiche (soprattutto la seconda).

Per ovviare al primo problema viene attuata una sofisticata elaborazione dei risultati dei circuiti collegati ai sensori; si tratta per lo più di una elaborazione software, che consiste nel filtrare e mediare i risultati.

Invece, per non violare le normative riguardanti la EMI (*ElectroMagnetic Interference*) gli impulsi utilizzati per eccitare il sensore (soprattutto nel caso di *QMatrix*) sono modulati secondo uno schema spread-spectrum (a spettro distribuito) che ne riduce drasticamente l'energia irradiata o convogliata.

In conclusione di questo articolo,

Fig. 5



possiamo dire che l'impiego di sensori tattili è ormai alla portata di tutti e "sotto le mani" del grande pubblico.

Prossimamente vedremo in pratica come realizzare "in cantina" un sensore di questo tipo e come utilizzarlo in pratica. ■



Un approccio integrato alla progettazione di circuiti

- Realizza un prototipo virtuale via software prima di raggiungere il laboratorio
- Confronta all'istante i dati simulati con le misure reali
- Ricevi feedback istantanei dagli strumenti virtuali integrati
- Utilizza le funzionalità integrate per la didattica
- Scegli l'opzione con oscilloscopio da 100MS/s
- Sistema da laboratorio compatto, robusto e versatile
- 12 strumenti integrati in un'unica piattaforma

>> Scarica le risorse gratuite su ni.com/academic/i/eep

02 41.309.1

IRS Ingegneria
Ricerca
Sistemi
Distributore unico per l'Italia
Tel.: +39 049 8706156 • info@irsweb.it

**NATIONAL
INSTRUMENTS**

NATIONAL INSTRUMENTS (USA) INC. • 20015 Avenida de las Americas, Suite 1300 • Austin, Texas 78741 • USA
PUBBLICITÀ: 02/101/0012 • CODICE MINISTERO 02/101/0012 • Pagine: 108 • Web: www.ni.com/academic/i/eep • 1/10
© 2010 National Instruments. Tutti i diritti riservati. L'uso di questo materiale, National Instruments, NI e le sue logo sono marchi registrati di National Instruments. Altri prodotti e nomi aziendali sono marchi registrati o marchi delle rispettive aziende. 5000



Conoscere e usare Arduino

dell'ing.
MIRCO
SEGATELLO

Scopriamo come gestire display a cristalli liquidi alfanumerici e grafici, vedendo alcuni sketch di esempio validi per i più comuni modelli reperibili in commercio.

Siamo giunti ormai alla terza puntata del corso su Arduino; questa volta l'argomento della lezione sono i display: non solo quelli di testo ma anche quelli grafici. Faremo una carrellata di applicazioni con diversi tipi di dispositivi, sempre in modo molto semplice e proponendo esempi pratici che riguardino la parte sia hardware che software. I primi dispositivi che andremo a descrivere sono i display di testo, immancabili in svariate applicazioni e molto semplici da utilizzare, grazie alle librerie già pronte. Prima di addentrarci nei dettagli, facciamo una breve intro-

duzione su questo tipo di display, non tanto dal punto di vista costruttivo, quanto riguardo al loro utilizzo. Partiamo dicendo che tutti i display LCD di testo si basano su un chip decoder, che, letti i dati in ingresso, provvede a gestire i pixel del display in modo appropriato. È proprio il tipo di decoder usato all'interno del display che ne determina l'utilizzo e le funzioni, non tanto la grandezza o il numero di righe; è quindi importante, quando si acquista un display LCD, sapere quale chip decoder lo governa, perché da questo dipenderà la sua gestione a livello software. A livello hardware,

Tabella 1

+5V	Alimentazione positiva
GND	Massa
VLCD	Tensione negativa usata per la regolazione del contrasto
Data	8 linee di dati
WR/RD	Linea per la lettura/scrittura per LCD
E	Comando di abilitazione display
RS	Linea per impostazione dati/comandi

invece, le cose sono più complicate, perché ogni casa produttrice, anche a parità di chip decoder montato, personalizza la piedinatura dei contatti, rendendo a volte inutilizzabili display privi di documentazione. Per questo corso useremo esclusivamente display LCD basati sul controller Hitachi HD44780 oppure Samsung ST7066U: si tratta di display con interfaccia parallela per i quali i più svariati ambienti di sviluppo per microcontrollori includono già i comandi per controllarli, il che risparmia la fatica di dover elaborare specifiche routine di controllo. L'interfaccia hardware è standard e prevede i contatti raggruppati nella **Tabella 1**.

Oltre a queste linee, potrebbero essere disponibili i contatti della retroilluminazione (BL+ e BL-) da utilizzarsi se è richiesta la visione in ambienti poco illuminati.

Ovviamente i nomi possono cambiare da produttore a produttore, così come il riferimento al pin del connettore al quale fanno capo, però la funzione rimane sempre la stessa.

Facciamo ora un esempio pratico, utilizzando un display basato su chip Hitachi che presenta il connettore posto in alto e la piedinatura riportata nella **Tabella 2**.

Tabella 2

Pin	Segnale	Funzione
1	VSS	Alimentazione (0V)
2	VDD	Alimentazione (5V)
3	VO	Drive LCD (0V rispetto VDD)
4	RS	(Alto) ingresso codici di istruzione (Basso) ingresso dati
5	R/W	(Alto) lettura dati (Basso) scrittura dati
6	E	Segnale di abilitazione
7	DB0	Linea di bus dati
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	BL-	Terminale di alimentazione LED (-)
16	BL+	Terminale di alimentazione LED (+)

In questo display la retroilluminazione (in inglese *backlight*) viene attivata semplicemente ponendo BL- a massa e BL+ al positivo dei 5 volt con una resistenza da 20÷50 ohm a seconda del livello di intensità luminosa desiderato. A questo punto siamo pronti al cablaggio fisico del display con la scheda Arduino; per fare questo ci sono diverse possibilità, ad esempio l'utilizzo della Protoshield oppure di una base sperimentale. L'importante è rispettare lo schema di collegamento illustrato nella **Fig. 1**. Facciamo ora un esempio pratico utilizzando il display di codice CDL4162 (Clover) distribuito dalla ditta Futura Elettronica, dal cui sito è possibile scaricare anche il data-sheet in lingua italiana, nel quale è riportata chiaramente la piedinatura, con nomi e funzioni di ogni singolo pin.

È inoltre riportata la mappa dei caratteri visualizzabili, che si basa su un codice ASCII rivisto per contenere numeri, lettere e simboli in diverse lingue.

La corrispondenza tra pin e funzioni è riportata nella **Tabella 3**.

Per la retroilluminazione potete fare riferimento alle indicazioni precedenti. Per il cablaggio dovete fare riferimento allo schema elettrico illustrato nella **Fig. 2**.

In ogni caso è necessario effettuare i seguenti collegamenti tra la scheda Arduino e il display LCD:

- LCD RS → pin digitale 12;

Tabella 3

Pin	Segnale	Funzione
1	BL+	Terminale di alimentazione LED (+)
2	BL-	Terminale di alimentazione LED (-)
3	GND	Alimentazione (0V)
4	VDD	Alimentazione (5V)
5	VO	Drive LCD
6	RS	(Alto) ingresso codici di istruzione (Basso) ingresso dati
7	R/W	(Alto) lettura dati (Basso) scrittura dati
8	E	Segnale di abilitazione
9	DB0	Linea di bus dati
10	DB1	
11	DB2	
12	DB3	
13	DB4	
14	DB5	
15	DB6	
16	DB7	

- LCD R/W → GND;
- LCD Enable → pin digitale 11;
- LCD D4 → pin digitale 5;
- LCD D5 → pin digitale 4;
- LCD D6 → pin digitale 3;
- LCD D7 → pin digitale 2.

Oltre a ciò bisogna provvedere a collegare l'alimentazione ed il trimmer per la regolazione del contrasto. I pin D0, D1, D2, D3 possono essere lasciati liberi oppure posti a massa.

A questo punto avete il vostro display cablo col microcontrollore, ma non ancora programmato; se lo accendete, non noterete nulla se non dei pixel parzialmente attivi.

Al momento non risulta tarato neppure il contrasto.

Per completezza riportiamo qui di seguito anche tutti i comandi disponibili in fase di programmazione e gli esempi disponibili nell'ambiente di sviluppo.

Function

- LiquidCrystal()
- begin()
- clear()
- home()
- setCursor()
- write()
- print()
- cursor()
- noCursor()
- blink()
- noBlink()
- display()
- noDisplay()
- scrollDisplayLeft()
- scrollDisplayRight()
- autoscroll()
- noAutoscroll()
- leftToRight()
- rightToLeft()

- createChar()

Examples

- Hello World
- Blink
- Cursor
- Display
- Text Direction
- Autoscroll
- Serial input

