

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

€ 5,00

Elettronica In

oltre l'elettronica

www.elettronica.in.it

Anno XV - n. 138
Giugno 2009



L'auto del
FUTURO

Irrigatore ad ENERGIA SOLARE



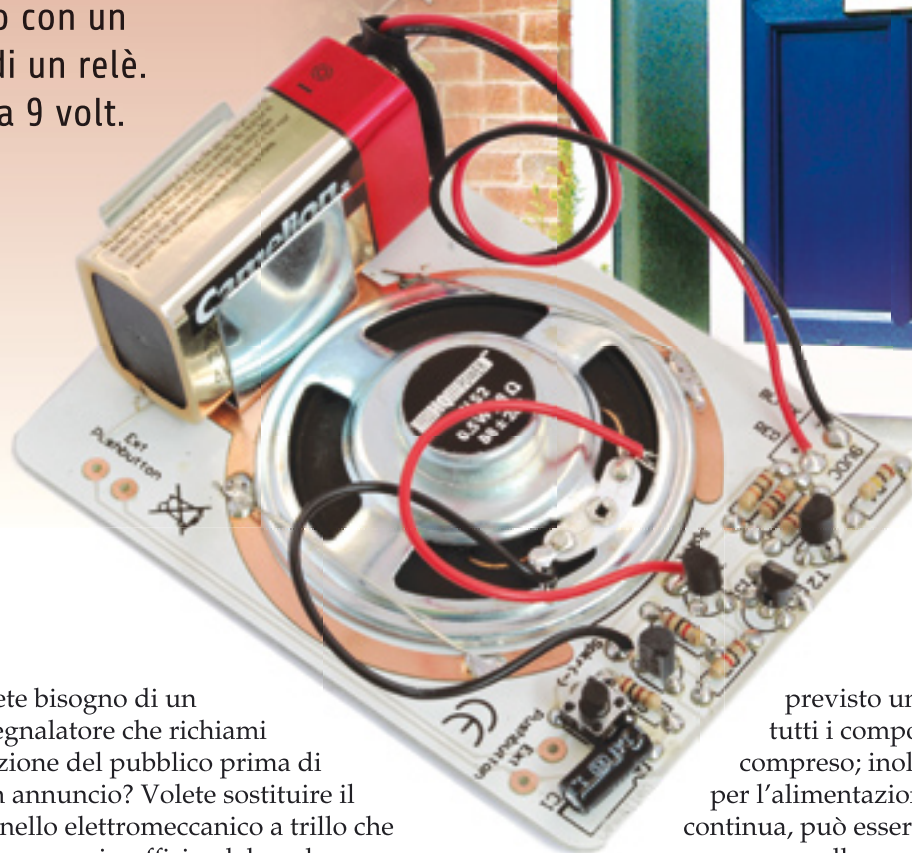
- Ricevitore GPS-USB
- DATA-LOGGER con VINCULUM VDIP1
- Ding-Dong elettronico
- 4 IN 4 OUT, espansioni e firmware
- VU-METER con barra a LED
- Registrare temperature con DTPIC18F67J50

ESCLUSIVO!
Corso dsPIC

DING-DONG ELETTRONICO

..... di DAVIDE SCULLINO

Avvisatore acustico allo stato solido che simula il suono di una campana con due tonalità; si comanda con un normale pulsante o con un contatto di un relè. Funziona a 9 volt.



Avete bisogno di un segnalatore che richiami l'attenzione del pubblico prima di fare un annuncio? Volete sostituire il campanello elettromeccanico a trillo che avete in casa o in ufficio, del quale non ne potete più? Eccovi dunque un progetto semplice e compatto, che va bene in tutte le situazioni in cui occorre un generatore di note acustiche di tipo ding-dong. Il circuito si presenta semplice e compatto, grazie anche al fatto che per esso abbiamo

previsto uno stampato contenente tutti i componenti, altoparlante compreso; inoltre, accontentandosi, per l'alimentazione, di appena 9 volt in continua, può essere impiegato non solo come campanello ma anche come avvisatore acustico generico in abbinamento con svariati apparati elettronici: ad esempio si può usare per segnalare lo scadere di un tempo, l'entrata di una persona o un'automobile in un luogo, il termine della cottura in un forno elettrico ecc. Proprio per esten-

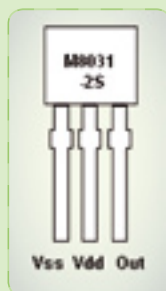
L'integrato M8031-2s

Si tratta di un piccolo chip customizzato prodotto dalla Bowin Electronic, progettato per generare il suono di un campanello a due campane (ding-dong) abbastanza fedelmente e similmente a quello prodotto dai campanelli da abitazione.

Il componente si presenta come un transistor in TO-92 ed ha quindi tre terminali, che sono l'alimentazione positiva, la massa e l'uscita del segnale audio; basta fornirgli da 1,3 a 4 volt (l'alimentazione ottimale è fra 1,3 e 3,3 V) in continua e fa tutto da solo. Il segnale sonoro (che

consta in una sequenza di due campane ripetuta due volte) viene generato non appena il chip riceve l'alimentazione, dopodiché l'M8031-2s si porta automaticamente in standby e può emettere un nuovo suono solo se privato dell'alimentazione per qualche istante.

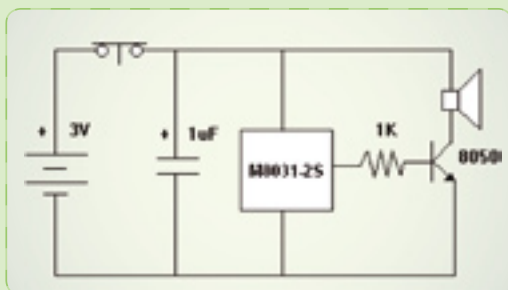
All'interno di questo inte-



- **Tensione di funzionamento: 1,3+3,3 V**
- **Assorbimento in stand-by: 0,5 µA max**
- **Assorbimento in funzione: 0,5 mA**
- **Frequenza oscillatore: 120 kHz**

grato troviamo un generatore di note basato su un oscillatore ad RC operan-

te a una frequenza tipica di 120 kHz (comunque per la tolleranza nel processo costruttivo e la deriva termica è ammissibile che la frequenza spazi tra 102 e 144 kHz) che genera la nota base, la quale viene poi modulata da una logica che produce il giusto involuppo dell'ampiezza e dello slittamento di frequenza.



derne le possibilità di utilizzo, il circuito è stato provvisto sia di un interruttore locale, sia di due ingressi di comando attivabili con contatti puliti o, all'occorrenza, con uscite a transistor, a patto di rispettare la polarità prevista. Nell'impiego come campanello, una volta installato il nostro "ding-dong" deve essere alimentato con un piccolo alimentatore da rete universale, da 500 mA, impostato per fornire 9 Vcc; una pila da 9 volt tornerà utile nei luoghi dove, pur non essendo presente la rete elettrica, desideriamo ugualmente un campanello.

Ma prima di parlare delle applicazioni e delle configurazioni, vediamo di cosa si tratta, andando a gettare uno sguardo allo schema elettrico.

SCHEMA ELETTRICO

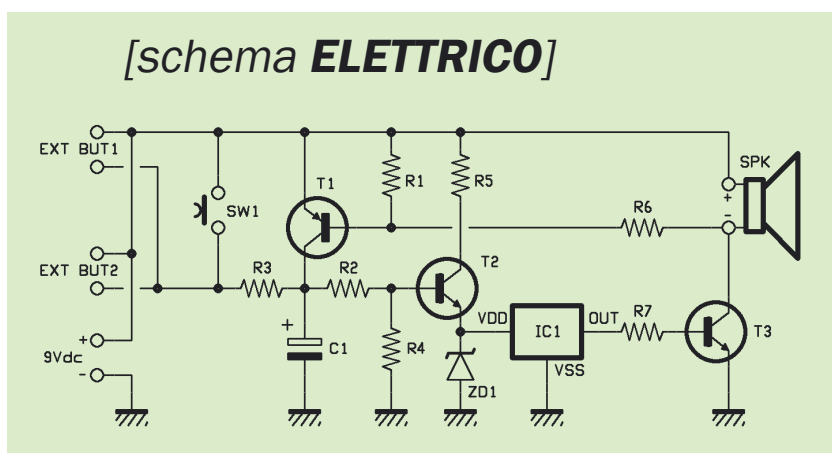
Come vedete, il circuito è l'essenza della semplicità, grazie soprattutto al fatto che le note acustiche vengono generate, cadenzate e riprodotte da un piccolo circuito integrato a tre terminali: l'M8031-2s della Bowin Electronic (<http://www.bowin-ic.com.hk/>); quest'ultimo è un generatore di suono bitonale capace di realizzare proprio il motivo prodotto da due campane, con il giusto involuppo: attivandolo emette il primo suono, che poi va in dissolvenza, mentre si sovrappone la seconda nota, la quale si dissolve anch'essa. L'M8031-2s contiene tutti i circuiti che servono, a partire dal generatore di note, che è un oscillatore temporizzato con una rete RC operante tipicamente

alla frequenza di 120 kHz; il segnale prodotto da questo viene modulato sia in frequenza che in ampiezza tramite una rete digitale che permette di ottenere l'involuppo desiderato. Il componente funziona a bassissima tensione (da 1,3 a 3,3 V cc) ed assorbe 0,5 mA in funzione e appena 0,5 µA in standby, ossia quando non genera il suono. A proposito di suono, l'M8031-2s entra in funzione non appena lo si alimenta e genera la propria sequenza acustica, sequenza consistente nell'emissione di due "ding-dong" consecutivi, spazati di circa mezzo secondo; poi si porta autonomamente in standby e per fargli riemettere il suono del campanello bisogna privarlo dell'alimentazione e alimentarlo nuovamente dopo qualche istante. L'integrato funziona senza richiedere componenti esterni, a parte quelli necessari a realizzare l'amplificatore occorrente a pilotare un altoparlante.

Nel nostro circuito il componente viene alimentato quando si preme il pulsante SW1 o si chiudono in cortocircuito uno o entrambi gli ingressi di comando EXT BUT1 ed EXT BUT2; quando avviene ciò, C1 si carica attraverso la resistenza R3 e, mediante il partitore formato da R2 ed



R4, polarizza l'NPN T2, che è montato nella configurazione a collettore comune. Il transistor applica così all'IC1 la tensione di alimentazione, che viene limitata a 3 V dal diodo Zener ZD1 in modo da presentare il valore idoneo alle esigenze dell'M8031-2s. Ora dovete notare che l'integrato inizia a produrre il suono delle campane e si prende tutto il tempo occorrente: infatti viene alimentato per un intervallo maggiore di quello consentito dalla carica data inizialmente a C1 dal pulsante; questo perché lo stesso transistor (T3, montato ad emettitore comune) utilizzato per amplificare il segnale di uscita e inviarlo a un piccolo altoparlante, con il proprio collettore polarizza ad impulsi la base del T1. Questo transistor a riposo è interdetto perché in condizioni di standby il collettore del T3, non avendo questo transistor segnale sulla base (a riposo l'OUT dell'IC1 è a zero volt) è interdetto; va in conduzione solo quando T3 riproduce il segnale. Ora va precisato che in pratica T1 funziona da amplificatore del segnale del campanello, ma in realtà a noi serve solo per ricaricare C1, giocando sul fatto che gli impulsi della componente audio danno un po' di carica, che basta a mantenere acceso IC1. Questo meccanismo consente all'M8031-2s di autosostenersi anche se il pulsante o gli ingressi di comando sono già stati rilasciati; naturalmente quando si completa la sequenza dei suoni, l'integrato va comunque in standby, perché la base del T1 non riceve più polarizzazione. Se il pulsante e gli ingressi di controllo non sono attivi, il circuito si porta a riposo e dopo un paio di secondi T2 va in interdizione; se invece SW1 è ancora premuto o EXT BUT1 o EXT BUT2 sono ancora chiusi, IC1 si spegne comunque



ed occorre rilasciare pulsante e ingressi per almeno un paio di secondi.

Questo intervallo serve a C1 a scaricarsi quanto basta a lasciar andare in interdizione T2 e privare l'M8031-2s dell'alimentazione; se non gli viene tolta la tensione, l'integrato non può eseguire alcuna sequenza acustica, nemmeno se si preme nuovamente SW1 o si chiude uno degli ingressi di comando esterno.

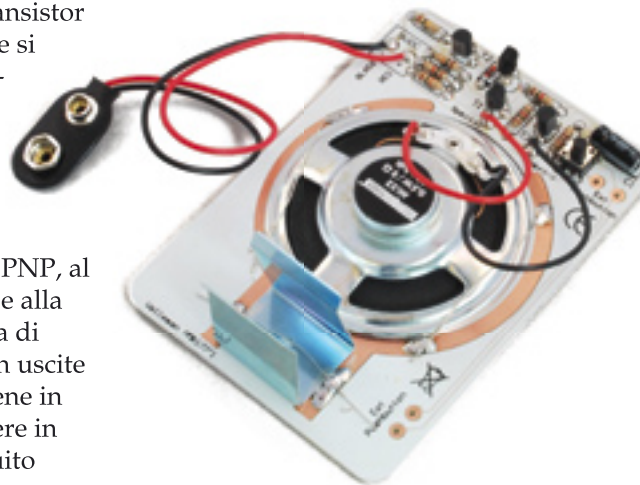
Completiamo la descrizione dicendo che gli ingressi EXT BUT1 ed EXT BUT2 sono stati pensati per consentire l'attivazione del campanello mediante pulsanti ausiliari (quindi quelli dell'impianto elettrico esistente) o uscite a contatto pulito (con relé) o anche a transistor; per collegare dei pulsanti non c'è alcun problema, mentre per i transistor occorre rammentare che se si tratta di NPN bisogna collegarli con il collettore al contatto positivo (quello corrispondente al + della pila) e l'emettitore all'elettrodo terminante su R3, mentre se sono dei PNP, al positivo ci va l'emettitore e alla R3 il collettore. Nulla vieta di comandare gli ingressi con uscite a livello di tensione, sebbene in questo caso occorra mettere in comune la massa del circuito

di comando con il negativo del campanello e applicare al punto d'ingresso corrispondente ad R3 da 7 a 10 Vcc quando si vuole avviare la sequenza sonora.

REALIZZAZIONE PRATICA

E con questo è tutto, almeno per la teoria; vediamo dunque come costruire il campanello, iniziando col dire che allo scopo è stato previsto uno stampato che ospiterà tutti i componenti, pila e altoparlante compreso; per realizzarlo scaricate il file contenente la traccia lato rame dal nostro sito www.elettronica-in.it e stampatelo su un foglio di carta da lucido o acetato, così da ottenere la pellicola per la fotoincisione.

Fatto ciò, sovrapponetelo al master così ottenuto a una basetta rama-

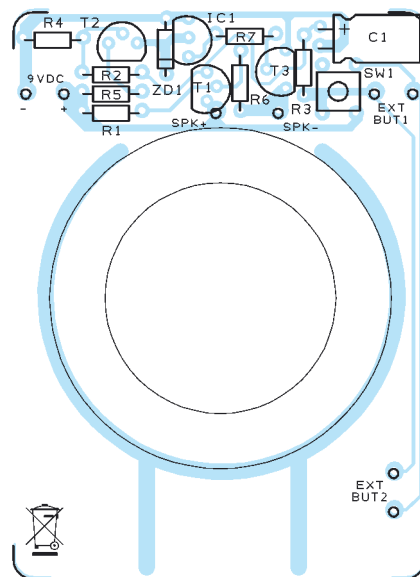
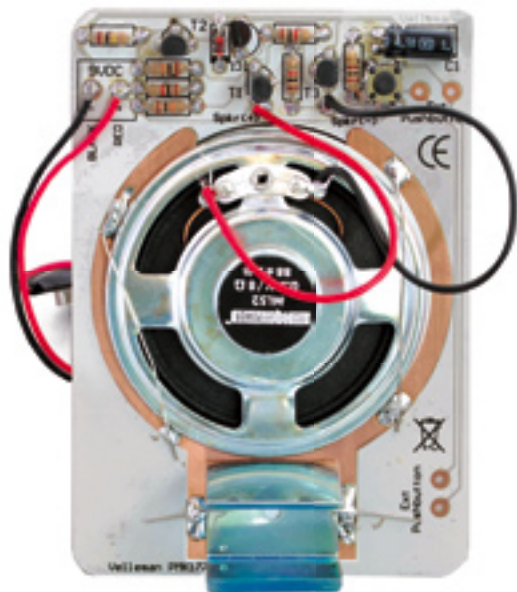


Elenco Componenti:

R1: 10 kohm
 R2: 10 kohm
 R3: 100 ohm
 R4: 100 kohm
 R5: 1 kohm
 R6: 1 kohm
 R7: 1 kohm
 C1: 10 μ F 35 VL elettrolitico
 ZD1: Zener 3V/400mW
 T1: BC557
 T2: BC547
 T3: BC547
 IC1: M8031-2S
 SW1: Microswitch

Varie:

- Altoparlante 0,5W/8 ohm
- Clip per batteria 9V
- Porta batteria 9V da cs
- Circuito stampato



ta monofaccia presensibilizzata e procedete all'impressione nel bromografo, dopodiché rimuovete la pellicola, inserite la basetta nel bagno di sviluppo e infine procedete all'incisione nell'apposita soluzione.

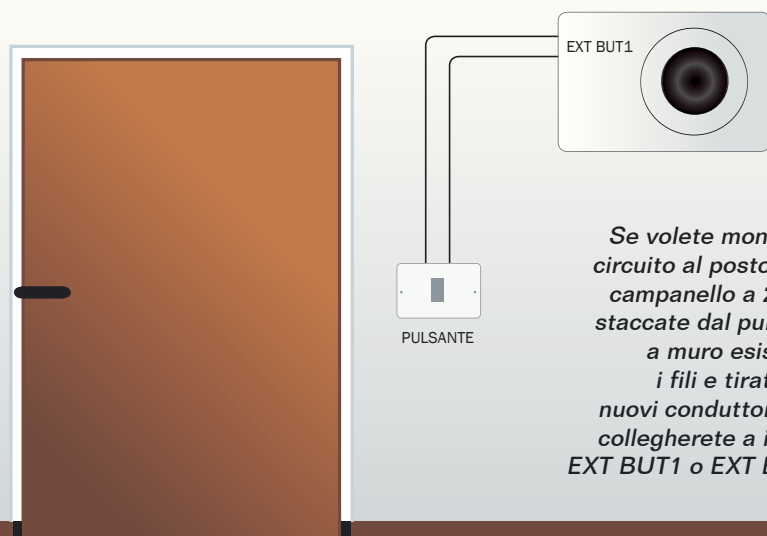
Ottenuta la basetta stampata foratela, rammentando che il buco dal quale deve affacciarsi la membrana dell'altoparlante conviene farlo realizzando tanti

piccoli fori con una punta da 1,5 mm di diametro lungo la circonferenza, quindi staccando l'interno e lavorando con una lima tonda la basetta in modo da eliminare le bave lasciate dal distacco. In alternativa, potete fare il foro usando una sega a tazza per plastica o ferro.

Fatto ciò, sullo stampato cominciate a disporre i componenti, badando che in questa realizzazio-

ne è previsto che siano montati dal lato delle piste; quindi introducetene i terminali dalla faccia ramata, stagnateli e tagliate l'eccedenza dal lato opposto. Iniziate con le resistenze e procedete con il diodo Zener, rammentando che il catodo di quest'ultimo corrisponde alla fascetta colorata sul suo corpo; inserite poi l'elettrolitico, prestando attenzione alla polarità indicata e dopo i tre transistor, per ognuno dei quali suggeriamo di verificare il verso d'inserzione nel disegno di montaggio illustrato in queste pagine. Completate l'insieme inserendo e saldando il pulsante miniatura da stampato e la clip per la pila da 9 volt; se pensate di alimentare il circuito con un piccolo alimentatore da rete, ai punti + e - 9Vdc collegate una presa plug da pannello facendo in modo che al positivo sia connesso l'elettrodo centrale.

L'altoparlante è previsto che sia collocato dal lato con la membrana rivolta verso il lato delle piste: va collegato alle rispettive piazzole mediante due sottili fili



Se volete montare il circuito al posto di un campanello a 220 V, staccate dal pulsante a muro esistente i fili e tirate due nuovi conduttori, che collegherete a i punti EXT BUT1 o EXT BUT2.

Il gatto suona sempre due volte

Stufi del gatto che gratta gli stipiti o del cane che abbaia per rientrare a casa? Insegnate al vostro compagno a quattro zampe a suonare alla porta! Nulla è più facile, da quando dall'America è arrivato un rimedio per salvaguardare nervi e stipiti delle porte, alla portata di tutte le tasche: un vero e proprio campanello da zampa. Realizzato in plastica e collegato via radio ad un ricevitore dotato di avvisatore acustico, si può fissare ad un'altezza raggiungibile dall'animale su una mensola o sul muro, oppure per terra. Ogni volta che il vostro animale domestico preme il pulsante, il ricevitore suona emettendo il classico suono del campanello elettronico o riproducendo il verso di un cane che abbaia. Le pile sono garantite per diecimila operazioni, più che sufficienti anche



per gli animali più irrequieti. Naturalmente il prodotto è corredato dalle istruzioni per insegnare al cane o al gatto a premere il pulsante con la zampa o con il naso. Se volete saperne di più visitate la pagina web <http://www.comforthouse.com/doorbells.html>.

di rame, possibilmente rispettando la polarità indicata; per fissarlo, potete stagnare alla pista che fa da bordo al foro grande due spezzoni di filo in rame rigido, che terrete ben tesi affinché possano tenere premuta sulla basetta la flangia dell'altoparlante stesso. Affinché il fissaggio risulti efficace, suggeriamo di tendere i fili uno da una parte e l'altro dalla parte opposta della flangia, come mostrato nelle fotografie del prototipo da noi realizzato. La stessa tecnica permetterà di fissare alla basetta una clip per la pila da 9 volt, che è previsto prenda posto sopra le piazzole ramate rettangolari che stanno sul lato corto della basetta opposto a quello occupato dai componenti.

Fatto ciò, non vi resta che chiudere il circuito in un contenitore plastico provvisto di feritoie per lasciar uscire il suono dell'altoparlante e magari di uno sportello per cambiare la pila; se pensate di far funzionare il campanello con un alimentatore da rete e avete collegato allo stampato la

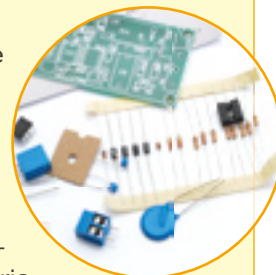
presa plug, fissate quest'ultima a una parete della scatola, opportunamente forata per consentire di introdurre lo spinotto. Per l'installazione in un locale dove c'era un campanello funzionante a 220 volt, bisogna collegare (del filo da 2x0,5 mmq basta e avanza...) indifferentemente l'ingresso EXT BUT1 o l'EXT BUT2 al pulsante vicino alla porta (se già esistente) dopo aver scollegato questo da ogni

eventuale filo di collegamento; l'isolamento dal vecchio circuito è necessario ad evitare che il nostro campanello venga sottoposto all'alta tensione. Per il livello sonoro che può dare, potete sistemare il circuito dove volete: sopra la porta o all'altezza degli interruttori; potete inserirlo anche in una cassetta di derivazione, nel cui coperchio dovrete realizzare le feritoie occorrenti a lasciare uscire il suono. ■



per il MATERIALE

Il progetto descritto in queste pagine è disponibile in scatola di montaggio. Il kit del ding-dong (cod. MK177, Euro 8,70) è prodotto dalla società Velleman e distribuito nel nostro paese da Futura Elettronica. La scatola di montaggio comprende tutti i componenti, la basetta forata e serigrafata, le minuterie ed il pannello frontale opportunamente forato; non sono compresi il pulsante esterno e la batteria di alimentazione.



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

DATA-LOGGER con VINCULUM VDIP1

Torniamo ad occuparci dei prodotti Vinculum della FTDI, realizzando un dispositivo in grado di memorizzare in una Pen-Drive USB i dati ricevuti tramite una porta seriale.



..... di MATTEO BOSIO

Come quasi sempre accade nella realizzazione di un progetto, ed in particolare per le applicazioni elettroniche, il test definitivo per la verifica del corretto funzionamento deve essere effettuato sul campo di lavoro. In questa situazione il progettista sovente si trova a non avere a disposizione tutti gli strumenti di cui dispone solitamente sul banco del laboratorio, quali oscilloscopio, analizzatore di stati logici, PC con apparecchiature di debugging ecc. Ecco che risulta molto utile avere a disposizione uno strumento portatile che possa essere lascia-

to collegato al sistema sotto esame durante una simulazione del funzionamento e sia in grado di tenere traccia degli eventi che si verificano, in modo da poter risalire alle cause di eventuali anomalie verificatesi e risolvere i problemi rilevati. Pensate ad esempio allo sviluppo di un regolatore di carica per pannelli solari, alla realizzazione di un localizzatore portatile o di un circuito che monitorizza dati ambientali; in tutti questi casi risulta conveniente collegarsi direttamente ad un computer al quale affidare l'analisi e registrazione periodica dei dati

Vinculum VDIP1

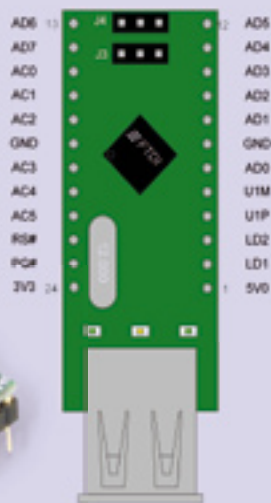
Il VDIP1 è un dispositivo costituito dall'integrato VNC1L della FTDI montato su un piccolo circuito stampato, dotato di connettore USB tipo A per periferiche Slave, che si presenta su uno zoccolo DIP a 24 pin a passo doppio; sui pin arrivano tutti i segnali del VNC1L necessari alla comunicazione con l'esterno, nonché l'alimentazione.

Il modulo è la soluzione ideale per la prototipazione e lo sviluppo di sistemi in cui si voglia disporre di un'interfaccia per la gestione di memorie USB VNC1L, ma può essere anche inserito in progetti finiti di basso e medio volume. Il nostro dispositivo presenta le seguenti caratteristiche tecniche:

- una porta USB principale provvista di connettore USB tipo A;
- seconda porta USB disponibile sui pin del modulo;
- selezione del tipo di interfaccia (UART, SPI o FIFO parallela);
- alimentazione a 5 V;
- regolatore di tensione interno con uscita 3,3 V ausiliaria che eroga fino a 200 mA;
- led indicatore di accensione e led indicatori di traffico dati sull'USB.

Sul lato componenti del modulo VDIP1 sono presenti due jumper a tre vie, denominati J3 e J4: sono

Il modulo Vinculum VDIP1



Pinout del VDIP1

quelli che permettono, a seconda della combinazione dei loro stati, di selezionare la modalità di comunicazione tra il modulo VDIP1 e il microprocessore con cui esso si interfaccia.

La **Tabella 1** riporta le possibili combinazioni.

In base alla modalità di comunicazione selezionata, il firmware presente all'interno del VDIP1 riconfigura i vari piedini. Nella nostra applicazione la comunicazione è FIFO parallela.

Nella Tabella 2 sono riportate le descrizioni dei vari pin per le modalità UART, Parallela, SPI.

Tabella 1 - Selezione interfaccia del VDIP1

J3	J4	Modalità I/O
1-2	1-2	Seriale UART
1-2	2-3	SPI
2-3	1-2	FIFO parallela
2-3	2-3	Seriale UART

Tabella 2 - Significato dei piedini del VDIP1.

PIN	PIN Name	UART	FIFO Parallela	SPI
6	AD0	TXD (output)	D0 (I/O)	SCLK (input)
8	AD1	RXD (input)	D1 (I/O)	SDI (input)
9	AD2	RTS (input)	D2 (I/O)	SDO (input)
10	AD3	CTS (input)	D3 (I/O)	CD (input)
11	AD4	DTR (output)	D4 (I/O)	-
12	AD5	DSR (input)	D5 (I/O)	-
13	AD6	DCD (input)	D6 (I/O)	-
14	AD7	RI (input)	D7 (I/O)	-
15	AC0	TXDEN (input)	RXD (output)	-
16	AC1	-	TXE (output)	-
17	AC2	-	WR (input)	-
19	AC3	-	RD (input)	-
20	AC4	-	-	-

di funzionamento, tuttavia non sempre è possibile farlo. Infatti usare un PC fisso posto in un locale comporta il dover realizzare la connessione cablata o wireless, mentre potendo installare un notebook sul posto la cosa risulta comunque poco pratica, sia perché è impossibile fare lunghi monitoraggi (dopo qualche ora le batterie del portatile si scaricano) sia per il fatto che lasciare un PC all'esterno può essere deleterio non solo per l'eventualità del verificarsi di precipitazioni atmosferiche ma anche perché un notebook è sempre un "boccone ghiotto" per qualche malintenzionato e difficilmente il giorno dopo lo si ritroverebbe al suo posto. Meglio, dunque, applicare all'apparecchiatura una più anonima e insignificante "scatolina nera" con un connettore seriale e una Flash-Pen USB.

Questo articolo propone proprio di realizzare uno strumento di questo tipo, in grado di ricevere i messaggi attraverso una seriale standard RS232 (o alternativamente con livelli TTL) e salvarli su una Flash-Pen USB, in modo che possano poi essere analizzati con calma in laboratorio.

Prima di vedere di che si tratta analizziamo le scelte fatte in fase di progetto: si è scelto innanzitutto di utilizzare una seriale (RS232 o TTL) come interfaccia tra l'apparecchiatura sotto test e il nostro strumento, in quanto è una porta disponibile in quasi tutti i tipi di microcontrollore, anche i più semplici, che può essere utilizzata come strumento di debug senza invadere troppo il reale funzionamento dell'apparecchiatura; infatti si potrebbe mandare una stringa seriale in modo periodico, che indica lo stato del sistema (se è strutturato secondo una macchina a stati, per esempio), il valore di alcu-

ne variabili determinanti, ecc., oppure si potrebbe mandare una stringa solo quando si verificano determinati eventi che vanno a modificare lo stato del sistema. Abbiamo inoltre deciso di utilizzare una Flash-Pen USB in quanto oggi si trovano dispositivi del genere con elevate capacità di memoria a costi molto ridotti: ad esempio una memoria da 4 GB si compera ormai a meno di 10 euro. Grandi capacità di memoria consentono di effettuare acquisizioni lunghissime, anche di giornate intere, fermo restando che bisogna fare i conti con il consumo di corrente e l'autonomia delle batterie, questione di cui parleremo in seguito. Altro vantaggio delle Pen-Drive è che sono facilmente leggibili dai PC senza dover utilizzare particolari interfacce: basta introdurle nelle prese USB ed eventualmente caricare i driver del caso (se si usa un sistema operativo più vecchio di Windows XP). Per il nostro data-logger abbiamo scelto di utilizzare il dispositivo Vinculum VDIP1 perché i componenti della famiglia Vinculum FTDI, basati sul chip VNC1L, permettono di implementare la funzionalità USB di tipo Host in un semplice microcontrollore dotato di limitate risorse hardware. FTDI ha proposto questa soluzione innovativa, con la quale si è aperta una porta a un segmento di mercato fino ad allora impraticabile, ossia quello delle porte USB Host su apparati diversi dai PC. Il VNC1L permette infatti di gestire due porte di tipo Host (che all'occorrenza possono essere configurate anche come "slave") con le quali interfacciarsi ad alcune delle più utilizzate periferiche USB, come le Pen-Drive; inoltre gestisce in modo totalmente trasparente la struttura dei file FAT, comunicando attraverso



Pen-Drive senza limiti

Nate per sostituire i poco capienti floppy-disk e i CD-RW e DVD-RW, delicati, ingombranti e lenti da scrivere, le Pen-Drive o USB-Pen stanno diventando un'alternativa agli hard-disk esterni; lo dimostra la disponibilità di supporti anche da 64 GB, come ad esempio la Flash Voyager della Corsair: un'unità la cui capacità è sufficiente a contenere una libreria di film in DVD e decine di migliaia di immagini digitali ad alta risoluzione. Oltre a una capacità di storage che solamente un anno fa era disponibile esclusivamente nei dischi rigidi, l'unità USB Flash Voyager da 64 GB garantisce robustezza, e resistenza all'acqua. Oltre ad avere la protezione dalla lettura non autorizzata, le Pen-Drive Flash Voyager da 64 GB sono avviabili, nel senso che possono essere utilizzate per avviare il sistema operativo da esse invece che dal disco rigido o dal CD, cosa utile per risolvere i problemi di sistema. Come tutte le novità, il prezzo è ancora relativamente elevato (in U.S.A. oltre 200 \$) ma scenderà a breve.

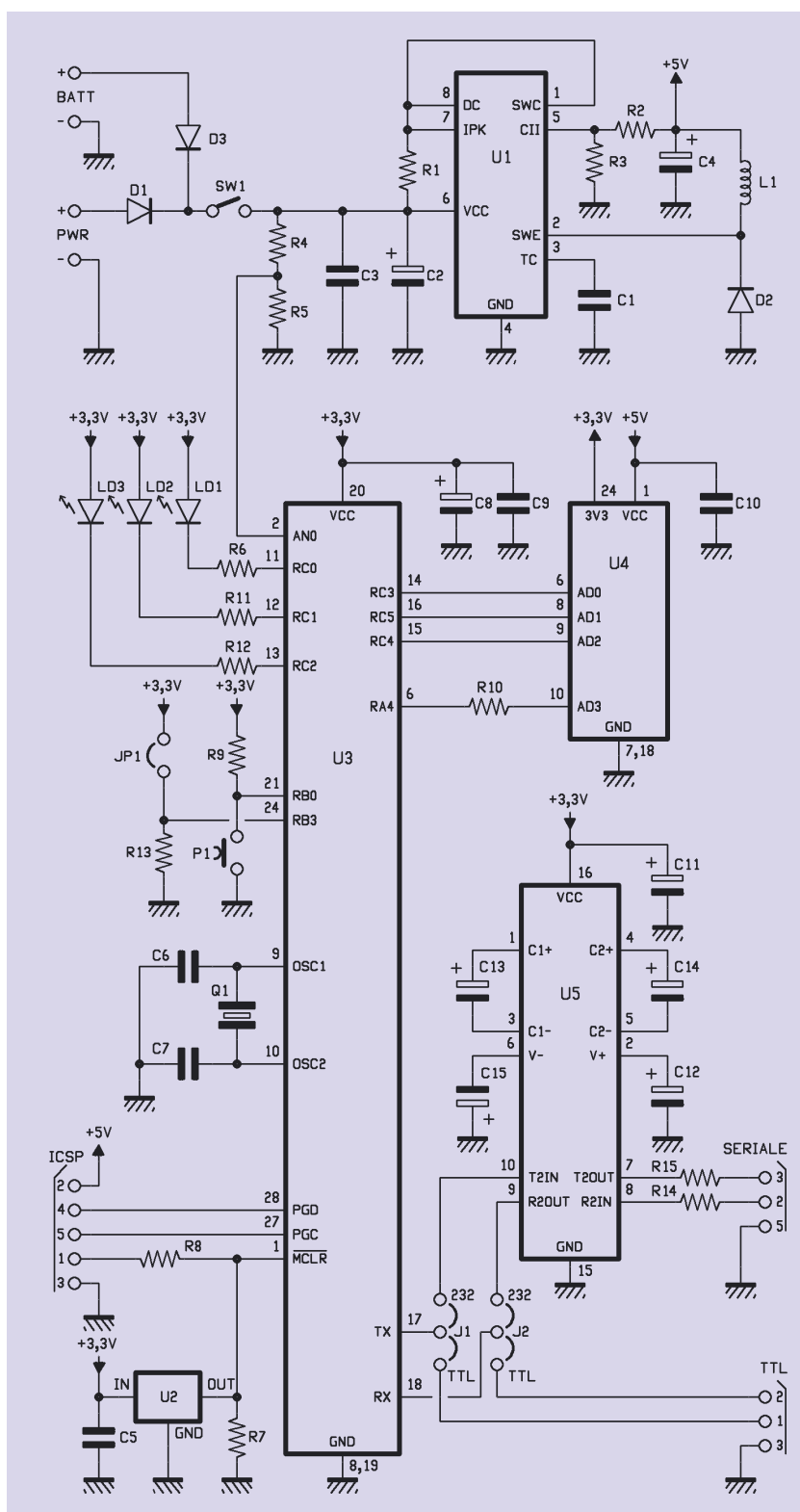
so UART, SPI o FIFO parallela, senza richiedere allo sviluppatore grandi nozioni software, in quanto tutto ciò che è richiesto è far comunicare il dispositivo con un semplice microcontrollore attraverso una delle interfacce appena citate. Quindi con grande semplicità è possibile aggiungere tutte le potenzialità della porta USB a sistemi con limitate risorse hardware, ma che dispongono almeno di una seriale.

La FTDI ha poi sviluppato una serie di moduli partendo dal chip VNC1L, programmati con firmware diversi a seconda dei casi, come il VDIP1, che presenta su zoccolo DIP 24 pin tutti i segnali del VNC1L, ma anche il VDRIVE2, pensato per essere aggiunto a prodotti finiti; non meno importante è il VF2F, un prodotto stand-alone progettato per far comunicare direttamente

due memorie USB senza passare attraverso il PC. Non possiamo poi dimenticare il VMUSIC2, da noi utilizzato nel fascicolo n° 123 per riprodurre file audio contenuti in una Pen-Drive USB e trasmettere l'audio in FM. Il nostro sistema verrà realizzato utilizzando il primo dei dispositivi citati, ossia il VDIP1, che abbiamo già ampiamente descritto nel fascicolo n. 119.

SCHEMA ELETTRICO

Prima di vedere l'utilizzo pratico del debugger, diamo uno sguardo allo schema elettrico che lo descrive, partendo dallo stadio di alimentazione: la scheda è stata predisposta per poter essere alimentata da una batteria a 9 V o da un alimentatore esterno che dia una tensione di $9 \div 15$ volt; allo scopo, i diodi D1 e D3 servono a garantire che sia la tensione



maggiore, tra quella fornita dalla batteria (BATT) e quella dell'alimentatore (PWR), ad alimentare il circuito. Il discorso è molto semplice: la tensione maggiore si presenta sui catodi, cosicché il

diodo polarizzato con il potenziale più basso resta interdetto e lascia che a dare l'alimentazione sia il dispositivo a tensione più alta; in tal caso il diodo interdetto impedisce che il dispositivo

che alimenta il circuito vada a scaricare corrente in quello a tensione minore. La tensione a valle dell'interruttore SW1 viene filtrata dai condensatori C2 e C3 e poi passa allo stadio regolatore di tensione, che è stato realizzato utilizzando l'integrato MC34063A (siglato U1) della Motorola; questo è un DC/DC converter di tipo switching che accetta in ingresso tensioni da 3 a 40 V (sebbene nel nostro caso il circuito sia predisposto per funzionare con all'ingresso 7,8÷15 V) anche non stabilizzate, in grado di erogare al massimo 1,5 A con una frequenza di lavoro di 100 kHz. Può essere configurato come step-up, step-down o voltage-inverting; nel nostro caso viene utilizzato come step-down, ossia abbassatore di tensione, per fornire in uscita 5 V ben stabilizzati. Il valore della tensione di uscita si ottiene dalla semplice formula:

$$V_{out} = 1.25 \cdot \left(1 + \frac{R_2}{R_3} \right) = 1.25 \cdot \left(1 + \frac{10K}{3.3K} \right) \approx 5V$$

Con i 5 volt alimentiamo direttamente il modulo Vinculum VDIP1, di cui sfruttiamo il regolatore interno a 3,3 V il quale, oltre ad alimentare gli stadi logici rende disponibili 3,3 volt continui sul piedino 24, che usiamo per far funzionare il resto del circuito, ossia il microcontrollore, il gestore di reset e il convertitore RS232. Dal regolatore interno al VDIP1 possiamo prelevare un massimo di 200 mA, quindi siamo più che tranquilli.

Adesso veniamo al cuore del sistema, ossia il microcontrollore: per necessità di numero di ingressi uscite e di memoria RAM disponibile, è stato scelto il PIC18F2620, che dispone di 64 kB di Flash a bordo, 4 kB di SRAM, 1 kB di EEPROM, interfaccia SPI, interfaccia UART e convertitori

AD a 10 bit integrati nel chip. Del micro utilizziamo l'ingresso analogico AN0 (pin 2) per monitorare la tensione della batteria di alimentazione attraverso la lettura del potenziale prelevato dal partitore resistivo formato da R4 ed R5; in realtà leggiamo genericamente la tensione a valle dell'interruttore SW1, che può essere sia quella della batteria, sia quella dell'alimentatore eventualmente collegato ai punti + e - PWR. La tensione riportata all'AN0 si ricava dalla formula:

$$(V_{an} = V_{in} \cdot \frac{R_5}{R_4 + R_5} = V_{in} \cdot 0,183)$$

dove V_{in} è la differenza di potenziale presente ai capi del condensatore C3, ovvero a valle dell'interruttore SW1. Con gli attuali valori di R4 ed R5 si ottengono 1,65V quando la batteria è carica a 9 V e 1,43 V quando la batteria è da considerarsi scarica, ossia fornisce una tensione attorno ai 7,8 V. Quest'ultimo valore è quello minimo ammissibile perché il circuito funzioni correttamente; al disotto di esso, il regolatore non riesce a dare più di 5 volt occorrenti al modulo VDIP1 per funzionare e per ricavare i 3,3 V da dare al microcontrollore. Nel circuito abbiamo previsto tre led di segnalazione, collegati

direttamente al microcontrollore. Osservate che solitamente, per non caricare il microcontrollore il comando dei led passa attraverso dei transistor; in questo caso, essendoci solo tre led e avendo il microcontrollore una massima corrente di "sink" di 25 mA, non ci sono problemi a pilotare i led direttamente con le uscite del PIC. R6, R11 e R14 vengono dimensionate per far passare nei led un massimo di 5 mA; supponendo una caduta di tensione di circa 2 V sui led, le resistenze di limitazione valgono:

$$R = \frac{3,3V - V_{LED}}{5mA} = \frac{1,3V}{5mA} = 260\Omega$$

Il valore più vicino della serie E12 è 270 Ω ed è poi quello che adottiamo.

La funzione dei led è la seguente: quello giallo è acceso se la tensione di alimentazione è maggiore di 7,8 V, quindi è sufficiente a garantire i 5 volt sul VDIP1) e, invece, spento se al di sotto; se all'accensione la tensione scende sotto i 7,8 V il dispositivo non si avvia (il micro non fa girare il programma) e il diodo resta spento. Il led rosso indica attività sulla seriale ed è acceso ogni volta che un byte viene ricevuto e memorizzato nella RAM del mi-

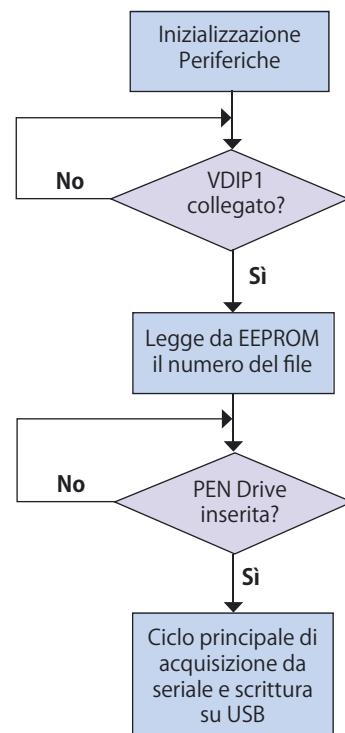
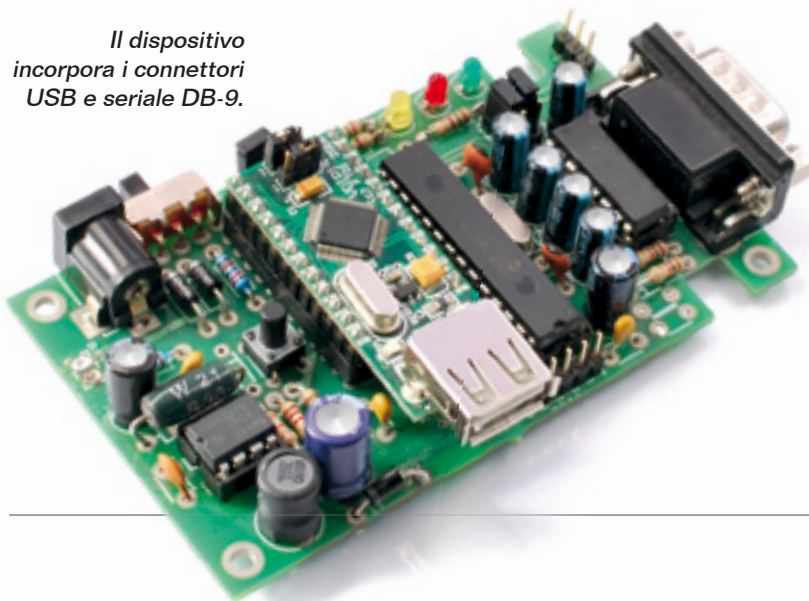


Fig. 1 - Il flow-chart della struttura del firmware.

Il dispositivo incorpora i connettori USB e seriale DB-9.



cro. Infine, il led verde indica che la Pen-Drive USB è collegata; per l'esattezza è acceso fisso se è inserita in maniera corretta, ovvero lampeggiante se la penna viene tolta (torna acceso a luce fissa reinserendola...) o inserita male. Nello schema troviamo, poi, il jumper JP1, con il quale si seleziona la modalità di funzionamento, sulla quale torneremo tra breve; il ponticello è letto dalla linea RB3 del micro e solo all'accensione. C'è poi il pulsante P1, il quale, una volta premuto, azzerà il contatore dei nomi dei file; il prossimo file creato sarà *F_00000.txt*. Premendo tale tasto, il led verde si spegne per un istante e poi si riaccende. Il connettore PROG a 5 poli serve per la connessione al programmatore/debugger della Microchip ICD2 e consente la programmazione del micro in-circuit e l'eventuale debug; all'ICSP giungono i pin 1 (MCLR) 2 (+5 V) 3 (GND) 4 (PGD) e 5 (PGC).

Il reset del microcontrollore vie-

ne realizzato mediante un gestore di reset (U2) di tipo LM809M3-2.93, che fornisce un impulso di reset a livello basso della durata di 300 ms quando la tensione di alimentazione sua e del micro scende sotto i 2,93 V. In questo modo ottimizziamo il reset del microcontrollore all'accensione e siamo certi che il PIC si avvia correttamente e si blocca quando la tensione che riceve non è più sufficiente a garantire la corretta esecuzione del programma. Detto ciò, torniamo al jumper JP1, per affrontare il discorso sulle due modalità di funzionamento: all'accensione, se il micro lo trova chiuso avvia la modalità 1, mentre se lo rileva aperto si porta nella modalità 2. Nella modalità 2 il data-logger effettua la registrazione ogni 100 caratteri ricevuti e dopo l'arrivo del centesimo presenta un tempo morto che tipicamente è di 500 ms. Ciò è dovuto al fatto che il circuito bufferizza i dati in arrivo e poi quando ha raggiunto il numero di caratteri (byte) previsto li va a memorizzare in un file nella Pen Drive; la memorizzazione dell'intero blocco richiede evidentemente un certo tempo, durante il

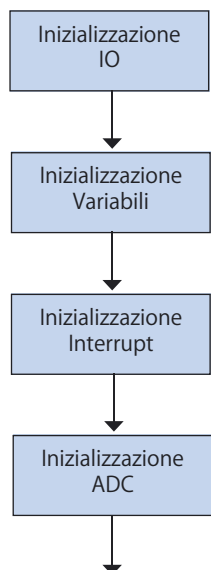


Fig. 2
Flow-chart
dell'inizializzazione
delle periferiche.

quale il microcontrollore risulta impegnato e non può ricevere nuovi caratteri. Può riprendere a farlo quando ha terminato la memorizzazione nella Pen Drive. Per ovviare a tale limitazione, abbiamo previsto la modalità 1, nella quale il circuito memorizza su file anche prima del raggiungimento dei 100 caratteri, basta che riceva sul canale seriale un carattere di INVIO (0x0D): più precisamente, l'unità che trasmette i dati da salvare può forzare in ogni momento la memorizzazione inviando un carattere di INVIO e ciò può farlo fino alla trasmissione di 99 caratteri consecutivi (anche se l'invio non avviene tutto nello stesso periodo). Quando la memorizzazione avviene dietro comando esterno (INVIO) il data-logger presenta ugualmente un

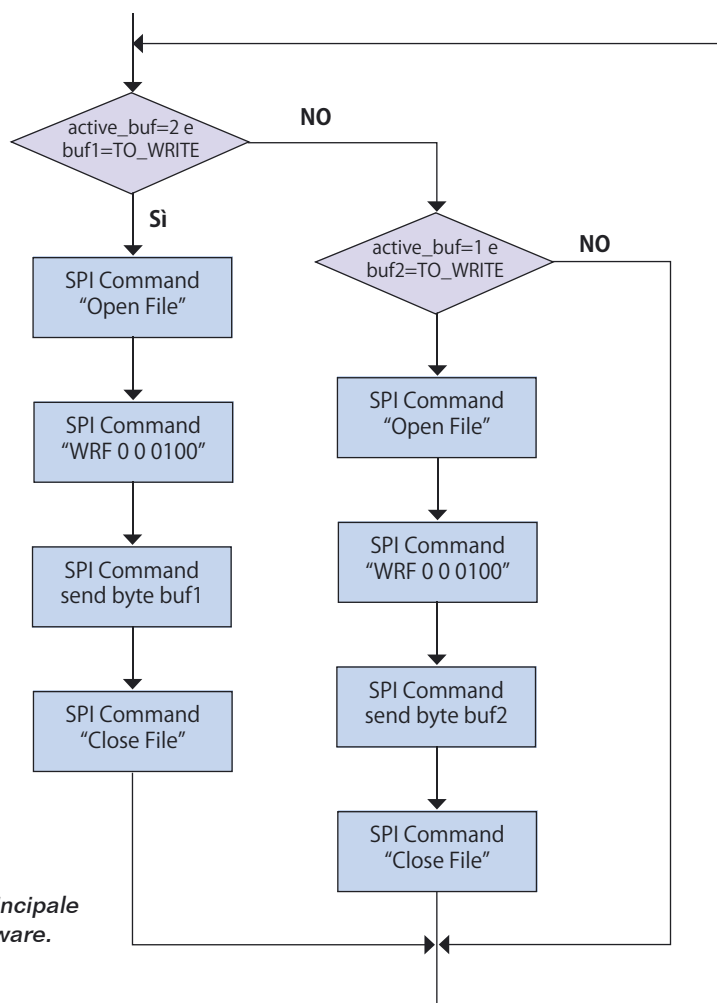


Fig. 3
Ciclo principale
del firmware.

tempo morto dovuto al fatto di essere impegnato nel salvataggio su file dei dati ricevuti prima del carattere di INVIO, ma tale tempo dipende dal numero di caratteri ricevuti dall'ultimo salvataggio e non è quindi fisso come nella modalità 2 o nel salvataggio automatico.

Nel caso non venga richiesta la memorizzazione forzata, una volta che il buffer avrà raggiunto un contenuto di 100 caratteri, il data-logger provvederà al salvataggio automatico come previsto nella modalità 2; qui, prima di inviare al data-logger nuovi caratteri occorrerà attendere i soliti 500 ms. Quanto alla connessione seriale, viene implementata dall'UART interno del PIC che può essere interfacciato con l'esterno in due modi: tramite l'integrato U5 (MAX202ECSE) che fa da conver-

ter TTL/RS232 oppure direttamente; la selezione si effettua mediante i jumper J1 e J2, che permettono di deviare le linee Rx e Tx seriali del microcontrollore verso U5 (che provvede alla conversione da TTL a livello RS232 e viceversa) o verso un connettore a 3 poli dal quale è possibile prelevare i segnali direttamente a livello TTL. Nel caso della linea RS232, i segnali vengono portati all'apposito connettore a vaschetta 9 poli maschio. Nel circuito è stato previsto anche un canale

USB per la comunicazione con il dispositivo VDIP1, realizzato attraverso le linee SPI del micro che sono RC3 (SPI Clock) usata come uscita, RC4 (SPI MISO, ossia Master Input Slave Output) usata da ingresso, RC5 (SPI MOSI, cioè Master Output Slave Input) inizializzata anche questa come uscita ed RA5 (SPI CS, ovvero Chip Select) che è ancora un'altra uscita.

IL FIRMWARE

Bene, prima di passare alle note

costruttive vediamo ora, nel dettaglio, la logica con cui è stato realizzato il firmware della scheda, che abbiamo sviluppato in linguaggio C18 utilizzando l'MPLAB IDE.

La prima operazione effettuata dal programma è l'inizializzazione delle periferiche, che descriviamo nel dettaglio in un secondo flow-chart (Fig. 2); in questo notate che dapprima avviene l'inizializzazione delle porte del microcontrollore usate come I/O, l'inizializzazione del bus SPI (attraverso il quale si comunica con il VDIP1) e dell'UART; poi vengono inizializzate le variabili di sistema e si avviano gli interrupt ed il convertitore A/D. Il sistema utilizza due interrupt: quello della seriale UART, utilizzato per memorizzare tutti i byte ricevuti; quello del Timer0, che interviene ogni 50 ms e che ha una priorità inferiore al precedente e viene utilizzato per gestire le varie temporizzazioni e i lampeggi del led. Tornando al flow-chart generale (Fig. 1) dove si parla di VDIP1 collegato s'intende chiarire che all'inizio il micro resta in attesa che il VDIP1 mandi, attraverso l'interfaccia SPI, una serie di stringhe; ciò indica che il modulo è connesso e che funziona regolarmente. Tali stringhe sono, in ordine, quella relativa alla versione (es. "Ver 02 08VDAPF On Line:"), la stringa "Device Detected", la stringa "No Upgrade" ed infine la stringa "D: \". Nel caso sia inserita una Flash-Pen USB, l'apposito led resta acceso fisso se essa resta inserita, mentre comincia a lampeggiare se viene tolta. Altra operazione iniziale è la lettura nella EEPROM interna del numero da assegnare al file in cui verranno registrati i dati acquisiti. Comprendete lo scopo di tale numero sapendo che il

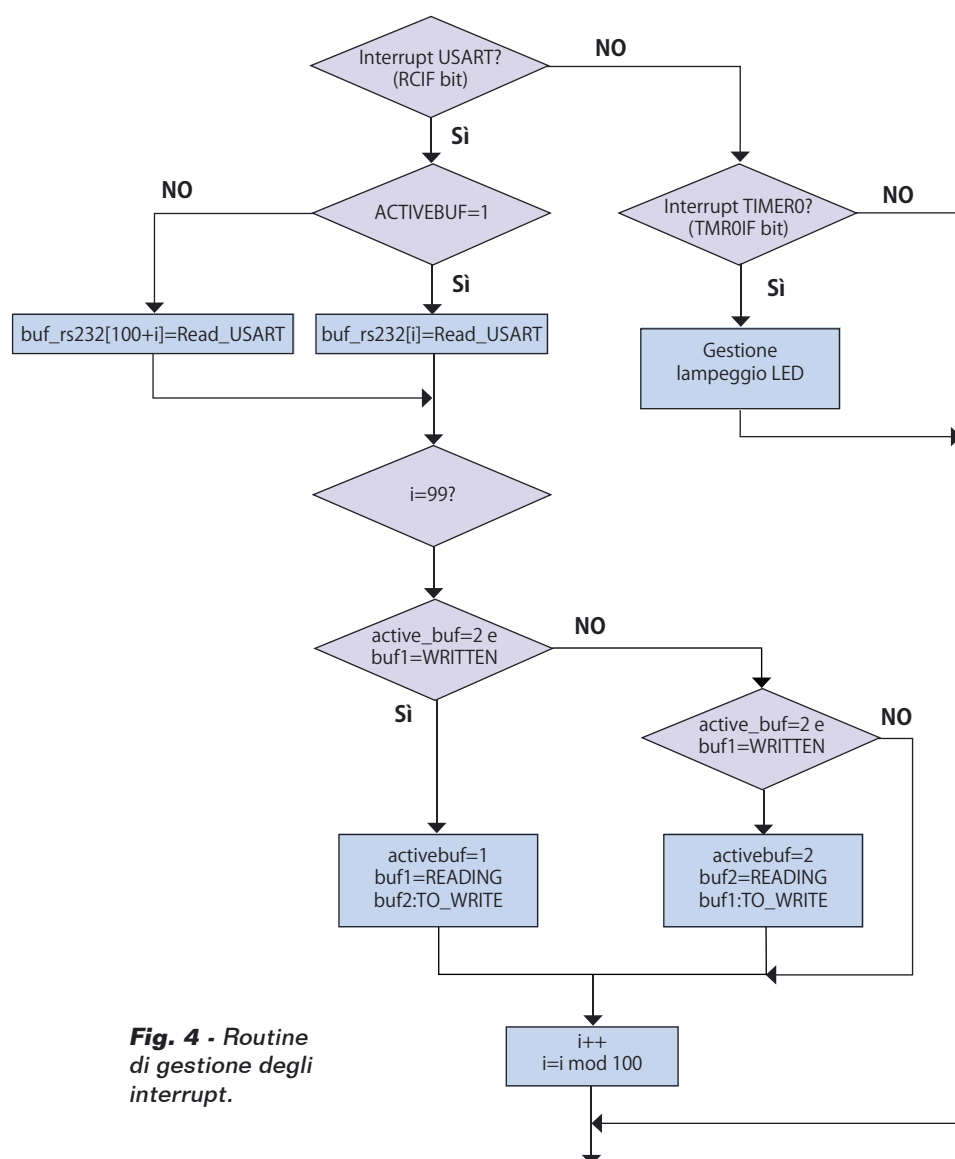


Fig. 4 - Routine di gestione degli interrupt.

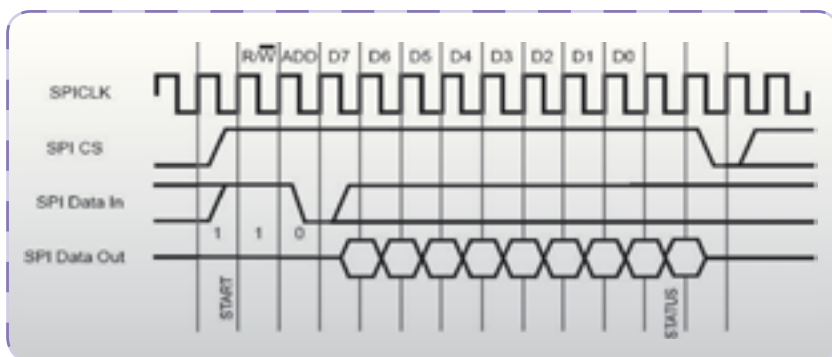


Fig. 5 Ciclo di lettura SPI.

data-logger memorizza i dati in un file che comprende le registrazioni avvenute dall'accensione allo spegnimento e che ad ogni nuova accensione crea un nuovo file per evitare di sovrascrivere i dati già esistenti. Per nominare i file in modo diverso, il sistema adotta una nomenclatura comprendente un numero che il firmware aggiorna ogni volta che il circuito viene alimentato. Il nome del file è del tipo F_XXXXX.txt. Dopo l'azzeramento mediante P1 si crea il file F_00000.txt; alla successiva accensione del data-logger la EEPROM conterrà il valore 00000 e il file che verrà creato sarà F_00001.txt. Quindi nella EEPROM sarà scritto il valore 00001 e alla successiva accensione il file creato sarà F_00002.txt e così via, fino a F_65535.txt. Giunti a 65.535, il numero riparte da 00000.

Vediamo ora il cuore del firmware, ossia il ciclo principale (Fig. 3) e la routine di gestione degli interrupt (Fig. 4). Come si può notare dal pezzo di diagramma visibile nella Fig. 3, quando si riceve un byte su seriale il sistema può essere in due stati, discriminati dalla variabile *activebuf*, che può valere 1 o 2; quando vale 1 i byte vengono salvati nella prima metà di *buf_rs232* (che è un buffer di 200 byte) mentre quando è 2 vengono memorizzati nella seconda metà di *buf_rs232*. Una volta acquisito il centesimo

byte, si va a gestire le variabili *buf1* e *buf2*, che possono valere WRITTEN (il buffer è stato scritto su USB), TO_WRITE (i 100 byte del buffer sono stati salvati e sono pronti per essere scritti) o READING (si sta acquisendo da seriale). Nel ciclo principale, quando *active_buf* vale 1 (si sta salvando nei primi 100 byte di *buf_rs232*) e *buf1* vale TO_WRITE (è pronto per essere scritto), si apre il file F_XXXXX.txt e si scrivono in esso i 100 byte; fatto ciò, si chiude il file e si pone la variabile *buf1* a WRITTEN, ad indicare che la scrittura è stata effettuata. Notate che abbiamo scelto di aprire e chiudere continuamente il file per diminuire la probabilità di spegnere il sistema a file aperto; la procedura prevede di spegnere il sistema una

volta che il flusso di dati lungo la seriale sia finito.

Lo stesso discorso vale nel caso in cui i byte acquisiti da salvare siano nella seconda metà di *buf_rs232*. Aprire il file ogni volta, salvare 100 byte e poi richiuderlo, obbliga a guardare le tempistiche in modo accurato. Facendo funzionare il bus SPI a 50 kHz e considerando la situazione peggiore, in cui dalla seriale si continua ad acquisire byte e quindi la routine di interrupt viene continuamente chiamata, per tali operazioni sono richiesti circa 250 ms. Ciò significa che al massimo si possono acquisire e salvare 400 byte al secondo, il che impone una velocità massima della comunicazione seriale di 3.200 bps; quindi usiamo la velocità standard inferiore più vicina, ossia 2.400 bps.

Ora diamo un'occhiata ai comandi inviati attraverso il bus SPI al VDIP1, che sono:

```
OPW nomefile
WRF 100
Byte 1
Byte 2
...
Byte 100
```

Listato 1

```

/*****
 * FUNCTION: SPI_Read
 * DESCRIPTION: non-blocking read of character on SPI bus
 * PARAMETERS: void
 * RETURN: TRANSFER_OK if data is received, TRANSFER_RETRY if not
 *****/
int SPI_Read(char *pspiData)
{
    return(SPI_Transfer(DIR_SPIREAD, pspiData));
}

/*****
 * FUNCTION: SPI_Write
 * DESCRIPTION: blocking write of character on SPI bus
 * PARAMETERS: pspiData - byte to be transmitted
 * RETURN: void
 *****/
void SPI_Write(char pspiData)
{
    while(SPI_Transfer(DIR_SPIWRITE, &pspiData));
}

```

Listato 2

```

/*****
* FUNCTION: SPI_Transfer
* DESCRIPTION: read or write transfer on SPI bus
* PARAMETERS: spi direction - DIR_SPIWRITE/DIR_SPIREAD
* spiData - pointer to data byte to write or read to
* RETURN: retData - TRANSFER_OK for succes, TRANSFER_RETRY to retry
*****/
int SPI_Transfer(int spiDirection, char *spiData)
{
    uchar retData;
    uchar bitData;

    //CS goes high to enable SPI communication
    SPI_CS=1;

    Delay100TCYx(2);

    //Clock 1 - Start state
    SPI_MOSI=1;
    SPI_delay();
    SPI_CLK=1;
    SPI_delay();
    SPI_CLK=0;

    //Clock 2 - Direction
    SPI_MOSI=spiDirection;
    SPI_delay();
    SPI_CLK=1;
    SPI_delay();
    SPI_CLK=0;

    //Clock 3 - Address
    SPI_MOSI=0;
    SPI_delay();
    SPI_CLK=1;
    SPI_delay();
    SPI_CLK=0;

    //Clock 4..11 - Data Phase
    bitData=0x80;

    switch(spiDirection)
    {
        case DIR_SPIREAD:
            //read operation
            retData=0;

            while(bitData)
            {
                SPI_delay();
                retData|=SPI_MISO?bitData:0;
                SPI_CLK=1;
                SPI_delay();
                SPI_CLK=0;
                bitData=bitData>>1;
            }
            *spiData=retData;
            break;

        case DIR_SPIWRITE:
            //write operation
            retData=*spiData;
            while(bitData)
            {
                SPI_MOSI=(retData & bitData)?1:0;
                SPI_delay();
                SPI_CLK=1;
                SPI_delay();
                SPI_CLK=0;
                bitData=bitData >>1;
            }
            break;

        default:
            break;
    }

    //Clock 12 - Status bit
    SPI_delay();
    bitData=SPI_MISO;
    SPI_CLK=1;
    SPI_delay();
    SPI_CLK=0;

    //CS goes low to disable SPI communication
    SPI_CS=0;

    Delay100TCYx(2);

    //Clock 13 - CS low
    SPI_delay();
    SPI_CLK=1;
    SPI_delay();
    SPI_CLK=0;

    return bitData;
}

```

Attesa risposta D:\
CLF nomefile

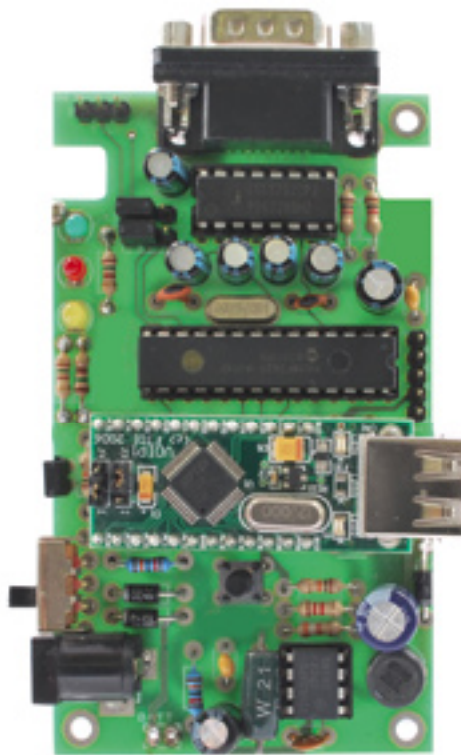
Per aprire un file nella Pen-Drive USB si invia il comando OPW nomefile, poi si manda il comando di scrittura di 100 byte WRF 100; in seguito si inviano i 100 byte da salvare, seguiti da 0x0D. Quando il VDIP1 finisce di scrivere, rispondendo con un "D:\:" alla richiesta di prompt, si chiude il file e si procede. Il Listato 1 illustra le funzioni a basso livello di lettura e scrittura di byte sul bus SPI, che si basano entrambe sulla funzione SPI_transfer.

Adesso analizziamo nel dettaglio la funzione SPI_transfer (Listato 2) che è il cuore della SPI_read e della SPI_Write. Come si può notare dal Listato 2, innanzitutto viene posto a livello logico alto il Chip-Select del VDIP1, ossia il segnale SPI_CS del microcontrollore, in modo da abilitare la comunicazione sul bus SPI. Viene poi portato a livello alto il segnale SPI_MOSI (SPI Master Out Slave In); ricordiamo che il Master SPI è il microcontrollore, e lo slave SPI è il VDIP1). Ora viene dato un impulso di clock (SPI_CLK). Il segnale SPI_MOSI viene posto ad 1 in caso di lettura

e a 0 in scrittura (indicato come bit R/W), poi si dà un altro impulso di clock (il dato è valido sul fronte di salita del clock). Viene poi mandato un bit 0 (ADD), quindi la funzione si differenzia a seconda che sia una SPI_read o una SPI_Write; in lettura si acquisiscono gli 8 bit del byte da leggere, mentre in scrittura si mandano gli 8 bit del byte da scrivere. Concludiamo l'analisi del firmware dicendo che il livello della batteria viene acquisito ogni secondo attraverso l'ADC interno del microcontrollore, che prende un campione mediante il partitore formato da R4 ed R5.

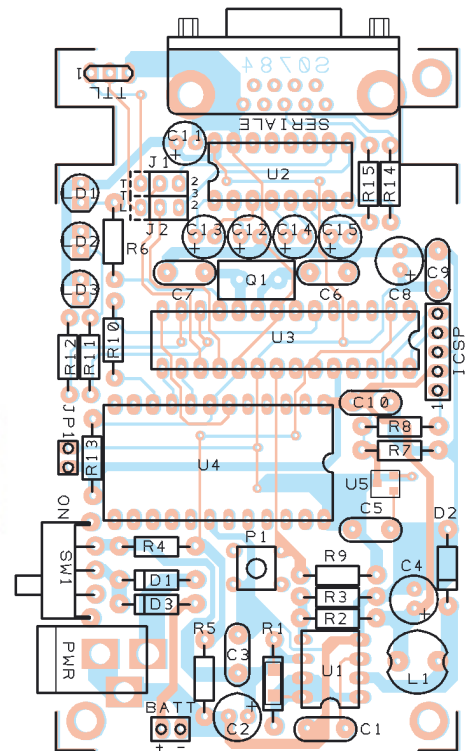
Elenco Componenti:

R1: 33 ohm 1W
 R2: 3,8 kohm 1%
 R3: 220 ohm
 R4: 120 kohm 1%
 R5: 27 kohm 1%
 R6: 150 ohm
 R7: 100 kohm
 R8: 470 ohm
 R9: 10 kohm
 R10: 0 ohm
 R11, R12: 150 ohm
 R13÷R15: 10 kohm
 C1: 470 pF ceramico
 C2: 100 µF 35 VL elettrolitico
 C3, C5: 100 nF multistrato
 C4: 470 µF 16 VL elettrolitico
 C6, C7: 15 pF ceramico
 C8: 10 µF 100 VL poliestere
 C9, C10: 100 nF multistrato
 C11÷C15: 1 µF 100 VL poliestere
 D1: 1N4007
 D2: 1N5819
 D3: 1N4007
 U1: MC33063AP1
 U2: MAX232CPZ
 U3: PIC18F2620-I/SP (MF784)
 U4: VDIP1
 U5: LM809M3-2.93
 LD1: Led 3 mm verde
 LD2: Led 3 mm rosso
 LD3: Led 3 mm giallo
 Q1: Quarzo 20 MHz



SW1: Deviatore a slitta 90°
 P1: Microswitch
 L1: Induttanza 220 µH 500 mA

Varie:
 - Zoccolo 4+4
 - Zoccolo 14+14



- Plug alimentazione
 - Strip maschio 2 poli
 - Strip maschio 3 poli (3 pz.)
 - Strip maschio 5 poli
 - Jumper (3 pz.)
 - Connettore DB9 maschio da CS
 - circuito stampato

REALIZZAZIONE PRATICA

Bene, possiamo ora alle note costruttive. Per realizzare il data-logger è necessario innanzitutto preparare il circuito stampato a

doppia faccia, del quale si possono scaricare le tracce direttamente dal nostro sito internet (www.elettronica.in) sotto forma di file immagine stampabile in scala 1:1.

Bisogna poi procurarsi tutta la componentistica, che è per montaggio tradizionale (PTH) ad eccezione dell'integrato LM809M3-2.93, che è in formato SOT-23 (per montaggio superficiale); tutti gli elementi, compreso il gestore di reset in SMD, devono prendere posto sul lato componenti. Il modulo Vinculum deve essere preferibilmente montato su uno zoccolo dip a 12+12 piedini a passo doppio. Per l'alimentazione sono previsti sia due asole per saldare una clip volante per batteria da 9 V, sia le piazzole per ospitare un connettore plug al quale collegare un alimentatore da rete, in

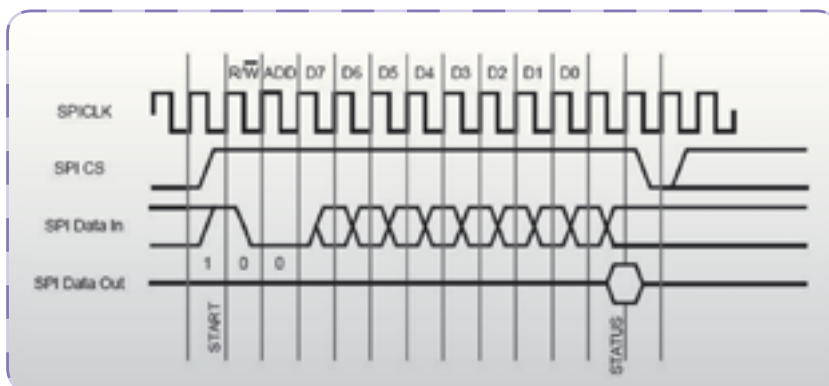
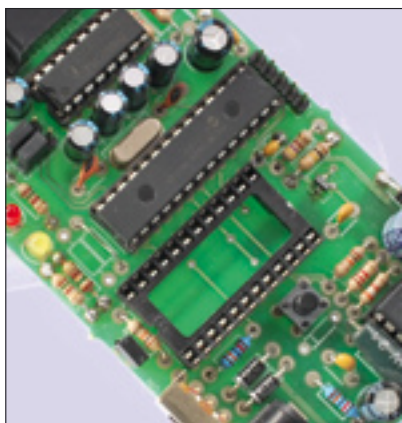


Fig. 6 Ciclo di scrittura SPI

grado di fornire una tensione stabilizzata compresa tra 8 V e 15 V e una corrente continua di circa 100 mA.

Se dovete acquisire dati provenienti da un apparato che ha l'interfaccia seriale TTL-compatibile, potete pure evitare di montare il connettore DB-9 (SERIALE) e limitarvi ad effettuare la connessione usando le tre piazzole siglate TTL; se, invece, il data-logger deve interfacciarsi con dispositivi che comunicano mediante una porta RS232, montate il connettore a vaschetta a 9 poli, che deve essere di tipo maschio. Per la connessione userete perciò un cavo femmina/femmina, os-



sia un null-modem (cavo incrociato DTE/DTE).

UTILIZZO PRATICO

Per usare correttamente questo circuito dovete tenere conto dei suoi limiti in termini di tempistiche di lavoro e di consumo: per un corretto funzionamento dell'acquisizione, l'apparecchiatura sotto test può mandare in continuo attraverso la seriale delle stringhe a 2400 bps, 8, N, 1; dalle prove effettuate su diverse Flash-Pen USB è emerso il fatto che sul mercato ne esistono varie tipologie caratterizzate da velocità di scrittura molto differenti tra loro. In pratica tutte le moderne

TTL o RS232?

Il data-logger dispone di una porta seriale che potete configurare perché si interfacci con dispositivi funzionanti a livelli TTL o a standard RS232 (le COM dei Personal Computer); per utilizzare l'uscita TTL-compatibile i jumper J1 e J2 devono essere chiusi sui punti siglati TTL (verso i led) mentre per collegare la seriale del microcontrollore al connettore DB9 (RS232) gli stessi vanno chiusi sui contatti chiamati 232 (verso l'integrato U2).



Pen-Drive supportano l'USB 2.0, ma poi praticamente alcune in scrittura non reggono il data-rate previsto e quindi lavorano molto più lentamente.

Per non incappare in brutte sorprese sarebbe meglio che il dispositivo che manda i dati da memorizzare (l'apparecchiatura sotto test...) attendesse circa 100 ms tra l'invio di due stringhe consecutive.

All'accensione del dispositivo, il led giallo deve illuminarsi, ad indicare che il livello di batteria è corretto; il led verde deve cominciare a lampeggiare. Inserendo la Flash-Pen USB, dopo circa 10÷15 secondi il datalogger riconosce la chiavetta e il led verde deve restare acceso fisso. Da questo momento in poi, ogni volta che si toglierà la chiavetta il led verde lampeggerà.

Ricordate che per non rischiare di corrompere un file aperto e perdere dati, non bisogna mai togliere la chiavetta mentre arrivano dati dalla seriale, ovvero quando il led rosso è acceso: prima di togliere la Pen-Drive bisogna fermare il flusso di dati seriale (anche staccando il cavo seriale) e attendere lo spegnimento del led rosso.

Quanto al discorso sui consumi, sappiate la scheda assorbe dai 15 mA in fase di stand-by ai 70 mA

nella fase in cui acquisisce dalla seriale e salva sull'USB; quindi, ipotizzando il peggiore dei casi, in cui acquisisce di continuo, vediamo quanto può durare una batteria da 9 V classica, che di solito è da 800 mA/h.

La durata possiamo ricavarla con la formula:

$$T = \frac{800 \text{ mA} \cdot h}{70 \text{ mA}} = 11.5h$$

Questa è l'ipotesi peggiore; nel caso più reale, di una acquisizione intervallata da fasi di inattività, l'autonomia può incrementare fino a 20 ore.



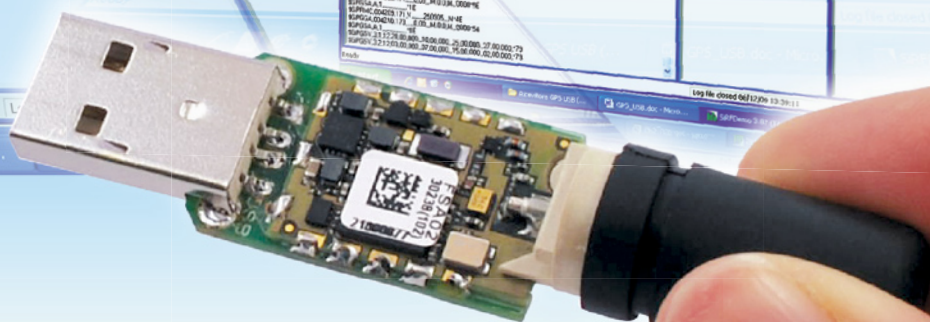
per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Il master dello stampato ed il firmware col quale programmare il micro sono scaricabili gratuitamente dal sito della rivista (www.elettronicaonline.it). Il micro programmato (cod. MF784) ed il modulo VDIP1 sono disponibili rispettivamente al prezzo di 15,00 e 46,00 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>

RICEVITORE GPS/USB



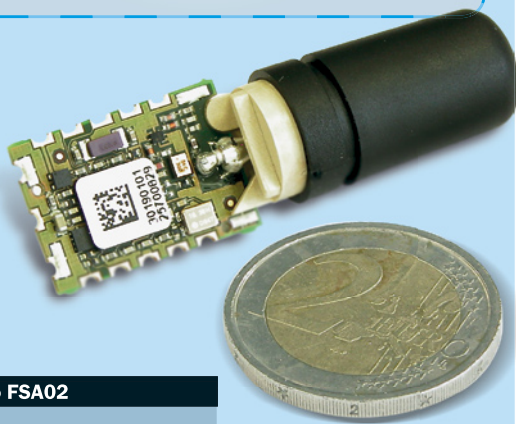
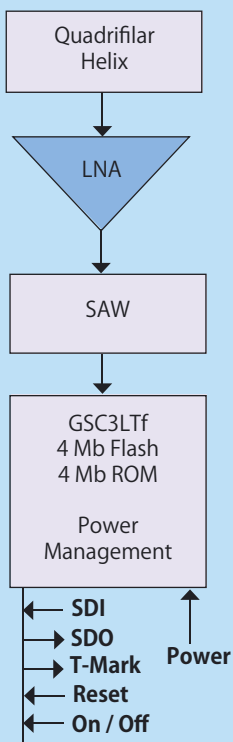
Molto compatto, dotato di antenna integrata e collegabile direttamente a un computer mediante l'Universal Serial Bus, grazie a un semplice software consente di valutare le informazioni in tracking e personalizzare i dati da rilevare; ideale per realizzare localizzatori e per l'abbinamento con PC notebook o palmari.

..... di DAVIDE SCULLINO

Nelle pagine di Elettronica In avete certamente visto passare molti dispositivi legati al mondo del GPS, però spesso i nostri progetti hanno sfruttato prodotti finiti o ricevitori ad uscita seriale TTL adattati per il collegamento al PC o dei quali ci limitavamo a leggere dati per funzioni di data-logging o per la trasmissione a distanza nei nostri localizzatori GPS/GSM. Stavolta vi parliamo di qualcosa di nuovo, ossia di un completo ricevitore miniaturizzato provvisto di antenna (elicoidale, nello specifico) e connessione USB, realizzato da

noi; naturalmente anche in questo progetto abbiamo impiegato un ricevitore già pronto, però l'abbiamo integrato in un circuito equipaggiato con un'interfaccia USB 2.0 compatibile che gli permette di dialogare con un qualsiasi computer e di prendere da questo l'alimentazione. Mediante un apposito software (SiRFDemo) scaricabile dal nostro sito Internet www.elettronica.in, il ricevitore può essere impostato per sfruttarne a pieno le ottime prerogative: nello specifico possiamo personalizzare il protocollo di comunicazione scegliendo

Il ricevitore Falcom FSA02



Caratteristiche tecniche del modulo FSA02

Ricevitore GPS	
Tempo di FIX (TTFF)	
Hot start (sensibilità 23 dBHz)	< 4 s. tipico
Warm start (sensibilità 28 dBHz)	< 35 s. tipico
Cold start (sensibilità 16 dBHz)	< 45 s. tipico
Sensibilità	
Acquisizione autonoma	-142 dBm
Tracking	-159 dBm
Ricevitore	
Banda	L1, CA code
Canali	20
Frequenza di refresh	1 Hz
Protocolli supportati	NMEA, SiRF binary
Formato dati GPS	
WGS84	
Processore	
CPU	ARM7/TDMI, 90 nm
FLASH	4Mb
ROM	4Mb
Caratteristiche elettriche	
Alimentazione	3,0 V+5,5V
Assorbimento modalità Full power	65 mW (3,3V)
Assorbimento modalità Trickle power	30 mW (3,3V)
Antenna integrata	
Frequenza	1.575,42 MHz
Guadagno	-3,5 dBic
Angolo di ricezione	120°
VSWR	<2,0:1

Per realizzare un dispositivo inseribile direttamente nella presa USB di un computer avevamo bisogno di un ricevitore piccolo, leggero e dotato di antenna integrata; lo abbiamo trovato nella produzione Falcom. Più precisamente, abbiamo scelto il modello FSA02, completo ricevitore a 20 canali con interfaccia seriale TTL ed antenna integrata elicoidale quadrifilare.

tra lo standard NMEA e il SiRF binary, ma anche le modalità di funzionamento.

Il programma demo è utilissimo per valutare le qualità del ricevitore, basato, come è facile immaginare, su un chipset SiRF Star III, un chipset che negli ultimi anni risulta sempre più apprezzato e impiegato nei localizzatori e navigatori satellitari; consente inoltre l'emulazione del funzionamento e vari test per la verifica della ricezione, della visione dei satelliti ecc.

Ma prima di analizzare il software, che pure è una parte determinante, diamo uno sguardo al circuito elettronico che realizza il ricevitore GPS/USB.

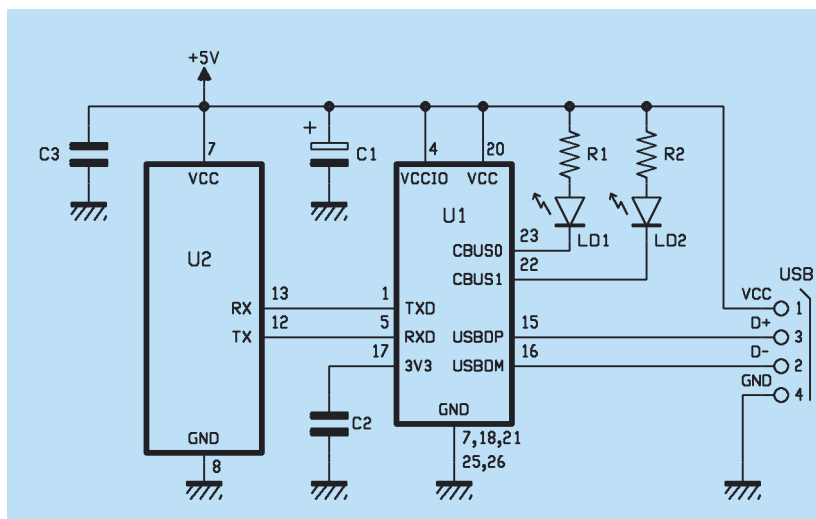
SCHEMA ELETTRICO

Come vedete, il circuito è molto semplice, anche perché tutto quello che serve è raggruppato in un modulo e un integrato: il ricevitore GPS e l'FT232RL, nello schema siglati rispettivamente U2 e U1. Sostanzialmente consta di un ricevitore Falcom FSA02 provvisto di un'interfaccia che gli permette di dialogare con dispositivi dotati di USB: infatti nel circuito la sua porta di comunicazione seriale a livello TTL, facente capo ai piedini 12 e 13 (rispettivamente TX ed RX) è interfacciata all'USB mediante un converter TTL/USB di tipo FT232RL.

Il ricevitore GPS è del tipo miniaturizzato ed è provvisto di antenna integrata elicoidale rivestita in gomma, capace di garantire una discreta sensibilità occupando uno spazio minimo, come si addice a un dispositivo fatto per essere inserito in una presa USB e che quindi deve presentare ridotte dimensioni; tutto l'insieme, ossia il ricevitore, il piccolo stampato contenente l'interfaccia USB e il connettore USB tipo A,

è grande come una Pen-Drive. Il modulo FSA02 è un buon ricevitore GPS per il segnale civile (portante L1) in grado di sfruttare anche i sistemi complementari di aumento della precisione quali DGPS, WAAS/EGNOS (i cosiddetti SBAS, ossia Satellite Based Augmentation System) e l'Extended Ephemeris, tipico della SiRF; ha tempi di avvio abbastanza ridotti, quantificabili in 4 secondi per l'Hot Start, 35 secondi per il Warm Start e 45 s per il Cold Start. La sensibilità con l'antenna in dotazione è dell'ordine dei -159 dBm in tracking (quando ha agganciato il numero minimo di satelliti) e -142 dB in acquisizione; sempre con l'antenna elicoidale, la copertura è intorno ai 120 gradi in verticale. I protocolli che il modulo supporta sono l'NMEA e il SiRF binary.

Il convertitore è un integrato prodotto dalla FTDI che più volte abbiamo utilizzato: l'FT232RL è un adattatore da RS232 (o TTL con dati in formato RS232) a USB 2.0, funzionante a 5 volt e provvisto di oscillatore di clock ed EEPROM interni; è sostanzialmente l'FT232 già visto in altre applicazioni, solo che ha l'oscillatore incorporato e quindi non necessita del quarzo esterno. L'alimentazione a 5 volt permette di farlo funzionare con i 5 V prelevati dall'USB, bus che, lo ricordiamo, consta di quattro linee: dato positivo (D+), dato negativo (D-), +5V e massa. Con gli stessi 5 volt prelevabili dalla porta USB (la corrente basta e avanza, dato che ogni porta USB standard deve poter erogare fino a 500 mA)



facciamo funzionare anche il ricevitore GPS, che si accontenta di una ventina di milliampere a piena potenza e circa 10 in modo power-save.

I pin CBUS0 e CBUS1 vengono programmati in fabbrica dal costruttore per gestire i due led di stato, che nel nostro circuito segnalano uno l'attività del canale dati in ricezione e l'altro la trasmissione.

Prima di passare alle note costruttive spendiamo qualche parola in più sul convertitore TTL/USB della FTDI: si tratta sostanzialmente di un DCE (*Data Communication Equipment*) ossia un apparato in grado di trasportare i segnali di una linea RS232 (sebbene a livelli di tensione TTL-compatibili) su una USB, gestendo gli eventuali criteri di controllo tipici di un UART (CTS, RTS, DTR ecc.); l'integrato è quindi paragonabile ad un modem, solo che affaccia una porta seriale su un bus USB. Può funzionare con una propria alimentazione, oppure prelevando i 5 volt dall'USB cui si collega; in quest'ultimo caso, però, la

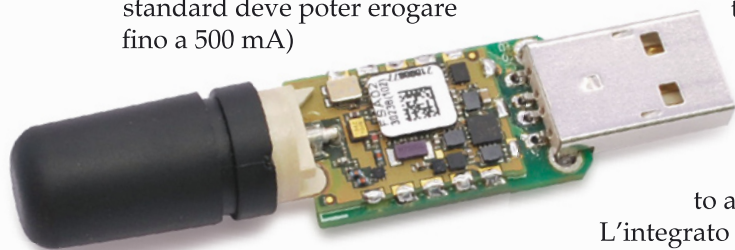
massa (negativo dell'USB) è in comune fra il computer e il circuito collegato all'FT232RL.

L'integrato della FTDI ha

internamente un regolatore di tensione che ricava i 3,3 V necessari alla propria logica; lo stesso potenziale viene reso disponibile per altri circuiti, dal piedino 17 al quale, nel nostro caso, è collegato il condensatore di filtro C2, necessario a filtrare la tensione ricavata internamente.