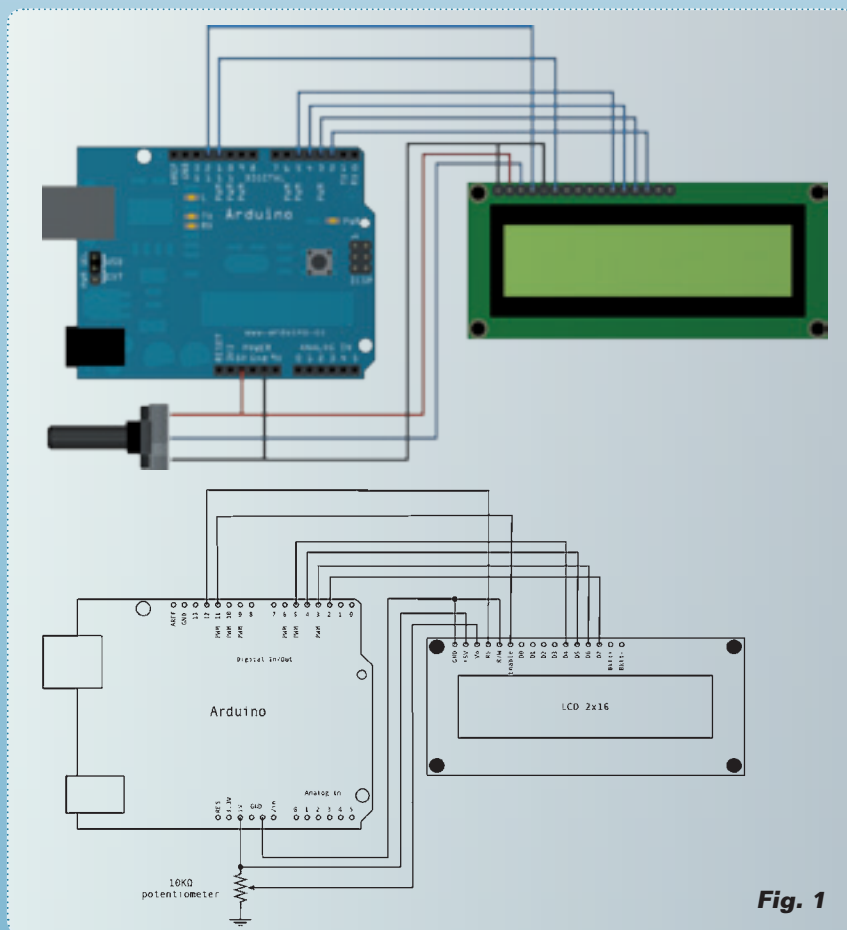


Fig. 1

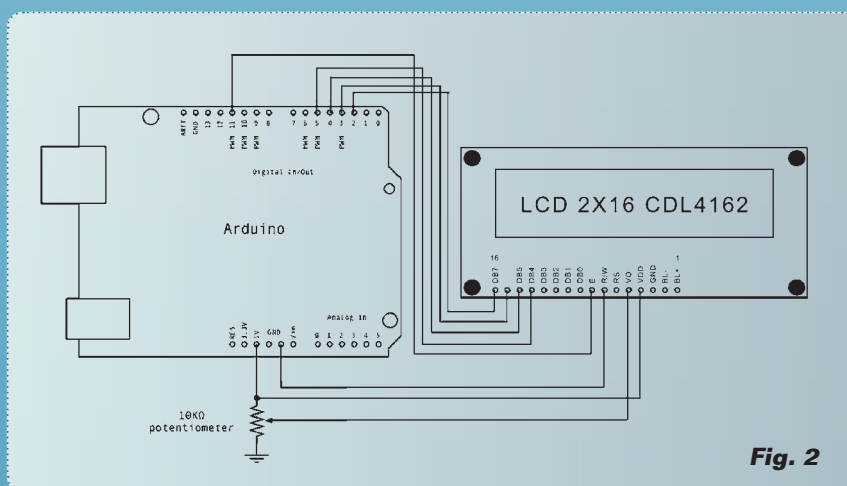


Fig. 2

Listato 1

```

/*
  Display_01

  Semplice scritta su display LCD di testo.
  Per il cablaggio hardware:

  * LCD RS pin to digital pin 12
  * LCD Enable pin to digital pin 11
  * LCD D4 pin to digital pin 5
  * LCD D5 pin to digital pin 4
  * LCD D6 pin to digital pin 3
  * LCD D7 pin to digital pin 2

  */
// include la libreria LCD:
#include <LiquidCrystal.h>

// inizializza il display con l'interfaccia
// hardware specificata
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // setta il numero di colonne e righe del display:
  lcd.begin(16, 2);
  // Scrive un messaggio.
  lcd.print("hello, world!");
}

void loop() {
  // Imposta il cursore all'inizio della seconda riga
  // (nota: il conteggio delle righe parte da zero:
  lcd.setCursor(0, 1);
  // Scrive il numero di secondi a partire
  // dall'istante di reset:
  lcd.print(millis()/1000);
}

```

- SetCursor
- Scroll

Adesso apriamo il software di sviluppo Arduino (la versione 18, nel nostro caso), creiamo un nuovo sketch oppure, più velocemente, utilizziamo un esempio già pronto disponibile nella cartella del programma *File/examples/liquidcrystal/helloworld*.

Sul nostro sito, a fine corso, saranno disponibili anche i programmi commentati in lingua italiana usati nelle varie puntate.

Lo sketch cui facciamo riferimento è descritto nel **Listato 1**.

Caricate ed avviate lo sketch, regolate il contrasto agendo sul trimmer sino ad evidenziare la scritta "hello, world!" ed il gioco è fatto.

Questo esempio è molto importante perché vi fa capire anche la modalità con la quale vengono visualizzate le variabili, in questo caso quella denominata *millis()*, che contiene il numero di millisecondi da quando Arduino sta eseguendo il programma; dividendola per 1.000 si ottiene il numero di secondi di esecuzione del programma.

Notate come la variabile scritta in memoria in forma binaria venga rappresentata in numero decimale cifra per cifra, con una conversione automatica. Altro punto importante da notare è la presenza della riga `#include <LiquidCrystal.h>` che permette di includere la libreria e quindi i comandi per gestire un display LCD. Allo stesso modo sarà possibile importare altre librerie, anche create da terzi, al fine di espandere le funzioni disponibili.

Potete sbizzarrirvi e provare anche tutti gli altri esempi inerenti ai display LCD: ci sono esempi per agire sulla scritta oppure sul cursore in vari modi. Per comodità riportiamo, nella **Tabella 4** di seguito, la sintassi con un esempio dei principali comandi per il display. Passiamo ora all'utilizzo dei display LCD grafici (GLCD) sicuramente più affascinanti e flessibili rispetto a quelli di testo; cercheremo di non addentrarci troppo nello specifico, rimandando, chi volesse approfondire il discorso sul controllo dei display grafici, all'apposito corso già pubblicato nei fascicoli dal n° 115 al n° 123.

La prima cosa da dire è che come per i display LCD di testo, i GLCD si suddividono a seconda del chip adibito al controllo e sono essenzialmente disponibili sul mercato due grandi categorie: la prima si basa sull'uso

Tabella 4 - Comandi per il display e loro sintassi.

comando	funzione
<code>LiquidCrystal lcd(12, 11, 5, 4, 3, 2);</code>	Inizializza il display e lo dichiara come oggetto di nome <code>lcd</code> da usarsi nel programma
<code>lcd.begin(cols, rows);</code>	Imposta per l'oggetto <code>lcd</code> il numero di colonne (<code>cols</code>) ed il numero di righe (<code>rows</code>)
<code>lcd.cursor();</code>	Attiva visualizzazione del cursore sulla posizione attuale
<code>lcd.noCursor();</code>	Disattiva visualizzazione del cursore sulla posizione attuale
<code>lcd.blink();</code>	Attiva il lampeggio del cursore
<code>lcd.noBlink();</code>	Disattiva il lampeggio del cursore
<code>lcd.display();</code>	Attiva visualizzazione sul display
<code>lcd.noDisplay();</code>	Disattiva visualizzazione sul display
<code>lcd.scrollDisplayRight();</code>	Trasla il testo del display di una posizione a destra
<code>lcd.scrollDisplayLeft();</code>	Descrizione: Trasla il testo del display di una posizione a sinistra

del chip KS0107B che però consente una sola risoluzione fissa pari a 128x64. Questo tipo di display è abbastanza comune ed economico, ma la risoluzione non troppo elevata lo rende poco flessibile. Una seconda categoria si basa sull'uso del cip T6963C, il quale permette di gestire display con le seguenti risoluzioni: 128x128, 240x64 e 240x128.

Esistono in commercio svariati altri modelli di GLCD, ma non sempre sono disponibili le librerie per il loro controllo; quindi, di fatto risultano più difficilmente gestibili.

Come primo esempio prendiamo in considerazione il display modello LM12864MBC distribuito dalla Futura Elettronica (codice 1446-LCD102B6B) e basato sul chip KS0107B, quindi con risoluzione 128x64 in bianco e nero. Come già spiegato per le prove con il display di testo, è necessario realizzare il cablaggio elettrico e poi programmare adeguatamente la scheda Arduino; i collegamenti sono quelli descritti nella **Tabella 5**.

Visto il carattere sperimentale dell'applicazione, anche in questo caso utilizziamo una breadbord su cui eseguire i collegamenti con dei fili rigidi. È possibile saldare sul display un connettore strip femmina come quelli presenti sulla scheda Arduino e da lì, sempre con dei filetti rigidi, eseguire i collegamenti. Sarebbe anche possibile saldare direttamen-

Tabella 5 - Collegamenti fisici tra il display LM12864MBC e Arduino Duemilanove.

Pin Arduino	Pin display	Nome	funzione
GND	1	VSS	GND
+5V	2	VDD	+5V
Pin2 (centrale) del trimmer	3	Vo	Contrasto LCD
17 (analog3)	4	RS	Data/instruction
16 (analog2)	5	R/W	Read/Write
18 (analog4)	6	E	Enable
8	7	DB0	Dato
9	8	DB1	Dato
10	9	DB2	Dato
11	10	DB3	Dato
4	11	DB4	Dato
5	12	DB5	Dato
6	13	DB6	Dato
7	14	DB7	Dato
14 (analog0)	15	CS1	Chip select 1
15 (analog1)	16	CS2	Chip select 2
Reset (porre a +5V)	17	RST	Reset
Pin1 del trimmer Pin3 del trimmer a GND	18	Vout	Tensione per Vo
+5V (se retro.zione ON)	19	BLA	Retroilluminazione anodo
GND	20	BLK	Retroilluminazione catodo

te dei fili sul display oppure utilizzare la Protoshield per eseguire tutti i collegamenti con dei filetti saldati. Ad ogni modo, dovete essere molto diligenti e seguire in modo scrupoloso la **Tabella 5**, ricontrrollando più volte il lavoro eseguito. Il trimmer, del valore di 10÷20 kohm, ha tre terminali: i due esterni vanno connessi uno a massa ed uno al pin 18 del display (Vout) mentre il terminale centrale

Tabella 9 - Collegamenti fisici tra il display ADM12864H e Arduino Duemilanove.

Pin Arduino	Pin display	Nome	funzione
GND	2	VSS	GND
+5V	1	VDD	+5V
Pin2 (centrale) del trimmer	3	Vo	Contrasto LCD
17 (analog3)	16	D/I	Data/instruction
16 (analog2)	15	R/W	Read/Write
18 (analog4)	17	E	Enable
8	4	DB0	Dato
9	5	DB1	Dato
10	6	DB2	Dato
11	7	DB3	Dato
4	8	DB4	Dato
5	9	DB5	Dato
6	10	DB6	Dato
7	11	DB7	Dato
14 (analog0)	13	CS1	Chip select 1
15 (analog1)	12	CS2	Chip select 2
Reset (porre a +5V)	14	RST	Reset
Pin1 del trimmer / Pin3 del trimmer a GND	18	Vout	Tensione per Vo
+5V (se retro.zione ON)	19	BLA	Retroilluminazione anodo
GND	20	BLK	Retroilluminazione catodo