REALIZZAZIONE PRATICA

Il nostro ricevitore GPS/USB si realizza molto semplicemente, a patto di avere a disposizione un buon saldatore dotato di punta sottile e del filo di lega saldante del diametro di 0,5 mm; la bassetta è molto piccola e a doppia ramatura e dovete realizzarla a partire dalle tracce lato rame scaricabili dal nostro sito www.elettronica.in.it. Una volta ottenuto il c.s. dovete lavorarlo per realizzare la cava nella quale dovrà entrare la parte plastica inferiore del ricevitore GPS; per farlo seguite il disegno di montaggio. Fatto ciò, realizzate le vie (i passaggi tra le piazzole comuni alle due facce) e poi sistemate i pochi componenti, che sono tutti in SMD; prestate attenzione all'FT232RL, che consigliamo di centrare bene nelle rispettive piazzole e bloccare stagnando uno dei terminali più esterni, quindi saldare stagnando alternativamente pin dei due lati. Il modulo Falcom va sistemato

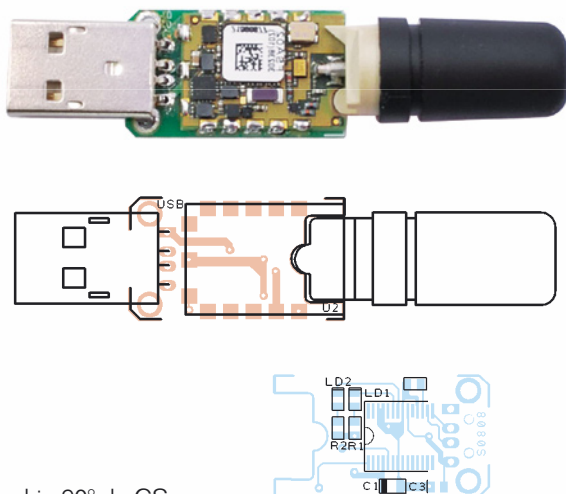


Elenco Componenti:

R1: 470 ohm (0805)
R2: 470 ohm (0805)
C1: 100 nF
multistrato (0805)
C2: 4,7 μ F 6,3 VL
tantalio (CASE-T)
C3: 100 nF
multistrato (0805)
U1: FT232RL
U2: Falcorn FSA02
LD1: Led verde (0805)
LD2: Led rosso (0805)

Varie:

- Connettore USB-A maschio 90° da CS
- Circuito stampato



dal lato opposto a quello dove si trova l'FT232RL; saldarlo è abbastanza facile perché ha i contatti sul bordo della propria basetta, quindi facilmente stagnabili e collegabili con le piazzole del sottostante c.s.

Per completare il montaggio stagnate il connettore USB di tipo A da circuito stampato, prestando particolare attenzione alle linguette di ancoraggio, indispensabili ad assicurare un fissaggio forte e stabile; se non stagnate le linguette negli appositi fori o lo fate male, dopo qualche inserzione ed estrazione del dispositivo è facile che il connettore USB si allenti.

A questo punto la costruzione è

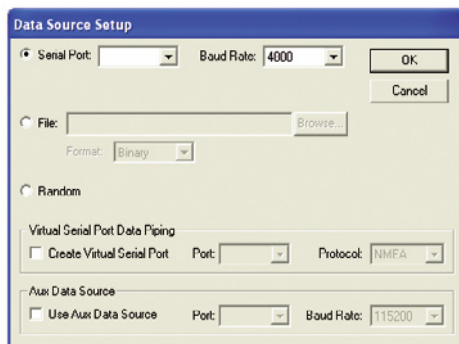


Fig. 1 - Impostazioni della seriale.

terminata; il dispositivo è pronto per l'uso, dato che non richiede regolazione alcuna o meglio,

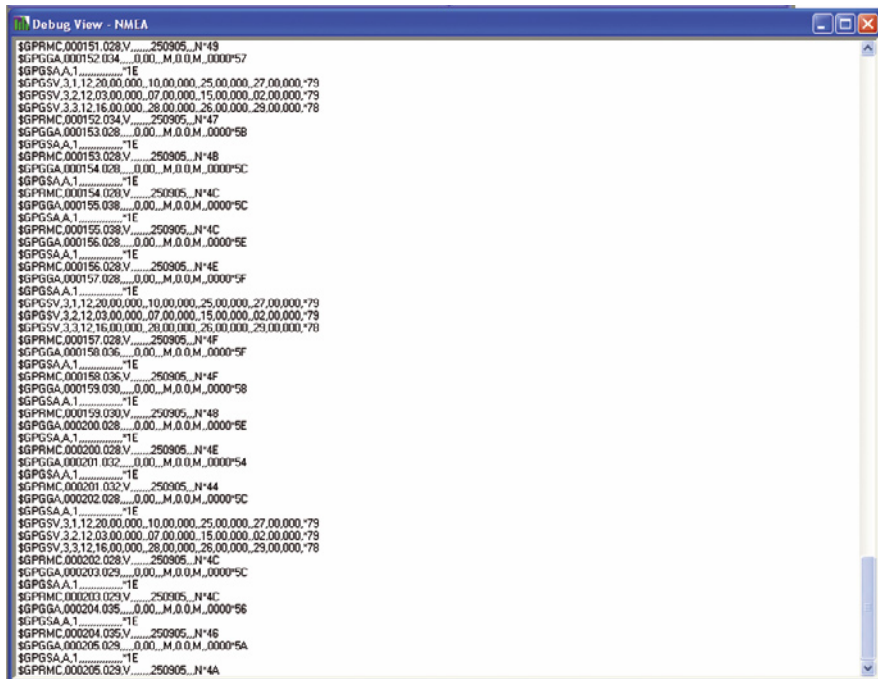


Fig. 2 - Finestra Debug View: è attiva impostando il protocollo NMEA.

che le regolazioni riguardanti il suo modo di funzionamento si fanno direttamente via USB con l'apposito software che ora descriveremo. Per usarlo corret-

tamente occorre preventivamente installare i driver di periferica, che poi non sono altro che quelli dell'FT232RL, disponibili gratuitamente per il download nel sito *www.ftdichip.com*; scaricato il file, scompattatelo e poi inserite il ricevitore GPS nella prima presa disponibile del vostro Personal Computer. Quando il sistema operativo vi avverte che è stato installato un nuovo hardware, avviate la ricerca dei driver indicando la cartella creata dalla scompattazione del file driver scaricato dal sito della FTDI; completata l'installazione potete iniziare a lavorare.

IL SOFTWARE SIRFDEMO

Il programma, scaricabile gratuitamente dal nostro sito o dalla pagina web <http://www.falcom.it>.

de/support/software-tools/sirf/, si installa facilmente in ambiente Windows (Windows 2000, XP, Vista) partendo dal file compresso scaricato. Una volta installato,

crea nella cartella *Programmi* e nella barra *Programmi* l'icona SiRF; per lanciare SiRF Demo puntate la voce *SiRF* e fate clic su *SiRF Demo*.

Avviato il programma, appare subito la finestra di dialogo *Data Source Setup* (Fig. 1) con la quale definire i parametri di comunicazione: in essa, nel menu a tendina *Serial Port* bisogna scegliere la COM virtuale associata al dispositivo USB; se non ricordate qual è la porta COM associata al ricevitore GPS, nel Desktop fate clic con il pulsante destro del mouse su *Risorse del computer* e impartite il comando *Proprietà*. Aperta la finestra di dialogo *Proprietà del sistema*, fate clic sulla scheda *Hardware* e qui sul pulsante *Gestione periferiche*; visualizzato l'elenco delle periferiche, fate clic su *Porte (COM e LPT)* e, nella struttura che si apre, vedete, tra parentesi, qual è la COM assegnata a *USB Serial Port* (se avete più dispositivi di comunicazione su USB, per evitare equivoci fate doppio clic sul loro nome e nella finestra di dialogo che si apre verificate di quale si tratta). Nella finestra di dialogo *Serial Port*, scelta la COM, dal menu a tendina *Baud Rate* scegliete la velocità di 4800, dato che il modulo GPS dialoga alla velocità di 4.800 bps. Confermate con *OK* e accedete alla schermata principale, composta, nella sua forma completa, da sei finestre, ognuna delle quali può essere chiusa o resa disponibile mediante un apposito comando nella barra dei comandi. Le finestre sono le seguenti: *Signal View*, che indica le caratteristiche dei segnali ricevuti dai singoli satelliti e dei satelliti stessi; *Map View*, che riporta le impostazioni del ricevitore e il numero di satelliti agganciati, oltre alla condizione di fix avvenuto. *Radar View*

mostra la posizione dei satelliti agganciati sul piano celeste, con il nome corrispondente; c'è poi

Debug View (Fig. 2) che visualizza i dati in arrivo dalla seriale del ricevitore GPS, stringa per stringa



Vino? Vigneti e vendemmie li cura il GPS

In Franciacorta agricoltura ed enologia sono da tempo computerizzate: la Berlucchi è stata tra le prime ad avvalersi del satellite per gestire i propri vigneti; gli appezzamenti di terra sono stati mappati in base alla tipologia di suolo e fotografati dal cielo, evidenziando con colori diversi le aree che presentano maggior vigore vegetativo (maturazioni più lente, maggiore acidità e minor grado zuccherino) e quelle con i filari dove il frutto presenta zuccheri più elevati e carichi produttivi inferiori. Sulla base delle mappe ricavate, l'incaricato dei campionamenti si sposta tra i filari con un palmare dotato di GPS, visualizza in tempo reale la sua posizione e preleva campioni. In Sudafrica alcuni produttori di vino impiegano un sistema GPS per determinare la posizione migliore dove piantare le viti di Cabernet-Sauvignon e, con le foto da satellite, identificano l'uva matura e guidano macchine automatiche nella raccolta. In

Germania la *Wagner Pflanzen Technik* ha realizzato una macchina trapiantatrice trainata da un trattore automatizzato guidato dal GPS. Nel settore c'è anche la *Leica*, che ha realizzato *Smart Wine*: è un sistema per la progettazione e la guida durante l'esecuzione di nuovi impianti di vigneto; consente di progettare il migliore impianto del vigneto da realizzare e si basa su ricevitori GPS Leica serie AX1200 a 20 campionamenti al secondo. Svolge il proprio compito in due fasi: nella prima acquisisce le coordinate del perimetro dell'area di impianto e dopo l'operatore creerà, in base alla propria esperienza, un progetto d'impianto, stabilendo orientamento, larghezza dei filari, capezzagna e distanza tra le piante. Il progetto verrà poi trasferito con un file di guida sul computer della trapiantatrice. Nella seconda fase questa macchina viene controllata dal GPS in modo che pianti le viti dove previsto dal progetto.

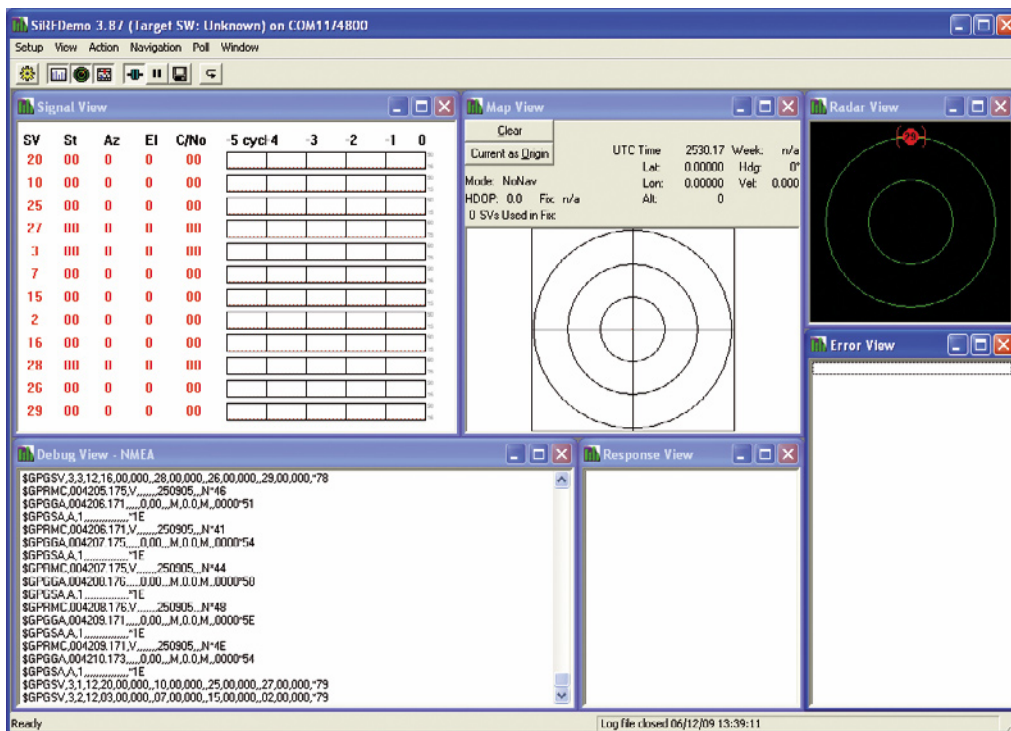


Fig. 3
Una sessione di lavoro con impostato il protocollo NMEA.

ga, in tempo reale. Le ultime due schermate sono *Response View* ed *Error View*, quest'ultima riservata alle comunicazioni di anomalia; *Response View* è attiva solamente nella modalità "sirf protocol", in alternativa a *Debug View*. Per attivare la connessione dati con il ricevitore GPS, dalla

schermata principale dovete aprire il menu *Action* e in esso fare clic su *Open Data Source*, ovvero, nella barra dei comandi, fare clic sul pulsante *Connect Data Source*. Trascorso il tempo necessario, la finestra *Radar View* mostrerà la costellazione dei satelliti agganciati indicando per ognuno la

posizione sull'orizzonte ed il numero identificativo, mentre *Signal View* illustrerà le condizioni dei satelliti e dei segnali corrispondenti. Prima di ogni sessione di lavoro, una volta stabilita la connessione aprite il menu *Action* e impartite il comando *Initialize Data Source*; nella finestra di

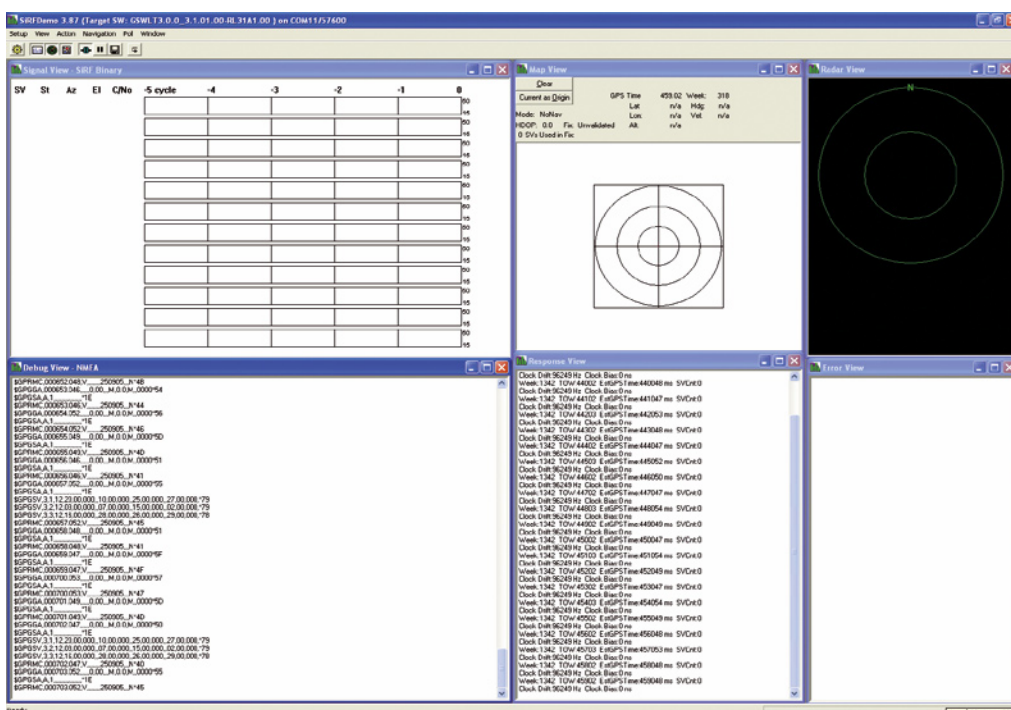


Fig. 4
Sessione di lavoro con attivato il protocollo SIRF.

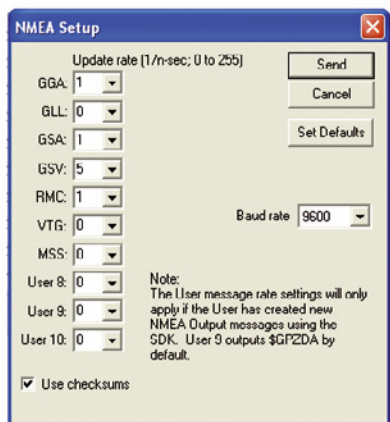


Fig. 5 - La finestra NMEA Setup.

dialogo che esso apre selezionate il solo pulsante d'opzione *Factory Reset* e fate clic su *OK*: ciò ripristina le impostazioni predefinite del ricevitore GPS e resetta il sistema. Il menu *Action* contiene utilissimi comandi tra i quali quelli per impostare il protocollo di comunicazione (*Switch to SiRF protocol* permette di impostare il protocollo SiRF, mentre *Switch to NMEA protocol* disattiva il SiRF e imposta il protocollo NMEA) e *Transmit serial message*, utile per inviare comandi manuali al ricevitore (allo scopo il programma apre una finestra di terminale); notate che attivando il protocollo NMEA vedrete le stringhe di dati in arrivo nella sola finestra *Debug View*, mentre usando il protocol-

lo SiRF (*Switch to SiRF protocol*) *Debug View* sarà inattiva mentre vedrete scorrere i dati in *Response View*.

Nel menu *Action* la gran parte dei comandi viene resa disponibile se il protocollo impostato è SiRF, altrimenti risulta non disponibile. Un dettaglio molto importante riguardante la finestra *Debug View* è che in essa possiamo decidere quali, tra le stringhe NMEA0183 uscenti dal ricevitore GPS, visualizzare: quando dal menu *Action* si impartisce il comando *Switch to NMEA protocol*, appare la finestra di dialogo **NMEA Setup** (Fig. 5) nella quale apposite caselle permettono di decidere il numero di stringhe di ciascun tipo (GGV, GGA, RMC ecc.) e quante volte devono ripresentarsi ad ogni ciclo di comunicazione; ebbene, se non desiderate vedere un certo tipo di dato, impostate a zero il valore nella casella corrispondente. Ad esempio, se volete solo le stringhe GPRMC lasciate il numero nella casella RMC e impostate 0 in tutte le altre. Il pulsante *Set Default* permette in ogni momento di ripristinare la configurazione predefinita.

Quando viene scelto il protocollo NMEA, diventa attiva la finestra *Signal View*, che dà indicazioni sullo stato dei segnali ricevuti (Fig. 6).

Nel menu *Navigation* possiamo impostare le modalità di uso del DGPS e dei sistemi di aumento della precisione SBAS, ovvero rinunciare all'ausilio di uno di essi o di entrambi. Dallo stesso menu è possibile impartire disposizioni sulla ricerca dei dati DGPS e impostare la modalità *Static Navigation*.

Quanto al menu *Poll*, contiene i comandi per visualizzare lo stato

di almanacchi, effemeridi e della funzione *SiRF Extended Ephemeris*; gli stessi comandi permettono di indicare al programma in quali file andare a cercare le informazioni sul DGPS, sulle effemeridi e sull'extended ephemeris, quando manchino dalla rete satellitare.

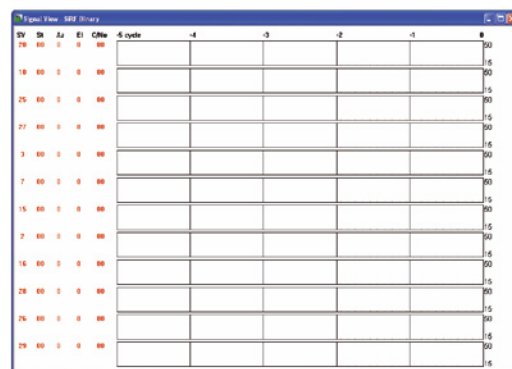


Fig. 6 - La finestra Signal View.

Notate che le funzioni dei menu *Action*, *Navigation* e *Poll* sono disponibili solo se da *Action* avete impartito il comando di scelta del protocollo SiRF, dato che sono caratteristiche della gestione dei chipset SiRF Star. Con NMEA attivo non sono utilizzabili. ■



Fig. 7
La
finestra
Radar
View



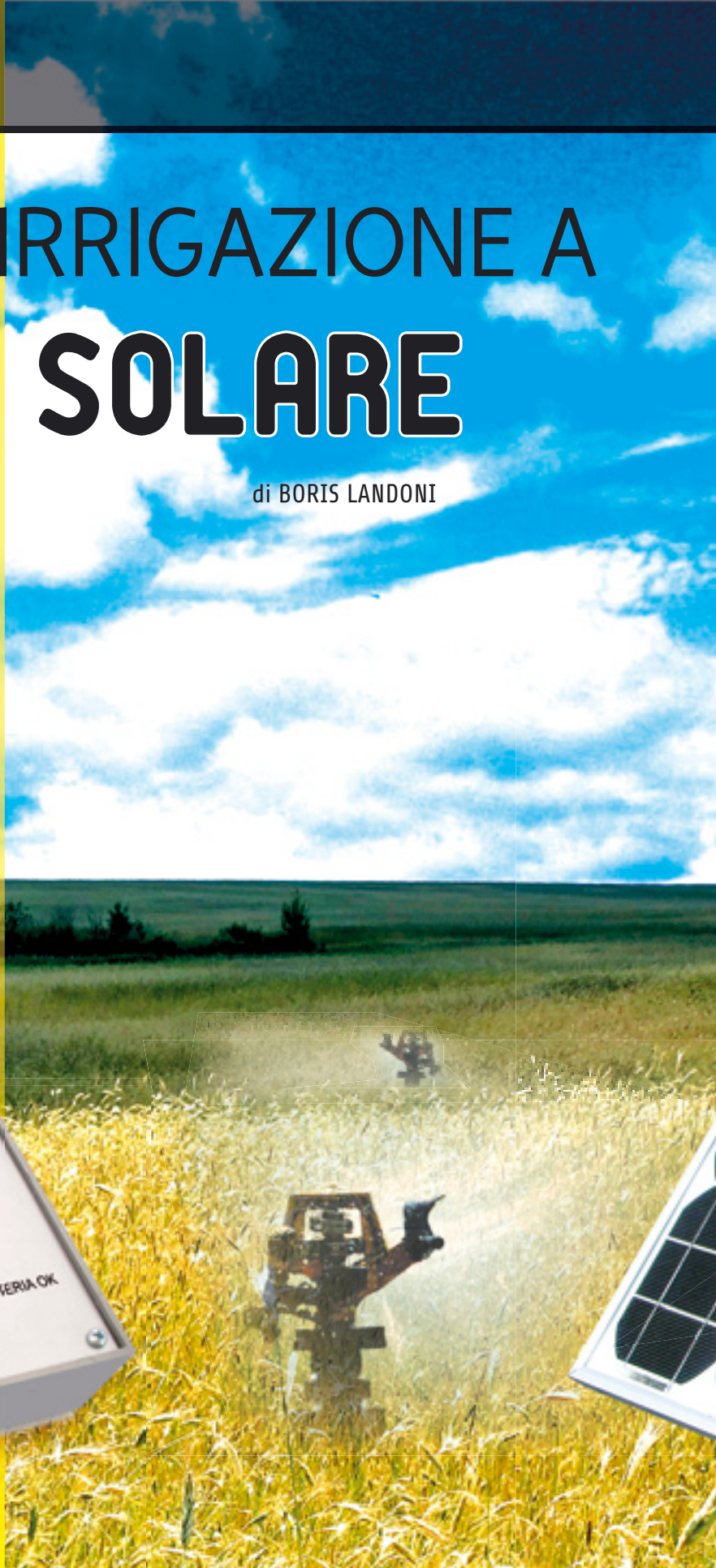
Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Il software applicativo e i driver per l'FT232RL possono essere scaricati gratuitamente dai siti della Falcom e dell'FTDI come indicato nel testo. L'integrato FT232RL costa 7,00 Euro mentre il ricevitore GPS con antenna integrata Falcom FSA02 costa 72,00 Euro.

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

SISTEMA DI IRRIGAZIONE A ENERGIA SOLARE

di BORIS LANDONI

Sistema programmabile per il comando di impianti di irrigazione, ideale per luoghi dove manca la rete elettrica o per la casa quando si parte e si preferisce staccare il contatore. Esegue un programma settimanale, non funziona in caso di pioggia, utilizza una batteria caricata da un pannello fotovoltaico e comanda elettrovalvole standard a 24 Vca.

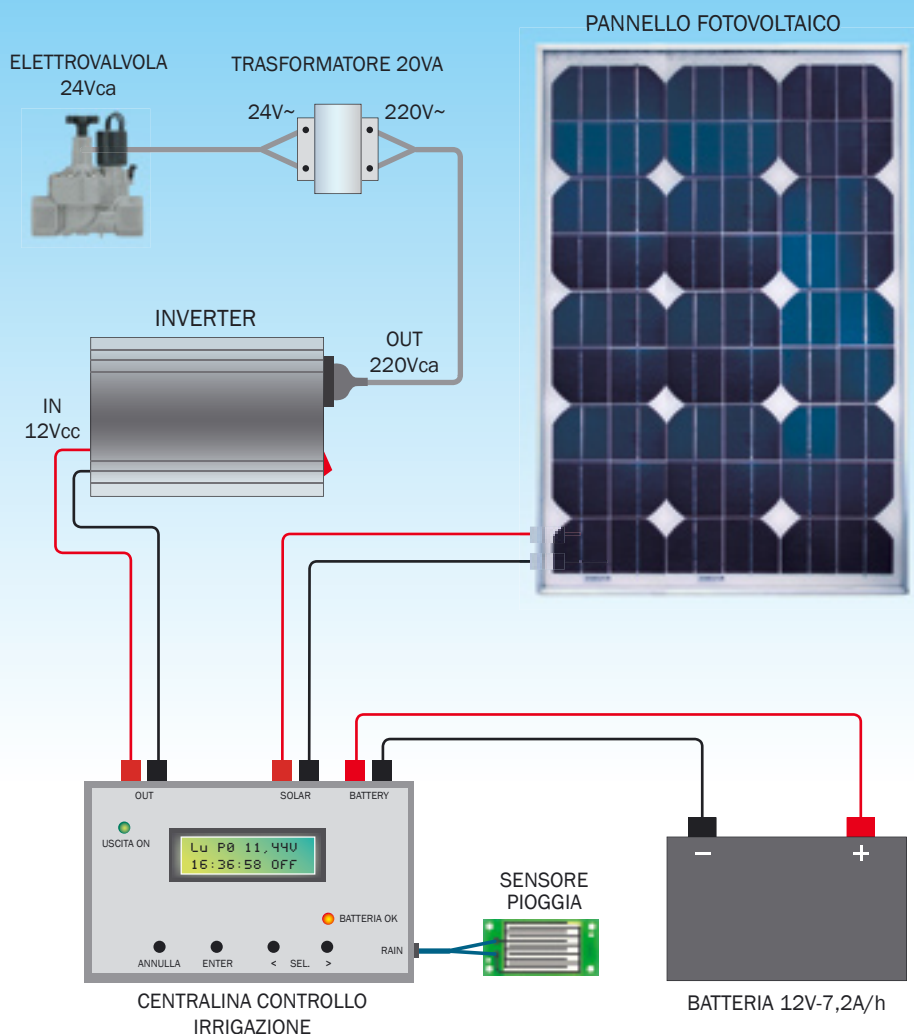




Quando partiamo per le vacanze, facciamo sempre la lista delle cose da fare: oltre a ciò che va in valigia, dobbiamo ricordarci quelle accortezze indispensabili per fare in modo che in nostra assenza la casa sia al riparo da incidenti causati dai servizi che durante l'anno tanto ci occorrono.

Quindi prima di girare la chiave della serratura chiudiamo il gas, l'acqua in tutto o in parte, e stacciamo il contatore dell'elettricità, perché sappiamo che i guasti sono sempre in agguato e possono costare caro: ad esempio un cortocircuito può facilmente (specie col caldo d'agosto) provocare un incendio. Se però abbiamo un bel giardino, come facciamo ad essere certi che venga regolarmente annaffiato? È vero, in commercio esiste una gran quantità di centraline per impianti di irrigazione che possiamo programmare per eseguire programmi diversi settimanalmente e quindi coprire tutto il periodo della vacanza, tuttavia normalmente le centraline funzionano con i 220 volt della rete elettrica e questo

significa dover lasciare attaccata la corrente in casa e quindi restare in balia dei rischi connessi. Per poter partire tranquilli, occorrerebbe una centralina del tutto autonoma sul piano dell'alimentazione elettrica; sicuramente ciò non sarebbe un grosso ostacolo, perché, seppur rare, esistono unità funzionanti a 12 Vcc e quindi a batteria. Però, come la mettiamo con le elettrovalvole per aprire e chiudere l'acqua, dato che solitamente funzionano a 220 Vca o a 24 Vca? Ebbene, ci siamo messi al lavoro e abbiamo trovato la soluzione, che poi è quella che potete vedere in queste pagine: un sistema di irrigazione autonomo funzionante a partire dalla tensione di una batteria da 12 volt di piccola capacità; con i 12 V facciamo funzionare la centralina, mentre per le elettrovalvole impieghiamo un inverter DC/AC e un trasformatore (l'inverter ricava 220 Vca e con un trasformatore da rete otteniamo i 24 Vca che ci occorrono). Per ricaricare la batteria impieghiamo un pannello solare che colleghiamo direttamente alla



Il nostro sistema abbinato a valvole a 24 Vca e ad un sensore di pioggia: la centralina funziona con la batteria, mantenuta carica dal pannello solare.

nostra centralina, la quale, oltre a fare da timer e a comandare le elettrovalvole, funziona da regolatore di carica. Ma questo ed altro lo approfondiremo analizzando lo schema elettrico. Il nostro sistema di comando dell'irrigazione è stato pensato non solo per la casa, ma anche e soprattutto per quei luoghi dove non sia disponibile la rete elettrica a 220 volt: ad esempio case di campagna, orti, terreni con serbatoi di accumulo dell'acqua piovana.

IL SISTEMA

Detto ciò, scendiamo sul piano prettamente tecnico e spieghiamo

com'è stato realizzato il sistema di irrigazione; in sede di progetto abbiamo ipotizzato di abbinarlo a due comuni elettrovalvole a 24 Vca capaci di garantire portate dell'ordine dei 50÷200 litri/minuto, facendole aprire per un'ora al dì. Da una ricerca condotta sui prodotti che si trovano presso gli specialisti del giardinaggio e nei supermercati del "fai da te", le elettrovalvole del genere che si trovano in commercio assorbono mediamente 200 mA a regime (poco più allo spunto) quindi diciamo pure che la potenza consumata è dell'ordine dei 4,8 watt a valvola e perciò poco meno di 10 watt a 24 Vca. Come accennato,

per far funzionare queste valvole ci serviamo di una batteria da 12 volt, che alimenta un inverter da 220 volt di uscita e di un trasformatore da 220/24 V, che dovrà essere da 10÷20 W; ammettendo un rendimento complessivo dell'insieme inverter/trasformatore di circa l'80 %, vediamo che la potenza che la batteria deve erogare ammonta a 12 watt. A 12 V, l'assorbimento dall'accumulatore sta intorno all'ampere, che per un'ora vuol dire consumare 1 A/h. Rapportata all'intera giornata, questa energia si traduce in 1/24, dato che pensiamo di far aprire le elettrovalvole per un'ora al giorno; ai circa 42 mA/h così ricavati va aggiunto il consumo medio in standby dell'elettronica di controllo, ossia la centralina con integrato il regolatore di carica, che ammonta a 35 mA, quindi 35 mA/h, dato che la centralina funziona 24 ore su 24.

Riassumendo, in totale la batteria deve cedere 77 mA/h, energia da reintegrare mediante l'apporto del pannello solare fotovoltaico. Quest'ultimo dovrebbe erogare proprio i 77 mA/h, tuttavia normalmente un pannello eroga corrente quando viene illuminato sufficientemente. Ipotizzando che possa lavorare con la massima insolazione per tre ore al giorno e quindi di poter prelevare corrente da esso solo per tale intervallo, dovrebbe sviluppare una corrente di picco (I_p) pari a:

$$I_p = 77 \text{ mA} \times 24/3 = 616 \text{ mA}$$

Dato che un pannello che carica una batteria da 12 V deve erogare mediamente 15÷16 volt di picco, possiamo dire che ce ne serve uno da circa 10 watt. Ora dobbiamo considerare che la piovosità media del periodo estivo non supera il giorno alla settimana, quindi in linea di massima

rinunciando alla ricarica per 4÷5 giorni al mese, il pannello deve caricare maggiormente nei giorni di sole, anche se in realtà quando piove l'irrigatore non serve che annaffi e quindi si risparmiano 42 mA/h (ossia oltre 5 watt). Per stare larghi, scegliamo un pannello che dia qualcosa di più e che quindi sviluppi 12 o 15 watt; così siamo certi che il sistema possa lavorare autonomamente teoricamente all'infinito, anche e soprattutto perché se è vero che il pannello solare sviluppa la potenza di picco per poche ore al giorno e in piena insolazione, in realtà genera corrente da mattina a sera. La batteria può essere anche abbastanza piccola, dato che deve fornire una corrente di picco non superiore all'ampere e che, anche ammettendo sette giorni di fila di pioggia (cosa che in estate è abbastanza inverosimile) il consumo non supera i 5,88 A/h (perché la centralina può non aprire le valvole se rileva che sta piovendo); scegliendo un accumulatore da 7,2 A/h siamo al riparo da tutto.

Prima di passare alla descrizione dello schema della centralina di controllo, facciamo chiarezza su un dettaglio: i più esperti tra i nostri lettori avranno certo da obiettare sul nostro modo di fornire i 24 V alternati, perché una tale tensione si sarebbe potuta ottenere da un ponte ad H composto da quattro transistor pilotati da una coppia di segnali rettangolari in opposizione di fase tra loro, senza andare a scomodare un inverter e un trasformatore da rete, soluzione che oltre a ingombrare comporta uno spreco di elettricità dovuto alle perdite di conversione nell'inverter e nel trasformatore.

Ebbene, se da un lato ciò è vero, va detto che generando sul posto i 24 Vca avremmo potuto coman-

dare solo elettrovalvole a 24 volt in alternata, quindi il nostro sistema non sarebbe stato utilizzabile da chi ha un impianto con valvole a 220 Vca o a 12 volt in continua. Invece facendo in modo che quando deve essere aperta l'acqua la centralina dia i 12 volt continui della batteria, possiamo con questi alimentare direttamente eventuali elettrovalvole a 12 Vcc, mediante un inverter 12Vcc/220ca elettrovalvole funzionanti a 220 V in alternata e infine, con inverter e trasformatore, valvole da 24 Vca.

LA CENTRALINA

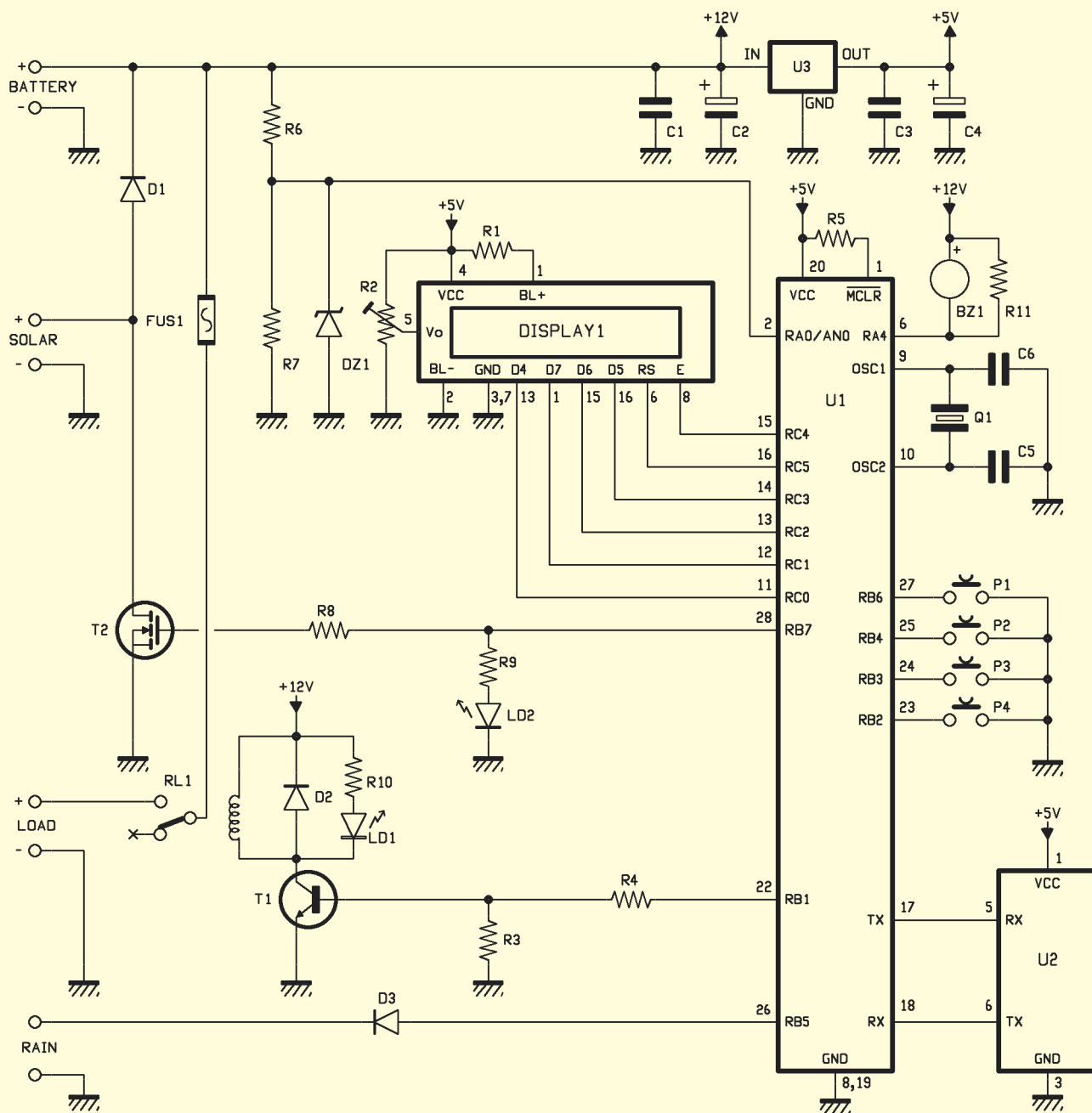
Per gestire l'intero sistema usiamo una centralina che in realtà è un timer settimanale programmabile molto semplicemente mediante quattro pulsanti e un display LCD a 16 caratteri su due righe, che permette di impostare un diverso orario di attivazione dell'irrigazione per ogni giorno della settimana; l'attuatore di questo temporizzatore è un relé, che fornisce 12 Vcc quando deve far attivare le elettrovalvole. Con i 12 volt alimentiamo direttamente le valvole a 12 Vcc o, mediante

inverter e trasformatore, quelle a 24 o 220 Vca. La stessa centralina è in grado di badare ai propri bisogni e si avvale della batteria da 12 volt, della quale gestisce la carica tramite un regolatore di carica di tipo shunt.

Il tutto è reso estremamente semplice dal fatto che sia i cicli di temporizzazione, sia il regolatore di carica, sono gestiti da un microcontrollore Microchip PIC16F876A.

Alla centralina fanno capo i morsetti della batteria, quelli di uscita (alimentazione delle valvole o dell'inverter) ossia lo scambio del relé, oltre a quelli del pannello solare. Ma non è tutto qui: per fare bene le cose, abbiamo pensato di dotare il circuito di un rilevatore di pioggia, realizzato mediante un sensore di presenza d'acqua tipo quelli impiegati nelle centraline meteo, in modo da bloccare il programma del timer, ovvero impedire l'attivazione dell'uscita (relé) e delle valvole se all'ora fissata per l'irrigazione sta piovendo o ha da poco smesso di piovere. Questo perché se il terreno è bagnato è superfluo annaffiarlo. Tale funzione permette di risparmiare energia e limita il consumo proprio quando ce n'è bisogno, dato che





se piove il pannello solare praticamente non carica la batteria. Per come funziona la parte timer, all'ingresso corrispondente è possibile collegare anche altri sensori, quali ad esempio un interruttore crepuscolare. Per ora, vi basti sapere che l'ingresso riservato al sensore di pioggia è normalmente aperto. Restando sullo schema elettrico,

diciamo che il microcontrollore gestisce i programmi di temporizzazione, l'ingresso di comando esterno, i pulsanti con i quali effettuiamo le configurazioni, il display e il gruppo di carica della batteria. Siccome ha già troppo da fare, solleviamo il PIC dall'onere di tenere l'ora, che viene fornita dal modulo RTC FT674M (U2) che già conoscerete perché

pubblicato nel fascicolo 117 della nostra rivista (al quale rimandiamo per le spiegazioni e la documentazione del caso); il modulo comunica l'orario di sistema al micro quando questo gli richiede l'aggiornamento. Più esattamente, genera ora e data di sistema e le mantiene anche in assenza dell'alimentazione del circuito, grazie ad un sistema di backup

interno che tiene i dati in memoria, anche in assenza dell'alimentazione principale, grazie ad una pila al litio a bottone. L'RTC comunica mediante una linea full-duplex facente capo ai suoi contatti 5 e 6, i quali terminano sui pin, rispettivamente, 17 e 18 dell'UART del PIC16F876A; la comunicazione avviene attraverso una porta seriale impostata per 9.600 bps, nessuna parità, 8 bit dati, 1 bit di stop.

E veniamo al display, che è del tipo a 2 righe per 16 caratteri, dotato di controller Hitachi HD44780 o compatibile; per gestirlo, il software del microcontrollore usa RC0 RC1, RC2, RC3 come uscite per l'invio dei dati ed RC4, RC5, ancora come output, ma per la gestione dei criteri di lavoro (RC4 comanda l'E e RC5 interviene sull'RS). La sua logica comunica sfruttando un bus di 4 o 8 bit (a seconda dell'impostazione data: 4 bit, nel nostro caso) e tre linee di comando chiamate R/W, RS, ed E; la prima determina se deve ricevere i dati o inviarne al dispositivo che lo pilota. Nel nostro progetto, dovendo semplicemente visualizzare scritte in arrivo dal microcontrollore, non gestiamo il piedino 7 ma lo lasciamo a massa, il che corrisponde alla condizione logica 0, quindi al modo di funzionamento "write": il dispositivo riceve semplicemente e il suo buffer di memoria viene usato solo per contenere i dati da visualizzare. Il protocollo di comando del display prevede di definire la posizione di ciascun carattere e la struttura del carattere stesso; in altre parole, per visualizzare qualsiasi simbolo bisogna sempre inviare una coppia di dati, descrittivi, i primi la posizione, e i secondi il contenuto del carattere. Allo scopo, la linea RS indica al display se i dati in

arrivo sul bus vanno interpretati come comandi o informazioni da visualizzare: per comando si intende un byte che fa muovere il cursore o azzerare l'LCD, mentre i dati sono, ovviamente, i caratteri da mostrare. Il micro pone RS a livello alto quando manda ai piedini del bus (RC0÷RC3) gli impulsi relativi a istruzioni che il

resetta sempre il controller con l'apposita istruzione (0000) che invia sui quattro bit relativi alle linee D4÷D7 (bus dati) dopo aver posto a zero E per abilitare il buffer e ad 1 logico RS in modo da comunicare alla logica che i dati sul bus vanno interpretati come comando (di azzeramento) e

Il comando esterno

La nostra centralina prevede la possibilità di sospendere l'attivazione dell'uscita (relé) in caso di particolari eventi; allo scopo abbiamo previsto l'ingresso RAIN, che in linea generale serve per collegare un sensore di pioggia col quale verificare se sta piovendo o è appena piovuto, in modo da evitare di attivare l'irrigazione. All'ingresso è associata la funzione del timer chiamata *Impostazioni Input*: scegliendo per questa il valore ON, quando l'ingresso viene cortocircuitato il relé si attiva indipendentemente da cosa impone il programma, mentre impostando OFF nella stessa condizione RL1 resta a riposo, anche se il timer dovrebbe attivarlo. La possibilità di scegliere come

si debba comportare l'uscita può servire ad utilizzare altri sensori: ad esempio un umidostato inserito nel terreno e che in giornate particolarmente calde, può anticipare l'irrigazione rispetto al programma. In questo caso *Impostazioni Input* deve essere ad ON.



display deve svolgere (per esempio l'avanzamento del cursore o della posizione in cui scrivere) e a 0 logico se, invece, invia caratteri da visualizzare.