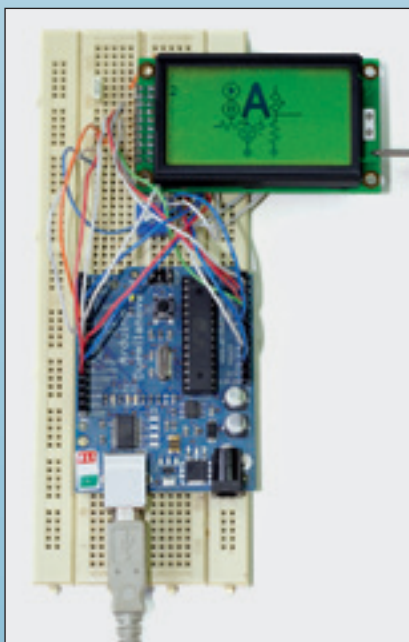


Fig. 3

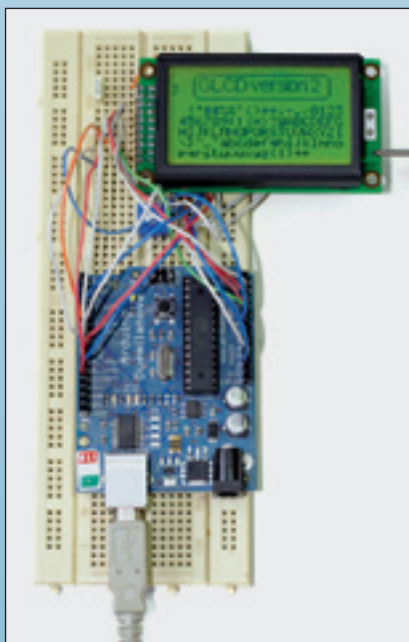


Fig. 4

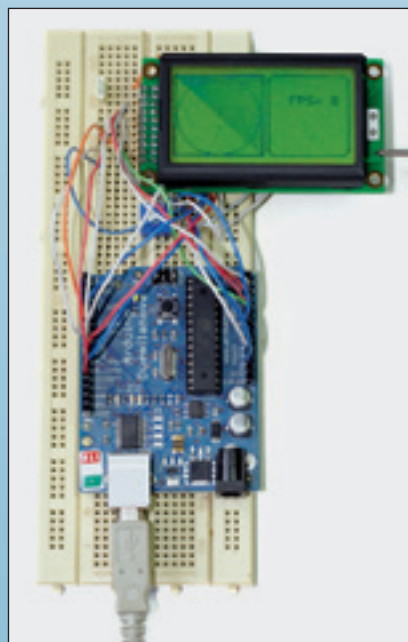


Fig. 5

va collegato al pin 3 (Vo) sempre del display. Non guasta inserire un piccolo condensatore da 100 nF tra i +5 V e il GND, il più vicino possibile al display per sopprimere eventuali disturbi elettrici presenti nei collegamenti. Fatto questo, si passa alla programmazione

della scheda Arduino, per la quale dobbiamo procurarci le librerie specifiche per il display grafico LCD con chip K0108, sviluppate in questo caso da terze parti. Sempre sul sito ufficiale in lingua inglese di Arduino, potete trovare una sezione dedicata a "KS0108

comando	descrizione
GLCD.Init(NON_INVERTED)	Inizializza la libreria per il disegno normale o invertito. Se normale ogni pixel viene disegnato (appare scuro), se invertito il pixel viene cancellato.
GLCD.GotoXY(x,y)	Posiziona il cursore nella posizione x e y, dove 0,0 è l'angolo superiore sinistro.
GLCD.ClearScreen()	Cancella lo schermo.
Graphic Drawing Functions: se color è WHITE cancella pixel; se BLACK attiva pixel.	
GLCD.DrawCircle(x, y, radius, color)	Disegna un cerchio alle coordinate x,y di raggio radius.
GLCD.DrawLine(x1,y1,x2,y2,color)	Disegna una linea dalle coordinate x1,y1 alle x2,y2.
GLCD.DrawVertLine(x, y, length, color)	Disegna una linea verticale dalle coordinate x,y di lunghezza length.
GLCD.DrawHorizLine(x, y, length, color)	Disegna una linea orizzontale dalle coordinate x,y di lunghezza length.
GLCD.DrawRect(x, y, width, height, color)	Disegna un rettangolo.
GLCD.DrawRoundRect(x, y, width, height, radius, color)	Disegna un rettangolo con gli angoli smussati.
GLCD.FillRect(x, y, width, height, color)	Disegna un rettangolo pieno.
GLCD.InvertRect(x, y, width, height)	Inverte i pixel del rettangolo specificato.
GLCD.SetInverted(invert)	Imposta il modo di disegno invertito.
GLCD.SetDot(x, y, color)	Disegna un pixel alla posizione x,y.
GLCD.DrawBitmap(bitmap, x, y, color)	Disegna un'immagine bitmap alle coordinate x,y.
Font Functions	
GLCD.SelectFont(font, color)	Imposta i font dei caratteri.
GLCD.PutChar(character)	Scrivi un carattere.
GLCD.Puts(string)	Scrivi una stringa.
GLCD.Puts_P(string)	Scrivi una stringa contenuta nella memoria del programma.
GLCD.PrintNumber(number)	Scrivi il valore decimale di una variabile numerica.
GLCD.CursorTo(x, y)	Imposta le coordinate di base per i font a larghezza fissa.

Tabella 6 - Comandi per la gestione dei display grafici, ripartiti per categoria: Graphic Drawing Functions riguardano il disegno di primitive grafiche e Font Functions è inerente ai caratteri.

Tabella 7 - Collegamenti fisici tra scheda Arduino e Graphic LCD serial backpack LCD-09352.

Pin Arduino	Pin LCD-09352	
Vin	Vin	Alimentazione positiva 6-7V
GND	GND	GND
TX	RX	Linea dati

Graphics LCD library” che spiega in dettaglio l'utilizzo di questa libreria e ne permette anche il download. I nostri lettori troveranno già disponibili questi file, arricchiti da un ulteriore esempio; la cartella di nome KS0108GLCD contiene sia le librerie, sia dei file di esempio e deve essere copiata all'interno della cartella *Libraries sottomirectory* della cartella principale *Arduino-0018*.

Aprirete lo sketch di nome *GLCDexample.pde*, che troverete nel percorso *File→example→Ks0108* e caricatelo sulla scheda Arduino. Una volta avviato il firmware, compariranno tre schermate in sequenza, la prima con un'immagine, la seconda con tutti i caratteri stampati e la terza con linee e cerchi, come mostrato nelle figure, rispettivamente, 3, 4, 5. Per vedere bene le rappresentazioni potrebbe essere necessario regolare il contrasto del display agendo sul trimmer.

Lo sketch appena descritto dimostra l'utilizzo

Tabella 8

funzione	Bytes [esadecimale]	Note
Cancellazione	0x7C + 0x00	
Demo mode	0x7C + 0x04	
Reverse Mode	0x7C + 0x12	
Set retroilluminazione	0x7C + 0x02 + R	R=valore illuminazione 0-100
Set/reset pixel	0x7C + 0x10 + X + Y + P	X=coordinata orizzontale Y=coordinata verticale P=0 resetta P=1 setta
Disegna Linea	0x7C + 0x0C + X1 + Y1 + X2 + Y2 + P	X1,Y1=coordinate di inizio X2,Y2=coordinate di fine P=0 cancella P=1 disegna
Disegna cerchio	0x7C + 0x03 + X1 + Y1 + R + P	X1,Y1=coordinate centro R=raggio P=0 cancella P=1 disegna
Disegno rettangolo	0x7C + 0x0F + X1 + Y1 + X2 + Y2 + P	X1,Y1=angolo alto sinistro X2,Y2=angolo basso destro P=0 cancella P=1 disegna
Modifica baudrate	0x7C + 0x07 + N	N="1" = 4800bps N="2" = 9600bps N="3" = 19,200bps N="4" = 38,400bps N="5" = 57,600bps N="6" = 115,200bps
Set coordinate	0x7C + 0x18 + X	X=nuova coordinata X
Set coordinate	0x7C + 0x19 + Y	Y=nuova coordinata Y

della libreria GLCD per KS0108 visualizzando a video sia immagini (bitmap in questo caso) sia testo e grafica. Se utilizzate i file scaricati dal nostro sito Internet, troverete anche un ulteriore esempio denominato *GLCDelettroni-*

Listato 2

```

/*****
/* Configuration for assigning LCD bits to Arduino Pins */
/*****
/* Arduino pins used for Commands
 * default assignment uses the first five analog pins
 */

#define CSEL1          14          // CS1 Bit   // swap pin assignments with CSEL2 if
left/right image is reversed
#define CSEL2          15          // CS2 Bit
#define R_W            16          // R/W Bit
#define D_I            17          // D/I Bit
#define EN             18          // EN Bit
// #define RES         19          // Reset Bit // uncomment this to control
LCD reset on this pin

/* option: uncomment the next line if all command pins are on the same port for slight speed & code size improvement */
#define LCD_CMD_PORT    PORTC      // Command Output Register for pins 14-19
/* Arduino pins used for LCD Data
 * un-comment ONE of the following pin options that corresponds to the wiring of data bits 0-3
 */
#define dataPins8to11 // bits 0-3 assigned to arduino pins 8-11, bits 4-7 assigned to arduino pins 4-7
// #define dataPins14to17 //bits 0-3 assigned to arduino pins 14-17, bits 4-7 assigned to arduino pins 4-7.
// (note command pins must be changed)
// #define dataPins0to3 // bits 0-3 assigned to arduino pins 0-3 , bits 4-7 assigned to arduino pins 4-7, this
// is marginally the fastest option but its only available on runtime board without hardware rs232.

/* NOTE: all above options assume LCD data bits 4-7 are connected to arduino pins 4-7 */

/*****
/*      end of Arduino configuration
*****/

```



*ca*in, che permette di visualizzare a video una semplice scritta. Lo sketch semplificato è più semplice da comprendere rispetto all'esempio originale. Un secondo esempio denominato *GLCDeinline* disegna due linee incrociate a tutto schermo, con al centro un cerchio. La libreria è molto nutrita e l'elenco completo di tutti i comandi è riportato, con le descrizioni del caso, nella **Tabella 6**. Come avete visto, il display grafico richiede 8 linee per i dati e 5 per il controllo; l'assegnazione di queste funzioni è esplicitata nel file *KS0108_Arduino.h* con le righe di programma

visibili nel **Listato 2**. È possibile modificare queste linee di codice per impostare in modo diverso le linee utilizzate per la gestione del display GLCD; a riguardo, si consiglia prima di leggere le note esplicative riportate nella pagina HTML di riferimento per questa libreria, nel sito di Arduino. Utilizzando un microcontrollore a 28 pin rimangono comunque poche linee disponibili per le applicazioni. Una seconda possibilità è offerta dal display grafico ADM12864H, anche questo reperibile presso la Futura Elettronica (codice LCD128x64) sempre basato sul chip KS0108 che presenta il vantaggio di avere i 20 pin in linea e quindi facilmente gestibili con una breadboard. Per il collegamento alla scheda Arduino potete fare riferimento alla **Tabella 9**. Una valida soluzione alle difficoltà di cablaggio ed al cospicuo numero di linee utilizzato, arriva con l'introduzione sul mercato dei display a controllo seriale, i quali necessitano di sole quattro linee (+Vcc, GND, TX, RX) che si comandano tramite semplici stringhe inviate in modalità seriale. In commercio sono anche

Listato 3

```
/*
  Display_02

  Semplice esempio utilizzo GLCD a comando seriale.
  Per il cablaggio hardware:
  Graphic LCD Serial Backpack connesso a GLCD-ADM12864H
  Usare le linee Vin, GND, TX.
*/

byte buf_erase[] = {0x7C, 0x00};           // Cancella schermo
byte buf_backl[] = {0x7C, 0x02, 0};        // (retroilluminazione 0=0% 100=100%)
byte buf_line[] = {0x7C, 0x0C, 0, 0, 127, 63, 1}; // Linea (X1,Y1) (X2,Y2) 1=disegna 0=cancella
byte buf_circle[] = {0x7C, 0x03, 63, 31, 20, 1}; // Cerchio (X1,Y1) raggio 1=disegna 0=cancella

void setup()
{
  Serial.begin(115200);                    // Imposta comunicazione a 115200 baud
}

void loop()
{
  Serial.write(buf_backl, 3);              // Retroilluminazione 0%
  delay(100);
  Serial.write(buf_erase, 2);              // Cancella schermo
  delay(1000);
  Serial.write(buf_line, 7);               // Disegna linea
  delay(1000);
  Serial.write(buf_circle, 6);             // Disegna cerchio
  delay(1000);
  Serial.print("ElettronicaIN");           // Scrive un testo
  delay(1000);
}
```

Listato 4

```

/*
  Display_03

  Semplice esempio utilizzo GLCD a comando seriale.
  Si usano i comandi a riga
  Per il cablaggio hardware:
  Graphic LCD Serial Backpack connesso a GLCD-ADM12864H
  Usare le linee Vin, GND, TX.
*/

void setup()
{
  Serial.begin(115200);      // imposta comunicazione a 115200 baud
}

void loop()                 // ripete all'infinito
{
  Serial.print(0x7C, BYTE); // Cancella schermo
  Serial.print(0x00, BYTE);
  delay(100);
  Serial.print(0x7C, BYTE); // Retroilluminazione 0%
  Serial.print(0x02, BYTE);
  Serial.print(0x00, BYTE);
  delay(100);
  Serial.print(0x7C, BYTE); // Disegna un cerchio
  Serial.print(0x03, BYTE); //
  Serial.print(100, BYTE);  // Coordinata X
  Serial.print(50, BYTE);   // Coordinata Y
  Serial.print(10, BYTE);   // Raggio
  Serial.print(1, BYTE);    // 1=disegna 0=cancella
  delay(1000);
  Serial.print(0x7C, BYTE); // Set coordinata X
  Serial.print(0x18, BYTE); // Coordinata X
  Serial.print(10, BYTE);   // Coordinata X
  Serial.print(0x7C, BYTE); // Set coordinata Y
  Serial.print(0x19, BYTE); // Coordinata Y
  Serial.print(40, BYTE);   // Coordinata Y
  Serial.print("ElettronicaIN");
  delay(1000);
}

```

disponibili dei convertitori seriale/parallelo per display GLCD, denominati Graphic LCD serial backpack: un esempio è il modello LCD-09352, perfettamente compatibile con il display grafico ADM12864H (foto nella pagina precedente in alto a sinistra). In questo caso il cablaggio è veramente minimo, perché sono sufficienti le tre linee descritte nella **Tabella 7**. La scheda aggiuntiva va innestata sul connettore del display; nel nostro caso abbiamo usato uno strip maschio sul display ed uno femmina sull'adattatore. Siccome la scheda aggiuntiva internamente ricava i 5 volt tramite un proprio stabilizzatore, va alimentata con una tensione leggermente più alta 6÷7 V (raccomandati); il massimo ammesso è 9 volt. Ecco perché la scheda Arduino dovrà venire alimentata tramite un alimentatore esterno, dato che la tensione della USB non è sufficiente. Per i nostri esperimenti abbiamo utilizzato un semplice alimentatore non stabilizzato da 0,5 A, impostato per ottenere una tensione di uscita di 6 volt; anche se può

sembrare poco, in realtà con tale impostazione in questo genere di alimentatori è facile prelevare anche 8÷9 volt. Sul pin Vin della scheda Arduino è presente la tensione appena uscita dall'alimentazione e non ancora stabilizzata a 5 volt; dovremo avere l'accortezza di fornire in ingresso non più di uno o due volt oltre detto valore. Le gestione di questo adattatore avviene semplicemente con l'invio di comandi seriali che la scheda Arduino gestisce in fase di programmazione tramite le due linee (TX ed RX) che sono le stesse utilizzate per la comunicazione con la USB (tramite il convertitore FT232). Riassumiamo, nella **Tabella 8**, i principali comandi da inviare al display, che devono essere composti secondo lo standard RS232, impostando come parametri di comunicazione 115200,N,8,1 ovvero 115.200 baud (valore predefinito ma modificabile), nessun bit di parità, 8 bit di dati ed un bit di stop. Nel **Listato 3** trovate il programma usato per testare il display, che permette di impostare la retroilluminazione a zero (spenta) cancella

Listato 5

```

/*
  Display_04

  Semplice esempio utilizzo GLCD a comando seriale.
  Si usano SubRoutine per i comandi
  Per il cablaggio hardware:
  Graphic LCD Serial Backpack connesso a GLCD-ADM12864H
  Usare le linee Vin, GND, TX.
*/

void setup()
{
  Serial.begin(115200);      // imposta comunicazione a 115200 baud
}

void loop()
{
  backlight(0);              // Retroilluminazione a 0%
  clearLCD();                // Cancella schermo
  cursorSet(40,30);          // Imposta posizione per il testo
  Serial.print("ElettronicaIN"); // scrive un testo nella posizione indicata
  line(10,10,50,50);         // Disegna una linea
  circle(10,10,10);          // Disegna un cerchio
  delay(2000);
}

// SubRoutine

// cancella LCD
void clearLCD(){
  Serial.print(0x7C, BYTE);
  Serial.print(0x00, BYTE);
}

// gestione retroilluminazione
void backlight(byte light){
  Serial.print(0x7C, BYTE);
  Serial.print(0x02, BYTE);
  Serial.print(light); // imposter zero per disabilitare la retroilluminazione
}

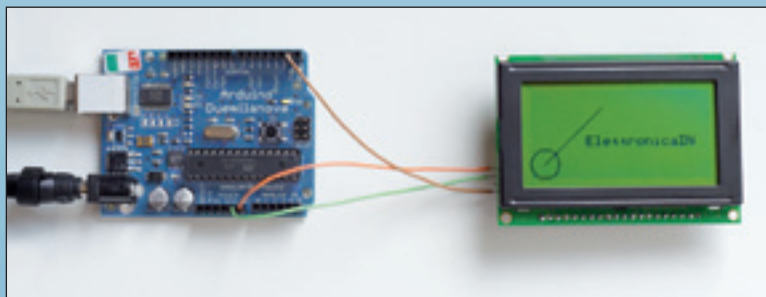
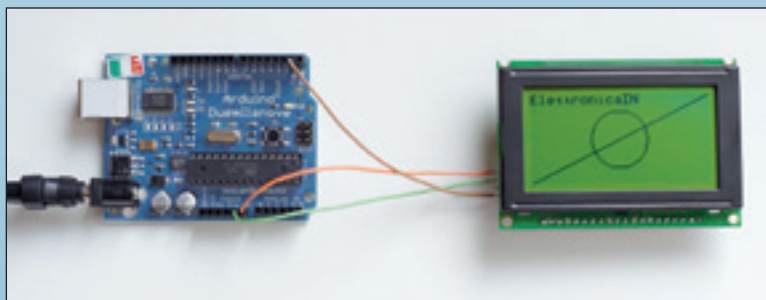
// disegna una linea
void line(byte x1, byte y1, byte x2, byte y2){
  byte buf_line[] = {0x7C, 0x0C, 0, 0, 127, 63, 1}; // Linea (X1,Y1)
  Serial.print(0x7C, BYTE); // Disegna un cerchio
  Serial.print(0x0C, BYTE);
  Serial.print(x1, BYTE); // Coordinata X1
  Serial.print(y1, BYTE); // Coordinata Y1
  Serial.print(x2, BYTE); // Coordinata X2
  Serial.print(y2, BYTE); // Coordinata Y2
  Serial.print(1, BYTE); // 1=disegna 0=cancella
}

// Disegna un cerchio
void circle(byte xpos, byte ypos, byte radius){
  Serial.print(0x7C, BYTE); // Disegna un cerchio
  Serial.print(0x03, BYTE); //
  Serial.print(xpos, BYTE); // Coordinata X
  Serial.print(ypos, BYTE); // Coordinata Y
  Serial.print(radius, BYTE); // Raggio
  Serial.print(1, BYTE); // 1=disegna 0=cancella
}

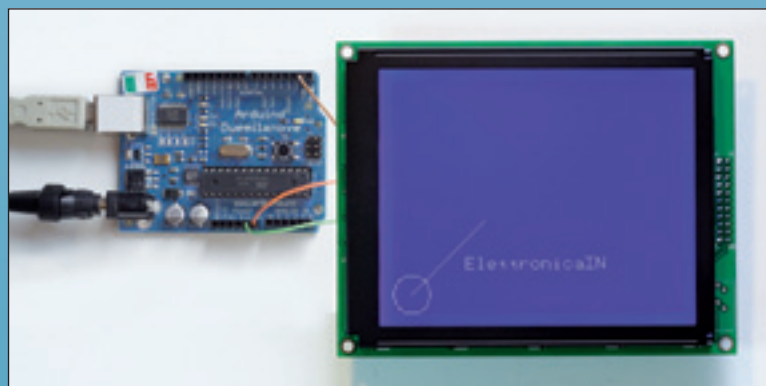
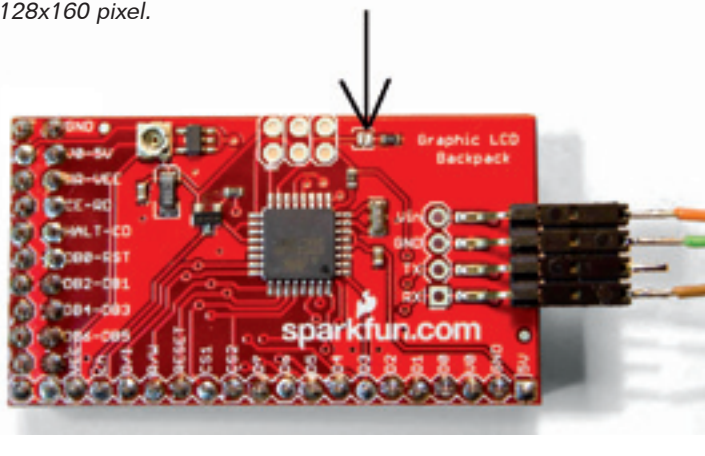
// muove il cursore alla posizione X, Y specificata
void cursorSet(byte xpos, byte ypos){
  Serial.print(0x7C, BYTE); // Set coordinata X
  Serial.print(0x18, BYTE); // Coordinata X
  Serial.print(xpos, BYTE); // Coordinata X
  Serial.print(0x7C, BYTE); // Set coordinata Y
  Serial.print(0x19, BYTE); // Coordinata Y
  Serial.print(ypos, BYTE); // Coordinata Y
}

```

lo schermo e disegna una linea, un cerchio e una scritta. Come vedete, le scritte sono immediate grazie al generatore di caratteri incluso nel backpack; è infatti sufficiente inviare il codice ASCII relativo al carattere da stampare a video. Ovviamente non dovrete fare alcuna conversione ma solo usare l'istruzione *Serial.print* seguita dalla stringa da stampare racchiusa tra virgolette. Un'alternativa all'invio dei comandi appena descritti è riportata nel **Listato 4**, in cui sono vedute una porzione di programma che spedisce i byte necessari uno alla volta. Vi è anche una terza alternativa, che consiste nel costruirsi delle subroutine da richiamare ogni qualvolta si voglia fare una funzione; questa soluzione la trovate meglio descritta nel **Listato 5**. Se avete modo di scaricare il data-sheet di questo backpack, noterete che risulta compatibile anche con un altro display "Huge Graphic LCD 160x128 pixel" di grandi dimensioni e con risoluzione maggiore. La gestione a livello software avviene nello stesso modo in quanto, ovviamente, la scheda Arduino dialoga ancora con il backpack, che riesce a gestire il display in completa autonomia. Con la retroilluminazione spenta la visibilità è ridotta, quindi abbiamo previsto un apposito sketch (display_05) che imposta anche una minima retroilluminazione. Il Backpack viene normalmente fornito compatibile con il display 128x64, ma è possibile impostarlo per funzionare con il display 128x160: allo scopo basta rimuovere il jumper posto sullo stampato. Tale jumper è davvero minuscolo ed è realizzato con una piccola saldatura (una goccia di stagno) che dovrà essere rimossa con molta attenzione. Bene, anche per questa puntata è tutto; avete tempo fino alla prossima per fare tutti gli esperimenti e le prove pratiche descritte. ■