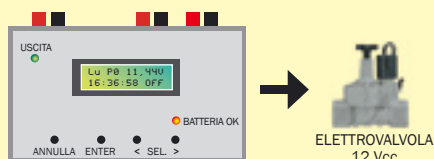
La linea E corrisponde all'Enable del display e serve ad abilitare (se posta a zero logico) o meno (1 logico) il componente.

Prima di scrivere uno o più caratteri, il microcontrollore

perciò non sono da visualizzare. Poi disattiva la linea Enable (la pone a 1 logico) mentre lascia i 4 bit inalterati. Dopo il comando di reset, il micro invia, una dopo l'altra, le coppie di istruzioni che spostano il cursore (un impulso sul pin RS comunica al driver HD44780 che deve ricevere le informazioni sul bus interpretandole come comandi) e scrivono il

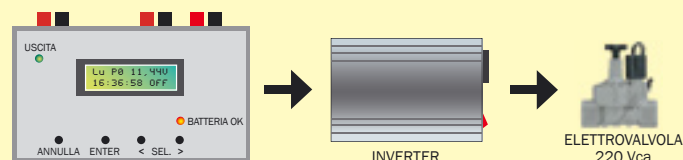
Le possibili configurazioni

Utilizzo con elettrovalvole a 12 Vcc

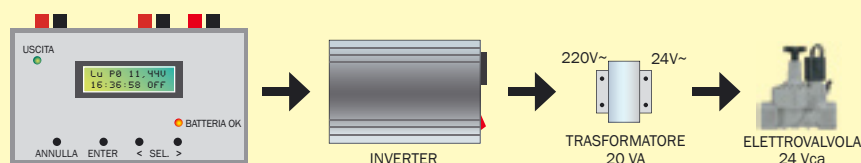


Nell'impiego con valvole a 220 o 24 Vca, l'inverter dovete sceglierlo in base alla potenza delle valvole; di solito un 50÷100 watt basta e avanza.

Utilizzo con elettrovalvole a 220 Vca



Utilizzo con elettrovalvole a 24 Vca



carattere (stavolta RS viene forzato a zero, comunicando al display che le operazioni si riferiscono a dati da mostrare sulla matrice LCD e non più a comandi). Come è d'obbligo, ogni singola istruzione viene scandita dall'impulso di Enable sulla linea E.

Oltre ai piedini già descritti, il display ne usa altri sei: due servono per l'alimentazione della logica (+5 volt riferiti a massa) uno è una massa secondaria e un altro va polarizzato con un potenziometro, alimentato a 5 volt, con il quale si regola il contrasto delle scritte; gli ultimi due sono il positivo (BL+, 5V) e il negativo (BL-) di alimentazione della retroilluminazione, grazie alla quale i caratteri possono essere visti anche in condizioni di oscurità. Riguardo alla retroilluminazione va detto che abbiamo dimensionato la resistenza di limitazione della corrente (R1) per un valore che consenta il minimo assorbimento pur mantenendo una visibilità accettabile.

L'alimentazione dell'intera centralina si applica ai punti siglati BATTERY; raggiunge direttamente l'ingresso del regolatore integrato U3 (un comune 7805) che provvede a ricavare i 5 volt occorrenti al microcontrollore, al modulo RTC e al display a cristalli liquidi. Alla linea di alimentazione principale sono collegati il relé ed il circuito di rilevamento dello stato di carica della batteria, realizzato mediante il partitore di tensione R6/R7 e l'ADC del microcontrollore; quest'ultimo campiona e digitalizza il potenziale presente sulla linea RA0/AN0 (assegnata all'inizializzazione al convertitore A/D) e viene protetto dal diodo Zener, il quale impedisce che la tensione ai capi della R7 superi i 5 volt. Il circuito di monitoraggio della tensione della batteria è parte del sistema di regolazione della carica e di gestione del pannello solare che vedremo a breve. Adesso completiamo la descrizione dello schema dicendo che RB2,

RB3, RB4 ed RB6 sono impostate come input con pull-up interno e servono a leggere i quattro pulsanti impiegati per l'impostazione del timer; RB1 è settata come output e provvede, quando serve, a mandare in saturazione il transistor T1, compito del quale è alimentare la bobina del relé e quindi far chiudere il contatto di uscita del circuito (condizione evidenziata dall'accensione del led verde) e fornire alimentazione all'utilizzatore.

Quanto alla RB5, viene inizializzata come ingresso con pull-up interno e serve a rilevare il comando esterno, che può essere impartito da qualsiasi contatto pulito (interruttore, pulsante, relé) ma anche da un transistor NPN connesso in modo open-collector con l'emettitore sulla massa. Infine, la linea RA4 viene usata come uscita per pilotare in modo sink il cicalino BZ, che serve ad assistere le procedure di programmazione e a ripetere acusticamente l'inizio e la fine delle temporizzazioni.

IL REGOLATORE DI CARICA

Prima di vedere la modalità timer e le impostazioni corrispondenti, spieghiamo come funziona il controllo della carica, ovvero come il micro mantiene in carica la batteria avvalendosi del pannello solare; per prima cosa il PIC legge la tensione applicatagli dal partitore R6/R7 e se la trova minore di 12,6 volt dà il via libera alla carica, che avviene semplicemente lasciando il pannello (connesso ai morsetti + e - SOLAR) libero di dare corrente alla batteria e, ovviamente, al resto della centralina. Quando la tensione tra i punti BATTERY raggiunge 13,7 V, il micro ritiene carica la batteria: pone a livello alto la propria linea RB7 e polarizza il gate del mosfet, il quale chiude in

cortocircuito i morsetti SOLAR e assorbe su di sé tutta la corrente che il pannello potrebbe erogare, impedendo così che la batteria riceva un'ulteriore carica, inutile e dannosa. Lo stato della RB7 viene evidenziato dal led rosso, il quale funge da spia di stato della batteria: acceso significa che la carica è completata, mentre quando è spento vuol dire che il pannello la sta ancora caricando o che comunque non ha raggiunto quei 13,7 V corrispondenti alla piena carica.

Il circuito di controllo della tensione di batteria viene utilizzato anche nella modalità timer, per evitare che il relé vada a dare corrente alle elettrovalvole (o all'inverter che le alimenta) se la batteria è considerata scarica: l'uscita viene staccata (anche se il timer dice di attivarla) se la batteria va sotto i 10,5 V e viene ricollegata (se il timer dice che l'uscita deve essere attivata) se supera gli 11,2 volt.

LA FUNZIONE TIMER

Per ciascun giorno della settimana si possono definire fino a tre programmi o cicli di attivazione e rilascio del relé e quindi di comando dell'utilizzatore da gestire. Ciò significa che per ogni giorno possiamo decidere, ad esempio, di attivare il carico dalle 10.00 alle 10.30, poi un'altra volta dalle 18.00 alle 18.30, quindi, ancora dalle 23.00 alle 24.00.

Nel caso dell'irrigazione, solitamente si attivano le elettrovalvole per un solo periodo al giorno, quindi basterebbe un'unico programma, tuttavia nulla vi impedisce di diversificare l'apertura dell'acqua in diversi momenti.

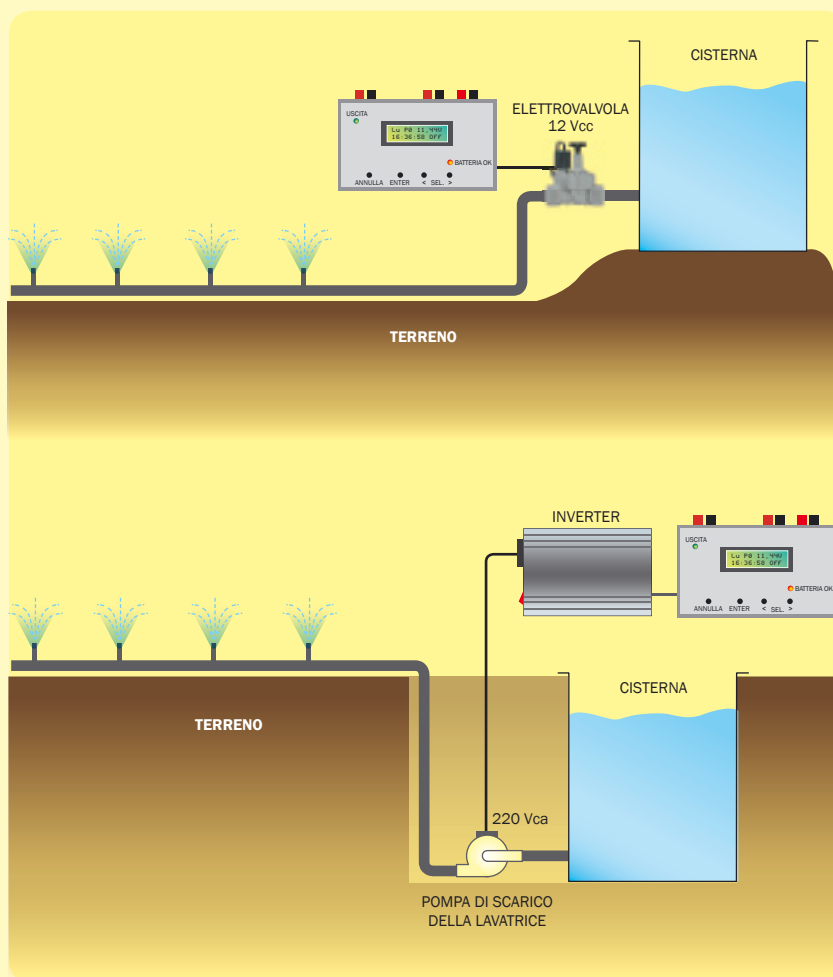
Oltre al controllo automatico, abbiamo aggiunto una funzione di comando esterno: mediante l'ingresso RAIN, attivo a zero logico

E se manca l'acqua corrente?

Oltre che per la casa, la centralina può essere utilizzata per gestire l'irrigazione programmata di orticelli e giardini dove non sia presente la rete elettrica, però si può impiegare anche dove manca la rete di distribuzione idrica: in questo caso l'acqua può essere prelevata da una roggia o da un serbatoio di raccolta dell'acqua

piovana. Per la distribuzione può bastare aprire e chiudere le elettrovalvole se l'acqua è disponibile su un piano rialzato rispetto a quello dove deve avvenire l'irrigazione, mentre se il punto di prelievo è più in basso vi occorre una pompa elettrica, azionabile, quest'ultima, con la tensione prodotta dal sistema; si potrebbe

benissimo impiegare una pompa di scarico della lavatrice, che consuma una cinquantina di watt e funziona 220 Vac, collegandola all'uscita dell'inverter 12/220 V. In questo caso non servono le elettrovalvole, dato che l'acqua può uscire dai getti solo quando la pompa entra in funzione.



(chiuso a massa) si può imporre un evento che, a seconda della configurazione desiderata, può essere l'eccitazione o il rilascio del relé. Questo comando può essere impartito durante l'esecuzione di un programma o una volta sola (*comando manuale*) nell'arco della giornata; a riguardo,

nel menu di impostazione sono previste due voci per definire il comportamento del relé. Durante un programma permette di forzare lo stato del relé quando RAIN viene attivato, mentre al di fuori consente l'attivazione una sola volta in base alle modalità definite nelle impostazioni.

Il comando esterno è riservato al sensore di pioggia: imponendo che l'evento associato alla chiusura dei punti RAIN sia l'apertura del relé, se all'orario previsto dal timer per l'attivazione dell'uscita il sensore rileva pioggia, RL1 resti a riposo e quindi le elettrovalvole non siano alimentate.

Il comando manuale si impartisce sempre con RAIN e permette di definire l'attivazione o il riposo forzato del relé in un certo intervallo nel giorno in corso fuori dai programmi impostati; l'eccezione non si ripete, nel senso che se si imposta per un lunedì, il lunedì seguente non viene più eseguita.

COME FUNZIONA L'INSIEME

Il programma che "gira" nel microcontrollore prevede che, dopo l'inizializzazione degli I/O, venga interrogato il modulo RTC e poi inizializzato e quindi comandato il display, al fine di fargli visualizzare il giorno della settimana e l'ora dedotti proprio dal modulo U2; fatto ciò, il programma legge ciclicamente i quattro pulsanti per vedere se qualcuno di essi viene premuto. Durante l'esecuzione di un programma di temporizzazione, una routine gira in loop testando l'ingresso RAIN,

ossia la linea RB3, per verificare se è forzato il comando esterno; in questa evenienza, il PIC imposta il relé di uscita come stabilito nella configurazione corrispondente (l'impostazione predefinita è *Accensione*, nel senso che, al verificarsi del comando, il relé viene attivato forzatamente). Notate che il relé permane nello stato deciso dalla configurazione fin quando RAIN è chiuso; appena il firmware del microcontrollore rileva il ritorno ad 1 logico della linea R3, lascia che il relé sia gestito in modo normale, ossia esclusivamente dai programmi impostati dall'utente. In tutte le fasi del funzionamento, un'altra routine interroga ciclicamente il modulo RTC per ottenere il refresh dell'ora e del giorno di sistema. Come accennato, il timer prevede la possibilità di impostare fino a tre programmi al giorno, distintamente per ognuno dei sette giorni che compongono una settimana; ogni programma consiste nell'attivazione del relé di uscita ad una determinata ora e nella successiva disattivazione all'ora programmata.

I pulsanti presenti nel circuito hanno le seguenti funzioni, almeno in programmazione: P1=DOWN (torna indietro di un passo o di un minuto nell'impostazione dell'ora o dell'orario di attivazione/spegnimento che si sta programmando); P2=UP (avanza di un passo o di un minuto nell'impostazione dell'ora o dell'orario di attivazione/spegnimento che si sta programmando); P3=ESC (uscita dalla procedura o voce di menu, ovvero ritorno alla precedente voce); P4=ENTER (conferma dato immesso ed eventuale uscita dalla procedura). Ogni volta che si preme un tasto, il cicalino emette una breve nota acustica di conferma.

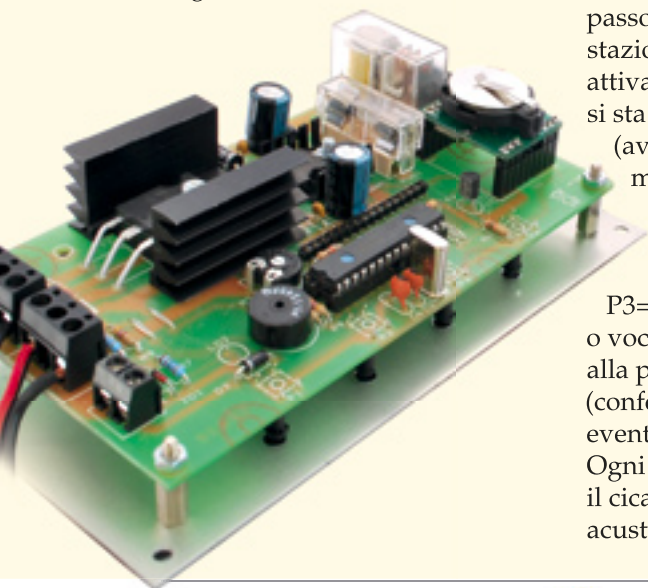
Detto ciò, vediamo come si effettuano le impostazioni del funzionamento.

LA PROGRAMMAZIONE

A riposo, il display mostra nella riga in alto e da sinistra a destra il giorno della settimana abbreviato (**Lu** per lunedì, **Ma** per martedì, **Me** per mercoledì, **Gi** per giovedì, **Ve** per venerdì, **Sa** per sabato e **Do** per domenica) il programma eventualmente in esecuzione e la tensione letta dall'ADC del micro ai capi della batteria; nella riga in basso visualizza l'ora (nel formato ore, minuti, secondi) oltre allo stato del relé (OFF = rilasciato; ON = eccitato). Un esempio è:

*Me P1 13,22V
11:00:52 ON*

dove **Me** sta per mercoledì, **P1** significa che è in esecuzione il primo programma, **ON** che il relé al momento è attivo, 13,22V è la tensione della batteria e 11:00:52 l'ora. Il programma in esecuzione, indica perché il relé è nello stato indicato nella riga in basso; per l'esattezza, **P0** indica che non è in esecuzione alcun programma (timer a riposo), **PM** vuol dire che è in esecuzione un comando manuale, mentre **PI** significa che la condizione di RL1 è dovuta a un comando esterno ricevuto dall'ingresso RAIN. Se è in esecuzione il programma 1, il 2 o il 3, sul display appaiono le scritte, rispettivamente, **P1**, **P2**, **P3**. Per accedere al menu di impostazione del timer occorre premere ENTER (P4) allorché il display mostra la scritta *Impostazioni*. Con UP e DOWN si scorre fra le voci di menu, scegliendo quella riguardante la configurazione da effettuare; premendo più volte UP si scorre fra *Impostazioni*, *Gestione Manuale* e *Visualizza*. Premendo DOWN



le voci si incontrano in ordine opposto.

In *Impostazioni*, si possono vedere giorno per giorno i programmi impostati, che possiamo modificare a piacimento; premendo una volta P4 si accede alla settimana (il display visualizza *Impostazioni Settimana*) e una volta ancora si entra nella programmazione del primo giorno (il display mostra *Lunedì*). Se si intende cambiare il giorno da vedere o modificare, basta premere UP o DOWN: ciò permette di scorrere fra lunedì, martedì, mercoledì, ecc. Per modificare un certo giorno, quando il display ne mostra il nome, bisogna premere P4; appare quindi il numero del programma: con ENTER si conferma che si vuole intervenire su quello, mentre per scorrere fra i tre programmi previsti si usano UP e DOWN. Entrati in un programma, il display propone l'orario di attivazione, scrivendo **Regola Ora ->** e l'orario al quale si intende attivare il relé, ovvero attivare l'irrigazione; per impostare l'ora, bisogna scorrere con UP e DOWN. Una volta impostata l'ora (1÷24) si conferma con ENTER; premendo ESC, si torna alla visualizzazione del giorno della settimana e nulla viene modificato, quindi se volete solamente verificare una certa impostazione, dopo che l'avete vista, premete su ESC e passate oltre. Dopo la conferma della modifica mediante la pressione di ENTER, il display mostra la dicitura

Regola Min -> e ciò vuol dire che vi viene chiesto di definire il minuto al quale deve scattare il relé; notate che ENTER si può premere anche quando si sta verificando un programma, senza modificarlo e, al solito, in tal caso si può uscire premendo due volte ESC (con una volta si torna a **Regola Ora ->**). Confermata una

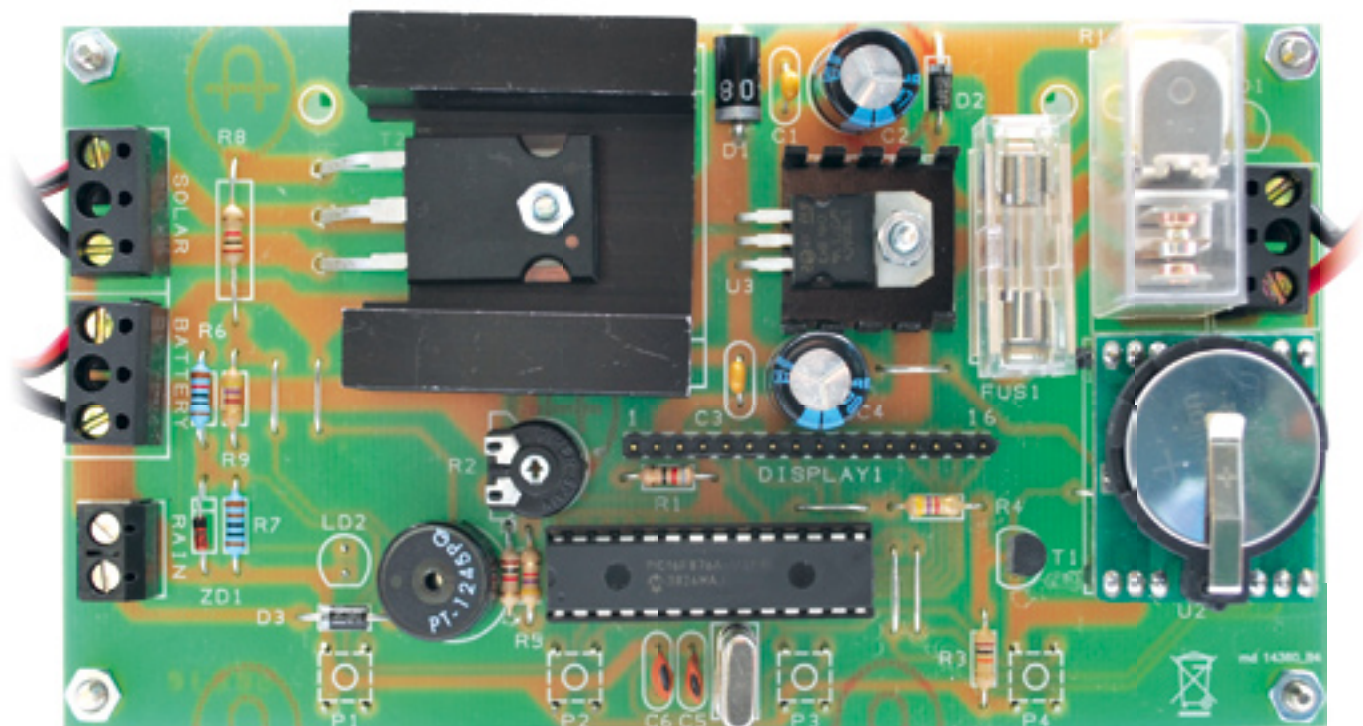
modifica, si accede alla programmazione dell'ora e del minuto di spegnimento (rilascio del relé) che si conduce in maniera analoga a quella appena descritta; il display mostra prima **Regola Ora <-** e poi **Regola Min <-**. Ogni volta che si preme ENTER quando il display visualizza **Regola Min ->** e **Regola Min <-**, avviene la memorizzazione di ore e minuti, confermata dalla comparsa sul

display, per un paio di secondi, della scritta **SALVATAGGIO**. Impostati ora e minuto di OFF, il sistema esce dalla procedura e il display ripropone orario, stato del relé, giorno della settimana e indicazione della condizione di esercizio (P0, PI, PM o P1, P2, P3). Per chiarezza, notate che il simbolo -> significa inizio della temporizzazione e quello <- , fine della temporizzazione stessa,



Il giardino fiorito più grande del mondo

Si trova in un parco in Olanda chiamato *Keukenhof* (letteralmente significa "giardino di cucina") e quest'anno compie il suo sessantesimo anniversario. In questa importante occasione è stato addobbato per commemorare Henry Hudson, capitano del vascello "De Halve Maen" della compagnia olandese delle indie orientali, che mise piede nel 1609 nella odierna Manhattan e fondò New Amsterdam, più tardi ribattezzata New York. Per celebrare questa particolare ricorrenza storica è stato anche allestito all'interno del parco un mosaico floreale formato da più di 50.000 bulbi che rappresentano la statua della libertà. All'ingresso, una sorridente ragazza in abito tradizionale olandese augura una buona visita ed un enorme paio di zoccoli di legno ancorati all'asfalto attende i visitatori per una particolare foto ricordo. All'interno del parco, oltre ai fiori, sono presenti punti ristoro, negozi di souvenir e padiglioni contenenti composizioni floreali in vaso che si avvicendano durante i due mesi di apertura. Tra i servizi offerti ai visitatori, un giro su un ecologicissimo battello elettrico nei canali circostanti il parco.



Elenco Componenti:

R1: 82 ohm
R2: Trimmer MO 4,7 kohm
R3: 10 kohm
R4, R5: 4,7 kohm
R6: 10 kohm 1 %

R7: 1 kohm 1 %
R8, R10, R11: 1 kohm
R9: 470 ohm
C1, C3: 100 nF multistrato
C2: 470 µF 35 VL elettrolitico
C4: 470 µF 16 VL elettrolitico

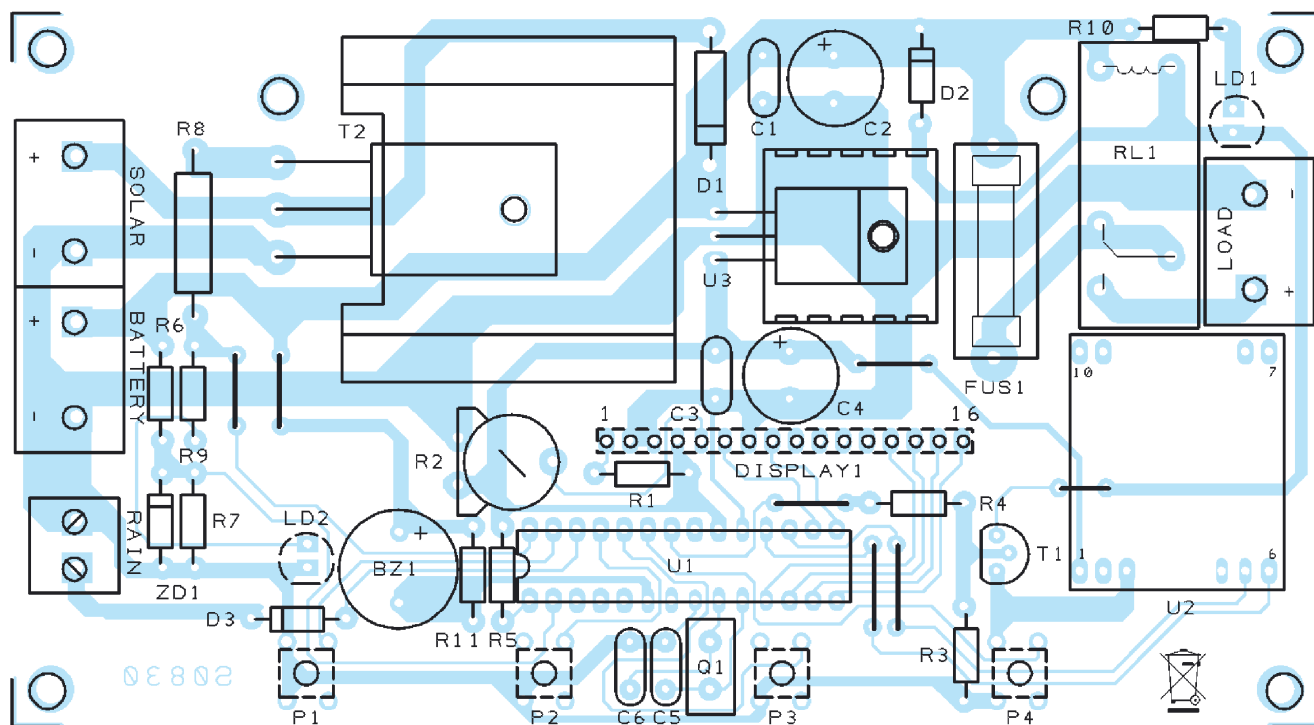
C5: 15 pF ceramico
C6: 15 pF ceramico
D1: BYW80-150
D2, D3: 1N4007
DZ1: Zener 5,1 V 400 mW
U1: PIC16F876A-I/P (MF830)

ovvero del programma. Per effettuare una nuova impostazione su un altro programma o giorno, bisogna passare dalla procedura già esposta: si preme ENTER finché il display non mostra *Impostazioni*, poi lo si preme un'altra volta finché non appare *Impostazioni Settimana*, poi una terza volta fino a vedere *Lunedì*; a questo punto, con UP e DOWN si sceglie il giorno voluto. Quando il display presenta il giorno, con ENTER si entra nella lista dei programmi e il display visualizza *Programma N 1*; se si intende vedere o modificare il programma 1, basta premere ENTER, altrimenti con UP e DOWN si scorre fino a visualizzare il programma voluto,

quindi con ENTER si accede alle impostazioni del caso (o semplicemente si vedono ore e minuti di ON e OFF).

Il menu *Impostazioni* comprende anche altre due voci, che sono *Impostazioni Orario/Giorno* e *Impostazioni Input*; a entrambe si accede scorrendo con UP o DOWN quando il display visualizza *Impostazioni*. La prima permette (premendo ENTER e UP/DOWN le volte che occorre) di impostare l'orologio di sistema, mentre la seconda riguarda la gestione dell'ingresso RAIN. Accedendo a quest'ultima (si entra premendo ENTER quando a display vedete la scritta *Impostazioni Input*) potete

scegliere, usando UP o DOWN, se quando date il comando all'ingresso RAIN il relé debba attivarsi o andare a riposo; la prima evenienza si verifica se fate comparire *Accensione* e confermate, mentre la seconda si ottiene confermando quando a display vedete *Spegnimento*. La conferma si dà con il solito tasto ENTER, allorché la procedura viene chiusa e il display torna a mostrare giorno, ora, ecc. Detto ciò, vediamo le funzioni accessibili dalla voce di menu *Visualizza*: si tratta di un'opzione che riassume, giorno per giorno, le impostazioni attualmente memorizzate per ciascun giorno; premendo ENTER, si accede ai



U2: Modulo RTC (FT674M)
 U3: 7805
 Q1: Quarzo 20 MHz
 P1÷P4: Microswitch
 T1: BC547
 T2: RFG70N06

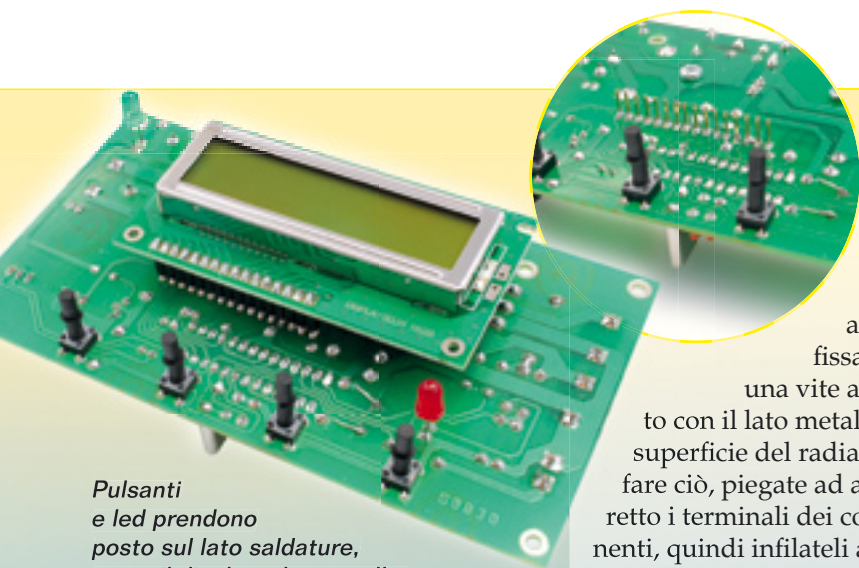
BZ1: Buzzer 12V con elettronica
 LD1: Led rosso 5 mm
 LD2: Led verde 5 mm
 DISPLAY1: Display LCD 16x2
 FUS1: Fusibile 10A
 RL1: Relé 12V 220Vac/10A

Varie
 - Zoccolo 14+14
 - Dissipatore 9,5 °C/W
 - Dissipatore 18 °C/W
 - Dado 3 MA (2 pz.)
 - Vite 12 mm 3 MA (2 pz.)

giorni e il display mostra *Lunedì*. A questo punto, per vedere le impostazioni del lunedì si preme ENTER, altrimenti se si intende visualizzare lo stato di un altro giorno, basta agire su UP e DOWN per sceglierlo. Arrivati al giorno voluto, con ENTER si entra nella visualizzazione: il display mostra la dicitura *Prog 1*, indicando che con una nuova pressione di ENTER si accede alla visualizzazione del primo programma del giorno in questione; per cambiare programma, al solito si usano UP e DOWN. Invece, per visualizzare i dati di un programma si agisce più volte su ENTER, in modo che il display mostri, nell'ordine, ora

di ON e ora di OFF; un'ulteriore pressione di ENTER riporta alla schermata di riposo (giorno, ora, stato del relé e indicazione mattino/pomeriggio). Vediamo, in ultimo, la funzione *Gestione Manuale*, che permette di definire l'eccezione alla programmazione in corso: con *Gestione Manuale* possiamo forzare il timer affinché ecciti il relé indipendentemente da cosa prevede il programma impostato. Entrati nella procedura, ossia quando il display mostra *Gestione Manuale*, si preme ancora ENTER: appare così la scritta *Accendi*; ora si può decidere se il relé deve attivarsi o andare a riposo (*Spegni*). La selezione si effettua mediante i tasti

UP e DOWN, che invertono la scelta ad ogni pressione. Deciso cosa il timer deve fare, un'ulteriore pressione di ENTER porta nell'impostazione corrispondente: se avete scelto *Accendi*, appare **Regola Ora ->**; qui, con UP e DOWN regolate l'ora di attivazione del relé. Invece, se avete optato per *Spegni*, appare **Regola Ora <-**. Dopo la definizione dell'ora, con ENTER si passa alla scelta del minuto (il display mostra **Regola Minuto ->**) se state impostando *Accendi* oppure **Regola Minuto <-** se state lavorando su *Spegni*). Premendo ancora ENTER, il display mostra la scritta **SALVATAGGIO** e poi passa alla definizione degli orari



Pulsanti e led prendono posto sul lato saldature, come il display, che va collegato mediante uno strip da 16 punte.

di accensione (se avete lavorato su *Spegni*) e spegnimento (se avete lavorato su *Accendi*). Ricordiamo che l'opzione *Accendi* significa che, indipendentemente dal programma impostato, all'orario scelto il relé si eccita, mentre *Spegni* determina, all'orario impostato, il rilascio del relé.

LA COSTRUZIONE

Giunti a questo punto possiamo passare alle note costruttive: tutti i componenti prendono posto su una basetta ramata monofaccia da realizzare, preferibilmente per fotoincisione, ricavando la necessaria pellicola da una stampa della traccia lato rame scaricabile dalla sezione download del nostro sito Internet www.elettronicaain.it. Inciso e forato lo stampato, bisogna disporvi i componenti iniziando dalle resistenze e dai diodi al silicio, quindi proseguendo con lo zoccolo per il microcontrollore e le strip femmina per ospitare il modulo RTC e il display LCD; è poi la volta dei pulsanti miniatura da c.s., del trimmer, dei condensatori (prima quelli non polarizzati, poi gli elettrolitici) del cicalino piezo, del quarzo e del regolatore integrato, che dovete montare sdraiato su un dissipatore di calore ad "U" da $18 \div 20$ °C/W. Un altro dissipatore, ma più grande (da non più di 10 °C/W) lo ri-

chiede il mosfet, che va anch'esso fissato con una vite appoggiata con il lato metallico alla superficie del radiatore. Per fare ciò, piegate ad angolo retto i terminali dei componenti, quindi infilateli a fondo nei rispettivi fori e, prima di effettuare le saldature, stringete insieme con una vite più dado da 3 MA, regolatore, dissipatore e stampato; per agevolare il trasferimento del calore, è preferibile spalmare uno strato di pasta al silicone tra la parte metallica del componente e il dissipatore. Per ultimo inserite e saldate il relé e le morsettiere da c.s. a passo 5 mm per i collegamenti di batteria, uscita, ingresso RAIN e pannello solare. Terminate con l'RTC e il PIC16F876, che dovete aver già programmato. Il display va fissato sulla basetta mediante una fila di punte a passo 2,54 mm, alta 15 mm, da stagnare per un'estremità nei fori corrispondenti alle sue connessioni; in alternativa, sul display saldate uno strip femmina da 16 contatti e unite questo e lo strip dello stampato base mediante una fila di punte. La basetta del display e lo stampato base devono essere uniti mediante quattro colonnine esagonali, alte 15 mm. Per tutte le fasi del montaggio, tenete d'occhio i disegni e le foto del prototipo: vi indicheranno come orientare i componenti polarizzati. Completato il montaggio e accertato che tutto sia in ordine, scegliete un contenitore adatto ad ospitare il circuito. Se ritenete di non dover effettuare frequenti ritocchi e considerato che la centralina starà quasi sicuramente all'esterno, potete scegliere un box a tenuta stagna, che va lavo-

rato superiormente allo scopo di creare una finestra dalla quale affacciare il display all'esterno; magari, se volete potete portare i pulsanti fuori dallo stampato e montarli su una delle facce del contenitore. Ai lati, prevedete i fori con i passacavi per i fili diretti all'utilizzatore (elettrovalvole) alla batteria ed al pannello solare: per questi due elementi i collegamenti si possono fare con del filo di rame in guaina (meglio con della piattina rosso-nera in modo da non sbagliare le polarità) da 1 mmq di sezione. Se prevedete di montare il sensore di pioggia, realizzate un quarto passaggio per il cavetto corrispondente o montate su un lato del contenitore una presa jack. Se il vostro impianto inverter e trasformatore, dovete sceglierli per una potenza di poco superiore al consumo delle elettrovalvole usate. ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Elettrovalvole e pompe per irrigazione possono essere acquistati presso i negozi di giardinaggio ed i consorzi agrari. Per quanto riguarda la centralina, il master del circuito stampato ed il firmware col quale programmare il microcontrollore possono essere scaricati dal sito della rivista (www.elettronicaain.it). Il microcontrollore è anche disponibile già programmato (cod. MF830, 12,00 Euro). L'inverter 12Vdc/220Vac 150W (cod. PSI150S) costa 34,00 Euro, il modulo RTC (cod. FT674M) costa 18,00 Euro mentre il sensore pioggia (cod. RN01) costa 8,50 Euro. 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>

4 IN 4 OUT ESPANSIONI & FIRMWARE

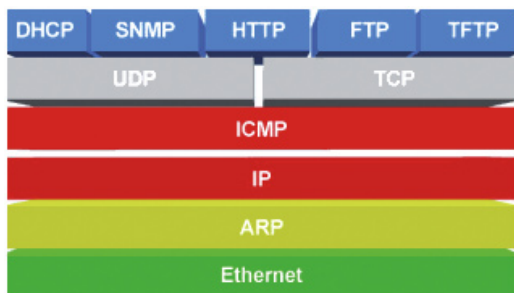
di ROBERTO PRESTIANNI

Dispositivo con interfacciamento Ethernet per la remotizzazione di I/O digitali su rete locale a 10 Mbit e Internet. Seconda puntata: configurazione delle espansioni e firmware.

Nella prima puntata, dopo una breve presentazione del dispositivo ci siamo occupati della descrizione del suo schema elettrico, della sua costruzione e di come eseguire le configurazioni di networking, fondamentali per l'interfacciamento dell'E4I4O con la LAN. In questa seconda ed ultima puntata proseguiremo con la descrizione delle configurazioni sia hardware che software inerenti alle schede di espansione FT473 ed FT488. Ricordiamo che queste due schede, presentate in passato, consentono l'ampliamento del

numero di linee di I/O digitali; ciascuna di esse aggiunge rispettivamente otto uscite a relè e otto ingressi. Come già spiegato, avendo la possibilità di impiegare l'E4I4O da solo (oltre che in coppia), sarà possibile collegare al massimo otto schede che possono essere tutte di un tipo o dell'altro, oppure combinazioni di esse; questo significa, ad esempio, che si potrebbero usare otto





Rappresentazione schematica del modello dello stack TCP/IP della Microchip: notate i livelli che compongono la struttura.

FT488, giungendo così a 68 linee, $8 \times 8 + 4 = 68$ (64 di espansione più quattro native dell'E4I4O). Oltre a spiegare come usare le espansioni, descriveremo il codice sorgente e quindi l'implementazione dell'applicazione basata sull'uso dei protocolli del TCP/IP scritti dagli sviluppatori della Microchip. A questo proposito va sottolineato il fatto che questo articolo non vuole assolutamente essere una descrizione esaustiva di tale stack, bensì una finestra sull'insieme delle potenzialità e caratteristiche offerte dalla combinazione di un microcontrollore PIC e dello stack fornito dalla Microchip. Ci si occuperà del livello applicativo dello stack che realizza le funzionalità custom che caratterizzano

l'E4I4O e che fa affidamento sul protocollo TCP. Sarà allora mostrato il meccanismo che consente l'interazione tra le pagine web memorizzate sulla EEPROM esterna e le risorse hardware del microcontrollore (in questo caso le linee di I/O), nonché la conversione dei listati HTML nel file *MPFSImg2.BIN*. Vedremo inoltre come servirci di un access-point wireless di tipo WiFi (Wireless Fidelity) per usare senza fili il nostro dispositivo, liberandoci così dalle limitazioni fisiche imposte dall'uso dei cavi ethernet.

CONFIGURAZIONI PER LE SCHEDE DI ESPANSIONE

Come anticipato, l'uso opzionale delle espansioni di I/O richiede alcune semplici configurazioni

preliminari, sia hardware che software: le prime sono necessarie per assegnare a ciascuna espansione un proprio indirizzo fisico; le configurazioni software, invece, vanno eseguite tramite l'apposita pagina web *Espansione I/O* ([expander.htm](#)). L'uso dell'E4I4O in stand-alone richiede una configurazione particolare, dal momento che la combinazione degli input di una scheda FT488 connessa al dispositivo "A", deve essere riportata esattamente sulle linee di output di una scheda FT473 connessa al dispositivo "B", e viceversa. Per meglio comprendere la procedura di configurazione, consideriamo un semplice esempio in cui due E4I4O (che chiameremo convenzionalmente "A" e "B"), vengono connessi alla stessa LAN (una volta configurati a dovere secondo le indicazioni fornite nella prima puntata). Una scheda di input FT488 verrà connessa ad "A", ed una di output FT473 sarà collegata a "B". L'obiettivo, quindi, è quello di remotizzare le otto linee di input della scheda



Controllare l'ambiente domestico con il pensiero non è fantascienza: lo dimostra la prima casa domotica controllata dagli impulsi cerebrali realizzata nei laboratori della *Fondazione Santa Lucia*. Una cuffia munita di elettrodi permette di captare

La casa esaudisce i desideri

l'onda P300 emessa dal cervello quando un oggetto cattura la nostra attenzione; un computer legge gli impulsi e invia il comando per azionare il dispositivo cui stiamo pensando o far muovere un piccolo robot. Lo sviluppo dell'ambiente domotico conosce in queste ricerche la sua applicazione più all'avanguardia, volta principalmente a migliorare la vita di chi versa in gravi difficoltà. Ma lo sviluppo di "case intelligenti" e lo studio delle tecnologie atte a migliorare la qualità degli spazi

abitativi propone anche soluzioni accessibili a tutti e di più largo consumo. Immaginate di uscire di casa pensando a quello che dovrete fare prima di varcare l'uscio: automaticamente si spengono le luci, si abbassano le tapparelle e vengono controllate tutte le finestre, si abbassa il riscaldamento, si attiva l'antifurto, viene tolta corrente alle prese di tutti gli apparati audio e video. Simili applicazioni domotiche servono anche ad aumentare gli standard di sicurezza e ad ottimizzare al massimo i consumi.



di espansione connessa ad "A" (più le quattro native dello stesso modulo), su "B", più esattamente sulla scheda di espansione di output ad esso connessa. Iniziamo allora con l'assegnazione degli indirizzi alle due schede di espansione. Sia la FT473 che la FT488 presentano tre jumper chiamati J3, J2, J1 (J3 è l'MSB, ossia Most Significant Bit), ognuno rappresentante uno dei tre bit dell'indirizzo (con tre bit si possono configurare otto indirizzi). Lasciando aperto uno di tali jumper, il relativo bit di indirizzo viene settato ad "1"; per contro, chiudendo il jumper il bit corrispondente è 0. Per fare una prima prova, supponiamo di voler assegnare l'indirizzo 1 alla scheda di input da connettere al dispositivo "A": J3 e J2 andranno chiusi mentre J1 va lasciato aperto (Indirizzo = 001). Lo stesso identico indirizzo va impostato sulla scheda di output che andrà collegata a "B": in questo modo il sistema assocerà automaticamente le due schede di espansione. Adesso dobbiamo però dire a ciascun E4I4O a quale indirizzo sarà presente la scheda di espansione che gli si andrà a collegare. Una volta eseguito il login tramite l'apposita pagina, cliccate sul link Espansione I/O ed entrerete così nelle configurazioni delle schede di espansione. La prima tabella è quella che ci consente di definire il tipo di I/O relativamente a ciascuno degli otto indirizzi (EXP7/... EXP0). Per quanto riguarda il

modulo "A", in corrispondenza dell'indirizzo EXP1 (il numero 1 è l'indirizzo della scheda) e tramite il relativo menu a tendina, selezioneremo "IN"; gli altri indirizzi li lasceremo tutti in "OUT", anche se effettivamente a questi non sono associate schede di espansione. Fatto ciò bisogna cliccare sul pulsante "Configura" per rendere effettiva l'impostazione. Questa configurazione è fondamentale affinché il microcontrollore venga informato della presenza di una scheda di input su un dato indirizzo, così che possa interrogarla periodicamente e rilevare quindi eventuali modifiche sulle sue linee di ingresso. Nel caso del modulo "B" invece, dal momento che si andrà a connettere una scheda di output, la configurazione della prima tabella dovrà essere lasciata inalterata. Ricordiamo che è consentito consultare e comandare sia "A" che "B" contemporaneamente, ovvero è possibile accedere alle relative pagine web da un'unica sessione del browser; la possibilità di assegnare un nome a ciascun dispositivo E4I4O (ad esempio "Modulo A" e "Modulo B") ci consente dunque di individuare ed accedere agevolmente dallo stesso browser, sia all'una che all'altra scheda (vedere Fig. 1). Ter-

minata questa prima configurazione, potremo connettere le due espansioni, una per ogni modulo E4I4O (secondo quanto previsto dall'esempio).