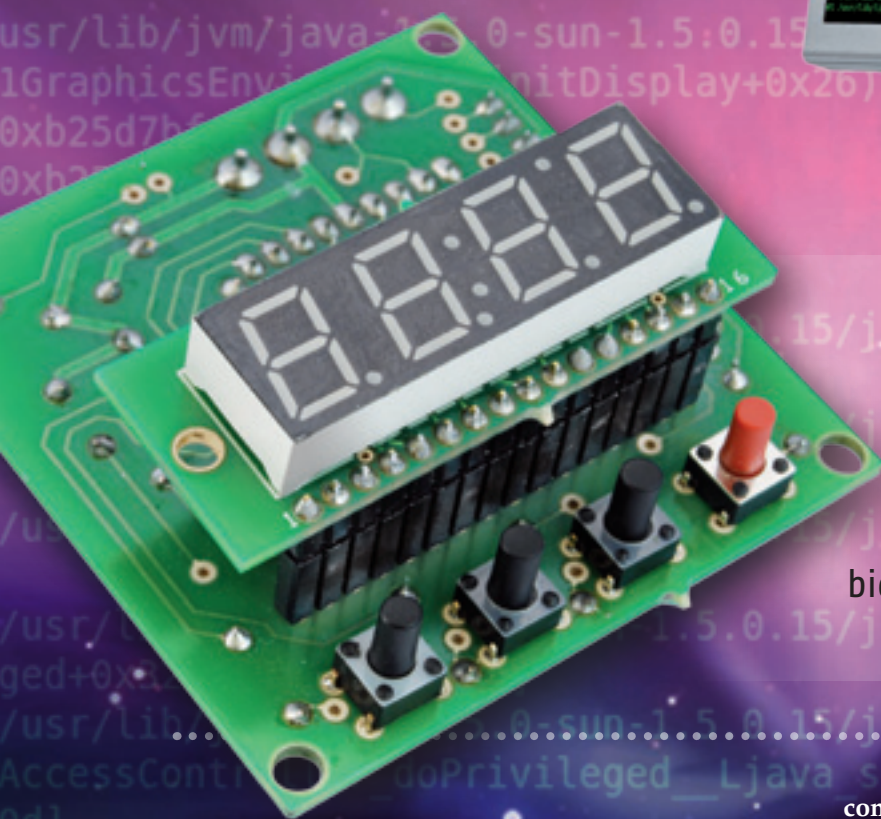


Le due piazzole da cui rimuovere il cortocircuito a stagno per impostare la scheda Arduino in modo da farle controllare un display da 128x160 pixel.



NANOTERM

IL PICCOLO TERMINALE UNIVERSALE



Terminale per svariati dispositivi a microcontrollore, ha quattro display a sette segmenti a LED per visualizzare scritte e numeri e altrettanti pulsanti per impartire comandi o attivare funzioni. La sua porta seriale TTL bidirezionale gli permette di interfacciarsi con tutti i microcontrollori e non solo.

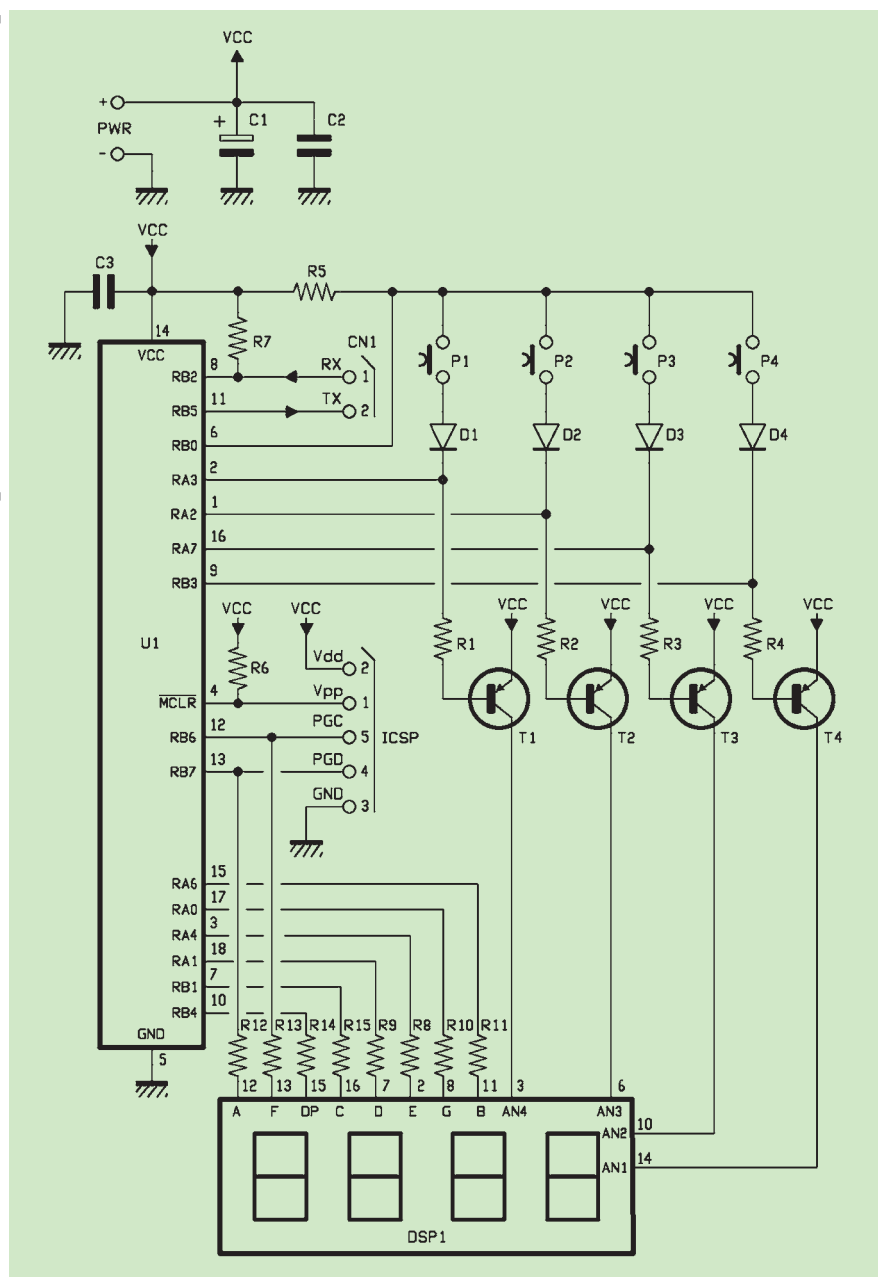
di CRISTIANO RUGGERI

Vi è mai capitato, nei vostri progetti elettronici, di dover visualizzare alcune informazioni in formato numerico e di non

avere a disposizione sufficienti linee di uscita sul vostro microprocessore per poter gestire un display? Oppure di necessitare di alcuni

comandi a pulsante e di ritrovarvi nella medesima situazione? In entrambi i casi la scelta è obbligata: si rinuncia ad alcuni pulsanti e/o alla visualizzazione, oppure si sceglie un micro-

controllore dotato di un numero maggiore di linee di ingresso e uscita, ma in questo caso il circuito si complica immancabilmente. Nello sviluppare un progetto ci capita sempre più frequentemen-



te di dover aggiungere pulsanti e un display dove visualizzare poche informazioni in formato testuale/numerico: ogni volta ciò comporta modifiche al firmware ed un lungo lavoro di debug del codice già scritto per altri progetti, ma che deve essere riadattato a seconda delle esigenze del caso. Indubbiamente ciò porta a perdite di tempo e lavoro che si potrebbero evitare se esistesse qualcosa di preconfezionato in grado di gestire pulsanti e display contemporaneamente. In

verità qualcosa del genere esiste da tempo: infatti quando i Personal Computer non erano ancora diffusi, nelle aziende esistevano i mainframe AS400, grossi elaboratori IBM centralizzati cui ogni utente accedeva facendo uso di un terminale, dotato di un monitor e di una tastiera simili a quelle degli odierni PC. Questi terminali più o meno standard (ad esempio VT100, TTY, Minitel, nomi ancora visibili nel setup di Hyper Terminal di Windows) all'inizio comunicavano col

mainframe attraverso una seriale veloce: per lungo tempo, intere multinazionali hanno basato il loro business su questa soluzione arcaica ma, nel contempo, talmente funzionale da essere utilizzata (chiaramente con le opportune evoluzioni tecnologiche) ancora ai giorni nostri.

Fatto sta che se IBM ha avuto l'idea dei terminali seriali, questa doveva essere (come si è dimostrato nel tempo) vincente. Riflettendo sull'architettura del genere ci siamo chiesti: "se esistesse un mini-terminale compatto ed economico, in grado di comunicare via seriale con qualsiasi microcontrollore in qualunque progetto, probabilmente il debug dei vari circuiti si ridurrebbe al verificare che i dati scambiati tra microcontrollore e terminale siano corretti". Purtroppo un dispositivo del genere non esiste, quindi abbiamo pensato di realizzarlo da noi e pubblicarlo in queste pagine. Il progetto di cui parliamo è quindi un piccolo terminale che dispone di quattro display a LED a 7 segmenti e di altrettanti pulsanti. Sui display è possibile visualizzare numeri e lettere (alcune come la W, la K e la X sono un po' approssimate, ma tutte le altre sono ben riconoscibili) trattino alto e basso, spazio, punto decimale e cerchietto alto (°C).

Relativamente ai pulsanti, il dispositivo riconosce gli stati di "premuto", "mantenuto premuto" e "rilasciato". La comunicazione avviene attraverso una seriale asincrona bidirezionale a 19.200 baud che comunica con un protocollo ad 8 bit dati, nessuna parità, 1 bit di stop.

La linea di ricezione viene utilizzata per ricevere i dati da visualizzare a display, mentre quella di trasmissione invia lo stato dei pulsanti e le due cose

CARATTERISTICHE TECNICHE

- **Display:** 4 gestiti in multiplexing;
- **Pulsanti:** 4 letti in multiplexing (contemporaneità supportata);
- **Tensione di alimentazione:** 3÷5,5 Vcc stabilizzati;
- **Corrente assorbita:** 50mA @ 3 Vcc; 120 mA @ 5,5 Vcc;
- **Frequenza di multiplexing** 48 Hz;
- **Comunicazione:** Seriale bidirezionale half-duplex, 9.600/19.200 baud, nessuna parità, 8 bit dati, 1 stop bit. Protocollo di comunicazione semplificato;
- **Frequenza di scrittura massima/consigliata:** 40/5 ricezioni seriali/scrittura a display al secondo;
- **Diagnostica:** Test pulsanti interno, abilitabile mediante procedura specifica.

sono svincolate, eccezion fatta per la modalità di test, che verrà spiegata più avanti.