Adesso consigliamo di eseguire alcuni test preliminari: innanzitutto verifichiamo che il modulo "A" sia in grado di leggere tutti gli input, quindi isoliamo il modulo "B" dal sistema sconnettendolo fisicamente dalla LAN. Assegniamo dei valori logici a questi ingressi (cortocircuitandoli o lasciandoli aperti se desideriamo assegnare, rispettivamente 0 o 1 logico) e, sempre dalla pagina Espansione I/O del modulo "A", tramite la seconda tabella (quella relativa agli Input-Expander), selezioniamo l'indirizzo "1" e facciamo clic su "Seleziona": immediatamente apparirà la configurazione degli

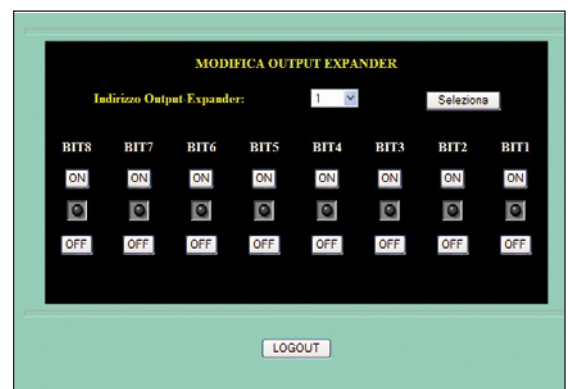
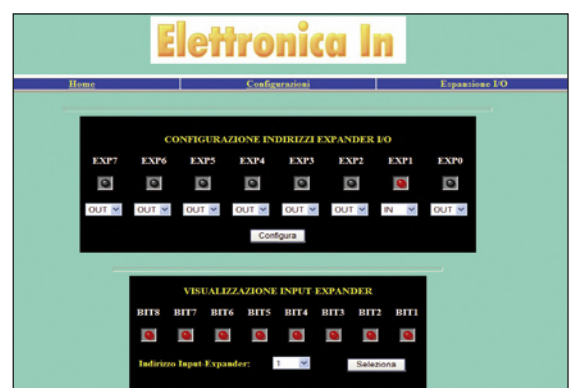


Fig. 2 - Pagine di configurazione e di controllo delle schede di espansione di I/O.



Usiamo un **access-point wireless**

Leggendo il paragrafo relativo allo stack Microchip, comprendiamo come il firmware che gira nel PIC implementa i protocolli di comunicazione a partire dall'IP (livello 3 - Rete), lasciando i livelli sottostanti all'hardware e quindi al controller ethernet integrato.

Questo significa che, in virtù della standardizzazione dei protocolli di rete e della loro struttura gerarchica, il microcontrollore è in grado di comunicare non solo grazie all'ethernet ma anche sfruttando tutti quei canali di comunicazione utilizzati dalla rete informatica, LAN o Internet. Ciò, ovviamente, richiede apposite interfacce che instradino il traffico Ethernet dell'E4140 da e verso altri mezzi fisici, come appunto il wireless.

Questa puntualizzazione potrebbe sembrare banale, ma ribadisce il fatto che una trasmissione basata sui protocolli standard dello stack TCP/IP viene svincolata dalla tecnologia ed i mezzi di trasmissione fisica usati, almeno ai livelli oltre il secondo. Ritornando al wireless, in questo caso la soluzione più semplice e affidabile prevede l'uso di un access-point di tipo WiFi, oggi sempre più

utilizzato e acquistabile a prezzi più che accessibili. La mansione più comune di tale apparecchio radio è la creazione di reti Wireless Area Network, ossia reti, private o pubbliche, senza fili. Gli access-point sono dotati di almeno una porta ethernet, con la quale possono essere messi in collegamento con la LAN o, nel nostro caso, direttamente con l'E4140; dunque, il nostro modulo andrà collegato direttamente alla porta ethernet dell'access-point tramite un cavo non incrociato, una volta configurati i suoi parametri di networking da PC. Qui sotto riportiamo un tipico esempio di collegamento che prevede l'uso di un solo E4140 per la lettura in remoto di linee digitali di input e il controllo di quelle di output, tramite PC provvisto di interfaccia WiFi. Supponiamo che il nostro elaboratore sia già connesso alla rete wireless creata dal nostro access-point;

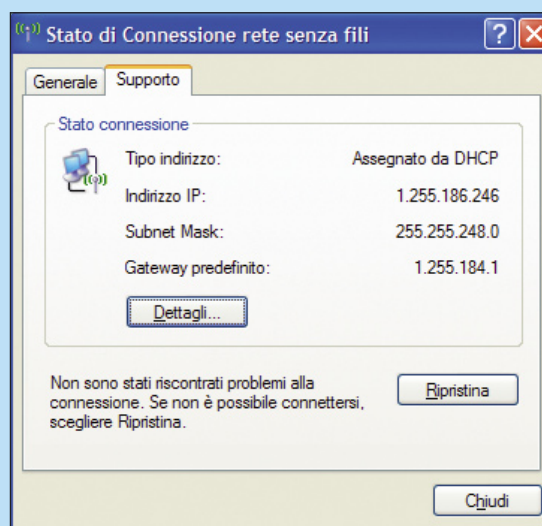


Fig. 6
Esempio di
parametri
di rete
di un
computer
connesso
ad un
Access
Point
Wireless.

basterà allora ottenere dal sistema operativo le impostazioni di rete relative a questo collegamento wireless, e memorizzarle sull'E4140 tramite l'apposita pagina *Configurazioni*. Andando sulle proprietà di rete (vedere Fig. 6) bisogna leggere il numero della "Subnet Mask" e l'indirizzo

IP del "Gateway predefinito" (l'access-point...) e riportarli così come sono sulla pagina di configurazione. Per quanto riguarda l'indirizzo IP da assegnare all'E4140, bisogna specificarne uno simile a quello del proprio computer, ma che differisca



ingressi applicati alla scheda FT488 (quella connessa al modulo "A").

Le spie luminose della tabella le vedremo accese se il bit è ad 1 logico e spente se, invece, si trova allo stato 0.

A questo punto siamo certi che il modulo "A" piloti correttamente la propria espansione di I/O, quindi passiamo alla verifica della scheda di output, che andrà connessa al dispositivo "B". Disconnettiamo ora "A" e ricollegiamo "B" alla LAN. Dalla pagi-

na *Espansione I/O* di quest'ultima e tramite la terza tabella (output-Expander) selezioniamo l'indirizzo 1. Cliccando sul pulsante "Seleziona" vedremo spegnersi gli otto led e quindi gli altrettanti relé della scheda; questo evento ci darà conferma della corretta comunicazione tra il modulo "B" e la sua scheda di output. Adesso, facendo clic sugli appositi pulsanti di "ON" e di "OFF" potremo modificare a piacere ciascuna linea. Notate che se selezionando una periferica di I/O

dalla pagina ESPANSIONE I/O appare il messaggio "**** ERRORE ****", significa che al relativo indirizzo non corrisponde una scheda di input o di output. Bene, a questo punto il sistema è pronto a funzionare, quindi ricollegando il modulo "A" alla LAN e modificando una o più linee di input, nel giro di pochi secondi dovremo vedere modificarsi lo stato dei relé del modulo "B". Nel caso in cui ciò non dovesse accadere, vi consigliamo di verificare le configurazioni di

per la sola ultima tripletta. Dal momento che questa assegnazione è del tutto arbitraria, poiché non realizzata in automatico dall'DHCP (Dynamic Host Configuration Protocol), per evitare di innescare conflitti bisognerà prestare attenzione a non utilizzare indirizzi già impegnati. Nella Fig. 6 viene mostrato un esempio in cui l'access-point ha assegnato l'indirizzo IP 1.255.186.246 al nostro computer, quindi si potrebbe assegnare al nostro dispositivo un IP 1.255.186.50, sempre che questo indirizzo non sia già impegnato da un altro host-computer. Una verifica in tal senso può essere fatta usando la funzionalità di rete "Ping". Un'altra alternativa all'uso di un semplice access-point è l'uso di un router ADSL WiFi; questa soluzione potrebbe risultare più semplice e pratica in quanto oggi sempre più compagnie telefoniche che offrono la connessione ad Internet danno in comodato un router ADSL WiFi. In questa eventualità, la configurazione è del tutto identica a quella dell'access-point, dal momento che il router è l'integrazione di un router ADSL e di un access-point WiFi. L'unica accortezza da avere riguarda l'eventualità che i parametri di rete dell'interfaccia ethernet del router possano essere differenti da quelli della rete wireless. Per ultimo facciamo presente che se adiamo a configurare adeguatamente il router ADSL, saremo in grado di "affacciare" su Internet l'E4I4O e quindi di pilotarlo e consultarlo da qualsiasi punto del mondo tramite Internet.

networking, badando particolarmente all'indirizzo IP del "modulo remoto" e al numero delle porte di comunicazione TX ed RX (vedere prima puntata); se non basta, spegnete e riaccendete i due E4I4O. Ribadiamo ancora una volta che le linee di I/O a bordo dell'E4I4O e quelle delle schede di espansione funzionano in maniera del tutto analoga; cambia solamente il modo con cui il sistema le gestisce. Sicuri della corretta remotizzazione dal modulo "A"

al modulo "B", si potrebbero adesso aggiungere altre schede che funzionino in maniera speculare: un'espansione di ingressi da

connettere a "B" ed una di uscite da collegare ad "A", entrambe da configurare col medesimo indirizzo hardware.

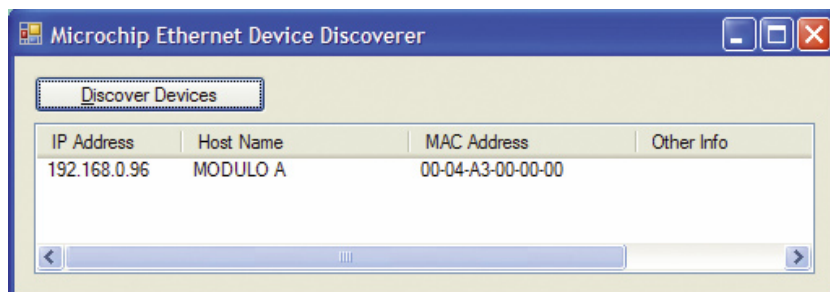


Fig. 3 - Interfaccia del tool "Ethernet Microchip Device Discoverer".

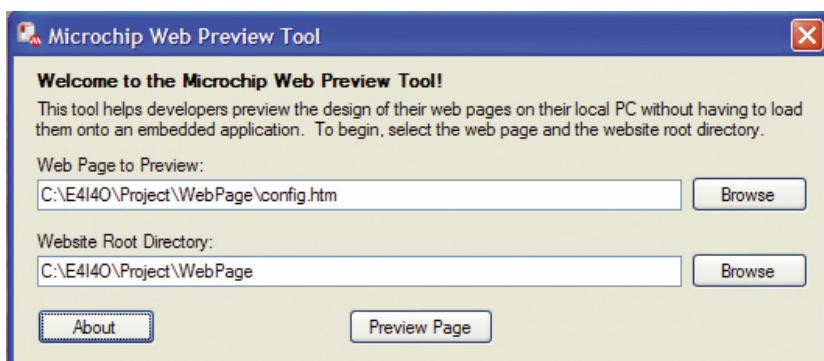


Fig. 4 - Interfaccia del visualizzatore "Web Preview".

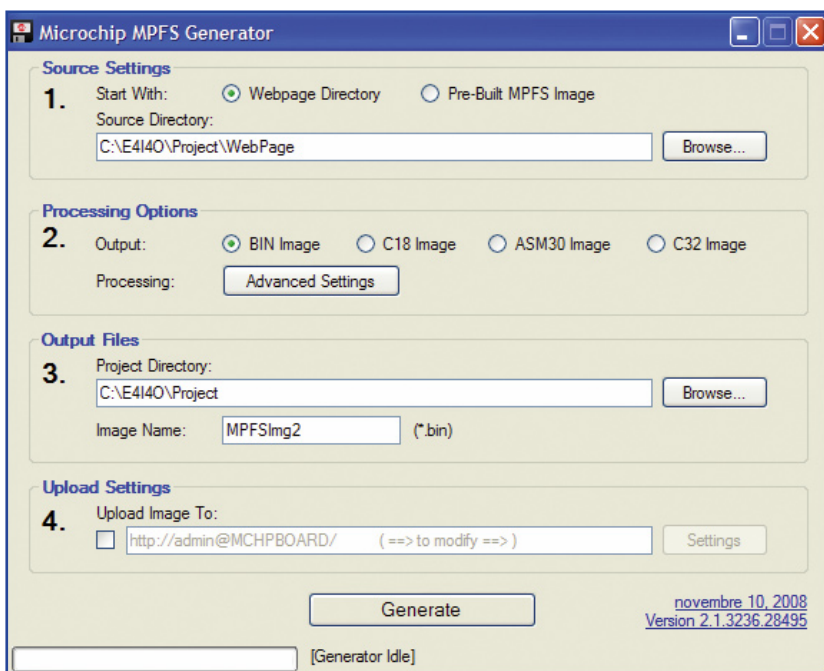
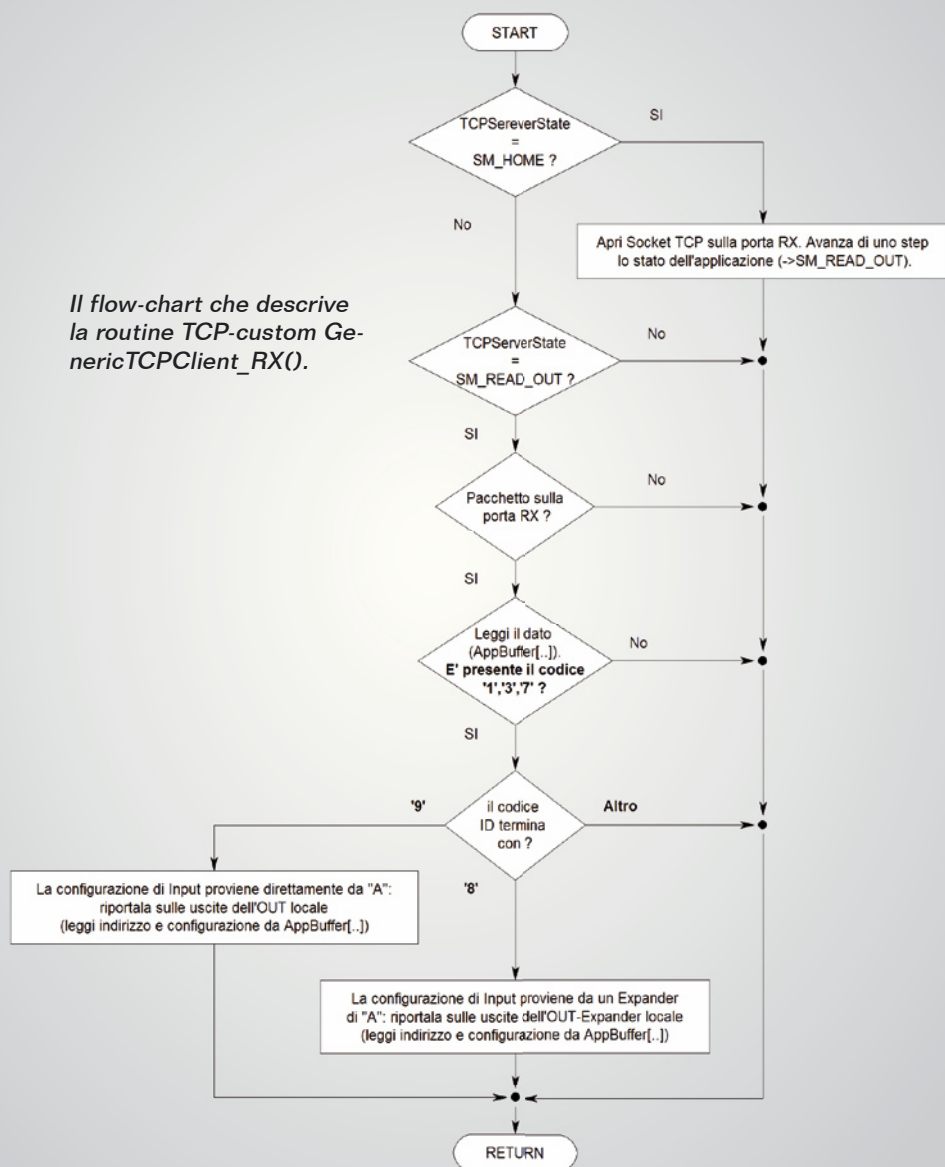


Fig. 5 - Interfaccia del tool "MPFS Generator".

Il flow-chart che descrive la routine TCP-custom GenericTCPClient_RX().



LO STACK TCP/IP MICROCHIP

In questo paragrafo sono richiamati alcuni importanti argomenti inerenti alla comunicazione in rete, come lo stack dei protocolli secondo il modello standard ISO/OSI (International Organization of Standardization/ Open Systems Interconnection), nonché la versione sviluppata dalla Microchip per i suoi microcontrollori (le famiglie PIC18, PIC24, dsPIC e PIC32); i concetti vi serviranno a comprendere come il firmware sia in grado di consentire all'E4I4O di interfacciarsi e comunicare in rete. Dalla teoria della comunicazione in rete di calcolatori, apprendiamo come i diversi protocolli

sono organizzati e strutturati secondo una gerarchia definita dal modello ISO/OSI: la catasta dei sette livelli previsti costituisce lo Stack TCP/IP. Ciascuno di questi livelli gerarchici lavora per servire il livello immediatamente superiore, occupandosi quindi dell'elaborazione di una specifica parte del lavoro di computazione svolto dall'intero stack. In ordine dal più basso a quello più alto, i sette livelli sono: *Physical*, *Data Link*, *Network*, *Transport*, *Session*, *Representation* e *Application*. Il livello *Physical* si occupa della comunicazione fisica attraverso un dato canale di comunicazione e secondo una particolare tecnologia. Salendo via-via di livello,

giungiamo al terzo, *Network*, che si occupa degli indirizzamenti, dell'organizzazione dei pacchetti, del routing eccetera. In altre parole, questo livello si preoccupa di rendere indipendenti quelli superiori dalla tecnologia e dal canale di comunicazione usato per i collegamenti di rete.

Al quarto livello risiedono i protocolli che consentono il trasporto dei pacchetti, vedendo solamente il punto di partenza e quello di arrivo; tutte le tappe intermedie sono organizzate dai livelli sottostanti, in particolare modo dal terzo e quindi dall'IP. Nel quarto livello risiedono l'UDP (User Datagram Protocol) e il TCP (Transfer Control Protocol). Quest'ultimo, utilizzato per la comunicazione dell'E4I4O, è il protocollo di trasporto dei dati più affidabile e sicuro, ed assolve al compito della frammentazione dei dati provenienti dai livelli applicativi e della gestione ottimale delle risorse di rete, al fine di evitare potenziali congestioni. L'ultimo livello è quello dell'applicazione e fornisce un determinato servizio usando la comunicazione in rete (Internet o LAN). La Microchip ha realizzato il suo stack TCP/IP che, periodicamente, aggiorna apportando migliorie e nuovi servizi interessanti. Ad esempio, nella penultima versione (v4.55) vengono implementati anche l'SMTP (Simple Mail Transfer Protocol) per la spedizione di e-mail ai server di posta elettronica, l'SSL (Secure Sockets Layer) per la crittografia di dati scambiati in rete, l'autenticazione mediante username e password per l'accesso alle pagine web, l'uso del servizio DynDNS (Dynamic Domain Name System) per l'aggiornamento automatico dell'IP globale, i Cookie, la criptazione MD5 (Message Digest algorithm

5) per l'upload dei file di configurazione di alcune applicazioni e molto altro.

La versione 4.55, impiegata in questo progetto e che potete scaricare dal sito della rivista, risale al novembre del 2008; chi di voi fosse interessato alla nuova versione (v5.00) pubblicata il 27 aprile di quest'anno, la troverà sul sito della Microchip alla pagina internet <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026439>, e alla voce Software.

Il file zippato contenente lo stack della Microchip contiene, oltre ad alcuni project relativi ad alcune schede dimostrative, anche dei semplici tool di supporto molto utili, ed alcuni di questi verranno

ora descritti. Uno di essi è l'*Ethernet Microchip Device Discoverer*, che consente di rintracciare tutti i moduli E4I4O presenti nella propria LAN e, più in generale, tutti i dispositivi ethernet Microchip. Come mostrato nella **Fig. 3**, la finestra del programma mostra, oltre che l'indirizzo IP di un E4I4O, anche l'Host Name, il MAC Address più altre info di secondaria importanza.

L'*Ethernet Discoverer*, che in ambiente Windows XP troverete seguendo il percorso *Start → Tutti i programmi → Microchip → TCP/IP Stack v4.55 → Ethernet Discoverer*, una volta installato lo stack è in grado di eseguire questi rilevamenti anche quando il computer che stiamo usando è

collegato alla LAN via wireless. Un'altra applicazione importante è l'*MPFS Generator*, che troverete seguendo lo stesso percorso dell'*Ethernet Discoverer*. Questo programma converte tutto il materiale HTML (e quindi tutte le pagine web), nel file binario *MPFSImg2.BIN* che andrà caricato sulla EEPROM esterna, così come descritto sulla prima puntata.

Come mostra la finestra di dialogo illustrata nella **Fig. 5**, al punto 1 viene specificata la cartella contenente le pagine HTML (WebPage) e quindi la sua posizione, al punto 2 viene richiesto il formato binario del file (BIN Image), mentre al punto 3 viene specificata la cartella di destina-

Diagramma di flusso che descrive la routine *GenericTCPClient_TX()*.

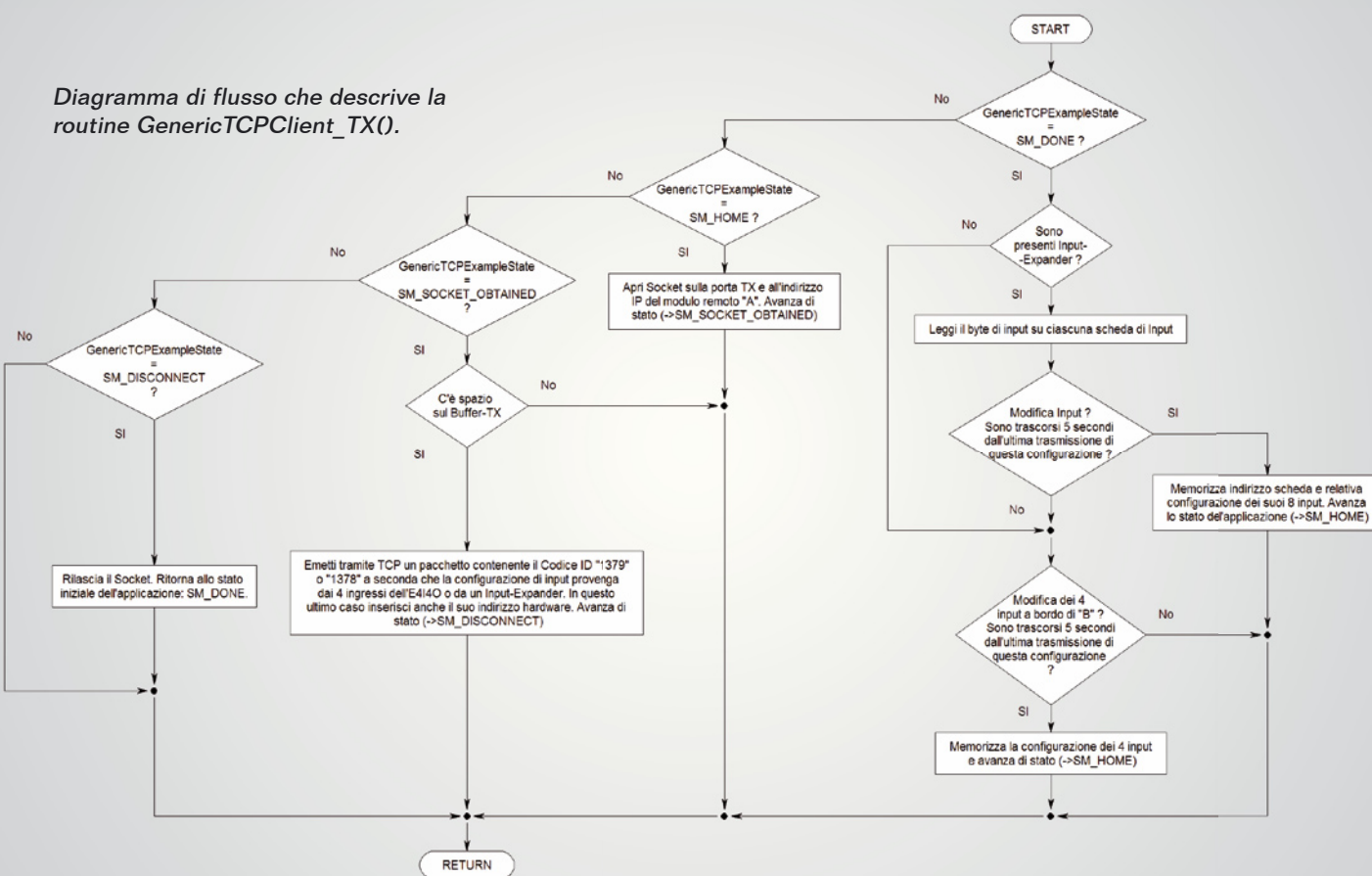
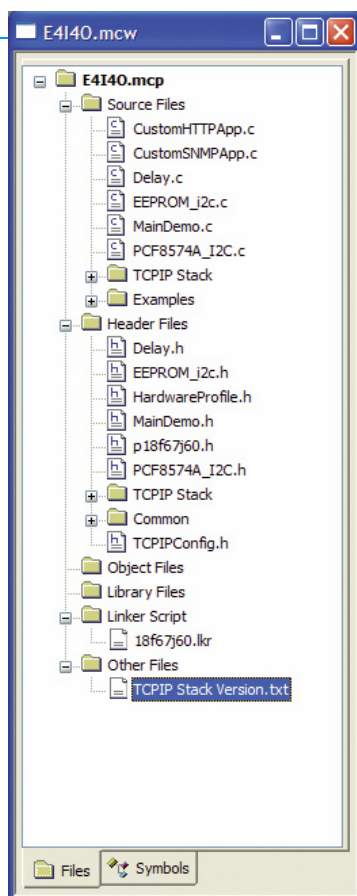


Fig. 7
Struttura
del Project
MPLAB.



zione del file .BIN (in questo caso è la stessa del project), nonché il suo nome. Infine, facendo clic sul tasto "Generate" il programma creerà il file voluto e stamperà a video una seconda finestra con alcune informazioni inerenti alla conversione.

Scaricando dal sito della rivista il project completo, noterete all'interno della cartella *Project* il file *MPFSImg2.BIN*; per quanto riguarda il suo nome, si è preferito mantenere quello predefinito, cosicché possiate rintracciarlo agevolmente tra tutti i file contenuti nello stack Microchip originale. Questo aspetto tornerà particolarmente utile a coloro che intendano personalizzare a proprio piacere le pagine Internet dell'E4I4O. A tal proposito ricordiamo che il file viene trasferito nella scheda in remoto semplicemente da uno degli host-computer presenti sulla stessa rete del modulo: basterà infatti caricare la pagina [*pload* \(nell'esempio 192.168.0.95 è l'indirizzo IP del dispositivo\) e specificare il nome ed il percorso del file; una volta premuto il tasto "Upload", computer e scheda inizieranno lo streaming e, contemporaneamente, il caricamento sulla EEPROM del file binario \(tramite comunicazione seriale PC\). Un altro degli applicativi che possono tornare utili è il "Web Preview", che consente una rapida visualizzazione di una pagina web sviluppata per il dispositivo Ethernet. Come mostrato nella Fig. 4, basta inserire il file .htm di quella pagina e la cartella che lo contiene.](http://192.168.0.95/mpfsu-</p>
</div>
<div data-bbox=)

IL FIRMWARE

Passiamo, adesso, alla descrizione del firmware: per prima cosa faremo una panoramica sull'intero progetto, del quale mostreremo la struttura generale e analizzeremo alcuni dei file, dopodiché proseguiremo con la descrizione di una serie di listati estratti sia dai file, sia dal codice HTML utilizzato nelle pagine web; concluderemo con lo studio dell'applicazione custom TCP, che realizza la remotizzazione delle linee di I/O tra due E4I4O. Per comprendere bene il meccanismo della remotizzazione degli input digitali, osservate innanzitutto i due flow-chart relativi alle routine *TCP-customGenericTCPClient_TX()* e *GenericTCPClient_RX()*.

Installando lo stack TCP/IP verrà creata la cartella *Microchip Solutions* nell'unità C (se non si specifica un'altra unità), al cui interno troverete poi diversi progetti demo stilati per l'MPLAB in linguaggio 'C' e catalogati in sottocartelle, ognuna di esse con i propri file .c e .h. Questi codici realizzano il firmware di alcune schede Microchip dimostrative, orientate alla comunicazione in

rete. Il progetto customizzato per la realizzazione del nostro dispositivo ethernet risiede nella cartella *TCPIP Demo App* (C:\Microchip Solutions\TCPIP Demo App). Qui troverete una serie di project MPLAB per diverse famiglie di PIC, ma quella di nostro interesse è il TCPIP Demo App-C18 (il microcontrollore utilizzato dall'E4I4O è il PIC18F67J60). Da questo project sono stati rimossi alcuni file non importanti ai fini del progetto, mentre altri sono stati ripuliti del codice superfluo e personalizzati secondo le specifiche dell'E4I4O, oppure aggiunti. Il risultato di queste modifiche le troverete sul project E4I4O scaricabile dal nostro sito. Nella Fig. 7 potrete osservarne la struttura, così come la presenza dei file *Delay.c*, *EEPROM_I2C.c* e *PCF8574A_I2C.c* nella cartella

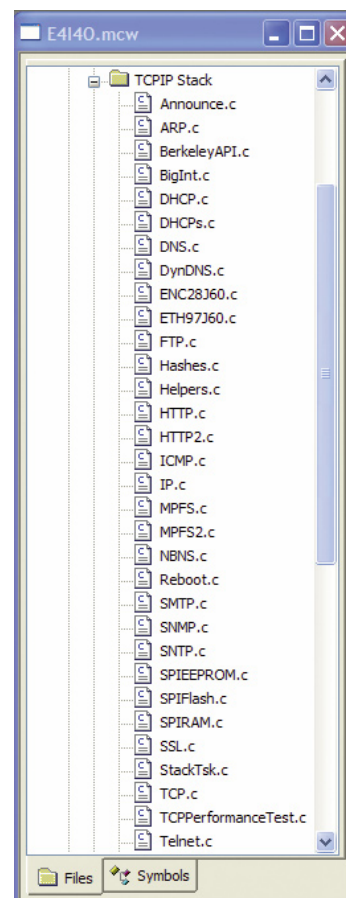


Fig. 8
Alcuni
file della
cartella
TCPIP
Stack.

Fig. 9
Assegnazione
di un nuovo
indirizzo IP.

Source Files e i relativi file .h nella Header Files. Il primo definisce alcune funzioni di ritardo impieganti i timer del micro, *EEPROM_i2c* realizza la comunicazione seriale per mezzo del modulo I²C integrato, mentre *PCF8574A_I2C* implementa la comunicazione con l'IC PCF8574A, necessaria al controllo delle schede di espansione, impiegante anche questa l'IC del PIC. Entrambe le cartelle Source e Header Files contengono a loro volta una sottocartella TCPIP Stack contenente tutti i file primari (e non solo) in cui sono definiti e dichiarati i protocolli dello stack e le loro specifiche routine (vedere la Fig. 8).

I file che ora andremo ad analizzare e che risultano di vitale importanza per il nostro firmware sono: *TCPIPConfig.h*, *MainDemo.c*, *HardwareProfile.h*, *CustomHTTPApp.c*, *HTTP2.c*, *HTTPprint.h*, *HTTP2.c* e *GenericTcpClient.c*. Osservate innanzitutto il listato del file *TCPIPConfig.h*: al suo interno vengono attivati e disattivati i moduli funzionali che realizzano le applicazioni necessarie al nostro sistema, semplicemente decommentando o commentando le linee di codice del tipo `#define STACK_USE_XXXX` dove al posto di XXXX sono riportati i nomi dei moduli. Tra quelli attivi abbiamo: *ICMP_SERVER* (e *CLIENT*) (*Internet Control Message Protocol* – quindi `#define STACK_USE_ICMP_SERVER`) essenziale al funzionamento del Ping, *HTTP2_SERVER* per la consultazione delle pagine web

(un applicazione HTTP), *GENERIC_TCP_CLIENT_EXAMPLE* che autorizza la *main()* alla chiamata delle routine per le applicazioni custom TCP (vedere nella *main()* il codice `#if defined(STACK_USE_GENERIC_TCP_CLIENT_EXAMPLE).....`). Tra le funzionalità disattivate ritroviamo: *HTTP_SERVER*, che rappresenta la primissima versione implementata da Microchip per la consultazione delle pagine web, DHCP per l'assegnazione automatica dell'indirizzo IP da parte della rete, SMTP per la trasmissione di e-mail e ancora DNS, Telnet eccetera. Seguono poi altre definizioni a visibilità globale come quella dell'MPFS (per il trasferimento del file .BIN), il tipo di EEPROM esterna (nel particolare la 24LC1025), e ancora le configurazioni di networking di default che ritroviamo nel sistema dopo la programmazione del PIC ed il reset, eccetera.

La *MainDemo.c* contiene innanzitutto la *main()* del progetto, in più altre routine di supporto quali la *InitAppConfig()* per la

configurazione iniziale dei parametri di networking (e non solo) oppure quelli precaricati sulla EEPROM, la *InitializeBoard()* che configura l'oscillatore, i timer ed altri moduli del micro e inizializza le linee di out e, per concludere, la *Reset_system()*, che viene chiamata tutte le volte che l'utente desidera ripristinare i parametri iniziali, una volta premuto l'apposito pulsante. La *main()* ha inizio con una immancabile fase di inizializzazione sia hardware che software (quella relativa ai protocolli e i task che ne fanno uso). Ecco che allora sono eseguite la *InitializaBoard()*, la *MPFSInit()* relativa alla procedura di mpfsupload del file binario, la *InitAppConfig()*, e poi la *StackInit()*, una funzione definita sul file *StackTsk.c* che inizializza i moduli TCP, HTTP, ed altri ancora sempre se attivati da listato *TCPIPConfig.c* (ad esempio MAC, ARP, UDP, SSL eccetera). Dopodiché la *main()* prosegue con il corpo principale, che viene reiterato continuamente. Al suo interno ritroviamo il cronometraggio dell'attesa per il rilascio della sessione di consultazione remota tramite pagine web (5 minuti), il lampeggio di LED0 ogni secondo, la chiamata della *Reset_system()* alla pressione del pulsante, ed altre azioni di maggiore importanza che adesso

Listato 1 Il form HTML che restituisce il nuovo indirizzo IP tramite query-string.

```
<form method="get" action="config.htm">
<td width="5%"></td>
<td><div align="center"><input type="text" name="ipadd0" maxlength="3" size="1" /></div></td>
<td><div align="center"><font color="yellow">.</font></td>
<td><div align="center"><input type="text" name="ipadd1" maxlength="3" size="1" /></div></td>
<td><div align="center"><font color="yellow">.</font></td>
<td><div align="center"><input type="text" name="ipadd2" maxlength="3" size="1" /></div></td>
<td><div align="center"><font color="yellow">.</font></td>
<td><div align="center"><input type="text" name="ipadd3" maxlength="3" size="1" /></div></td>
<td><div align="center"><font color="yellow">.</font></td>
<td><div align="center"><input type="submit" value="Salva"></div>
</form>
```

```

//***** config.htm *****
if(!memcmppgm2ram(filename, "config.htm", 10))
{
    if(login_ok == TRUE)        tick_login = TickGet(); //Reset Timeout
    //***** Modifica Indirizzo IP *****
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"ipadd0");
    if(ptr)
    {
        if((*ptr)!='\0')
        {
            IPnum0=ASCIIToNumeric(ptr); //Modifica effettiva della tripletta IP
            AppConfig.MyIPAddr.Val &= 0xFFFFF00;
            AppConfig.MyIPAddr.Val |= ((DWORD)IPnum0);
            AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
            SaveAppConfig(); //Salvataggio nuovo IP
        }
    }
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"ipadd1");
    if(ptr)
    {
        if((*ptr)!='\0')
        {
            IPnum1=ASCIIToNumeric(ptr); //Modifica effettiva della tripletta IP
            AppConfig.MyIPAddr.Val &= 0xFFFF00FF;
            AppConfig.MyIPAddr.Val |= (((DWORD)IPnum1)<<8);
            AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
            SaveAppConfig(); //Salvataggio nuovo IP
        }
    }
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"ipadd2");
    if(ptr)
    {
        if((*ptr)!='\0')
        {
            IPnum2=ASCIIToNumeric(ptr); //Modifica effettiva della tripletta IP
            AppConfig.MyIPAddr.Val &= 0xFF00FFFF;
            AppConfig.MyIPAddr.Val |= (((DWORD)IPnum2)<<16);
            AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
            SaveAppConfig(); //Salvataggio nuovo IP
        }
    }
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"ipadd3");
    if(ptr)
    {
        if((*ptr)!='\0')
        {
            IPnum3=ASCIIToNumeric(ptr); //Modifica effettiva della tripletta IP
            AppConfig.MyIPAddr.Val &= 0x00FFFFFF;
            AppConfig.MyIPAddr.Val |= (((DWORD)IPnum3)<<24);
            AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
            SaveAppConfig(); //Salvataggio nuovo IP
        }
    }
}
//*****

```

analizzeremo. Per prima troviamo la **StackTask()**, anche questa definita nel file **StackTsk.c**, che controlla l'eventuale presenza di pacchetti, ne identifica il tipo ed avvia la loro elaborazione dando la parola al task appropriato. Fa seguito la **StackApplications()** (anch'essa definita in quel file) che chiama all'esecuzione uno dopo l'altro, i diversi task di rete: ad esempio **HTTPServer()** per la consultazione delle pagine web, **FTPServer()** e **SMTPTask()** se attivi e, in modo particolare, le due routine **GenericTCPClient_TX()** e

GenericTCPClient_RX() definite nel file **GenericTCPClient.c**. Queste sono le due routine custom che consentono rispettivamente l'emissione periodica e al cambiamento della configurazione degli input, e la modifica degli output ad immagine degli input del modulo remoto (discorso valido sia per gli I/O a bordo dell'E4I40 che per quelli delle schede di espansione). Prima di entrare nel merito di queste importanti routine analizziamo gli altri file modificati. La **HardwareProfile.h** assegna gli

pseudonimi alle diverse linee di I/O e ai relativi TRIS (quindi gli ingressi INx, le uscite OUTx_IO e le linee SCL ed SDA dell'I2C) e definisce la configuration-bits. E poi la **HTTP2.c** che definisce tutte le routine del protocollo HTTP e contenente una piccola porzione di codice relativa alla procedura del login. Adesso vediamo come sia possibile modificare il contenuto delle pagine web quando inseriamo nuovi parametri, oppure leggiamo lo stato degli ingressi e andiamo a modificare quello

delle uscite. Prima di andare oltre però è doverosa una precisazione: quando desideriamo consultare una certa pagina Internet, dobbiamo digitare il relativo indirizzo nell'apposito campo del nostro browser: se ad esempio vogliamo consultare il sito della nostra rivista digitiamo <http://www.elettronica.in.it> o, più semplicemente www.elettronica.in.it. Lo stesso facciamo ogni volta che vogliamo consultare le pagine web dell'E4I4O e quindi <http://indirizzoIP>, ad esempio <http://192.168.0.95/>. A questo indirizzo possiamo aggiungere la pagina che ci interessa, nel caso delle configurazioni digiteremo la stringa <http://192.168.0.95/config.htm>.

Ora facciamo un altro passo in avanti: se a questo indirizzo aggiungiamo una stringa del tipo `?campo1=valore1&campo2=valore2&campo3=valore3...` otterremo come risultato la spedizione di informazioni testuali da parte del browser, al server che stiamo consultando; nel nostro caso, l'E4I4O. Quest'ultimo, ricevendo ciascuno dei campi accompagnati dai relativi valori (se ce ne sono), eseguirà allora una certa azione. La stringa oltre il simbolo `?` è detta query-string, ed è grazie a questa che il browser, e quindi l'utente, riesce ad impartire i comandi desiderati al microcontrollore interfacciato in rete. Questa stringa può essere scritta manualmente, ma il codice HTML delle pagine web la crea in automatico ogni volta che clicchiamo un pulsante o compiamo un'altra azione. Esaminiamo a questo punto il seguente URL (Uniform Resource Locator): <http://192.168.0.95/config.htm?ipadd0=192&ipadd1=168&ipadd2=0&ipadd3=50>. La sua query-string viene generata in automatico quando sulla pagi-

na *Configurazioni* scriviamo un nuovo indirizzo IP sulle quattro caselle di testo da tre caratteri (192.168.0.50) e clicchiamo sul pulsante "Salva" (vedere figura

9). Ecco che l'E4I4O riceverà le seguenti informazioni: `ipadd0 = 192, ipadd1 = 168, ipadd2 = 0, ipadd3 = 50`. Infatti, osservando il **Listato 1** estratto dal codice

Listato 3

Il form HTML per il controllo degli OUT a bordo dell'E4I4O.

```
<table width="400" border="0" align="center" bgcolor="black">
<tr><td colspan="4" height="8"></td></tr>
<tr>
<form method="get" action="index.htm">
<td><div align="center"><input type="submit" name="out4on" value="ON"></div></td>
<td><div align="center"><input type="submit" name="out3on" value="ON"></div></td>
<td><div align="center"><input type="submit" name="out2on" value="ON"></div></td>
<td><div align="center"><input type="submit" name="out1on" value="ON"></div></td>
</form>
</tr>
<tr><td colspan="4" height="8"></td></tr>
<tr>
<td><div align="center"></div></td>
<td><div align="center"></div></td>
<td><div align="center"></div></td>
<td><div align="center"></div></td>
</tr>
<tr>
<tr><td colspan="4" height="8"></td></tr>
<form method="get" action="index.htm">
<td><div align="center"><input type="submit" name="out4off" value="OFF"></div></td>
<td><div align="center"><input type="submit" name="out3off" value="OFF"></div></td>
<td><div align="center"><input type="submit" name="out2off" value="OFF"></div></td>
<td><div align="center"><input type="submit" name="out1off" value="OFF"></div></td>
</form>
</tr>
<tr><td colspan="4" height="12"></td></tr>
</table>
```

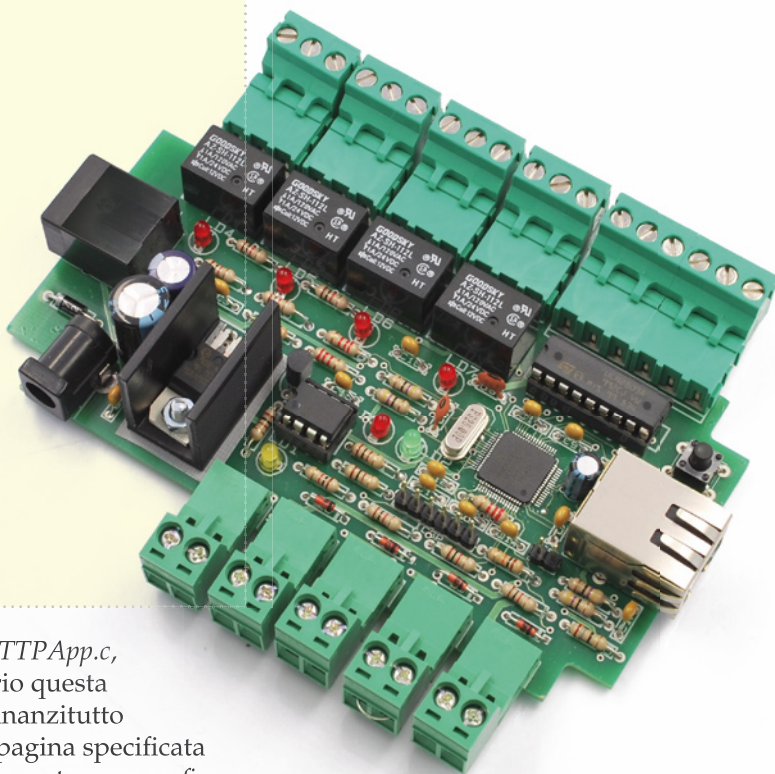
Listato 4

Lettura del comando di ON/OFF degli OUT dell'E4I4O.

```
//***** index.htm *****
if(!memcmppgm2ram(filename, "index.htm", 9))
{
    if(login_ok == TRUE)    tick_login = TickGet(); //Reset Timeout

    //***** Modifica Linee Output *****
    //OUT 4
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"out4on");
    if(ptr) OUT4_IO = 1;
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"out4off");
    if(ptr) OUT4_IO = 0;
    //OUT 3
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"out3on");
    if(ptr) OUT3_IO = 1;
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"out3off");
    if(ptr) OUT3_IO = 0;
    //OUT 2
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"out2on");
    if(ptr) OUT2_IO = 1;
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"out2off");
    if(ptr) OUT2_IO = 0;
    //OUT 1
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"out1on");
    if(ptr) OUT1_IO = 1;
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"out1off");
    if(ptr) OUT1_IO = 0;
    //*****
}
```

```
//Visualizzazione tramite immagine gif del Nibble-Output
void HTTPPrint_outgif(WORD num)
{
    // Determine which switch
    switch(num)
    {
        case 1:
            num = OUT1_IO;
            break;
        case 2:
            num = OUT2_IO;
            break;
        case 3:
            num = OUT3_IO;
            break;
        case 4:
            num = OUT4_IO;
            break;
        default:
            num = 0;
    }
    // Print the output
    TCPput(sktHTTP, (num?'1':'0'));
    return;
}
```

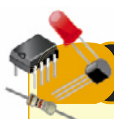


HTML della pagina *config.htm*, noterete il form che esegue le assegnazioni dei quattro (o meno) valori inseriti dall'utente sui rispettivi campi *ipadd0/1/2/3*. Questo semplice meccanismo è valido in linea generale e quindi interviene laddove vi sono comandi e valori da passare al nostro modulo remoto. Il firmware è dunque in grado di ispezionare l'intera query-string e di estrarre le informazioni, ovvero i comandi impartiti dall'utente. Adesso, allora, esaminiamo questo aspetto, ovvero la lettura e l'interpretazione del nuovo indirizzo IP da assegnare e salvare sul dispositivo. Il codice contenuto nel **Listato 2**, tratto dal

file *CustomHTTPApp.c*, svolge proprio questa mansione: innanzitutto identifica la pagina specificata dall'URL, in questo caso *config.htm*, grazie alla routine chiamata *memcmppgm2ram(filename, "config.htm", 10)*. Questa non fa che comparare il filename dell'url con il nome della pagina (in tutto 10 caratteri). In caso di riscontro positivo, setaccia la query-string e cerca per prima la stringa *ipadd0* e lo fa con l'istruzione *HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"ipadd0")*. Se questa è presente e il relativo valore non è nullo ('\0') preleva la stringa/valore e dopo la conversione ASCII/valore numerico

lo va a salvare sulla EEPROM. Lo stesso discorso vale per le altre tre triplette dell'indirizzo IP appena digitato dall'utente. Con lo stesso meccanismo, dalla pagina *index.htm* è possibile accendere e spegnere uno dei quattro relé a bordo della scheda.

Qui i campi sono del *out4on* e *out4off*, mentre i possibili valori sono ON e OFF. Ad esempio, quando clicchiamo sul pulsante ON di OUT4 per accendere il relé numero quattro, l'URL verrà completato con la stringa *index.htm?out4on=ON*. In realtà in questo caso non è necessario il valore ON/OFF, in quanto basta il campo per esprimere il comando e specificare la linea di out. Il **Listato 3** riporta la porzione di codice HTML che realizza la spedizione di questo comando, mentre il **Listato 4** illustra il codice sorgente del firmware che interpreta ed attua a livello hardware tale controllo. Osservate adesso nuovamente il **Listato 3**, in particolare la sua parte centra-



per il MATERIALE

Tutti i componenti di questo progetto sono facilmente reperibili in commercio; il master dello stampato ed il firmware col quale programmare il micro sono scaricabili gratuitamente dal sito della rivista (www.elettronica.in.it). Per facilitare la realizzazione di questo progetto, abbiamo previsto un kit di montaggio comprendente tutti i componenti, le minuterie, ed il microcontrollore; la scatola di montaggio (cod. FT823K) costa 55,00 Euro. La stessa apparecchiatura è anche disponibile già montata e collaudata allo stesso prezzo (cod. FT823M). Sono anche disponibili separatamente il circuito stampato con serigrafia e solder (cod. S823, Euro 9,00) ed il microcontrollore vergine PIC18F67J60-I/P (Euro 9,80).

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

```

//***** Configurazione Out Expander *****
ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"expselout");
if(ptr)
{
    expout_use = *ptr-'0';           //Salvataggio per visualizzazione indirizzo selezionato
    if(*ptr == '9')
    {
        data_exp_out=0;              //None
        message_exp_out[0]='\0';     //Messaggio Nullo
    }
    else
    {
        exp_add_out=*ptr-'1';
        bitpoint=1;
        for(counter=1;counter<(*ptr-'0');counter++) bitpoint *=2;      //Selezione bit
        if((AppConfig.EXP_Config & bitpoint) == 0)                      //Riconoscimento output
        {
            data_exp_out=0;        //Reset uscite expander
            //Scrittura da Expander I2C
            PCF8574A_I2C_RW_Data_Random(exp_add_out, &data_exp_out, WRITE_I2C);

            message_exp_out[0]='\0';    //Messaggio Nullo
        }
        //Memorizzazione messaggio di errore
        else
        {
            memset(message_exp_out, ' ', 17);        //Reset variabile testo
            //Caricamento stringa da visualizzare
            memcpypgm2ram((void*)message_exp_out, (ROM void *)"*** ERRORE ***", 14);
            message_exp_out[14]='\0';                //Fine stringa
        }
    }
}
//*****

```

le: in essa sono presenti quattro righe di codice che consentono la visualizzazione di immagini in formato gif: ad esempio ``. Il testo compreso tra le due tilde, cioè `outgif(4)`, viene sostituito dal firmware con un valore binario che è l'espressione logica dello stato della linea di out, nell'esempio OUT4. Questo significa che se OUT4 è a 1 (a livello logico alto) l'immagine sarà `led1.gif`, mentre se è a 0 (livello basso) sarà `led0.gif`. Queste due gif, se andate a controllare nella cartella *WebPage*, vedete che rappresentano due spie luminose virtuali, una accesa e l'altra spenta, rispettivamente. Anche queste ultime, come le pagine HTML, vengono convertite nel file binario *MPFSImg2.BIN* e memorizzate sulla EEPROM.

Questa sorta di completamento delle pagine web viene utilizzato tutte le volte che bisogna visualizzare delle informazioni non statiche, ovvero informazioni suscettibili di cambiamenti: i parametri di networking, oppure gli stati di input e di output, eccetera.

Ciascuna di esse però necessita di una specifica routine di stampa che emetta la corretta sequenza ASCII di completamento; ciascuna di queste viene dichiarata in automatico sul file *HTTPPrint.h*, generato ogniqualvolta utilizziamo il tool di conversione "MPFS Generator".

Considerate allora il **Listato 5**: qui ritroviamo la routine *HTTPPrint_outgif(WORD num)*, la quale a parte il codice iniziale, riporta la stessa stringa che tro-

viamo tra le due tilde del codice HTML, ovvero *outgif(WORD num)*, dove il parametro 'num' specifica il numero della porta di output (4/3/2/1).

In base al valore di questo parametro, il codice campiona lo stato logico dell'uscita e tramite la funzione *TCPPUT()* rilascia la stringa di completamento, in questo caso per la definizione del nome dell'immagine gif.

Ribadiamo che il codice sorgente che si occupa di questo interfacciamento con le pagine web è stilato sul file *CustomHTTPApp.c* che, come dice il suo stesso nome, definisce un'applicazione customizzata basata sull'HTTP.

Bene, con questo abbiamo concluso anche con il firmware; non ci resta che augurarvi buon lavoro!

REGISTRARE TEMPERATURE CON DTPIC18F67J50

di LUCA PASCARELLA

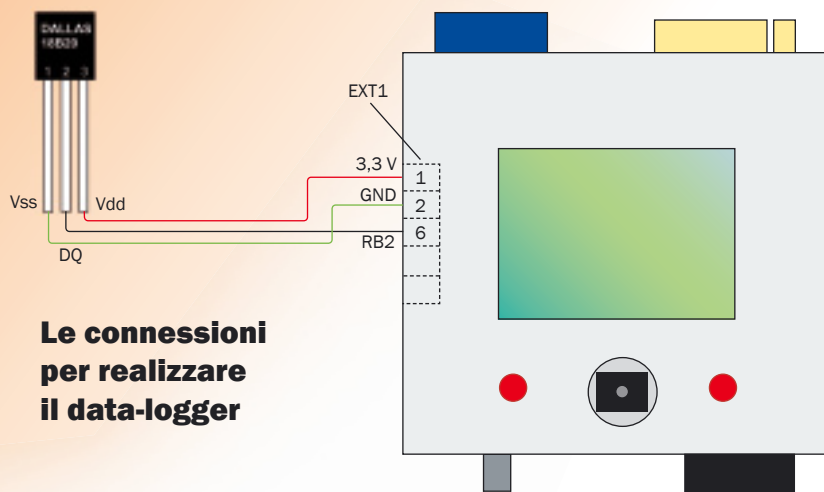
Ecco un'altra applicazione della demoboard Olimex: un data-logger per valori di temperatura che possiamo utilizzare anche come lettore USB di SD-Card.

Nel fascicolo n° 137 abbiamo presentato la demoboard DTPIC18F67J50 della Olimex, una pratica scheda di sviluppo basata sul microcontrollore Microchip PIC18F67J50 ed equipaggiata con un accelerometro a tre assi e quattro livelli di sensibilità, un connettore per SD-Card, una porta USB e una seriale, un joystick, un display LCD monocromatico da 48 x 84 pixel ed altro ancora; la sua dotazione di periferiche permette di sfruttarla per emulare applicazioni software che prevedono l'uso di periferiche per il rilevamento di posizioni e accelerazione, di periferiche Human Interface, dell'interfaccia USB e del display a matrice di punti, ma anche la lettura e la scrittura di

dati in SD-Card.

Il mese scorso ci siamo lasciati dopo aver descritto due semplici giochi che sfruttavano l'accelerometro e il display, oltre ad un'applicazione per visualizzarne i valori letti (sempre dall'accelerometro); a corredo del progetto abbiamo fornito le librerie per controllare il display e l'MMA6260Q, che sono scritte in linguaggio





Le connessioni per realizzare il data-logger

ANSI C e presentano delle API, molto intuitive, tali da garantire brevi tempi di start-up. Ci siamo lasciati promettendo di ritornare sull'argomento per completare la panoramica sulla demoboard con nuove idee e nuovi progetti. In queste pagine vogliamo presentarvi una nuova applicazione: si tratta di un data-logger per la temperatura, capace di memorizzare una quantità pressoché illimitata di campioni di misura. La lettura della temperatura è affidata a delle sonde DS18B20 connesse in parallelo su un bus

One-Wire comune implementato nella demoboard con la linea RB2 del microcontrollore montato in essa; i sensori campionano la temperatura e ne presentano il valore in formato digitale. Per memorizzare i campioni così catturati si è scelto di sfruttare lo slot per memorie SD con una comune SD-card preventivamente formattata. Per rendere il progetto interessante e "user-friendly", abbiamo previsto la possibilità di collegare direttamente la scheda a un PC tramite la porta USB; in questo caso lo facciamo sfruttan-

do lo Stack USB fornito gratuitamente dalla Microchip. Con tale configurazione la board può essere rilevata da un comune PC come memoria di massa (USB Mass Storage) e di conseguenza gestita come una semplice chiavetta o unità per schede di memoria USB con file-system FAT16 o FAT32. Perciò la demoboard in questa applicazione può essere usata non solo come data-logger, ma anche da lettore esterno di memory-card per computer. Tra le varie combinazioni possibili è stata scelta l'USB Mass Storage in quanto il protocollo corrispondente permette di utilizzare qualsiasi dispositivo senza la necessità di dover installare driver, ottenendo così innumerevoli vantaggi per l'utente finale, che potrà usare un comune PC per archiviare il buffer di campioni salvati sulla SD-card. Il tutto con pochi e semplici passaggi. In Windows 2000/XP/Vista i driver sono già nel sistema operativo. Il progetto, creato nell'ambiente di sviluppo MPLAB-IDE, può essere interamente scaricato, come

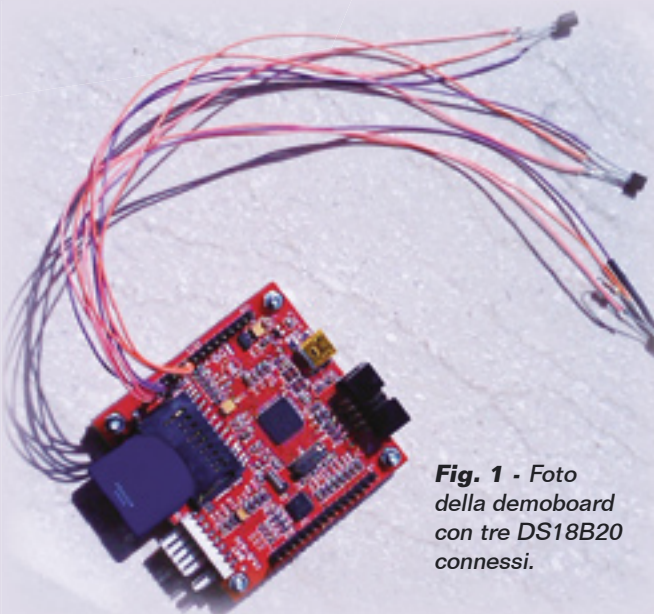


Fig. 1 - Foto della demoboard con tre DS18B20 connessi.

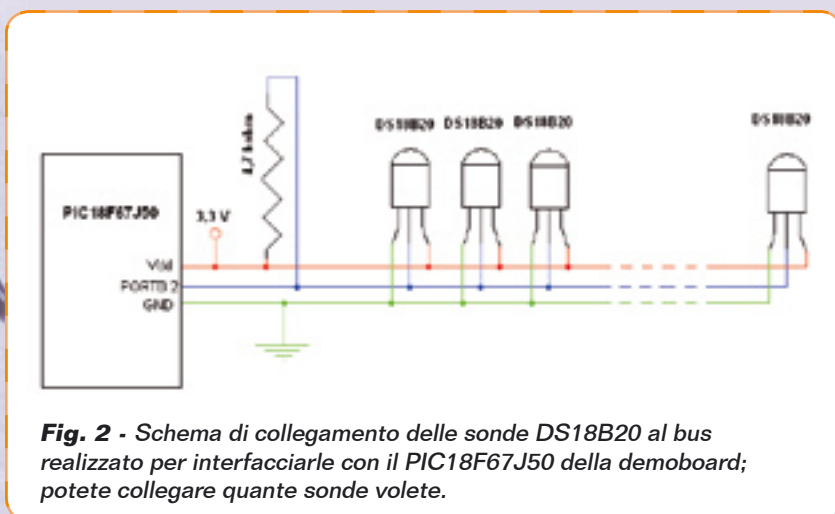


Fig. 2 - Schema di collegamento delle sonde DS18B20 al bus realizzato per interfacciarle con il PIC18F67J50 della demoboard; potete collegare quante sonde volete.

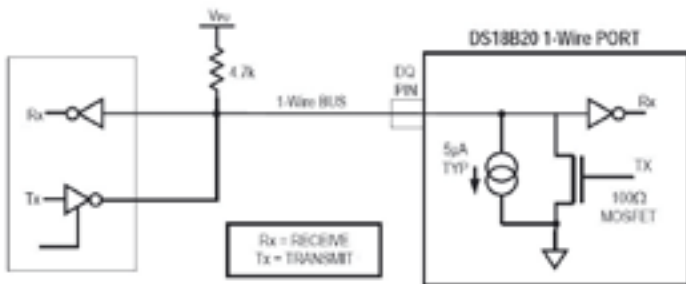


Fig. 3 - Modalità di collegamento con l'unità master nel protocollo 1-Wire.

sempre, dal nostro sito; potete utilizzare i file sorgente sia per motivi di studio, sia come punto di partenza per realizzare un progetto personalizzato.

IL SENSORE DI TEMPERATURA

La sonda utilizzata per leggere la temperatura è la DS18B20 della Dallas Semiconductor e viene prodotta in due formati distinti: TO-92 e microSOP; per facilità di utilizzo abbiamo preferito la prima versione (contenitore a tre pin simile a un comune transistor). Dallo schema elettrico si intuisce che le connessioni sono estremamente semplici, infatti il tutto si riduce a collegare tre terminali, che sono i due pin

d'alimentazione (Vdd, GND e quello della linea dati, che è bidirezionale); infatti la sonda della Dallas comunica sfruttando il protocollo One-Wire, che contempla la comunicazione dati mediante un solo filo (riferito alla massa, ovviamente). Nel nostro caso, prevedendo di usare tre sonde, tutti i pin delle linee dati si affacceranno su un unico filo a costituire il bus per i dati verso il PIC18F67J50. Perché tutto funzioni correttamente, il bus necessita di una resistenza di pull-up da 4,7 kohm: tale valore viene indicato dalle specifiche della Dallas ed è indifferente rispetto al numero di dispositivi connessi allo stesso bus.

Essendo la comunicazione con

i sensori di tipo digitale, si ha il vantaggio di sistemare le sonde anche un po' lontane dalla demoboard, in modo da formare delle reti di sensori distribuiti, dove una comune linea analogica sarebbe soggetta a disturbi ed interferenze provenienti dall'ambiente circostante e difficilmente eliminabili.

Per completare la descrizione dell'hardware, è bene spendere due parole sulla configurazione dei pin che si interfacciano al bus One-Wire: il protocollo permette di realizzare un'architettura multi-punto, in cui il dispositivo master (microcontrollore) deve avere una configurazione di uscita open-drain o tri-state, realizzabile con un PIC ponendo il pin interessato in alta impedenza ($TRISBbits.TRISB2 = 1;$), mentre le unità slave (dispositivi che interrogati devono rispondere) avranno un'uscita open-drain che permette di portare a 0 il segnale sul bus, tenuto costantemente alto dalla resistenza di pull-up da 4,7 kohm. La struttura è illustrata nella Fig. 3.

USO DEL BUS 1-WIRE

La comunicazione ha inizio con una sequenza di reset che solo il master può inviare; la sequenza di inizializzazione prende il nome di *reset-pulse* e consiste nel tenere a livello basso la linea (0 V) per 480 μ s ($OW_reset_pulse();$). Questa sequenza è rilevata da tutti gli slave connessi al bus, i quali a loro volta rispondono con un segnale di presenza detto *presence-pulse*, che ha una durata variabile tra i 60 ed i 240 μ s ed ancora una volta consiste nel porre la linea di comunicazione a livello basso.

Da questa prima analisi si intuisce che i livelli logici 0 ed 1 non corrispondono a valori di tensioni, ma piuttosto devono essere

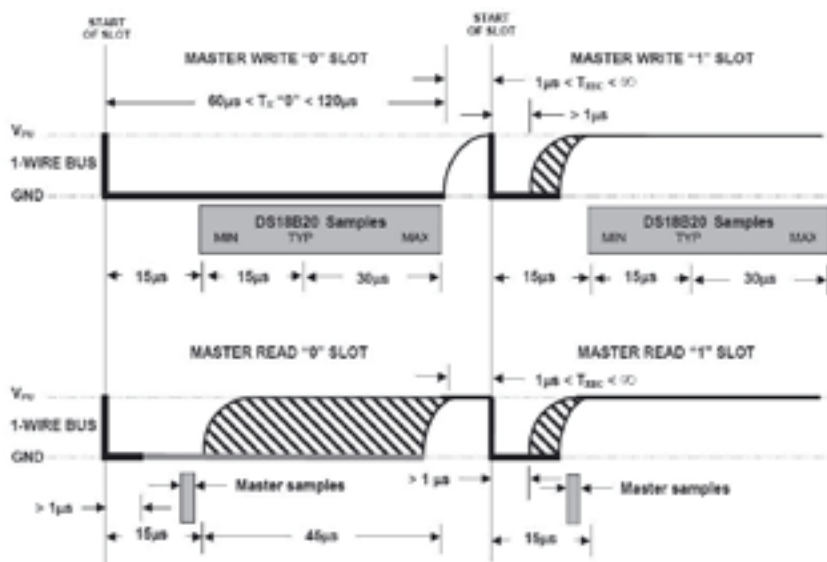
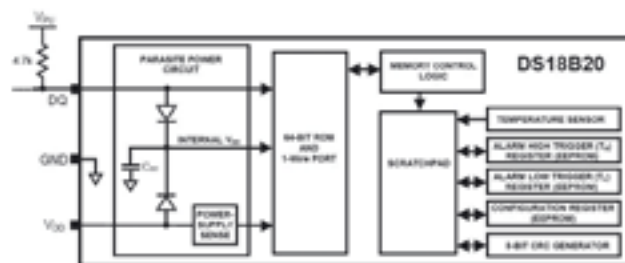


Fig. 4 - Temporizzazioni tipiche di una sessione di comunicazione su 1-Wire.

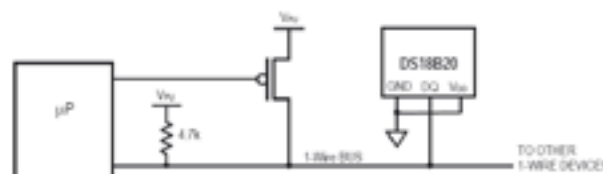
Il sensore DS18B20

L'integrato DS18B20 è un preciso sensore di temperatura con risoluzione regolabile tra 9 e 12 bit, che fornisce il risultato della misura in gradi Celsius, espresso in formato binario; è molto pratico perché non richiede procedure di taratura. Inoltre è dotato di una funzione d'allarme programmabile secondo una soglia di trigger. Il range di temperatura misurabile oscilla tra -55 °C e +125 °C e la precisione nell'arco tra -10 e +85 °C è garantita in mezzo grado centigrado. L'integrato comunica le proprie letture mediante un protocollo chiamato One-Wire (1-Wire) e sviluppato dalla Dallas Semiconductor per l'interfacciamento di dispositivi tramite un singolo canale di comunicazione; il protocollo è stato creato per interagire con dispositivi in cui la velocità di comunicazione e la quantità di dati da scambiare non è elevata. Nella modalità normale, ogni integrato invia i propri dati sul contatto DQ rispetto al terminale GND. Ma è anche prevista una speciale mo-

dalità nata per contenere il consumo di elettricità e per interrogare sensori collocati in apparati privi di alimentazione a riposo o collegati mediante due soli fili: appunto, DQ e GND. Tale modalità è chiamata Parassite Power e permette di creare reti ad-hoc per i casi in cui si necessita di pochi attimi di funzionamento seguiti da periodi brevi o lunghi di stand-by. Il protocollo One-Wire è stato pensato per realizzare bus di comunicazione su cui affacciare più dispositivi, dei quali uno solo deve essere il master (un microcontrollore o microprocessore) mentre gli altri (slave) si limitano a rispondere quando interrogati; la comunicazione viene sempre iniziata dal master e i dispositivi slave possono essere interrogati singolarmente, ognuno indirizzato grazie a un codice univoco impresso in fabbrica (64-Bit Lasered ROM code). Quanto alla modalità Parassite Power, la tecnica adottata è concettualmente simile a quella adoperata dai Tag



Schema a blocchi del sensore DS18B20.



Connessione delle sonde DS18B20 in modalità parassite power.

passivi per RFID: in questo caso l'antenna ricevente del Tag RFID cattura in una prima fase l'energia con la quale carica un condensatore; nella seconda fase, con l'energia del condensatore alimenta la logica, che trasmette il proprio codice cortocircuitando periodicamente l'antenna. Lo scambio di informazioni non può durare per tempi molto lunghi, ma è vincolato alla quantità d'energia

immagazzinata nel condensatore. Nel Parassite Power i DS18B20 sono collegati solo con Vss e DQ (linea dati): quando il master deve interrogarli, applica alimentazione al bus, cosicché ciascun integrato possa caricare un condensatore che ha all'interno; poi sospende la Vdd e inizia a interrogare i DS18B20 (come previsto dal protocollo One-Wire) i quali rispondono sfruttando l'energia immagazzinata.

associati a durate temporali di un particolare evento, similmente a ciò che accade nelle comunicazioni in radio frequenza codificate in PDM (ad esempio con l'MM53200).

Per convenzione, i livelli logici sono così definiti: lo 0 (*OW_write_bit(0);*) consiste nel tenere a livello basso la linea per un periodo pari ad uno slot temporale di durata variabile tra 60 e 120 µs, al termine del quale la linea viene rilasciata per un periodo di almeno 1 µs; similmente l'1 logico (*OW_write_bit(1);*) viene rappresentato tenendo la linea dati a zero logico per un perio-

do non superiore ai 15 µs. Nel tempo mancante a colmare lo slot temporale (60÷120 µs) la linea dati deve essere rilasciata e la resistenza di pull-up deve fornire il giusto valore di tensione di riferimento.

Il grafico con le temporizzazioni è illustrato nella Fig. 4.

In genere tra master e slave vengono scambiati byte d'informazione (*OW_write_byte(byte);* e *OW_read_byte();*) e per rendere la comunicazione affidabile, al termine di ogni transizione viene inviata una sequenza di CRC (*cyclic redundancy check*) che permette di verificare se la

conversazione è stata esente da errori nei bit.

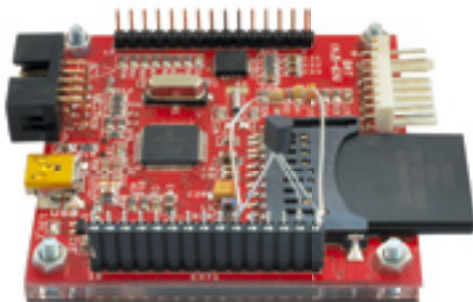
Come già accennato, il protocollo One-Wire contempla la presenza di più unità slave, grazie al fatto che ogni DS18B20 può essere identificato dal dispositivo master (il microcontrollore) perché ha scritto in fabbrica un codice univoco da 64 bit, anche detto 64-BIT Lasered ROM code (viene letto con *ReadSerialNumber(serial);*).

Nel redigere questo progetto abbiamo previsto per semplicità realizzative il collegamento di tre sonde DS18B20; dimensionando opportunamente l'array SMD è

possibile in qualsiasi momento espandere il numero di elementi One-Wire. Il modello di comunicazione proposto permette di associare in cascata elementi anche eterogenei tra di loro; se volete saperne di più potete visitare il link <http://www.maxim-ic.com/products/1-wire/> per conoscere le diverse tipologie di chip esistenti (EEPROM e controller per batterie).

IL FIRMWARE

Ancora una volta il linguaggio di programmazione scelto per questo progetto è un linguaggio di alto livello ed esattamente l'ANSI C, che è ormai uno standard ben affermato nell'ambito dell'elettronica embedded; il motivo di



tale scelta ricade su diversi fattori, tra cui la possibilità di sfruttare lo stack USB fornito gratuitamente della Microchip. È evidente che con un linguaggio di basso livello, tipo l'assembler, è possibile ottimizzare al meglio le prestazioni della CPU, ma questo implicherebbe costi di sviluppo non sempre giustificati. Questa premessa aiuta a capire perché le librerie per lo stack USB, ma anche quelle per il One-Wire, sono state scritte in linguaggio C e non in Assembler. A sostegno di questa tesi c'è anche da considerare l'aiuto offerto dal compilatore nella fase di traduzione, il quale

```
if(sw2 == 0 && cond == 0){
    while(sw2 == 0);
    cond = 1;
}
if(cond == 1){
    //Let's do a soft detach to insure that the PC isn't trying to
    // write the files the same time we are.
    LUNSoftDetach(0);
    if(logFile != NULL){
        for(cont=0; cont<numDS; cont++){
            ConvertTemp((char*)&SND[cont][0]);
            temp = ReadTemp((char*)&SND[cont][0]);
            temp2 = temp;
            temp = temp - temp2;
            temp3 = temp * 10000;

            if(cont==0)
                FSfprintf(logFile, "\r\nTemperatura Sonda
Numero %d: % d.-%-4d °C\r\n",cont+1, temp2, temp3);
            else
                FSfprintf(logFile, "Temperatura Sonda Numero
%d: % d.-%-4d °C\r\n",cont+1, temp2, temp3);
        }
    }
    else{
        logFile = FSfopenpgm("log.csv",WRITE);

        mLED_1_On();
    }
}
else
{
    if(logFile != NULL)
    {
        FSfclose(logFile);
        logFile = NULL;

        //Now that we are done writing to the drive we can reattach
        // so that the PC can read the files that we just wrote.
        LUNSoftAttach(0);
    }
    mLED_1_Off();
}
//}
}
else{
    if(MDD_SDSPi_MediaDetect() == TRUE){
        initResults = FSInit();
    }
}

if(sw2 == 0 && cond == 1){
    mLED_1_Off();
    while(sw2 == 0);
    cond = 0;
    Delay100TCYx(10);
}
```

I due controlli *if(sw2 == 0 && cond == 0)* e *if(sw2 == 0 && cond == 1)* permettono di monitorare costantemente lo stato dello switch, il quale serve ad abilitare o disabilitare il datalogger; il controllo sfrutta una semplice implementazione di una logica bistabile con l'uso della variabile booleana *cond* inizializzata a 0.

provvede ad effettuare sul codice un minimo di controlli logici utili per scovare potenziali bug. Dopo questa breve premessa, torniamo ad occuparci del nostro progetto: come anticipato, il software permette di raccogliere periodicamente dei valori di temperatura e di conservarli su un dispositivo di memoria, che nel nostro caso è una comune SD-Card. La Flash viene gestita dal PIC adattando le proprie routine al tipo di file-system presente in essa. Alla stessa SD si può accedere mediante la porta USB della demoboard, ma bisogna prestare

attenzione al fatto che, essendo il file-system della SD accessibile sia dalla USB che dai processi di campionamento della temperatura, bisogna imporre dei vincoli al software in modo tale da disattivare le richieste provenienti dall'USB quando la memoria è già impegnata dalla scrittura dei dati da parte del PIC. Questo meccanismo diventa necessario per ovviare ai problemi di inconsistenza sui dati che ne potrebbero derivare e viene svolto dalle funzioni *LUNSoftAttach(0);* e *LUNSoftDetach(0);* che, rispettivamente, sbloccano e bloccano le richieste da parte dell'USB.

Tutti conosciamo l'USB (Universal Serial-Bus) che è un'interfaccia di comunicazione ormai divenuta uno standard affermato a livello mondiale sia nell'elettronica di consumo che nelle applicazioni industriali, grazie alla sua flessibilità ed affidabilità. Dalla sua prima versione (USB 1.0, caratterizzata da un data-rate di 1,5 Mbit/s, velocità adeguata a periferiche lente come mouse e tastiera...) l'USB ha subito numerose modifiche per essere sempre competitiva con le nuove tecnologie. La limitazione nella velocità è stata

presto superata con la versione USB 1.1, quest'ultima capace di garantire una velocità massima di 12 Mbit/s, ulteriormente innalzata con lo standard USB 2.0, che garantisce un data-rate teorico di 480 Mbit/s. A breve saranno disponibili sul mercato prodotti con la nuova versione USB 3.0, capace di una velocità pari a 4,800 Gbit/s, cioè 10 volte più veloce dello standard attuale. La compatibilità con le precedenti versioni ha giocato un ruolo fondamentale nello sviluppo dell'USB, perché permette di utilizzare vecchie periferiche su Host nuovi. Per spiegare perché è possibile far funzionare la demoboard o una Pen-Drive senza il bisogno di installare i driver specifici, bisogna far luce sul concetto di classi, spesso ignorato; diciamo innanzitutto che le periferiche USB

possono essere raggruppate in categorie: mouse e tastiera fanno parte della classe HID (Human Interface Devices), le periferiche di archiviazione di massa rientrano nella classe USB Mass Storage devices ed ancora, modem o simili faranno parte del gruppo CDC (Communication Devices Class). Raggruppando i devices in categorie, si ha la possibilità di utilizzare le funzioni base delle periferiche con dei driver comuni, magari pre-installati nel sistema operativo. Dovendo solo copiare o cancellare i file registrati dal datalogger, sono sufficienti le funzionalità di base offerte del sistema, e quindi con uno Stack lato client opportunamente configurato possiamo sfruttare queste potenzialità già presenti nel sistema operativo, senza dover ricorrere a driver di terze parti.

Da un primo esame del **Listato 1** appare evidente che le prime operazioni svolte dal PIC siano quelle riguardanti l'inizializzazione del sistema; osservando il file **main.c** si scopre che la prima routine viene svolta chiamando la funzione **InitializeSystem()**; presente nel **main()**, quindi si passa all'inizializzazione delle periferiche gestite tramite le API delle varie librerie. Per avviare la SD-Card, il PIC18F67J50 invoca **FSInit()**; che gli permette di identificare il file-system secondo cui essa è stata formattata (FAT16

o FAT32). Per rendere operativi tutti i dispositivi connessi al bus One-Wire si utilizzerà la funzione **initOWDevices()**; che, oltre all'inizializzazione delle porte di comunicazione, si occuperà di ricercare tutti i dispositivi connessi al bus, catalogando nel vettore **char SND[4][8]**; (array bidimensionale) il numero di serie a 64 bit univoco di ogni DS18B20; a questo punto, avendo in memoria il ROM-code di ogni dispositivo sarà possibile accedere individualmente ad ogni sensore DS18B20 affacciato sul bus. Infine ci sono le ultime due inizializzazioni riguardanti i led e lo switch: **mInitAllLEDs()**; e **mInitSwitch2()**;

Nella funzione **InitializeSystem()** viene richiamata anche **USBDeviceInit()**; che provvede ad avviare lo stack USB; al termine di questa procedura la periferica USB viene riconosciuta dall'Host, il quale entrerà in sincronizzazione con il PIC. Tutte le comunicazioni sono bufferizzate ed è quindi necessario chiamare periodicamente **USBDeviceTasks()**; che provvede a gestire le richieste. **USBDeviceTasks()**; deve essere invocata almeno ogni 100 microsecondi e la sua invocazione può essere gestita

in due modi, selezionabili nel file **usb_config.h** semplicemente definendo una delle due voci seguenti: **#define USB_POLLING** o **//#define USB_INTERRUPT**. Tra il polling e l'interrupt l'unica differenza sta nel fatto che nel primo sappiamo con certezza quando il controllo passerà al gestore dell'USB, mentre nel secondo caso i tempi sono affidati ad un timer che risulta asincrono rispetto al nostro codice; se non ci crea problemi, questo secondo metodo è da preferire rispetto al polling.

Nel **Listato 1** si può vedere un

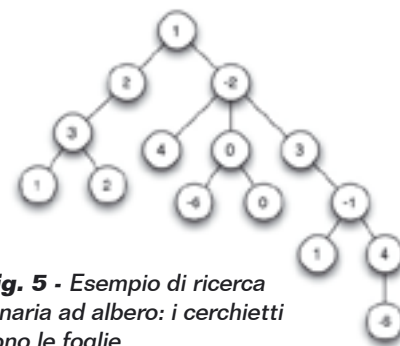


Fig. 5 - Esempio di ricerca binaria ad albero: i cerchietti sono le foglie.

Tabella 1

Bit	Completo bit	Informazioni
0	0	Nell'albero c'è sia il BIT 0 che il BIT 1.
0	1	Nell'albero c'è solo il BIT 0.
1	0	Nell'albero c'è solo il BIT 1.
1	1	Non ci sono dispositivi partecipanti

DOVE ACQUISTARE LA DEMOBOARD

La demoboard descritta in queste pagine ed utilizzata per proporre varie applicazioni, viene prodotta dalla Olimex (PIC-LCD3310) ed è commercializzata in Italia dalla ditta Futura Elettronica di Gallarate (VA) con il codice 8100-DTPIC18F67J50 al prezzo di 88,00 euro. Chi volesse acquistarla può visitare il sito Internet www.futurashop.it, dal quale è possibile scaricare anche il data-sheet in formato pdf contenente tutte le informazioni tecniche dettagliate.

frammento del codice presente nel **main()**: questo pezzo di software si preoccupa di leggere la temperatura delle sonde e registrarla in memoria; in dettaglio, la funzione **ConvertTemp((char*)&SND[i][0])**; avvia la conversione della temperatura della sonda avente per numero di serie quello passato come argomento. Al termine della conversione **temp = ReadTemp((char*)&SND[i][0])**; pone nella variabile **temp** (float) il valore della temperatura letta. La funzione **FSprintf** svolge due ruoli: crea la stringa accettando i valori int da convertire in codice ASCII e successivamente stampa la stringa creata nel file. Le librerie standard **printf** e **sprintf** del C18 e di conseguenza **FSprintf** non supportano l'argomento **%f** e quindi non è possibile convertire in un sol colpo il valore di una variabile float in stringa; perciò facciamo ricorso a due variabili temporanee di tipo "int" (**temp2** e **temp3**) che prendono rispettivamente la parte intera e la parte decimale del valore della temperatura.

RICERCA DEI DISPOSITIVI ONE WIRE

La ricerca dei dispositivi One Wire connessi allo stesso bus viene svolta dalla funzione **OWSearch()**; visibile nel **Listato 2**, che implementa in linguaggio C un algoritmo di ricerca binaria ad albero (**Fig. 5**); l'algoritmo percorre ogni ramo fino al riconoscimento di un dispositivo o dopo aver raggiunto le foglie. L'algoritmo inizia inviando sul bus One-Wire l'impulso di reset e l'analisi delle risposte permette di trovare la radice dell'albero binario. Per ottenere il numero di serie di ogni dispositivo, il master invia il comando di ricerca **0xF0**, al quale tutti i dispositivi presenti sul bus rispondono contempora-

```
int OWSearch()
{
    int id_bit_number;
    int last_zero, rom_byte_number, search_result;
    unsigned char id_bit, cmp_id_bit;
    unsigned char rom_byte_mask, search_direction;

    // initialize for search
    id_bit_number = 1;
    last_zero = 0;
    rom_byte_number = 0;
    rom_byte_mask = 1;
    search_result = 0;
    crc8 = 0;

    // if the last call was not the last one
    if(!LastDeviceFlag){
        // 1-Wire reset
        if(!DetectSlaveDevice()){
            // reset the search
            LastDiscrepancy = 0;
            LastDeviceFlag = FALSE;
            LastFamilyDiscrepancy = 0;
            return FALSE;
        }
        // issue the search command
        OW_write_byte(0xF0);
        // loop to do the search
        do{
            // read a bit and its complement
            id_bit = OW_read_bit();
            cmp_id_bit = OW_read_bit();
            // check for no devices on 1-wire
            if((id_bit == 1) && (cmp_id_bit == 1))
                break;
            else{
                // all devices coupled have 0 or 1
                if(id_bit != cmp_id_bit)
                    search_direction = id_bit; // bit write value for search
                else{
                    // if this discrepancy if before the Last Discrepancy
                    // on a previous next then pick the same as last time
                    if(id_bit_number < LastDiscrepancy)
                        search_direction = ((ROM_NO[rom_byte_number] & rom_byte_mask) > 0);
                    else
                        // if equal to last pick 1, if not then pick 0
                        search_direction = (id_bit_number == LastDiscrepancy);

                    // if 0 was picked then record its position in LastZero
                    if(search_direction == 0){
                        last_zero = id_bit_number;

                        // check for Last discrepancy in family
                        if(last_zero < 9)
                            LastFamilyDiscrepancy = last_zero;
                    }
                }
                // set or clear the bit in the ROM byte rom_byte_number
                // with mask rom_byte_mask
                if(search_direction == 1)
                    ROM_NO[rom_byte_number] |= rom_byte_mask;
                else
                    ROM_NO[rom_byte_number] &= ~rom_byte_mask;

                // serial number search direction write bit
                OW_write_bit(search_direction);

                // increment the byte counter id_bit_number
                // and shift the mask rom_byte_mask
                id_bit_number++;
                rom_byte_mask <<= 1;

                // if the mask is 0 then go to new SerialNum byte rom_byte_number and reset mask
                if (rom_byte_mask == 0){
                    docrc8(ROM_NO[rom_byte_number]); // accumulate the CRC
                    rom_byte_number++;
                    rom_byte_mask = 1;
                }
            }
        } while(rom_byte_number < 8); // loop until through all ROM bytes 0-7

        // if the search was successful then
        if(!((id_bit_number < 65) || (crc8 != 0))){
            // search successful so set LastDiscrepancy, LastDeviceFlag, search_result
            LastDiscrepancy = last_zero;

            // check for last device
            if (LastDiscrepancy == 0)
                LastDeviceFlag = TRUE;

            search_result = TRUE;
        }
    }
    // if no device found then reset counters so next 'search' will be like a first
    if(!search_result || !ROM_NO[0]){
        LastDiscrepancy = 0;
        LastDeviceFlag = FALSE;
        LastFamilyDiscrepancy = 0;
        search_result = FALSE;
    }
    return search_result;
}
```

SD-Card nei sistemi EMBEDDED

Le SD-card stanno diventando un "must" nelle applicazioni embedded: sempre più dispositivi elettronici le impiegano sia per immagazzinare grandi quantità di dati, sia per compiere aggiornamenti al firmware tramite la tecnica del *bootloader*. Di sicuro una delle domande che circonda frequentemente nella testa è perché le SD stanno conquistando quote di mercato a discapito di altre soluzioni di memorizzazione. Per fornire una risposta che spieghi questa diffusione in larga scala, bisogna analizzare i pro e i contro di tutte

le alternative: dal punto di vista economico, il supporto che sembra essere il più conveniente è la EEPROM /data-flash da saldare sul circuito stampato; vantaggi di questa soluzione sono senza dubbio il basso costo del singolo chip, l'assenza di connettori e la possibilità di usare la memoria anche senza un file-system sofisticato. Per contro, gli svantaggi sono rappresentati dall'impossibilità di rimuovere la memoria e trasportarla, oltre che dalla necessità di utilizzare un microcontrollore per accedere al suo contenuto

o modificarlo. Una soluzione simile alle EEPROM, per numero di pin richiesti, ma con i vantaggi della removibilità, è rappresentata dalle Compact Flash, che sono più datate delle SD, più ingombranti e in linea di massima più costose delle SD, anche se al momento i prezzi sono scesi di molto. Va però detto che le Compact Flash sono più veloci delle SD-Card. Un'alternativa all'ormai vecchia Compact Flash è la Pen-Drive USB, che costa pochissimo; però non tutti i processori e microcontrollori sup-



portano la modalità USB Host, necessaria a gestire le Pen-Drive.

La SD-card è il supporto di memorizzazione che condensa quasi tutti i vantaggi: ha dimensioni ridotte (ancor più nella versione miniSD usata nei cellulari) è removibile, ha un ridotto numero di contatti e comunica serialmente mediante un bus basato sul diffusissimo protocollo SPI, che oramai tutti i microcontrollori implementano.

neamente, inviando il primo bit (meno significativo) del proprio identificativo a 64 bit. Essendo la comunicazione temporizzata, le risposte dei DS18B20 che si sovrappongono formano una AND del bit richiesto. Siccome dopo il primo bit i sensori invia-

no lo stesso

bit complementato, il master, analizzando le due informazioni secondo le regole esposte nella **Tabella 1**, può determinare lo stato logico del primo bit. Solo nel primo caso della tabella, il bit viene scelto arbitrariamente e alla successiva iterazione il bit sarà complementare.

Dopo aver determinato i bit, il master li invia sul bus, così che al successivo bit potranno rispondere solo i dispositivi con il numero di serie corrispondente.