SCHEMA ELETTRICO

Come potete constatare, lo schema elettrico è piuttosto semplice: il cuore del circuito è un microcontrollore PIC16F88, scelto in quanto dispone di una seriale hardware e di un numero di linee di ingresso e uscita sufficienti per lo scopo. Questi comunica con il mondo esterno attraverso le linee seriali TX (pin 11, RB4) ed RX (pin 8, RB2); la resistenza di pull-up R7 garantisce il livello logico 1 sulla linea RXD qualora i dati ricevuti provenissero da una seriale open-collector.

I dati sono elaborati, convertiti in formato a 7 segmenti e quindi visualizzati sui display, questi ultimi gestiti in multiplexing. Con questa tecnica, i display vengono accesi ciclicamente uno per volta, in sequenza. Le linee RB3, RA7, RA2 e RA3 comandano, attraverso i transistor PNP BC327 siglati T4, T3, T2 e T1 le linee di accensione dei quattro anodi dei display a sette segmenti ad anodo comune, mentre le linee RB7,

RA6, RB1, RA1, RA4, RB6, RA0 e RB4 gestiscono i sette segmenti da A a G e il punto decimale di tutti i display in parallelo.

In pratica, dovendo visualizzare ad esempio la scritta 1234, il micro cambia lo stato delle linee dei sette segmenti per realizzare la cifra 4 (segmenti B, C, F e G a OFF) quindi pone per un istante la linea di comando RA3 a livello logico basso, polarizzando la base il transistor T4 mediante la resistenza R4. Il PNP satura alimentando l'anodo e facendo così accendere il primo display, su cui viene visualizzata la cifra 4. Il PIC spegne quindi il display, cambia nuovamente lo stato dei segmenti per avere la cifra 3 (segmenti A, B, C, D e G a OFF) e attiva la linea RA2, accendendo così il secondo display e facendo apparire su di esso la cifra 3. La procedura si ripete per tutti i display e, giunto all'ultimo, il ciclo riparte dall'inizio.

Di fatto, le cifre non vengono visualizzate tutte contemporaneamente, bensì in sequenza, ma ad una velocità sufficientemente alta (in questo progetto, 48 volte al secondo) da ingannare l'occhio umano che, pertanto, vede i quattro display sempre accesi.

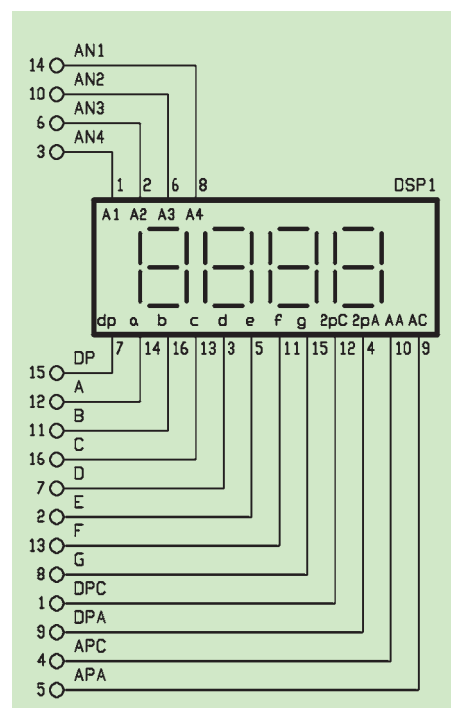
Le linee RA4, RB6, RA0 e RB4 vengono utilizzate anche per rilevare lo stato dei pulsanti P1÷P4, attraverso i diodi D1÷D4. Notate che i pulsanti sono collegati, da una parte, tutti all'ingresso RB0, e dall'altra ognuno a un proprio diodo.

Quando viene premuto un pulsante, la linea di comando display cui esso è associato viene chiusa tramite il diodo sull'ingresso RB0; in questo modo, lo stato della linea di comando viene riportato pari-pari sull'ingresso. In pratica, se la linea RA4 è a zero, significa che il display 4 è attivato: se in questa condi-

zione anche l'ingresso RB0 è a zero, significa che il pulsante P4 è premuto, altrimenti è rilasciato. Lo stesso discorso vale per tutti gli altri pulsanti. Di fatto, quando il micro attiva il primo display, legge anche lo stato del pulsante 1. Se scrive nel display 2, legge anche il pulsante 2 e così via. Lo stato dei pulsanti viene inviato alla trasmissione seriale, in modo da poter essere gestito esternamente.

Nel circuito è presente l'immanicabile connettore ICSP, che ci permette la programmazione in-circuit del terminale.

Non sono, invece, presenti regolatori di tensione, poiché il dispositivo è stato pensato per l'impiego in circuiti dotati di un proprio regolatore, ossia per essere alimentato dal circuito cui verrà collegato. A tal proposito, tenete presente che il terminale qui proposto può funzionare con una tensione compresa fra 3 e 5,5 V applicati tra i morsetti + e - PWR. A 3 V consuma circa 50 mA massimi, mentre a 5,5 V necessita di



[schema elettrico DISPLAY]

circa 120 mA. Le resistenze di limitazione della corrente dei vari segmenti (tutte da 180 ohm) sono state ottimizzate per far scorrere in essi una corrente di circa 16 mA a 5 V (le uscite del micro arrivano a 20 mA), valore che diviso per quattro (ogni display è acceso una sola volta per ogni scansione su quattro display) dà come risultato 4 mA medi, intensità sufficiente per qualsiasi display a LED ad alta efficienza. Se vorrete ridurre la tensione di alimentazione senza perdere in luminosità, il valore da utilizzare per le resistenze potrà essere ottenuto con la formula:

$$R_{seg} = (V_{lim} - V_{sat} - V_{led}) / (I_{led} \times 4)$$

i cui termini significano:

R_{seg} = valore della resistenza in serie, in ohm;

V_{lim} = tensione di alimentazione, in volt;

V_{sat} = tensione di saturazione del transistor, circa 0,2 V;

V_{led} = caduta di tensione del led del display (1,6 V per il rosso, 1,9 V per il verde, 2,2 V per il blu eccetera) in volt;

I_{led} = corrente nominale nel led (3÷5 mA per i led ad alta efficienza) in ampere.

Se, ad esempio, volessimo utilizzare il dispositivo alimentandolo a 3 V e con dei display verdi, che hanno una caduta di tensione diretta (per segmento) di circa 1,9 V

a 4 mA, il valore delle resistenze di caduta sarebbe:

$$R_{seg} = (3 \text{ V} - 0,2 \text{ V} - 1,9 \text{ V}) / (0,004 \times 4) = 56 \text{ ohm.}$$

Analogamente, usando quelli a LED blu (2,2 V) a 4,5 V (tre batterie AA in serie) necessiterebbero di resistenze da:

$$R_{seg} = (4,5 - 0,2 - 2,2) / (0,004 \times 4) = 131 \text{ ohm}$$

Valore approssimabile, per i componenti della serie E12, a 120 oppure a 150 ohm.

IL FIRMWARE

Questo dispositivo, per quanto semplice nell'uso e nel concetto, necessita di un firmware piuttosto complesso, che trovate schematizzato nel diagramma di flusso visibile nella Fig. 1. Durante la fase di avvio, il programma inizializza in modo opportuno le linee di I/O del micro e tutti i registri coinvolti nel normale funzionamento, quindi invia sulla seriale il comando di BOOT (0xFE), con cui comunica ad un eventuale micro esterno di essere in avvio e pronto per emettere dei messaggi di boot. Si pone quindi in attesa, per un secondo, di un comando di SKIP (0xFD), comando che potrebbe provenire dal micro esterno per forzare il terminale a non generare i messaggi di benvenuto e ad evitare il controllo dei pulsanti per l'avvio del modo di test (TEST MODE).

A comando di SKIP ricevuto in modo corretto, corrisponde un salto al loop principale senza emissione di alcun messaggio di benvenuto a display. Nel caso di carattere di SKIP non ricevuto o ricevuto errato, vengono, scritti a display tre messaggi di benvenuto modificabili solo dal sorgente;

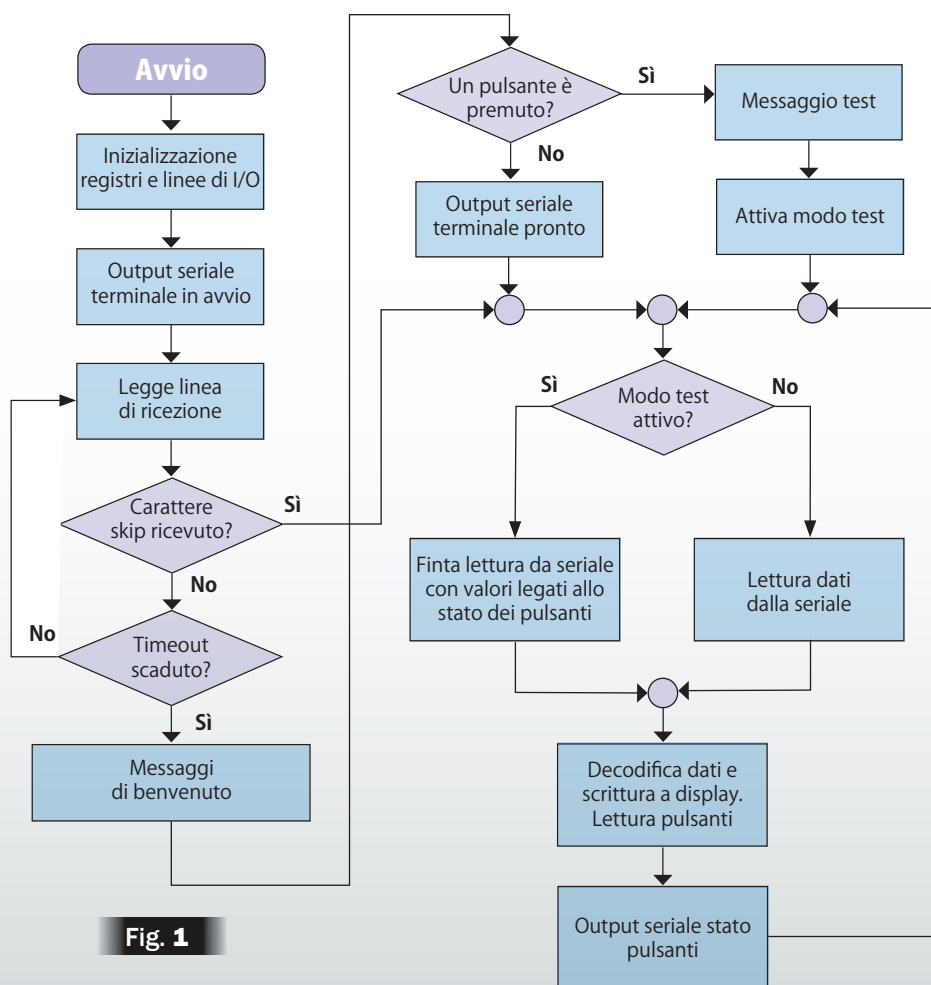


Fig. 1

Gestione dei display in multiplexing

Nella gestione di display a 7 segmenti in multiplexing, tutti i segmenti dello stesso tipo (tutti i segmenti a, tutti i segmenti b, tutti i punti decimali, ecc.) sono collegati in parallelo tra loro e pertanto, ipotizzando di alimentare il comune di ogni display, dando tensione ad ogni linea si accenderebbero tutti i segmenti collegati ad essa. Per evitare ciò, occorre pilotare i display con un'elettronica di controllo in grado di agire contemporaneamente sulle linee sia dei singoli segmenti che di abilitazione dei vari display (digit 1, digit 2 ecc.). Per visualizzare a display cifre differenti, occorre agire per fasi successive: si invia lo stato dei segmenti al primo display e si abilita questo per un certo tempo, poi lo si disabilita. Quindi si manda al secondo display lo stato dei segmenti, si abilita lo stesso display e poi lo si disabilita; si ripete quindi il ciclo per gli altri display. Se la velocità di questa scansione è sufficientemente alta (48 Hz per esempio) l'occhio umano non si accorge della sequenzialità e percepisce i display come sempre accesi. 48 Hz corrispondono a circa 20,8 ms per una scansione completa, pertanto ognuno dei quattro display rimane abilitato per 5,2 ms. Dato che ogni display rimane