FUNZIONAMENTO DEL DATA-LOGGER

Prima di utilizzare la demoboard come data-logger, bisogna introdurre nel lettore SD una card già formattata con FAT16

o FAT32 (dalle prove che abbiamo fatto siamo certi che la demoboard e il nostro firmware possono comodamente gestire SD-Card di capacità fino a 2 GB); potete comunque effettuare la formattazione direttamente con la scheda, perché, come abbiamo detto, a tutti gli effetti viene vista dal PC e dai sistemi operativi Windows come periferica di memorizzazione di massa USB (USB Mass Storage Device). Dunque, se disponete di una SD-Card non ancora formattata introdurla pure nel lettore e date l'alimentazione alla demoboard, dopodiché il firmware caricato nel micro farà tutto il resto; l'alimentazione può essere fornita sia tramite batteria che mediante la porta USB, laddove la board sia collegata ad un computer. Se dovete formattare la SD, da Windows aprite Risorse del computer e fate clic con il pulsante destro del mouse sull'icona corrispondente all'unità esterna USB (la demoboard) quindi impartite il comando *Formatta*; completata

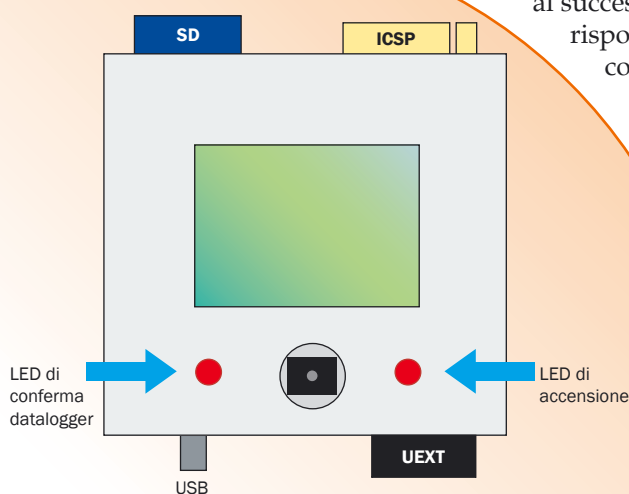


Fig. 6 - Comandi della demoboard nella modalità data-logger.

la formattazione, il data-logger è pronto.

Che la SD-Card venga inserita già formattata o che la formattazione venga fatta on-board, per avviare la registrazione della temperatura basta premere il joystick; a questo punto il led di sinistra si accende per segnalare

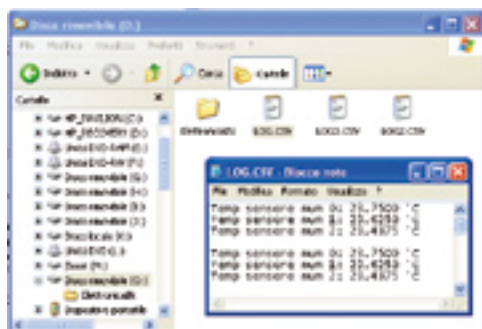


Fig. 7 - Contenuto della SD-Card letta tramite la demoboard.

che la demoboard è in modalità data-logger. Per terminare la registrazione basta premere nuovamente lo stesso pulsante fin quando il led si spegnerà. Quando il dispositivo è connesso al PC (via USB) e non sta effettuando il salvataggio dei dati, è possibile accedere alla memoria interna e compiere tutte le operazioni consentite dal sistema operativo.

Se viene premuto il joystick mentre la board è connessa al PC,

l'USB verrà momentaneamente bloccata, per ritornare in funzione non appena termina la registrazione dei dati.

I dati campionati verranno salvati nel file *LOG.TXT* ed avranno il seguente formato:

T N°1: 24.2500 °C
T N°2: 24.3125 °C
T N°3: 24.3125 °C

T N°1: 23.4725 °C
T N°2: 24.3125 °C

Temperature da record

Le temperature, specie quelle ambiente, sono sempre state oggetto dell'attenzione degli studiosi e primeggiano tra i record; ecco alcuni esempi: la temperatura più alta, ben 57,8°C, fu registrata ad Al-'Aziziyah (Libia) il 13 settembre 1922; la più bassa a Vostok (Antartide) il 21 luglio 1983 vennero registrati - 89,5° C. Il luogo più caldo della Terra è Dallol (Etiopia): la temperatura media dal 1960 al 1966 è stata di 34,5°C; il luogo più freddo è stato la Stazione Sovetskaja (Antartide) dove la temperatura media nel 1957/58 è stata di - 57°C. La più ampia escursione termica nell'arco di 24 ore fu negli: a USA Browning, nel Montana, tra il 23 e il 24 gennaio 1916 la temperatura scese da 6,5 a -49°C.

T N°3: 25.3145 °C
T N°1: 23.120 °C
T N°2: 24.3130 °C
T N°3: 26.3750 °C

T N°1: 22.8210 °C
T N°2: 24.8125 °C
T N°3: 28.2105 °C

Per leggere le risultanze delle registrazioni periodiche, vi basterà accedere alla SD-Card dal PC e aprire il file *LOG.TXT* con un qualsiasi editor di testo. ■

RM ELETTRONICA

Forniture per hobbisti, scuole e industria

Distributore autorizzato:



**FUTURA
ELETTRONICA**



**ELSE
Kit**



ELETTRONICA

Via Val Sillaro 38 • 00141 Roma • Tel: 06-8104753

Il tuo punto
di riferimento
per l'elettronica
a ROMA



Nella famosa serie televisiva Supercar, il protagonista combatteva quotidianamente le forze del male con l'aiuto di KITT (*Knight Industries Three Thousand*), una speciale Pontiac Trans Am dotata di intelligenza artificiale con persino la possibilità

di provare sentimenti e stati d'animo diversi. Ma non solo: la vettura era in grado di comunicare con un sintetizzatore vocale, poteva collegarsi con tutti i data-base americani per scoprire in breve tempo informazioni su chiunque, era in grado di calcolare qualunque

Non più un sistema a sé stante ma un nodo all'interno di una rete: è l'auto del futuro, sempre connessa e in grado di fornirci tutte le informazioni che ci servono. Ma anche di interagire con le altre vetture e con le infrastrutture a terra e magari, come Supercar, di guidare da sola.

L'AUTO DEL FUTURO VIAGGERA' IN RETE

di ARSENIO SPADONI

percorso tra due località valutando la strada migliore e disponeva persino di numerosi sensori in grado di osservare e analizzare l'ambiente circostante. Trent'anni dopo, molte delle fantastiche caratteristiche della vettura sono realtà o lo saranno tra breve. Una moderna automobile dispone di circa

200 sensori in grado di misurare qualsiasi cosa, dalla pressione dei pneumatici alla temperatura interna o esterna, dalla presenza di ghiaccio sulla strada alla forza con cui si verifica un ipotetico incidente. Una Mercedes di ultima generazione utilizza circa 85 microprocessori ma anche la più economica vettura

al mondo, la Tata Nano, ne monta almeno una dozzina. I navigatori satellitari con guida vocale sono utilizzati quotidianamente da milioni di automobilisti mentre i sistemi di localizzazione wireless sono in grado di stabilire la posizione della vettura in caso di furto; sullo stesso principio funzionano anche i

sistemi di gestione e controllo delle flotte di autocarri e autobus nonché le assicurazioni con contratti *pay-per-use*: più si utilizza la vettura, più alto è il premio e viceversa; i *telepass* consentono di transitare ai caselli autostradali senza fermarsi, addebitando l'importo del pedaggio sul conto corrente

Auto ed elettronica: un binomio sempre più stretto

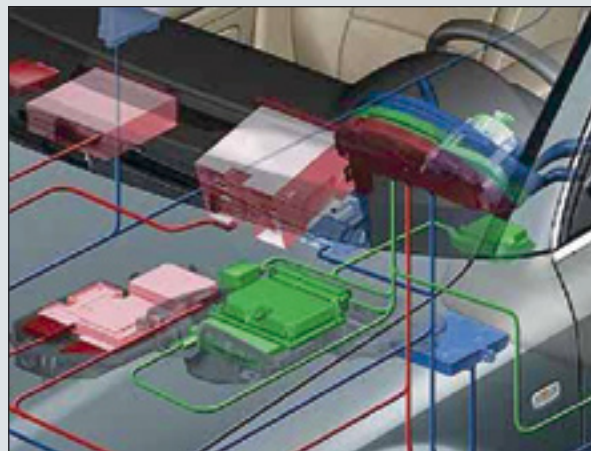
Molto prima che le automobili si avviassero a diventare dei veri e propri nodi wireless all'interno di una rete, all'elettronica erano state affidate numerose funzioni in campo automobilistico al punto che oggi una moderna automobile utilizza dai 50 ai 100 microcontrollori, oltre ad alcune centinaia di sensori. Soltanto un massiccio impiego dell'elettronica può consentire all'industria automobilistica di soddisfare i requisiti di sicurezza, il livello di confort e il contenimento dei consumi e delle emissioni nocive richiesti dal mercato e dalle norme vigenti, regole che si fanno ogni giorno più stringenti e che comporteranno in futuro un ulteriore aumento dei sistemi elettronici di bordo. L'impiego sempre più massiccio dell'elettronica è fondamentale anche per una riduzione dei tempi di produzione ed un miglioramento del servizio di assistenza e manutenzione che diventerà sempre più automatizzato. Attualmente l'elettronica in auto svolge numerose funzioni:

- controllo degli organi meccanici e di trazione (motore, freni, ruote: ciò al

fine di ottenere le migliori prestazioni in ogni condizione di marcia, col minore consumo di carburante e la più bassa emissione di sostanze inquinanti);

- eliminazione e semplificazione di organi meccanici (es: cavo dell'acceleratore e, in futuro, piantone del volante e valvole di aspirazione e scarico);
- sostituzione di attuatori elettromeccanici con analoghi dispositivi allo stato solido (insensibili alle sollecitazioni meccaniche e più rapidi);
- sostituzione dei sistemi di segnalazione luminosa con led allo stato solido (più robusti e con una durata maggiore);
- aggiunta di servocomandi per sostituire l'azione manuale di conducente e passeggeri (dagli alzacristalli agli specchietti retrovisori, dai sedili elettrici al tettuccio apribile);
- implementazione di funzioni di assistenza alla guida (sensori di retromarcia, sistemi di prevenzione dei colpi di sonno e di controllo della marcia);
- integrazione di sistemi di intrattenimento (lettori audio/video);
- implementazione di sistemi wireless di infomobilità (radio/TV, navigatori satellitari, sistemi di comunicazione/controllo e sicurezza basati su reti GSM, accesso a Internet, tecnologie di scambio dati con altri veicoli e con infrastrutture a bordo strada, sistemi intelligenti di controllo della marcia e di prevenzione degli incidenti, per finire con i dispositivi di guida assistita e perfino automatica).

E' proprio quest'ultima serie di innovazioni che trasformerà la nostra autovettura in un mezzo completamente differente da quello a cui siamo stati abituati oggi. L'elettronica di bordo può essere considerata come un sistema a intelligenza distribuita composto da



tantissimi sottosistemi embedded che prendono il nome di Ecu (Electronic Control Unit) e che comunicano tra loro mediante un apposito bus. Ciascuna Ecu svolge una specifica funzione utilizzando dei sensori per rilevare grandezze analogiche (spostamenti, temperature, velocità, ecc.), uno o più microcontrollori per elaborare le informazioni e dei driver di potenza per azionare specifici servocomandi. Tutte le Ecu comunicano tra loro per mezzo di una rete che nella maggior parte dei casi utilizza il protocollo CAN (Controller Area Network): si tratta di un bus seriale di comunicazione dati progettato per applicazioni real-time. Nato

bancario o sulla propria carta di credito. Una volta semplici dispositivi meccanici, le automobili stanno diventando sempre più digitali. Le automobili connesse in grado di accedere ai sistemi di navigazione satellitare e alle reti di comunicazione e, molto presto, anche di comunicare con gli altri veicoli, trasformeranno il nostro modo di guidare, eliminando la possibilità di perderci, restare bloccati nel traffico o coinvolti in incidenti stradali. Una maggior capacità

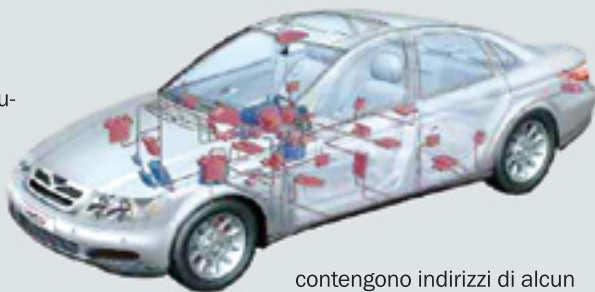
di connessione consentirà anche di accedere più facilmente ai contenuti multimediali diversi dai canali radiofonici, rendendo più confortevoli e meno noiosi i tragitti in automobile. È stato calcolato che un automobilista trascorre mediamente circa 45 ore al mese in auto: se il conducente e, soprattutto, i passeggeri potessero destinare questo tempo ad altre attività, perlomeno il livello di stress calerebbe. Le nuove tecnologie applicate all'automobile aprono nuove

opportunità e modelli di business, molti dei quali ancora tutti da esplorare. Le tariffe assicurative *pay-per-use* sono un esempio recente così come il *car-pooling* ed il *car-sharing* che hanno trovato nella localizzazione veicolare la possibilità di razionalizzare la logistica e, di conseguenza, i costi. Ci sono tantissimi nuovi modelli di business che ruotano attorno alle automobili connesse, perfino nel campo della pubblicità basata sulla localizzazione, con offerte mirate in funzione del

originariamente per l'industria automobilistica, il CAN-bus si è diffuso anche nell'automazione industriale per le sue caratteristiche di robustezza ed affidabilità. Questo protocollo è stato sviluppato da Robert Bosch nel lontano 1986 su richiesta della Mercedes. L'esigenza di far comunicare i molti dispositivi elettronici presenti all'interno delle automobili (Antilock Braking System, Traction Control, Air Conditioning Control e chiusura centralizzata sono solo alcuni esempi) nonché la loro complessità, avrebbe portato ad un aumento insostenibile di collegamenti dedicati ed una duplicazione dei sensori necessari, con conseguente aumento dei costi di produzione e soprattutto notevole ingombro fisico. Per questi motivi è nato il CAN-bus, il quale consente a controllori, sensori e attuatori di comunicare tra loro ad una velocità fino a 1Mbit/sec, offrendo anche:

- bassi costi di progettazione e implementazione;
- funzionamento in ambienti ostili;
- facilità di configurazione e modifica;
- rilevamento automatico degli errori di trasmissione.

Il protocollo CAN si colloca all'interno della pila ISO/OSI nei livelli 1 fisico e 2 data link, lasciando totale libertà per il livello 7 applicazione. Il nucleo del protocollo è nel livello data link, e nella particolare politica di accesso al mezzo di trasmissione. I pacchetti trasmessi da una stazione CAN non



contengono indirizzi di alcun genere e al loro posto troviamo un identificatore del contenuto del messaggio (giri al minuto, temperatura motore, temperatura abitacolo ...) unico sull'intera rete. Un nodo ricevitore può così verificare il contenuto del messaggio e filtrare i soli pacchetti a cui è interessato, ignorando gli altri. Questo modo di operare è detto Multicast. Per garantire la comunicazione real-time il CAN ricorre ad un metodo di accesso al mezzo CSMA/CD modificato: due o più stazioni che iniziano a trasmettere competono per l'accesso al bus sul loro valore di priorità determinato proprio dall'identificatore. Quello con il valore numerico più basso vince la competizione per il canale. Ogni conflitto viene risolto in accordo con il meccanismo del wired-and secondo il quale lo stato dominante (0 logico) sovrascrive lo stato recessive (1 logico). Ciascun nodo è in grado di rilevare il proprio malfunzionamento e di autoescludersi dal bus se questo è permanente. Questo è uno dei meccanismi che consentono alla tecnologia CAN di mantenere la rigidità delle temporizzazioni, impedendo che un solo nodo metta in crisi l'intero sistema. La larga diffusione del protocollo CAN ha determinato un'ampia disponibilità di chip ricetrasmittitori, di microcontrollori che integrano porte CAN nonché di tools di sviluppo.

luogo in cui ci troviamo, dell'ora, del nostro stile di guida e delle condizioni del nostro veicolo: se, ad esempio, i sensori rilevano che siamo quasi a secco, il localizzatore ci indicherà la stazione di servizio, tra quelle in zona, che pagherà per questo tipo di servizio (magari, se è anche ora di pranzo, decantandoci le delizie del ristorante annesso o proponendoci qualche oggetto in promozione). Dobbiamo abituarci a pensare all'automobile non più come a un sistema autonomo

ma come un nodo all'interno di una rete: solo così riusciremo ad anticipare i futuri scenari.

LA SVOLTA DEL GPS

Sicuramente, il momento in cui ci si è resi conto che anche per l'automobile le cose sarebbero cambiate radicalmente, sono state le prime applicazioni con il GPS (*Global Positioning System*) e, subito dopo, l'integrazione di questa tecnologia con i sistemi di comunicazione cellulare (GSM, GPRS E UMTS). Nato prevalen-

temente per scopi militari e per la navigazione marittima, questo sistema ha reso possibile, grazie anche alla miniaturizzazione, alla riduzione dei costi ed alla disponibilità di data-base stradali dettagliati, lo spostamento da una località ad un'altra senza problemi, la guida in zone sconosciute, il raggiungimento di un qualsiasi punto di interesse (albergo, ristorante, stazione di servizio, aeroporto, ecc.) anche senza conoscere l'indirizzo. Inizialmente quasi tutti i navigatori satellitari erano installati in maniera fissa sulla vettura, costavano tantissimo ed erano difficili da aggiornare. Successivamente il mercato si è orientato verso i modelli portatili (da fissare sul parabrezza o sul cruscotto) ed è stato un vero e proprio boom: attualmente si calcola che nel nostro paese un automobilista su tre possieda un navigatore satellitare. Contemporaneamente nascevano le prime applicazioni wireless basate sull'impiego dei GPS e sulla possibilità di conoscere - da remoto - la posizione della vettura. A questo dato (la posizione) si sono aggiunte col tempo numerose altre informazioni (dalla velocità all'accelerazione, dai parametri di funzionamento del motore alla temperatura esterna) che hanno reso possibile la nascita e lo sviluppo di aziende di servizi di vario genere. E' evidente che la raccolta e l'aggregazione di un elevato numero di informazioni possono essere utilizzate per tantissimi scopi. Nel nostro paese il caso più significativo è quello di Octo Telematics che, tramite una piccolissima unità remota GPS/GPRS installata nella vettura, gestisce i servizi assicurativi *pay-per-use* di numerose compagnie; attualmente la società ha accordi con 36 compagnie assicurative, tra le quali

Sistemi a prova di ingorgo

Il 18 aprile scorso è entrato in funzione presso il CCISS (Centro Coordinamento Informazioni Sicurezza Stradale del Ministero dei Trasporti) la nuova piattaforma IMA Datex, uno dei più avanzati sistemi europei di gestione delle notizie sul traffico. La nuova Centrale gode di spazi attrezzati per circa 1000 metri quadrati, oltre 60 operatori, un sistema informativo all'avanguardia, una piattaforma software tra le più avanzate a livello mondiale in materia di infomobilità. Questa nuova struttura dovrebbe garantire ciò che in passato, spesso, il CCISS non è stato in grado di fornire, ovvero delle tempestive informazioni in caso di ingorghi, incidenti, deviazioni. Quante volte ci è capitato di ascoltare messaggi relativi ad una coda dopo che c'eravamo finiti in mezzo da almeno una o due ore? Oppure, al contrario, la segnalazione di un ingorgo che, al nostro arrivo, era già stato risolto? La maggior parte di questi ritardi era dovuta al fatto che le informazioni venivano trasmesse telefonicamente ed immesse a mano nel sistema. Oggi, grazie anche alla nuova piattaforma, le informazioni vengono elaborate in maniera automatica in virtù del fatto che ormai, nella maggior parte dei casi, provengono direttamente da sensori, telecamere e soprattutto dai dispositivi wireless di cui molti mezzi sono muniti. Attualmente nel nostro paese moltissime vetture sono dotate di localizzatori GPS/GPRS utilizzati quali antifurto, nei sistemi assicurativi pay-per-use, per l'assistenza tecnica remota, eccetera. La gestione dei dati provenienti dalle vetture fa capo a pochissime grandi società la più importante delle quali è Octo Telematics che gestisce circa 800.000 sistemi remoti. Questa società dispone di una piattaforma che consente, oltre a fornire i servizi al singolo utente, anche di avere - in maniera completamente anonima e in tempo reale - la situazione complessiva della velocità



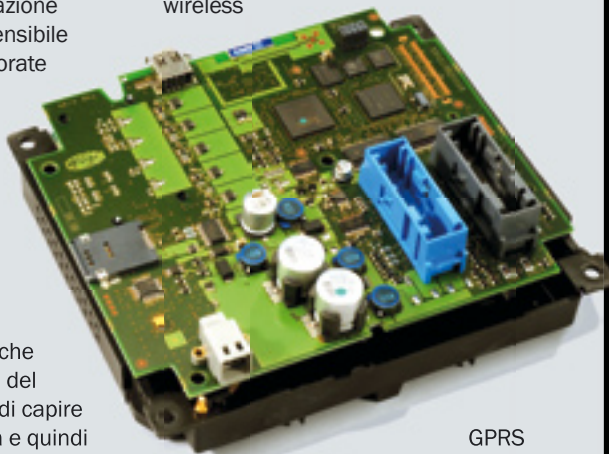
i primi cinque gruppi europei. Complessivamente gli abbonati ai servizi di Octo Telematics sono oltre 800.000 con l'obiettivo di un milione di utenti per fine anno. Le tipologie delle

polizze variano da compagnia a compagnia ma sono sempre legate ai dati forniti dal GPS di bordo: più strada si percorre e più si paga, anche se l'importo può variare in funzione di dove

andiamo, dove parcheggiamo e del nostro stile di guida. Ma i dati provenienti dalle vetture gestite da Octo Telematics vengono anche utilizzati (in forma anonima e aggregata) per sapere

con cui si muovono i mezzi controllati. Essendo il numero di vetture abbastanza significativo, questi dati forniscono un quadro esatto del traffico sulle strade italiane. Queste informazioni giungono in tempo reale ai fornitori dei vari servizi, dal CCISS a Infoblu, che le integrano con gli altri dati e provvedono, sempre automaticamente, a renderle disponibili per gli utenti tramite Internet, radio, TV e TMC. Octo Telematics mette a disposizione gratuitamente queste informazioni mediante il proprio sito Internet segnalando anche quanti sono i mezzi attivi in quel momento e il numero di rilevazioni elaborate per ricavare i dati finali. A lato riportiamo due schermate che si riferiscono alla situazione generale della rete autostradale italiana ed a quella, più specifica, di un tratto dell'Autostrada dei Laghi in Lombardia. La situazione del traffico è facilmente comprensibile grazie all'impiego di mappe colorate che vanno dal verde, al blu, al rosso, al nero. Se un tratto di strada è contraddistinto dal colore verde significa che le autovetture viaggiano su quel tratto ad oltre 90 chilometri orari e quindi sul percorso non ci sono problemi; viceversa se il colore è rosso, le autovetture sono quasi ferme, a significare che c'è un problema. Le dimensioni del tratto rosso consentono anche di capire quant'è la lunghezza della coda e quindi l'entità del problema. Sottolineiamo ancora una volta che tutto ciò avviene in maniera completamente automatica ed i dati ricavati sono pertanto in tempo reale: un passo avanti enorme rispetto al passato, che sarà ancora più significativo nel momento in cui il numero dei veicoli facenti parte di questa "rete" aumenterà in maniera significativa. E' evidente che se i mezzi fossero molti di più (attualmente la media è di una vettura "connessa" ogni 30), le informazioni sarebbero molto più precise e potrebbero essere utilizzate in maniera significativa anche per tratti stradali con scarso traffico. Le informazioni ottenute con questi sistemi possono rappresen-

tare anche una utilissimo archivio per l'elaborazione di previsioni sul traffico, specifiche per un determinato giorno e ora di una specifica zona. E' evidente che più numerosi sono i dati, più precise saranno le previsioni. Come detto in precedenza, l'utilizzo dei dati provenienti dalle unità wireless ha consentito di migliorare notevolmente questo genere di servizi; il passo successivo sarà sicuramente quello di utilizzare la stessa unità wireless per aggiornare i sistemi di navigazione. In questo modo si potrà creare rapidamente un circolo virtuoso che migliorerà il servizio. Se infatti i localizzatori fossero muniti di sistema wireless



GPRS
(... i primi

sistemi stanno già per essere prodotti), un unico dispositivo potrebbe sia fornire al server centrale le informazioni sul traffico, sia ricevere gli avvisi riguardanti i problemi che eventualmente dovessero verificarsi (incidenti, code, interruzioni, ecc.) lungo il percorso impostato. Oltre, naturalmente, ad effettuare un aggiornamento delle mappe, dei punti di interesse e delle postazioni degli autovelox mobili. Questo potrebbe essere solo l'inizio, infatti basterebbe connettere il sistema al CAN-bus per implementare nuovi altri servizi relativi soprattutto alla manutenzione del veicolo.



questi dati vengono "girati" a numerose altre società ed enti tra cui anche il CCISS e Infoblu, che li integrano con le altre informazioni in loro possesso per poi fornire i bollettini sul traffico diffusi tramite radio, TV e Internet. Segnaliamo che i dati originali per tutta Italia di Octo Telematics sono disponibili gratuitamente sul sito della società in forma grafica, compresi i dati relativi al numero di utenti attivi e di rilevazioni effettuate (www.octotelematics.it). La società, in base ai dati storici è anche in grado di effettuare previsioni per lo stato del traffico di un particolare giorno, in un'ora particolare e per una specifica area.

Utilizzano la stessa tecnica, società come Openstreetmaps (www.openstreetmaps.org) stanno creando mappe open-source raccogliendo i dati forniti dai GPS di bordo. Queste informazioni sono fondamentali anche per i servizi di car-sharing che sempre di più si stanno diffondendo nel mondo e in Italia. Per ottimizzare questo tipo di attività è fondamentale l'uso della tecnologia: senza i sistemi wireless sarebbe necessario un parco auto tre volte superiore per offrire lo stesso tipo di servizio agli abbonati. Il Car Sharing (www.isccarsharing.it) offre un approccio inedito alle quattro ruote: si compra l'uso effettivo del mezzo anziché il mezzo stesso. Un'alternativa per chi percorre pochi chilometri o per chi non intende rinunciare all'auto di proprietà, ma deve fare i conti con esigenze di mobilità che oggi sono risolte con

a che velocità viaggiano le vetture riuscendo così a ricostruire in tempo reale e in maniera completamente automatica un quadro esatto del traffico sulle strade italiane. Il numero di vet-

ture è già sufficiente per avere un dato abbastanza preciso del traffico anche se è evidente che all'aumentare dei veicoli gestiti le informazioni saranno sempre più precise. Attualmente

La tecnologia TMC

Acronimo di Traffic Message Channel, è una tecnologia che permette di visualizzare sui navigatori satellitari predisposti le informazioni sul traffico (interruzioni, code, ecc.) in maniera completamente automatica ed in real-time. Nel nostro paese le informazioni vengono fornite principalmente da Infoblu (tramite RTL102.5) e CCISS (tramite le fre-



quenze della RAI). Le informazioni di queste due società provengono dalle fonti usuali, ovvero dai dati di OCTO Telematics, dalla rete di telecamere e sensori della società Autostrade, ecc. Le informazioni vengono trasmesse su frequenze FM, attraverso il canale RDS, e utilizzano il protocollo TMC che definisce le regole di codifica. Ogni segnalazione viene inviata come messaggio indipendente, costituito da un codice evento, un codice localizzazione ed alcune informazioni aggiuntive, come l'ora di scadenza della segnalazione, ossia l'orario al quale si può ritenere che la segnalazione non sia più da prendere in considerazione. Sulla base dei dati ricevuti il localizzatore può modificare il percorso calcolato al fine di evitare i problemi segnalati. I messaggi sono codificati secondo lo standard *Alert C* e possono eventualmente essere crittografati, secondo le specifiche definite dal protocollo TMC. Il sistema ricevente decodifica i messaggi e mostra le segnalazioni sulla mappa, attraverso il protocollo TMC e le tabelle delle localizzazioni già integrate nelle mappe dei principali fornitori. Inoltre il messaggio può essere

inviato anche in forma testuale, tradotto nella lingua dell'utente. Le segnalazioni sono generate dal servizio TMC.IT (www.tmcitaly.it) ed inoltrate tramite le frequenze di RTL 102.5, emittente che copre circa l'80% del territorio Italiano. Il servizio fornisce informazioni sullo stato del traffico ma anche altre notizie che possono condizionare il viaggio (situazioni meteo, incidenti, chiusure di strade). Le informazioni di traffico fornite da Infoblu sono anche disponibili via GPRS: il servizio al momento è disponibile solo per i terminali Nokia.



l'acquisto di una seconda o terza macchina. La ricerca del veicolo desiderato è resa più rapida e semplice dal sistema di localizzatore di bordo che consente alla centrale operativa di avere sempre sotto controllo la posizione delle varie vetture rendendo più efficace il sistema di prenotazione del mezzo da parte dell'abbonato, via Internet o telefono. L'auto viene aperta dall'utente tramite una chiave Rfid che viene avvicinata al cruscotto. La possibilità di sapere esattamente la posizione di ogni singola autovettura dell'intero parco macchine consente una elevata flessibilità. Il *Car Sharing* è uno dei nuovi strumenti della mobilità sostenibile: è stato calcolato che una vettura di questa flotta equivale - in quanto a risorse utilizzate - a 20 vetture tradizionali di utenti con percorrenza non superiore a 10.000 chilometri annui. In altre parole se il car sharing fosse diffuso capillarmente il numero di vetture immatricolate potrebbero ridursi di 20 volte e lo stesso utente potrebbe godere degli stessi servizi garantiti da un'automobile di proprietà con un risparmio di 3-4.000 euro l'anno. I servizi di localizzazione remota sono attualmente implementati anche nelle vetture messe a disposizione dei clienti dalle principali società di autonoleggio: al momento solamente per prevenire i furti ma, a breve, anche per offrire una serie di servizi innovativi. Anche nel campo dei servizi di telediagnosi e teleassistenza stanno per prendere il via numerosi progetti promossi dalle principali case automobilistiche.

Con tutte queste iniziative è evidente che uno dei problemi che sorgeranno nei prossimi

anni sarà quello della compatibilità dei vari sistemi. Da questo punto di vista Magneti Marelli (www.magnetimarelli.com), società leader in Italia per la fornitura di sistemi automotive hi-tech, ha recentemente stretto due accordi: uno con Navteq (leader mondiale di dati cartografici digitali per la navigazione satellitare e soluzioni basate sulla localizzazione) e l'altro proprio con Octo Telematics. Nel primo caso l'accordo si propone di compiere un passo tecnologico verso il traguardo della navigazione integrata a bordo auto (*Automotive Integrated Navigation*), una navigazione, cioè, realmente interconnessa con il veicolo in tutte le sue parti e impostazioni e adeguata alle caratteristiche fisiche dell'ambiente che circonda il veicolo. Un esempio è il *"Green Routing"*, un primo passo verso la navigazione ecologica. Questa applicazione consente di definire itinerari "verdi", che richiedono cioè la minore quantità possibile di carburante il consumo del quale viene determinato in base alla conformazione del veicolo, alle informazioni sul traffico e alle caratteristiche della strada integrate nelle mappe Navteq: curve, dislivelli, limiti di velocità, eccetera. Grazie ad una connessione wireless, Magneti Marelli ha sviluppato un sistema in grado di calcolare l'itinerario più adatto combinando i dati relativi al motore del veicolo con le informazioni presenti nelle mappe. Un altro esempio è il *"Green Driving"*, una forma di guida responsabile con adattamento dei parametri in tempo reale: il sistema integrato a bordo auto di Magneti Ma-



relli consente l'adattamento di parametri quali i comandi dell'iniezione e dell'acceleratore, oltre alla gestione intelligente del cambio. Di conseguenza, è possibile ottimizzare tutti i sistemi integrati, e il conducente ha l'opportunità di guidare in maniera più responsabile sulla base dei suggerimenti sulla velocità e sul cambio. Ovviamente questi nuovi sistemi offrono connettività in tempo reale consentendo l'aggiornamento di applicazioni e software, oltre al download di aggiornamenti cartografici e contenuti dinamici quali ad esempio le informazioni sul traffico. L'architettura del software consente anche l'aggiornamento di nuovi contenuti all'interno dell'applicazione di navigazione. Magneti Marelli ha previsto anche l'introduzione di altre tecnologie avanzate che consentono di accedere in modo migliore alle diverse fonti di dati, grazie soprattutto all'utilizzo di una telecamera che funge da "terzo occhio" per aiutare il conducente a riconoscere la segnaletica. Il sistema Magneti Marelli combina i dati cartografici Navteq e le informazioni fornite dalla telecamera per comunicare al guidatore informazioni precise sulla segnaletica

stradale. Ma ci sono anche i fanali "intelligenti" che si orientano sia in verticale che in orizzontale in base alle informazioni trasmesse dalla cartografia migliorando automaticamente l'illu-

minazione in prossimità delle curve e dei dossi.

Il secondo accordo, quello con Octo Telematics, prevede la convergenza dei rispettivi protocolli di comunicazione e piattaforme telematiche. L'obiettivo è quello di favorire, attraverso l'utilizzo di uno standard comune, lo sviluppo del mercato legato ai dispositivi infotelematici per le automobili e la relativa fruizione dei servizi telematici assicurativi.

Col nuovo protocollo di comunicazione la piattaforma infotelematica per autoveicoli di Magneti Marelli sarà in grado di dialogare con il Centro servizi di Octo Telematics; anche sulle vetture dotate di "T-Box" Magneti Marelli, sarà quindi possibile fruire dei servizi assicurativi di Octo Telematics. In base all'accordo, inoltre, le società metteranno a punto un protocollo di comunicazione su canale bluetooth per far dialogare anche la scatola telematica



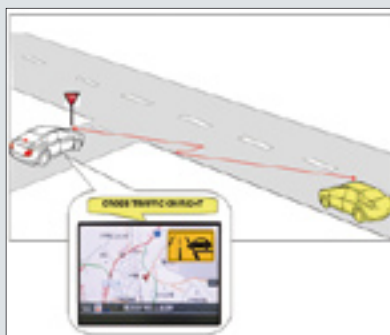
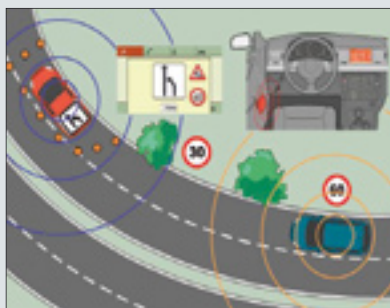
“Clear Box” di Octo Telematics con il navigatore portatile di Magneti Marelli, in modo da rendere disponibili i servizi di Octo Telematics anche su PND (*Portable Navigation Device*). La nascita di questi nuovi sistemi renderà presto obsoleto l'unico sistema automatico di visualizzazione sui navigatori satellitari (che debbono essere predisposti) delle informazioni sul traffico (interruzioni, code, ecc.): il TMC, acronimo di *Traffic Message Channel*. Questa tecnologia utilizza le informazioni che vengono fornite da Infoblu (tramite RTL102.5) e CCISS (tramite le frequenze della RAI) e che vengono inviate mediante il sistema RDS della radio FM. Sulla base dei dati ricevuti il localizzatore può modificare il percorso calcolato al fine di evitare i problemi segnalati.

VEICOLI CHE PARLANO CON ALTRI VEICOLI

Il passo successivo nell'evoluzione dell'auto digitale è quello di fare comunicare tra loro le vetture con una rete V2V (*Vehicle-to-Vehicle*) e con le infrastrutture a bordo strada (caselli, semafori, ecc.) con una rete V2I (*Vehicle-to-Infrastructure*). A questo scopo è stato messo a punto un nuovo standard denominato DSRC (*Dedicated Short Range Communication*) una sorta di Wi-Fi per automobili che si basa sullo standard IEEE 802.11a e che opera alla frequenza di 5,9 GHz. Il DSRC consente elevate velocità di trasferimento dati tra veicoli entro un raggio di 200 metri e data-rate più bassi (utilizzati per segnalazioni di emergenza e sicurezza) fino ad una distanza massima di un chilometro tra veicoli nonché tra veicoli e infrastrutture a bordo strada. Come abbiamo visto,

L'Italia è stato uno dei primi paesi al mondo a sperimentale e mettere in pratica la tecnologia V2I (vehicle-to-infrastructure, da veicolo ai sistemi a terra), anche se solo per il pagamento del pedaggio. Stiamo parlando del Telepass, attivo da quasi 20 anni ed installato ad oggi su più di sei milioni di veicoli.

Finora per quanto riguarda la sicurezza ed il confort dei passeggeri sono stati escogitati e resi obbligatori i sistemi che tutti noi conosciamo, dalle cinture agli air bag, dagli esp ai sistemi frenanti con abs. Poco o nulla è stato invece fatto sfruttando le possibilità offerte dai sistemi di comunicazione radio tra veicoli o tra questi e le infrastrutture lungo il bordo strada. Negli ultimi anni, tuttavia, i progetti in questo settore si sono moltiplicati ed è probabile che in poco tempo assisteremo ad una rapida diffusione dei sistemi di comunicazione automatica V2V (vehicle-to-vehicle, da veicolo a



questa tecnologia viene utilizzata da tempo per il pagamento automatico dei pedaggi ma è evidente che le possibili applicazioni sono ben altre, specie se consideriamo lo scambio dati tra vetture. La tecnologia DSRC potrebbe essere utilizzata, ad

veicolo) e V2I, da veicolo a infrastruttura a terra. A questo scopo è stato messo a punto un nuovo standard denominato DSRC (*Dedicated Short Range Communication*) una sorta di Wi-Fi per automobili che si basa sullo standard IEEE 802.11a e che opera alla frequenza di 5,9 GHz. Il DSRC consente elevate velocità di trasferimento dati tra veicoli entro un raggio di 200 metri e data-rate più bassi (utilizzati per segnalazioni di emergenza e sicurezza) fino ad una distanza massima di un chilometro tra veicoli nonché tra veicoli e infrastrutture a bordo strada. Come abbiamo visto, questa tecnologia viene utilizzata da tempo per il pagamento automatico dei pedaggi ma è evidente che le possibili applicazioni sono ben altre, specie se consideriamo lo scambio dati tra vetture. La tecnologia DSRC potrebbe essere utilizzata, ad esempio, per avvisare chi segue di una improvvisa frenata a causa di un ostacolo sulla strada o dello scoppio di una ruota o, ancora, della presenza di ghiaccio: in questo modo le vetture che seguono (e che in molti casi sono fuori visuale) potrebbero rallentare ed evitare il pericolo. Altre applicazioni prevedono la segnalazione dell'avvicinarsi di veicoli di emergenza (polizia, vigili del fuoco, ecc.), il cui DSRC potrebbe, tra l'altro, agire anche sugli impianti semaforici per dare la priorità al loro passaggio. I risultati offerti dalla tecnologia V2V dipendono in gran parte dal numero di veicoli dotati di questo sistema: se sono pochissimi gli effetti non possono che essere scarsi. Da studi condotti è tuttavia emerso che buoni risultati si possono ottenere con una diffusione di appena il 3-5% delle vetture circolanti e con la condivisione di soli quattro parametri: posizione, velocità, direzione e ora.

Per quanto riguarda la tecnologia V2I, gli studi riguardano la possibilità di

esempio, per avvisare chi segue di una improvvisa frenata a causa di un ostacolo sulla strada o dello scoppio di una ruota o, ancora, della presenza di ghiaccio: in questo modo le vetture che seguono (e che in molti casi sono fuori visuale) potrebbero

La tecnologia DSRC e i sistemi V2V e V2I

interagire con gli impianti semaforici e più in generale la possibilità di ricevere con congruo anticipo avvisi di pericolo e segnalazione di allarme: ad esempio, prima di approcciare un incrocio o attraversare un passaggio a livello. Purtroppo dotare di infrastrutture di questo tipo tutta la rete stradale è molto costoso e soprattutto inutile se poi i veicoli non dispongono delle apparecchiature in grado di sfruttare queste risorse. E' stato calcolato che rendere "intelligenti" col DSRC tutti gli impianti semaforici degli USA costerebbe la bellezza di 4,5 miliardi di dollari. Nel settore dei sistemi di comunicazione V2V, una delle Case automobilistiche che ha investito di più in ricerca e sviluppo è stata la General Motors che ha messo a punto e sta sperimentando un hardware relativamente semplice che sfrutta la tecnologia DSRC. Auto dotate di questa tecnologia possono comunicare e scambiarsi informazioni come posizione e velocità: i guidatori possono essere avvisati in anticipo se un altro veicolo è fermo in un punto dove è difficile o impossibile vederlo o si sta approssimando allo stesso incrocio. A seconda della situazione, il sistema trasmette questi allarmi tramite messaggi visivi e sonori oppure facendo vibrare il sedile del guidatore. Gli autoveicoli si collegano fra loro quando sono a po-

che centinaia di metri l'uno dall'altro scambiandosi informazioni sulla posizione, velocità, accelerazione e direzione di marcia.

Scopo ultimo di questi sistemi è quello di ridurre al minimo la possibilità di incidenti, specie in prossimità di incroci pericolosi, traffico intenso, strade dissestate e scarsa visibilità.

Di questi temi si occupa anche "Easy Rider" il progetto di Magneti Marelli primo classificato tra quelli che hanno partecipato al programma "Industria 2015". Easy Rider prevede la realizzazione di un sistema per l'integrazione di infrastrutture stradali, veicoli e servizi per il miglioramento dell'infomobilità e della sicurezza stradale.

La parte più interessante riguarda i segnali stradali intelligenti, in grado di trasmettere informazioni all'automobilista insieme al monitoraggio del profilo stradale da comunicare in tempo reale ai veicoli in transito. Il progetto che vede come capofila Magneti Marelli, verrà realizzato nei prossimi anni da diversi consorzi, composti in tutto da 250 imprese e 100 centri di ricerca delle più importanti società del settore (Fiat, Anas, ST Microelectronics e Art).

Magneti Marelli è anche impegnata con Telecom Italia in "tema.mobility", un consorzio partecipato dalle due aziende al 50%, che ha l'obiettivo di creare soluzioni



innovative per il collegamento delle autovetture alle reti cellulari e fornire nuovi servizi di infomobilità a bordo. Grazie alla reciproca collaborazione, le due aziende hanno realizzato un'applicazione innovativa che, attraverso una "telematic box" dotata di una SIM card cellulare, interagisce con il guidatore tramite il vivavoce o un display dedicato e collega l'auto alle reti UMTS ad alta velocità, garantendo la connessione alla larga banda mobile durante il viaggio e il conseguente accesso a servizi di infomobilità.

La soluzione tecnologica di tema.mobility si caratterizza per essere "aperta" a tutti i service provider in grado di fornire nuove applicazioni per l'automobilità, proprio per favorire lo sviluppo di servizi innovativi pensati per il mondo dei trasporti pubblici e privati, per le assi-

curazioni automobilistiche e per la gestione delle flotte di veicoli aziendali.

Per quanto riguarda il trasporto pubblico e privato, il guidatore avrà a disposizione, servizi informativi su limitazioni di accesso, presenza di ZTL, segnalazione di eventi che interferiscono con il normale flusso di traffico, la possibilità di pagare pedaggi e parcheggi, la telediagnosi del veicolo in caso di guasto, servizi di sicurezza come la chiamata di emergenza, l'e-call e l'antirapina, compresa la localizzazione dell'autovettura in caso di furto.

L'applicazione permette inoltre di memorizzare i dati di utilizzo del veicolo consentendo ai titolari di polizze assicurative "pay per use" di ridurre i costi e alle assicurazioni di avere un rapporto sempre più "personalizzato" con i clienti.

rallentare ed evitare il pericolo. Altre applicazioni prevedono la segnalazione dell'avvicinarsi di veicoli di emergenza (polizia, vigili del fuoco, ecc.), il cui DSRC potrebbe, tra l'altro, agire anche sugli impianti semaforici per dare la priorità al loro passaggio.

I risultati offerti dalla tecnologia V2V dipendono in gran parte dal numero di veicoli dotati di questo sistema: se sono pochissimi gli effetti non possono che essere scarsi. Da studi condotti è tuttavia emerso che buoni risultati si possono ottenere con

una diffusione di appena il 3-5% delle vetture circolanti e con la condivisione di soli quattro parametri: posizione, velocità, direzione e ora.