Cifra	Seg. a	Seg. b	Seg. c	Seg. d	Seg. e	Seg. f	Seg. g	Seg. DP	Cifra	Seg. a	Seg. b	Seg. c	Seg. d	Seg. e	Seg. f	Seg. g	Seg. DP
0	0	0	0	0	0	0	1	1	Kk	1	1	1	1	0	0	0	1
1	1	0	0	1	1	1	1	1	Ll	1	1	1	0	0	0	1	1
2	0	0	1	0	0	1	0	1	Mm	0	0	0	1	0	0	1	1
3	0	0	0	0	1	1	0	1	Nn	1	1	0	1	0	1	0	1
4	1	0	0	1	1	0	0	1	Oo	1	1	0	0	0	1	0	1
5	0	1	0	0	1	0	0	1	Pp	0	0	1	1	0	0	0	1
6	0	1	0	0	0	0	0	1	Qq	0	0	0	1	1	0	0	1
7	0	0	0	1	1	1	1	1	Rr	1	1	1	1	0	1	0	1
8	0	0	0	0	0	0	0	1	Ss	0	1	0	0	1	0	0	1
9	0	0	0	0	1	0	0	1	Tt	1	1	1	0	0	0	0	1
Aa	0	0	0	1	0	0	0	1	Uu	1	0	0	0	0	0	1	1
Bb	1	1	0	0	0	0	0	1	Vv	1	1	0	0	0	1	1	1
Cc	0	1	1	0	0	0	1	1	Ww	1	0	0	0	0	0	0	1
Dd	1	0	0	0	0	1	0	1	Xx	0	1	1	0	1	1	0	1
Ee	0	1	1	0	0	0	0	1	Yy	1	0	1	1	0	0	0	1
Ff	0	1	1	1	0	0	0	1	Zz	0	0	1	0	0	1	0	1
Gg	0	1	0	0	0	0	1	1	-	1	1	1	1	1	1	0	1
Hh	1	0	0	1	0	0	0	1		1	1	1	0	1	1	1	1
Ii	1	1	1	1	0	0	1	1	spazio	1	1	1	1	1	1	1	1
Jj	1	0	0	0	1	1	1	1	.	0	0	1	1	1	0	0	1

La tabella mostra la corrispondenza tra il carattere da visualizzare ed i segmenti da accendere per visualizzarla.

accesso solamente per $\frac{1}{4}$ del tempo di scansione, per garantire una luminosità adeguata esso necessita di una corrente di pilotaggio quattro volte superiore a quella che necessiterebbe se fosse costantemente acceso. Posto ad esempio, di accendere ogni segmento con 10 mA, in multiplexing dovremmo calcolare le resistenze di limitazione per una corrente quattro volte superiore, ossia 40 mA.



Se non facessimo così, la luminosità sarebbe molto bassa. In linea di massima, la corrente di pilotaggio di ogni segmento è x volte superiore a quel-

la di alimentazione tipica (10 mA) e x rappresenta il numero di cifre da cui la nostra matrice di display è composta (nel nostro caso x vale 4).

il messaggio predefinito è NANO TERM_1.00. Immediatamente dopo l'invio dell'ultimo messaggio, viene verificato lo stato dei pulsanti; se almeno uno di essi è premuto (richiesta di attivazione MODO TEST), viene visualizzato il messaggio TEST e si attiva il MODO TEST. Se, invece, sono tutti a

riposo, viene inviato il comando di terminale pronto (0xFF) ed avviato il ciclo principale. Durante questo ciclo, il programma controlla sempre lo stato del modo test. Se non è attivo, vengono letti i dati dalla seriale; se lo è, il terminale ignora la seriale e genera internamente i dati da visualizzare in base allo stato dei

pulsanti. Il pulsante che si trova a riposo è sempre evidenziato con il trattino alto, mentre quello premuto viene indicato con il numero del pulsante scritto sul display corrispondente. Se, ad esempio, P1 fosse premuto, verrebbe scritto 1 sul primo display e trattino su tutti gli altri display. Se fossero premuti P1 e

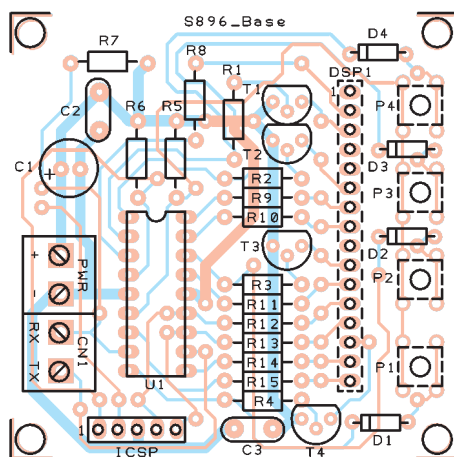
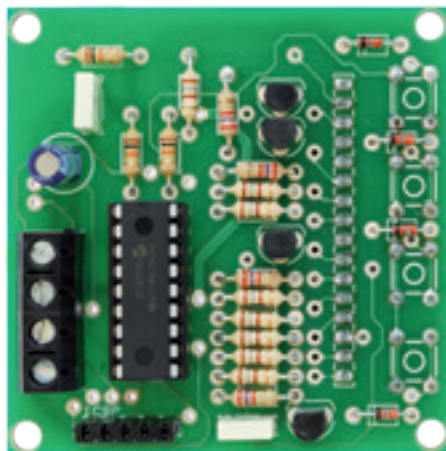
[piano di montaggio **BASE**]

Elenco Componenti:

R1: 2,7 kohm
 R2: 2,7 kohm
 R3: 2,7 kohm
 R4: 2,7 kohm
 R5: 10 kohm
 R6: 10 kohm
 R7: 10 kohm
 R8: 180 ohm
 R9: 180 ohm
 R10: 180 ohm
 R11: 180 ohm
 R12: 180 ohm
 R13: 180 ohm
 R14: 180 ohm
 R15: 180 ohm
 C1: 10 μ F 16 VL
 elettrolitico
 C2: 100 nF 63 VL
 poliestere
 C3: 100 nF 63 VL poliestere
 U1: PIC16F88-I/P (MF896)
 D1 ÷ D4: 1N4148
 T1: BC327
 T2: BC327
 T3: BC327
 T4: BC327
 P1 ÷ P4: Microswitch

Varie:

- Morsettiera 2 poli passo 5 mm (2 pz.)
- Strip femmina 5 poli
- Strip femmina 16 poli
- Zoccolo 9+9
- Circuito stampato



P4 vedremmo a display "4 - - 1", eccetera. Per uscire dal modo test, occorre spegnere e riaccendere il nostro terminale. Quale che sia la provenienza dei dati (seriale o autogenerazione in modo test) essi vengono decodificati e quindi scritti sul display. Durante la fase di scrittura dei dati, viene anche letto lo stato dei pulsanti, stato che è inviato alla seriale immediatamente dopo aver aggiornato il display. Per non complicare il diagramma di flusso, sono state volutamente

omesse le fasi relative alle routine che conteggiano il tempo di pressione del pulsante e il successivo ritorno a riposo, utilizzate per generare un solo output distinto contenente le informazioni di pulsante premuto, mantenuto premuto, e tornato a riposo. In pratica, la scrittura su seriale avviene ad ogni cambio di stato del pulsante, tranne che per la segnalazione dello stato di riposo, generata solo quando tutti i pulsanti tornano ad OFF. Qualora vogliate saperne di più

sul software, potete scaricare dal nostro sito il sorgente, che è ben commentato in ogni linea.

IL PROTOCOLLO

Il modo con cui il nostro dispositivo comunica con il mondo esterno non deve essere complicato, perché altrimenti si perderebbe uno dei requisiti base che hanno portato alla realizzazione del progetto, ossia la facilità d'uso.

Trasmissione

Tutte le trasmissioni sono effettuate con un solo byte, su cui vengono inseriti gli stati di pulsante premuto, mantenuto premuto, rilasciato e terminale pronto. Il formato è il seguente:

*Bit0: 0 = P1 a riposo;
 1 = P1 premuto.
 Bit1: 0 = P2 a riposo;
 1 = P2 premuto.
 Bit2: 0 = P3 a riposo;
 1 = P3 premuto.
 Bit3: 0 = P4 a riposo;
 1 = P4 premuto.
 Bit4: non usato.
 Bit5: non usato.
 Bit6: 0 = Non rilevante;
 1 = Pulsanti mantenuti premuti.
 Bit7: non usato.*

In pratica, ogni volta che uno o più pulsanti vengono premuti, il terminale invia un carattere in cui il bit relativo ad ogni pulsante premuto viene posto a 1. Dopo un secondo, se almeno un pulsante è ancora premuto, viene inviato un secondo byte in cui sia il bit relativo al pulsante che il bit 6 sono ad 1. Quando il pulsante viene rilasciato, il terminale invia un byte a zero (0x00) per comunicare che tutti i pulsanti sono a riposo. Nel caso di pressione simultanea di più tasti, lo zero viene inviato solo quando tutti tornano a riposo. Facciamo alcu-

ni esempi per chiarire la cosa:

*%00001000, 0x08 vuol dire pulsante 4 premuto;
%01001000, 0x48 corrisponde a pulsante 4 mantenuto premuto;
%00000000, 0x00 indica pulsanti a riposo.*

Esistono ancora due segnalazioni, ossia quella di terminale in avvio (BOOT, 0xFE) inviata immediatamente dopo il reset del circuito, e quella di terminale pronto (READY, 0xFF) inviata una sola volta durante la fase di accensione, terminata la messaggistica di benvenuto. Se dopo l'invio del messaggio di BOOT (0xFE) il terminale riceve entro un secondo il messaggio di SKIP (0xFD), allora esso invia immediatamente il messaggio di pronto READY (0xFF), salta tutta la messaggistica di benvenuto e la richiesta di modalità test, per passare direttamente al loop principale. Nel caso in cui il carattere di SKIP non venga ricevuto o sia ricevuto sbagliato, allora l'avvio procederà regolarmente con i messaggi di benvenuto e la verifica di richiesta di test mode, come vedremo più avanti.

Ricezione

La ricezione può essere di due tipi: quella durante la fase di BOOT (avvio) e quella nel corso della normale attività del sistema. Nel primo caso il circuito attende un solo carattere definito di SKIP (0xFD, 253, salto dei messaggi e della verifica di modo test) che deve arrivare entro un secondo dalla trasmissione del segnale di BOOT (0xFE) da parte del terminale. Ricevuto questo, tutti i messaggi di benvenuto generati dal terminale vengono evitati. Ciò diventa utile quando il terminale viene utilizzato con un sistema che prevede delle sue

Nuovi terminali per Android

Sono sempre più i costruttori di smartphone che adottano Android, il sistema operativo per terminali mobili evoluti creato da Google e basato su kernel Linux, quindi open-source. Dopo HTC, nota azienda produttrice di terminali mobili evoluti (che circa un anno fa ha immesso sul mercato il primo telefonino equipaggiato con Android) anche grandi nomi come Sony Ericsson, Huawei, LG, stanno "sforando" prodotti sviluppati per lavorare con Android, alcuni dotati di tastiera Swipe (l'innovativa tastiera che permette di scrivere più velocemente facendo scorrere

le dita di lettera in lettera). Il successo di Android sta nel fatto che è open-source e quindi si presta ad essere migliorato, modificato ed integrato con programmi ed hack scritti dagli utenti e dai programmatori di tutto il mondo. Al fascino di Android hanno ceduto anche grandi gestori come TIM e Vodafone, che già offrono terminali dell'HTC.



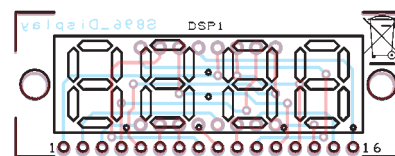
segnalazioni di BOOT, le quali hanno diritto di prelazione sulla messaggistica del terminale. Nel secondo caso, quello di normale funzionamento, la ricezione è costituita da una stringa di 6 byte, secondo la struttura seguente:

SYNCH,DATO4,DATO3,DATO2

,DATO1,PUNTO_DECIMALE.

In essa SYNCH è il carattere di sincronismo di stringa, in questo caso predefinito come "*" (asterisco). Può essere cambiato a piacimento da sorgente, dato che esso è stato definito in una costante chiamata SYNCH. DATO4÷1 rappresenta il carattere che verrà

[piano di montaggio **DISPLAY**]



Elenco Componenti:

DSP1: Display 4x7 segmenti anodo

comune

- Strip maschio 16 poli
- Circuito stampato

I comandi seriali del dispositivo.

HEX	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Note
0x00	0	0	0	0	0	0	0	0	Tutti i pulsanti tornati a riposo.
0x01+0x0F	0	0	0	0	x	x	x	x	P1+P4 premuti.
0x41+0x4F	0	1	0	0	x	x	x	x	P1+P4 mantenuti premuti.
0xFE	1	1	1	1	1	1	0	0	Terminale in avvio (BOOT).
0xFF	1	1	1	1	1	1	1	1	Terminale pronto (READY).

visualizzato sul display 4÷1; tale carattere deve pervenire in formato ASCII e può essere un numero, una lettera qualsiasi, sia maiuscola che minuscola, lo spazio, il trattino basso “_” e quello alto “-” e l’apostrofo “'” per il grado centigrado. PUNTO_DECIMALE rappresenta il display vicino al quale verrà visualizzato il punto decimale. Questo campo ha valore numerico e può valere da 0x00 a 0x03 secondo la logica seguente:

*0x00 = nessun punto decimale;
0x01 = punto decimale a sinistra del primo display;
0x02 = punto decimale a sinistra del secondo display;
0x03 = punto decimale a sinistra del terzo display.*

Facciamo alcuni esempi che aiutano a comprendere: volendo scrivere “12,34”, la sequenza

da inviare al terminale sarebbe “*1234”, 2, che in PicBasic può essere tradotta come HSEROUT [“*1234”,2]. Analogamente, per scrivere “CIAO” dovremmo inviare “*CIAO”, 0, equivalente ad HSEROUT [“*CIAO”,0]. La massima frequenza di aggiornamento consigliata è circa 20 Hz, che corrispondono a 20 visualizzazioni al secondo. In realtà, l’apparato funziona anche con velocità maggiori: è stato provato fino a 40 aggiornamenti al secondo, ma la velocità è tale da non permettere alcun apprezzamento visivo. Sperimentalmente, la miglior resa è stata ottenuta con 3÷5 aggiornamenti al secondo. Prestate attenzione ad un particolare molto importante: il microcontrollore PIC16F88 lavora con il clock interno di 8 MHz, che è piuttosto basso per velocità di comunicazione seriale come quella da noi adottata; pertanto il baud-rate di 19.200 baud è affetto da un errore inevitabile, dovuto ai divisori di clock interni al PIC usati dal generatore di baud-rate. Questo errore è documentato persino nel data-sheet del PIC ed in pratica determina che i 19.200 baud teorici risultano essere in realtà 17.730 baud. Ciò significa che quando si utilizzerà il NanoTerm con un altro micro, occorrerà fare in modo che anche il baud-rate di quest’ultimo venga generato con lo stesso errore. Se utilizzerete un secondo PIC16F88 a 8 MHz, l’errore sarà identico e pertanto non ci saranno perdite di dati. Invece se impiegherete microcontrollori differenti, dovrete necessariamente calibrare di nuovo il clock del baud-rate. Per esempio, se state

usando un PIC12F683, potrete utilizzare, per l’output seriale, il comando PicBasic:

```
SEROUT2 TXD,BR_GEN,[“Dati da trasmettere...”]
```

ponendo BAUD_RATE = 37. In PICBASIC, infatti, il parametro BR_GEN viene calcolato con la formula:

$$BR_GEN = (1000000 / \text{Baudrate}) - 20$$

Nel nostro caso si ottiene $BR_GEN = (1000000 / 17730) - 20 = 57 - 20 = 37$.

Altrimenti potrete scendere a 9.600 baud (ricompilando il codice del NanoTerm): la velocità è sempre affetta da un errore, ma di minor entità, per cui si percepirà un leggero sfarfallio del display ma non si avranno errori di trasferimento dati per effetto di baud-rate affetti da inevitabili errori di approssimazione. Quando realizzerete il dispositivo, potrete comunque scegliere se scaricare dal nostro sito il codice compilato per i 9.600 (MF896_96) o (MF896_192) per i 19.200 baud.

REALIZZAZIONE PRATICA

Il dispositivo è realizzato su due circuiti stampati distinti: il primo integra solamente i quattro display, con i catodi opportunamente collegati in parallelo. Il secondo, invece, include tutti gli altri componenti. Iniziate scaricando dal nostro sito www.elettronica.in.it i master in formato 1:1 dei due circuiti a doppia faccia, stampateli su fogli di

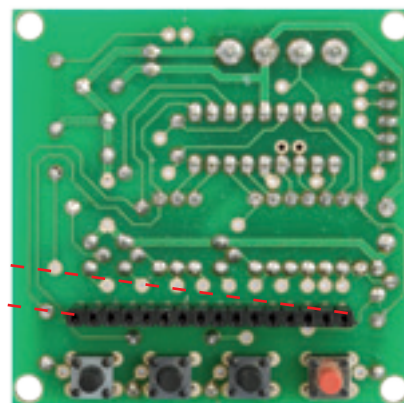


per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista così come il firmware utilizzato per programmare il microcontrollore PIC16F88. La versione non programmata di quest’ultimo componente costa 5,00 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>

Lo stampato con il display va dotato di un pin-strip e inserito nel connettore SIL della scheda base.



acetato trasparente e realizzate le due basette con la tecnica della fotoincisione. Incisi e forati gli stampati, iniziate il montaggio dalle vie, ossia i collegamenti tra piazzole poste su lati opposti del circuito. Procedete con le resistenze, con lo zoccolo del microcontrollore e con i transistor. Montate per ultimi i condensatori, i connettori, la morsettiera e i display sul circuito specifico per essi (decidete voi se li volete montare su zoccolo o direttamente sullo stampato). Saldare infine sul c.s. della scheda display un pin-strip maschio a 16 pin e su quello dell'altra scheda un pin-strip femmina, sempre a 16 pin. Eventualmente, per la disposizione aiutatevi con le immagini visibili nel piano di montaggio. Verificate con un tester, impostato sulle misure di continuità, che non vi siano cortocircuiti tra il positivo e il negativo della morsettiera e tra il TX e l'RX. Montate il PIC sul suo zoccolo e alimentate il circuito; in questa fase non preoccupatevi di eventuali accensioni casuali di segmenti, che sono del tutto normali. Usando il connettore ICSP, programmate (ad esempio tramite il programmatore PicKit 2) il microcontrollore con il firmware che trovate nel nostro

sito www.elettronica.in.it. Appena terminata la programmazione, il dispositivo parte automaticamente, visualizzando i tre messaggi di benvenuto (NANO TERM _1.00) e ponendosi in attesa di dati, stato questo segnalato dai quattro trattini che appaiono nel display. A questo punto, potete verificare il funzionamento dei pulsanti avviando la modalità di test: spegnete e riaccendete l'apparato mantenendo premuto uno qualsiasi dei pulsanti fino a far comparire la scritta TEST dopo il terzo messaggio di benvenuto (versione FW 1.00). Premete i pulsanti e verificate che il numero del pulsante premuto compaia sul relativo display. Potete collegare la sonda di un oscilloscopio (base tempi di 250 microsecondi, sensibilità d'ingresso 2 V/div.) tra la linea TX e la massa: ogni volta che premete o rilasciate un pulsante, vedrete un treno di impulsi corrispondente a un dato seriale. Mantenendo premuto un pulsante per almeno 1 s, viene

inviato un secondo dato seriale differente dal precedente per lo stato del bit 6, posto a 1 anziché a 0, per dare l'informazione di pulsante mantenuto premuto. Spegnete e riaccendete per uscire dal MODO TEST: il terminale è pronto per essere utilizzato nei vostri progetti.

NOTE CONCLUSIVE

Questo piccolo terminale si presta molto bene per realizzare la parte di visualizzazione in molti vostri apparati, purché disponiate di una porta seriale bidirezionale (reale o emulata che sia). Nei prossimi fascicoli vi mostreremo due progetti in cui lo abbiamo impiegato e precisamente: un misuratore di corrente/tensione/potenza da applicare ad alimentatori che ne sono sprovvisti; un alimentatore da laboratorio a tensione e corrente d'uscita regolabili. ■



Mostra Mercato Mondo Elettronica.net
4ª EDIZIONE

RAVENNA

25-26 SETTEMBRE 2010

Tel. e Fax 0545 27548 - Cell. 348 9030832 - www.mondoelettronica.net

FIERA DELL'ELETTRONICA
INFORMATICA E RADIANTISMO
ORARIO 9,00/18,30 - PARCHEGGIO GRATUITO

PALA DE ANDRÈ
Viale Europa 1

VALE COME RIDOTTO
ANCHE IN FOTOCOPIA