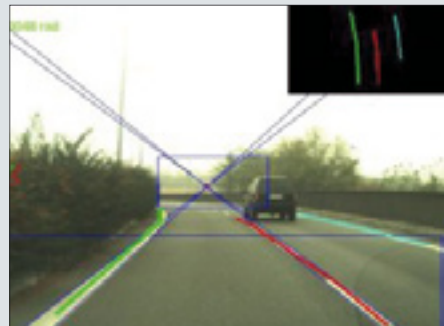
Per quanto riguarda la tecnologia V2I, gli studi riguardano la possibilità di interagire con

L'ultima frontiera: l'auto senza conducente

Dopo i sistemi di assistenza e informazioni sul traffico fruibili dal conducente, dopo le informazioni in tempo reale che interagiscono direttamente con il computer di bordo dei veicoli e i sistemi di scambio di informazioni Veichle to Veichle, l'auto sembra essere giunta al passo finale della sua evoluzione: la circolazione senza conducente. I tempi sono ormai maturi e ce lo dimostra BRAiVE, una

leggere correttamente le informazioni raccolte dalle telecamere e dai sensori di bordo potrebbe spaventarci ed è difficile pensare che una macchina possa muoversi senza la supervisione dell'uomo. E in effetti, come per tutti i sistemi automatici c'è il timore di cosa accadrebbe se un sensore fornisse un'indicazione errata o la CPU elaborasse scorrettamente un dato o, ancora, se un attuttore (ad

che potrebbero derivare da un errore di un sistema automatico di guida: statistiche alla mano, il 93 % degli incidenti (che causano 1,6 milioni di morti in tutta Europa da una trentina d'anni ad oggi) deriva da errori o comportamenti rischiosi del conducente e solo il 7 % è imputabile a guasti, cedimenti strutturali o cause non chiarite. Il progetto BRAiVE nasce essenzialmente con l'obiettivo di limitare gli incidenti stradali e propone due soluzioni: supportare il conducente distratto o troppo stanco avvertendolo se si sta avvicinando troppo a un'auto in colonna, se un veicolo lo sta superando o gli si sta avvicinando lateralmente, se c'è un ostacolo o un pedone sbucato all'improvviso sulla carreggiata; sostituire il conducente alla guida, governando da sé il veicolo. La Sonata preparata nell'ambito del progetto BRAiVE è un'auto capace anche di fare a meno del conducente, di procedere in sicurezza nella nebbia, suonare il clacson per avvertire gli altri conducenti o i pedoni, ma anche sterzare e frenare fino a fermarsi. Per gestire tutti questi sistemi l'auto-robot si affida a 10 telecamere, 3 laser a singolo piano, 1 laser a 4 piani, 16 fasci laser, DGPS e IMU (Inertial Measurement Unit) per la percezione dell'ambiente circostante e dei parametri di movimento del veicolo (velocità, accelerazione, direzione, rollio e beccheggio) e ad un sistema drive-by-wire con il quale può viaggiare autonomamente. La gestione del veicolo e l'assistenza alle manovre



di guida nella circolazione stradale sono da anni nel mirino dei ricercatori: già la Bosch alcuni anni fa aveva sviluppato un sistema di assistenza in grado di controllare acceleratore e freno dell'auto per mantenere automaticamente la distanza di sicurezza, rilevata anche in condizioni di scarsa visibilità mediante un efficace radar.

Il VisLab di Parma si è spinto oltre, realizzando numerosi sistemi di assistenza alla guida tutti concentrati nella Sonata, il cui computer di bordo può gestire quattro livelli: ausilio alla guida, semiautomatico, altamente automatico, totalmente automatico.

I sistemi di bordo usano telecamere per vedere ostacoli e veicoli e software di riconoscimento di immagine per identificare cosa le telecamere vedono e riconoscere i segnali stradali in modo da adeguare il comportamento del veicolo alle prescrizioni del Codice della Strada: ad esempio se vedono uno STOP cercano di localizzare la linea di arresto e fermano la vettura poco prima di essa. Le tecnologie sfruttate sono:

- LKS (Lane Keeping and Steering); il computer di bordo analizza i dati delle telecamere frontali, identificando le linee che delimitano le corsie al fine di mantenere,



Hiunday Sonata elaborata dal VisLab dell'Università di Parma guidato dal prof. Alberto Broggi, con il contributo della Fondazione Cassa di Risparmio di Parma e dei circa 1,8 milioni di euro che l'ERC (European Research Council, <http://erc.europa.eu/>) ha elargito a colui che ritiene uno tra i migliori ricercatori Europei nel campo dell'incremento della sicurezza stradale tramite sistemi di guida assistita e guida automatica. BRAiVE è un progetto ambizioso che ha dimostrato come ormai i tempi siano maturi perché le autovetture viaggino da sole o assistano il conducente nella quasi totalità dei momenti della circolazione stradale. Certo, di primo acchito l'idea di far muovere un veicolo contando sulla capacità di un computer di

esempio il comando idraulico del freno o dello sterzo) non compisse correttamente l'azione richiestagli. Incidenti aerei come quello dello sfortunato volo AF440 Air France, che pare sia precipitato per un errore nel rilevamento della velocità che avrebbe tratto in inganno sia il computer di bordo che il pilota, fanno riflettere sull'affidabilità dei sistemi di guida automatica. Ma la realtà, almeno sulle strade, è che alla guida l'uomo fa più danni di quelli



intervenedo sullo sterzo, il veicolo sulla propria strada;

- ACC (Adaptive Cruise Control); un metro a laser misura la distanza dai veicoli che si trovano davanti e il computer di bordo agisce sull'acceleratore e sul freno per mantenere la distanza di sicurezza;
- APPS (Automatic Pedestrian Protection System); telecamere e un sofisticato software di riconoscimento d'immagine verificano continuamente la strada davanti e lateralmente, quindi se un pedone o un altro veicolo attraversa improvvisamente la strada, il computer di bordo emette un segnale acustico per avvertire il conducente, suona automaticamente il clacson e, se verifica che malgrado ciò la vettura non riesce ad evitare la collisione perché il conducente non reagisce, comanda autonomamente il freno;
- LDW (Lane Departure Warning); usando la telecamera e il software di riconoscimento d'immagine, determina la posizione del veicolo nella propria corsia e avverte il conducente se sta uscendo di strada o invadendo altre corsie;
- S&G (Stop & Go); progettato per la guida in colonna, comanda autonomamente acceleratore, freni e sterzo in modo da seguire, senza collisioni, il veicolo che precede;
- AD (Autonomous Driving); avvalendosi di tutti i sensori e i sistemi di bordo e di un ricevitore GPS, guida completamente la vettura fino a portarla alla destinazione programmata;
- ARV (All Round Vision); impiega l'insieme delle telecamere per vedere davanti, dietro e di lato e digitalizza le immagini rendendole disponibili per il computer di bordo;
- BSM (Blind Spot Monitoring); mediante due telecamere integrate nei retrovisori tiene sotto controllo i cosiddetti "angoli morti" e fornisce al sistema di controllo informazioni sull'avvicinamento di veicoli, ad esempio in sorpasso, classificando come pericolosi quelli che cambiano posizione (che si spostano o si avvicinano) e ignorando gli altri;
- FCM (Frontal Collision Mitigation); mediante le telecamere e il riconoscimento d'immagine identifica

i veicoli che si trovano davanti e avverte il conducente dell'eventuale rischio di tamponamento o frena addirittura;

- Adaptive Lighting; mediante telecamere e riconoscimento d'immagine gestisce i fari anabbaglianti e anabbaglianti in base alla presenza di veicoli che giungono nella direzione opposta quando è buio;
- ICW (Intersection Collision Warning); sfrutta telecamere collocate lateralmente per controllare, negli incroci, le strade



lateralmente ed avvertire il conducente del sopraggiungere di veicoli;

- PAiD (Park Aid); mediante telecamere e riconoscimento d'immagine identifica le zone dove l'auto può essere parcheggiata e tramite metro a laser aiuta nel parcheggio;
- TSR (Traffic Sign Recognition); è certamente una delle funzionalità più importanti per l'auto-robot, in quanto mediante le telecamere e il software di riconoscimento d'immagine identifica la segnaletica stradale (per il momento, quella italiana).

Nel livello ausilio alla guida, BRAiVe utilizza i sistemi LDW, BSM, FCM, ICW, PAiD e TSR; nel semiautomatico invece coinvolge LKS, ACC, APPS e Adaptive Lighting. Più interessante è il livello altamente automatico, dove si combinano LKS+ACC e la tecnica S&G. Il livello più avanzato è la guida autonoma (totalmente automatico) che fa uso di tutti i sistemi e delle informazioni sul posizionamento ottenute da GPS e DGPS e delle segnalazioni di pericolo prodotte dai sistemi Veichle to Veichle e da quelle riguardanti la prossimità di intersezioni o punti di attenzione generate da sistemi in fase di sviluppo.

gli impianti semaforici e più in generale la possibilità di ricevere con congruo anticipo avvisi di pericolo e segnalazioni di allarme: ad esempio, prima di approcciare un incrocio o attraversare un passaggio a livello. Purtroppo dotare di infrastrutture di questo tipo tutta la rete stradale è molto costoso e soprattutto inutile se poi i veicoli non dispongono delle apparecchiature in grado di sfruttare queste risorse. E' stato

calcolato che rendere "intelligenti" col DSRC tutti gli impianti semaforici degli USA costerebbe la bellezza di 4,5 miliardi di dollari.

Nel settore dei sistemi di comunicazione V2V, una delle Case automobilistiche che ha investito di più in ricerca e sviluppo è stata la General Motors che ha messo a punto e sta sperimentando un hardware relativamente semplice che sfrutta la tecnologia DSRC. Auto dotate di questa tecnologia possono comunicare e scambiarsi informazioni come posizione e velocità: i guidatori possono essere avvisati in anticipo se un altro veicolo è fermo in un punto dove è difficile o impossibile vederlo o si sta approssimando allo stesso incrocio. A seconda della situazione, il sistema trasmette questi allarmi tramite messaggi visivi e sonori oppure facendo vibrare il sedile del guidatore.

Gli autoveicoli si collegano fra loro quando sono a poche centinaia di metri l'uno dall'altro scambiandosi informazioni sulla posizione, velocità, accelerazione e direzione di marcia. Scopo ultimo di questi sistemi



e quello di ridurre al minimo la possibilità di incidenti, specie in prossimità di incroci pericolosi, traffico intenso, strade dissestate e scarsa visibilità.

Di questi temi si occupa anche "Easy Rider" il progetto proposto da Magneti Marelli che è risultato primo classificato per il tema della Mobilità Sostenibile a livello nazionale tra quelli (oltre 50) che hanno partecipato al programma "Industria 2015".

L'obiettivo del progetto è implementare una rete interconnessa di veicoli e infrastrutture stradali intelligenti che abiliti l'erogazione di servizi di mobilità e sicurezza, per la gestione integrata del traffico pubblico e privato in ambito metropolitano ed extraurbano, al fine di decongestionare il traffico, ridurre le emissioni, aumentare la sicurezza stradale e diminuire i costi legati all'incidentalità.

Il progetto che vede come capofila Magneti Marelli, verrà realizzato nei prossimi anni da diversi consorzi, composti in tutto da 250 imprese e 100 centri di ricerca delle più importanti società del settore (Fiat, Anas, ST Microelectronics e Art, ecc.). Il coinvolgimento di numerosi partner ha la finalità di costruire una filiera rappresentativa degli interessi nazionali sull'infomobilità applicata all'efficienza del trasporto e alla sicurezza stradale.

Il partenariato, infatti, è costituito da rappresentanti di tutti i portatori di interesse del

settore: gestori dell'infrastruttura stradale e della mobilità urbana, componentisti, service e content provider. Al momento partecipano più di 30 aziende, sei organismi di ricerca, cinque Università e otto Enti Pubblici. Ambiente e sicurezza sono dunque i due obiettivi principali del Progetto Easy Rider.

Il primo obiettivo è perseguito attraverso la regolazione del traffico, la riduzione dei tempi di percorrenza e di ricerca del parcheggio, il controllo dello stile di guida e l'indirizzamento del veicolo su percorsi a minimo consumo.

Le innovazioni messe in campo per la sicurezza sono il rilievo e l'invio ai veicoli di informazioni sulla presenza di pericoli sulla strada, la chiamata di emergenza, l'aumento dell'efficienza e dell'efficacia dei soccorsi in caso di incidente.

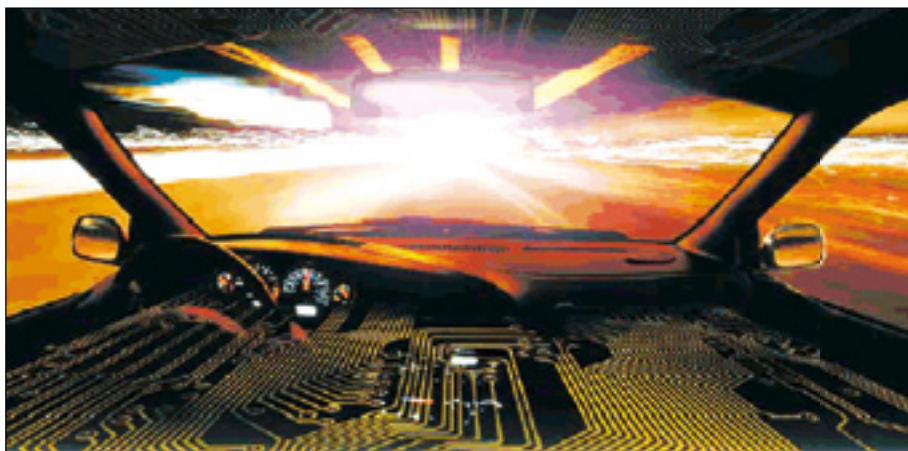
La parte più interessante riguarda sicuramente i segnali stradali intelligenti, in grado di trasmettere informazioni all'automobilista insieme al monitoraggio del profilo stradale da comunicare in tempo reale ai veicoli in transito.

Ad esempio, per quanto riguarda i cantieri stradali, la gestione è stata affidata alla società Aitek (www.aitek.it) che ha realizzato

un sistema che implementa le seguenti funzionalità:

- Rilevamento della posizione del personale in cantiere tramite l'utilizzo di tecnologie video e dispositivi in RF UHF che dovranno far parte dell'attrezzatura obbligatoria;
- Rilevamento delle condizioni del traffico in prossimità del cantiere;
- Rispetto della distanza di sicurezza nella zona interessata dal cantiere;
- Segnalazione ai veicoli in transito di eventuali situazioni di rischio;
- Invio ai veicoli di un protocollo di avvicinamento e transito in prossimità del cantiere;
- Controllo della velocità;
- Monitoraggio meteo;
- Invio segnalazioni ad un centro di controllo remoto.

L'architettura del sistema progettato da Aitek è basata su cinque componenti principali: una Road Side Unit (il centro di gestione della sicurezza); i sensori dell'infrastruttura (telecamere, laser-scanner); il client sui dispositivi mobili dei lavoratori in cantiere; la messaggistica fissa in loco (Pannelli a Messaggio Variabile, Lampeggiatori, Allarmi sonori) ed infine il client sui veicoli (ovvero il dispositivo V2I presente sui veicoli in transito).



VU-METER CON BARRA A LED

Progettato per completare il booster in classe H proposto il mese scorso, si adatta tranquillamente a qualsiasi finale di potenza BF.

dell'Ing. MARCO LANDONI

Un buon amplificatore è quasi sempre corredato da un level-meter o vu-meter che dir si voglia, ossia da un indicatore visivo della potenza emessa in uscita: ciò essenzialmente per due ragioni, la prima delle quali è che vedere una lancetta che oscilla (o una stringa luminosa che si allunga e si accorcia) a ritmo di musica rende meglio l'idea e dà quel tocco coreografico che chi ama l'ascolto ad alto volume non disdegna; la seconda ragione è che, se ben tarato, il level-meter fornisce un'indicazione del livello della potenza erogata agli altoparlanti e ci

permette di mantenere il volume al disotto della soglia alla quale si verifica la distorsione del segnale. Ebbene, l'articolo di queste pagine tratta proprio la realizzazione di un vu-meter, che in origine abbiamo progettato per abbinarlo all'amplificatore/booster basato sull'integrato TDA1562 presentato nel numero 137 della nostra rivista; infatti, oltre a poter essere utilizzato in modo

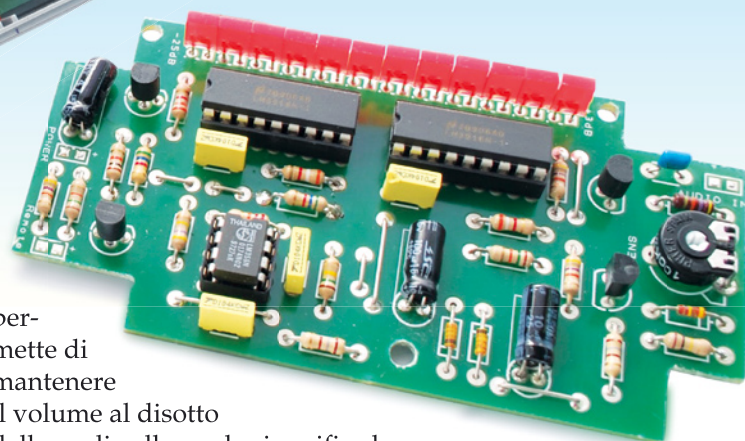
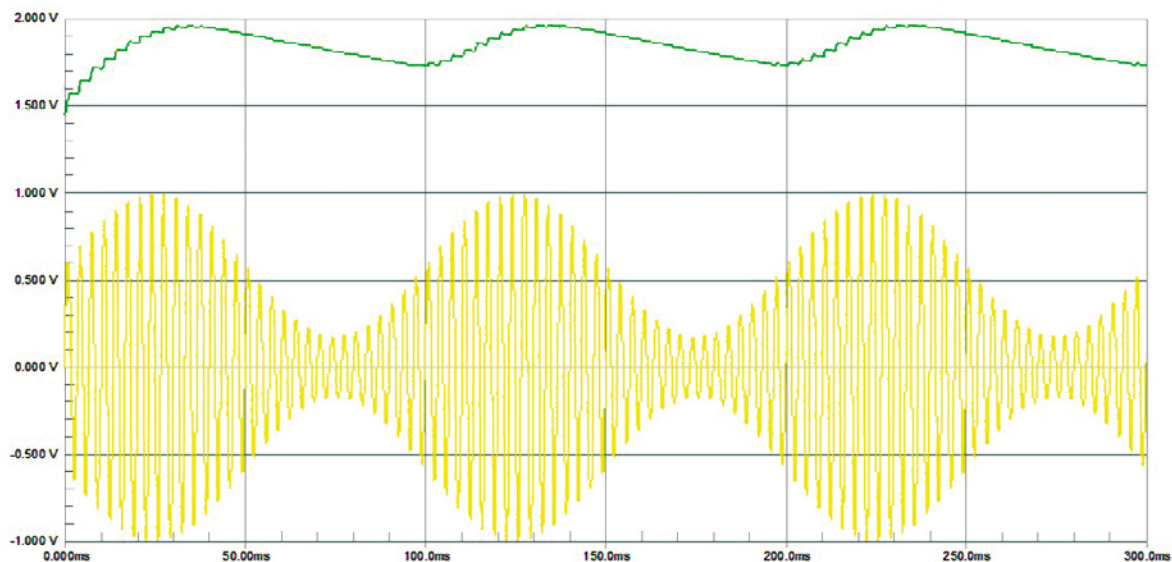


Fig. 1
In giallo,
l'andamento
della tensione
d'ingresso
del circuito;
in verde,
la tensione
ai capi del
condensatore
C5, che è
unidirezionale
e segue
l'involuppo
della
componente
d'ingresso.



indipendente in amplificatori da casa e da installazione fissa, il circuito può tranquillamente funzionare in automobile poiché si alimenta con una tensione di 12 V e le sue dimensioni sono tali da poterlo inserire nello stesso mobile plastico da noi previsto per il booster in classe H. Se sfogliando queste pagine a un primo sguardo lo schema elettrico vi potrà sembrare quello di un canonico circuito vu-meter basato sugli altrettanto canonici integrati della serie LM391x, vi garantiamo che in realtà non è così, poiché il circuito qui propo-

sto presenta caratteristiche non comuni: infatti, oltre ad essere in grado di pilotare ben 12 diodi luminosi (che può anche comandare in dissolvenza) consuma appena 15 mA.

SCHEMA ELETTRICO

Entriamo dunque nel dettaglio, esaminando lo schema elettrico e partendo dai diodi D2 e D3, ai capi dei quali è presente una tensione (*Ref*) pressoché costante di circa 1,3 V, utilizzata come riferimento per gli integrati U1 e U2 e come polarizzazione per lo stadio d'ingresso composto dai

transistor bipolari T1 e T2. Questi ultimi sono entrambi collegati in configurazione-emitter follower (o collettore comune) che, come ben saprete, funge da buffer di tensione; ciò vuol dire che il guadagno in tensione è praticamente unitario, ma si ottiene il disaccoppiamento di corrente ingresso-uscita, nel senso che la corrente erogata è praticamente indipendente da quella assorbita dall'ingresso. Il motivo per il quale è stata adottata la configurazione che vedete, è che permette di evitare la perdita di tensione base-emettitore (di 0,6÷0,7 V)

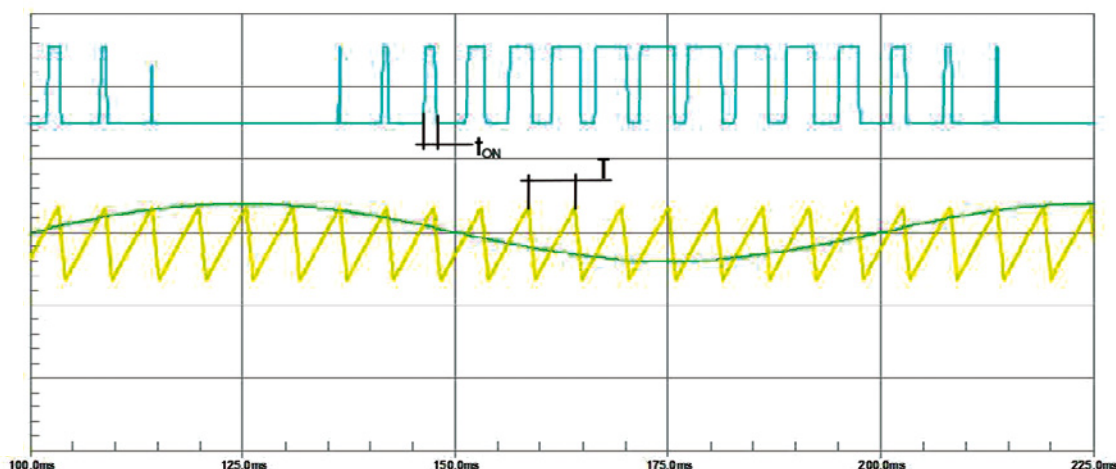


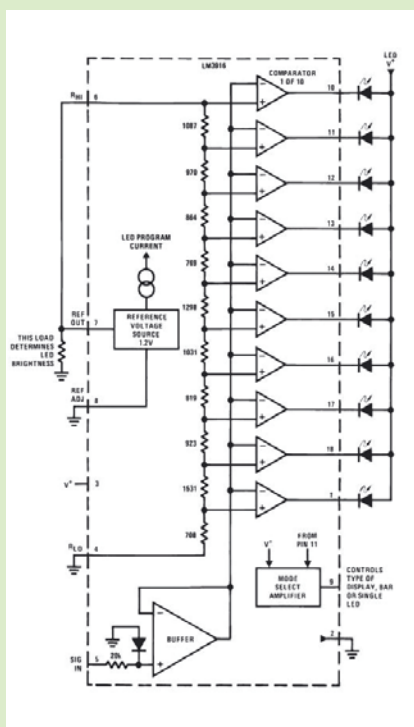
Fig. 2 - In verde, il segnale a dente di sega generato da U3 ed utilizzato come riferimento per U1 ed U2; in giallo, vedete il segnale d'ingresso dei due driver. Osservando il segnale azzurro (uscita del comparatore) si nota come varia il tempo di accensione dei led (*tON*) in funzione del livello d'ingresso.

che l'uso di un semplice stadio imporrebbe; in altre parole, se T1 fosse stato un NPN, sul suo emettitore avremmo avuto 0,6÷0,7 V in meno del potenziale di base. Collegando in cascata un PNP e un NPN, sull'emettitore di quest'ultimo otteniamo una tensione ben più alta.

Lo stadio di ingresso ha la duplice funzione di disaccoppiare l'ingresso (cosa utile non tanto quando usiamo il vu-meter con un finale di potenza, ma soprattutto se vogliamo collegarlo ad un preamplificatore o altro dispositivo ad alta impedenza d'uscita) e fare da rilevatore di picco, dato che, essendo l'emettitore di quest'ultimo collegato direttamente al condensatore C5 tramite la resistenza R3, sarà in grado di amplificare in corrente solamente quei segnali che istante per istante avranno un valore di tensione superiore a quello presente ai capi di C5. Quindi ai capi di questo condensatore sarà presente un segnale che seguirà l'involuppo del segnale audio d'ingresso ed avrà un valore minimo pari alla tensione *Ref*, ossia 1,3 V (Fig. 1).

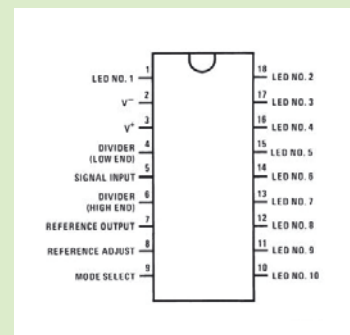
Tale segnale viene applicato direttamente ai pin d'ingresso dei due circuiti integrati U1 e U2 (LM3916) che pilotano direttamente i diodi luminosi ad essi collegati. Come si vede chiaramente dallo schema elettrico, metà dei led è affidata ad U1 e la metà restante all'U2.

L'M3916 prevede due possibilità di controllo: a punto luminoso e a barra; nel nostro caso abbiamo configurato U1 e U2 per il funzionamento a punto luminoso e mettendo i led in serie. Ciò significa che le uscite si attiveranno una sola alla volta, però sarà ugualmente possibile vedere illuminarsi una fila di led perché abbiamo adottato una speciale



Per realizzare il nostro vu-meter abbiamo utilizzato due integrati LM3916, progettati esplicitamente per realizzare voltmetri a barra di 10 led; ogni LM3916 contiene una serie di comparatori di tensione che ricevono dal piedino 5 e poi da un buffer il segnale da confrontare con i potenziali di riferimento determinati da un partitore multiplo

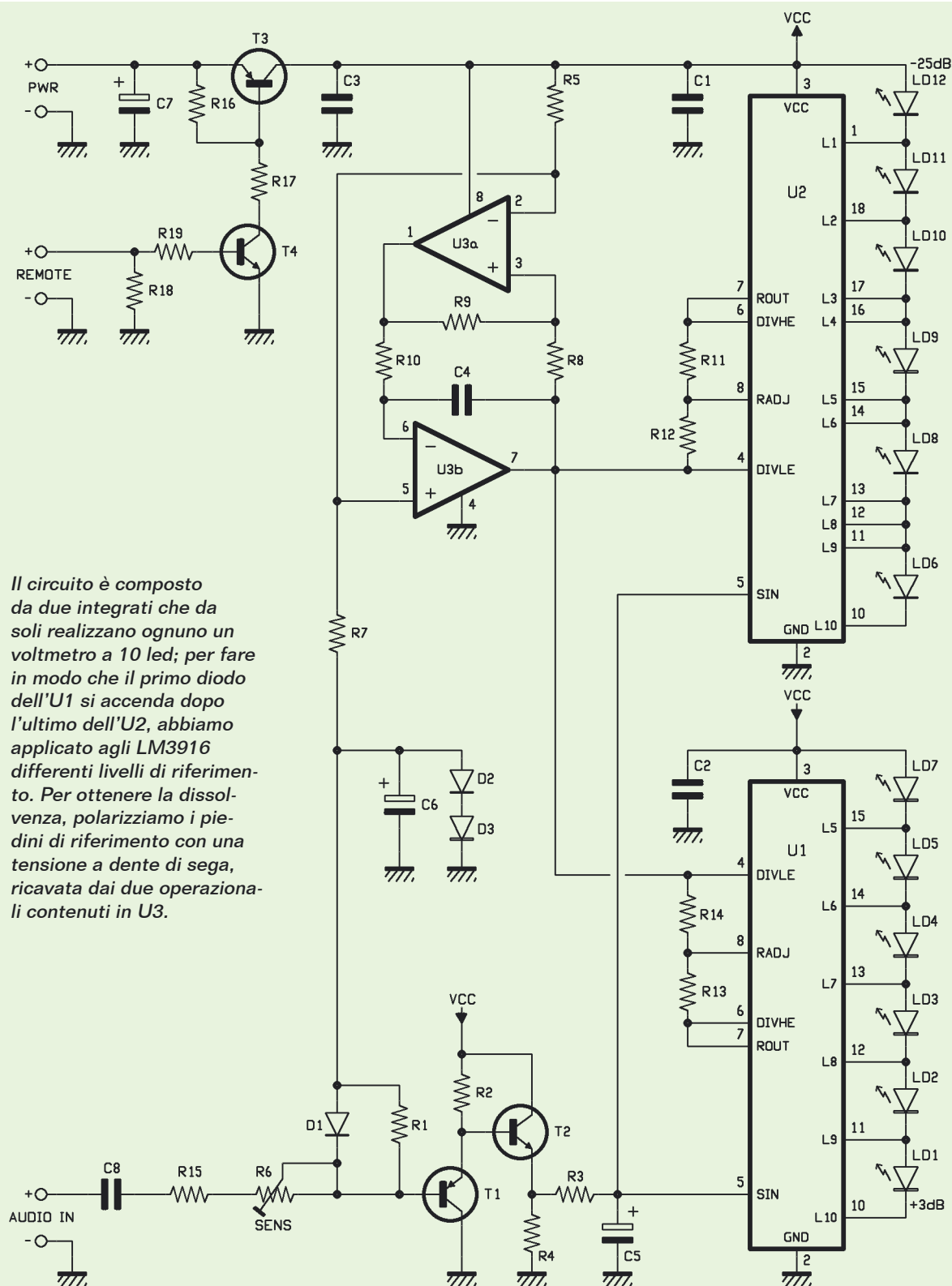
Il driver LM3916



a scala. Dal basso verso l'alto, i comparatori commutano a una tensione via-via crescente e la distanza tra la tensione che fa accendere il led connesso al comparatore più in basso (pin 1) e quella che determina l'accensione del led più in alto (connesso al pin 10) è fissa. Dal potenziale applicato al piedino 8 dipende la minima ampiezza da applicare per fare illuminare il led connesso al pin 1. Nella nostra applicazione usiamo una configurazione serie con sei soli led, che consente un notevole risparmio di corrente; i diodi si accendono uno alla volta, dato che il pin 9 è fisso a massa.

configurazione che è detta, appunto, serie: rispetto a quella parallela, che vede ogni led pilotato da un'uscita, qui i diodi sono in serie dalla prima all'ultima uscita. In questo modo, quando il livello del segnale fa attivare la prima uscita, il pin 1 viene posto a zero logico e si illumina LD12; se è la seconda uscita ad attivarsi, il pin 18 si porta a zero e l'1 resta ad alta impedenza, quindi si





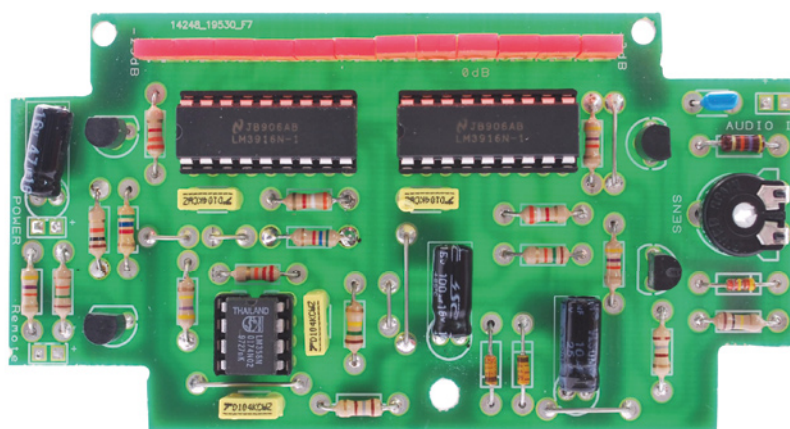
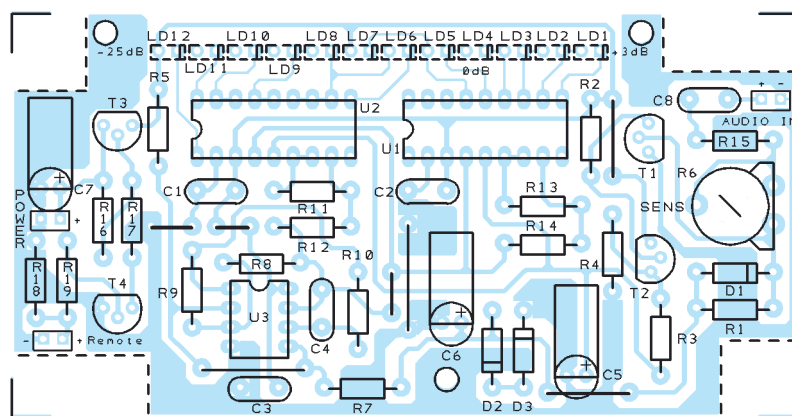
accendono insieme LD11 ed LD12. Con la terza uscita, restando i pin 1 e 18 in alta impedenza, si accenderanno LD10, LD11 e

LD12, che risulteranno in serie. L'utilizzo di U1 e U2 in modalità serie presenta il vantaggio di consumare pochissima corrente

rispetto alla modalità parallela: infatti nel primo caso la corrente è comune a tutti i diodi collegati allo stesso integrato mentre nel

secondo ogni diodo luminoso assorbirebbe una propria corrente. Supponendo di far funzionare i led con 5 mA ciascuno, una fila di sei accesi assorbirebbe 30 mA, contro i 5 della configurazione serie. L'unico svantaggio della serie è che richiede una tensione d'alimentazione alta abbastanza da alimentare tutta la serie di led. Nel nostro caso 12V sono sufficienti per due serie da sei diodi, poiché, considerando 1,8 V di caduta per ogni led, già 11 V bastano; inoltre i diodi sono divisi in due serie gestite ognuna da un integrato alimentato da 12 volt. Le due serie di led sono indipendenti ma i livelli sono concatenate in modo da rappresentare la scala voluta con le divisioni volute: così facendo avremo una dinamica complessiva di 25 dB e contemporaneamente gli ultimi sette diodi della barra si illumineranno a passi di 1 dB. Allo scopo l'integrato U1 ha una tensione di fondo scala (imposta tramite le resistenze R13 ed R14) che è 5 dB superiore a quella di fondo scala di U2; così con il medesimo segnale applicato all'ingresso (pin 5) l'ultimo led di U1 (LD1) si accende con un livello di 5 dB superiore a quello dell'ultimo led di U2 (LD6). Noterete che i terminali di riferimento di U1 e U2, R12 ed R14 sono collegati all'uscita dell'operazionale U3b: tale amplificatore, unitamente ad U3a, realizza un generatore a dente di sega. Ciò serve per far variare velocemente nel tempo la tensione di riferimento della cascata di comparatori interni ad U1 e U2, che pilotano i led. Applicando un riferimento non costante ma variabile, otteniamo l'effetto dissolvenza; per farvi capire la cosa supponiamo che per un certo livello d'ingresso costante avremo LD 10, ma che lo stesso livello

[piano di MONTAGGIO]



Elenco Componenti:

R1: 100 kohm
 R2, R4, R15, R18: 47 kohm
 R3, R7: 180 ohm
 R5: 3,3 kohm
 R6: Timmer 100 kohm MO
 R8: 22 kohm
 R9: 470 kohm
 R10: 150 kohm
 R11, R13: 3,9 kohm
 R12: 7,5 kohm
 R14, R19: 15 kohm
 R16: 10 kohm
 R17: 5,6 kohm
 C1 ÷ C4: 100 nF 63 VL poliestere
 C5, C7: 10 μ F 25 VL elettrolitico
 C6: 100 μ F 16 VL elettrolitico
 C8: 470 nF 50 VL poliestere
 D1 ÷ D3: 1N4148
 LD1 ÷ LD10: Led rosso
 5x2 mm rettangolare

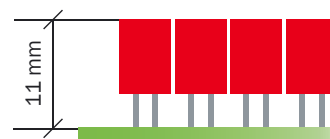
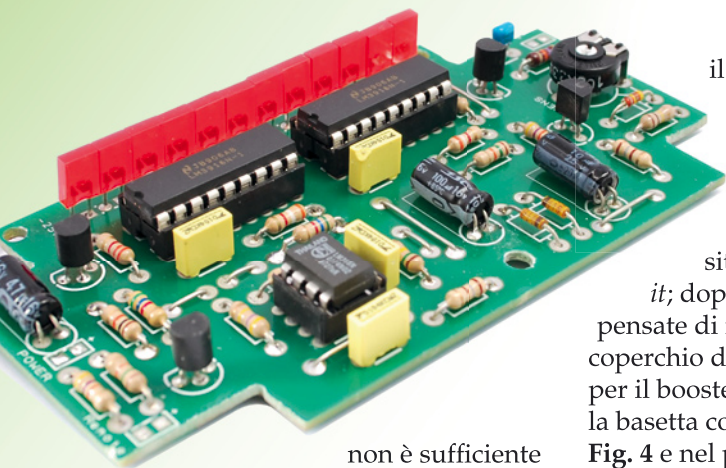


Fig. 3 - I led devono essere montati ad un'altezza di 11 mm.

T1, T3: BC556
 T2, T4: BC546
 U1, U2: LM3916N
 U3: LM358

Varie:

- Strip maschio 2 poli (3 pz.)
- Strip femmina 2 poli (3 pz.)
- Zoccolo 9+9 (2 pz.)
- Zoccolo 14+14
- Circuito stampato



non è sufficiente per far accendere anche il diodo successivo (LD9). Variando però la tensione di riferimento dei comparatori secondo l'andamento del segnale generato da U3 (Fig. 2) in ogni periodo T per un certo tempo t_{ON} LD9 risulterà comunque acceso. Poiché la frequenza con cui l'onda a dente di sega si ripete è molto maggiore della persistenza dell'occhio umano, avremo la sensazione che LD9 risulti acceso solo in parte.

Detto ciò, chiudiamo la descrizione dello schema con i transistor T3 e T4, il cui scopo è consentire l'accensione del circuito mediante un segnale elettrico di comando, cosa utile nell'uso in auto, dato che ci consente di accendere il Vu-meter con lo stesso segnale *remote* in arrivo dall'autoradio e usato per il booster. Alimentando il dispositivo tra i punti + e - PWR, non viene assorbita corrente se non quella di perdita del C7 e dei T3 e T4; quando l'ingresso *remote* riceve almeno 6÷7 volt, T4 va in saturazione e porta nella stessa condizione T3, il quale lascerà passare la corrente da + e - PWR al resto del circuito. Privando *remote* dell'alimentazione, il circuito può restare sotto tensione ma è spento.

REALIZZAZIONE PRATICA

La prima cosa da fare per costruire il vu-meter è preparare

il circuito stampato, che è monofaccia e si realizza per fotoincisione dopo aver scaricato la traccia dal nostro sito www.elettronica.in.

it; dopo la foratura, se pensate di inserire il circuito nel coperchio del contenitore usato per il booster, dovete sagomare la basetta come indicato nella Fig. 4 e nel piano di montaggio. Fatto ciò, iniziate il montaggio realizzando (con del filo di rame o degli scarti di reofori di componenti) i ponticelli e poi procedete inserendo e saldando gli zoccoli per i tre circuiti integrati U1, U2 e U3. Fate attenzione a non eccedere con lo stagno, per non cortocircuitare piedini adiacenti. Saldati gli zoccolini, procedete inserendo e saldando una ad una tutte le resistenze. Montate i tre diodi al silicio, badando a posizionare correttamente la fascia che contraddistingue il catodo; fate attenzione che in alcuni tipi 1N4148 potreste trovare, anziché una fascia sola, una serie di fasce colorate: giallo, nero, giallo, grigio. In tal caso il catodo è indicato dalla prima fascia gialla. Sistemati i diodi, dedicatevi ai condensatori in poliestere, che sono tutti da 100 nF tranne C8, il quale risulta essere da 470 nF. Montate ora gli elettrolitici, posizionando il loro corpo in orizzontale: per fare ciò è conveniente preformare

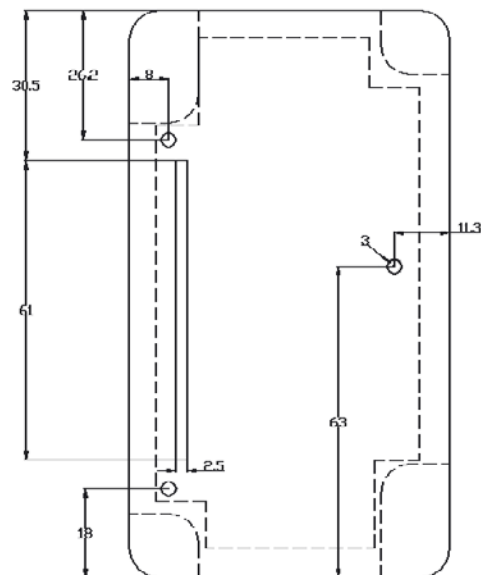


Fig. 4 - Piano di foratura del pannello frontale del mobile plastico.

a "L" i reofori dei condensatore prima di inserirli e saldarli. Proseguite montando il trimmer R6 e quindi i quattro transistor T1, T2, T3 e T4; per limitare lo spessore del circuito, teneteli distanti dalla basetta non più di 3 mm.

Rimangono a questo punto da montare i dodici diodi luminosi, che dovete sistemare in fila e in modo che la loro altezza dal piano della basetta sia 11 mm esatti (Fig. 3); per non sbagliare, saldate il primo e l'ultimo, quindi inserite quelli intermedi, che si livelleranno automaticamente appoggiando lo stampato sui led.

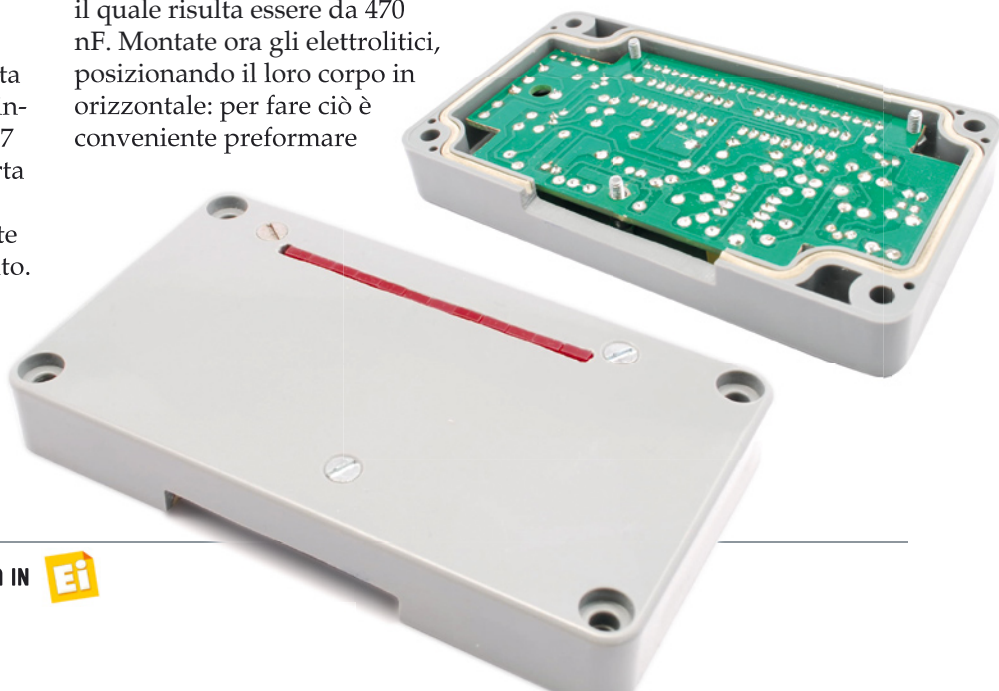
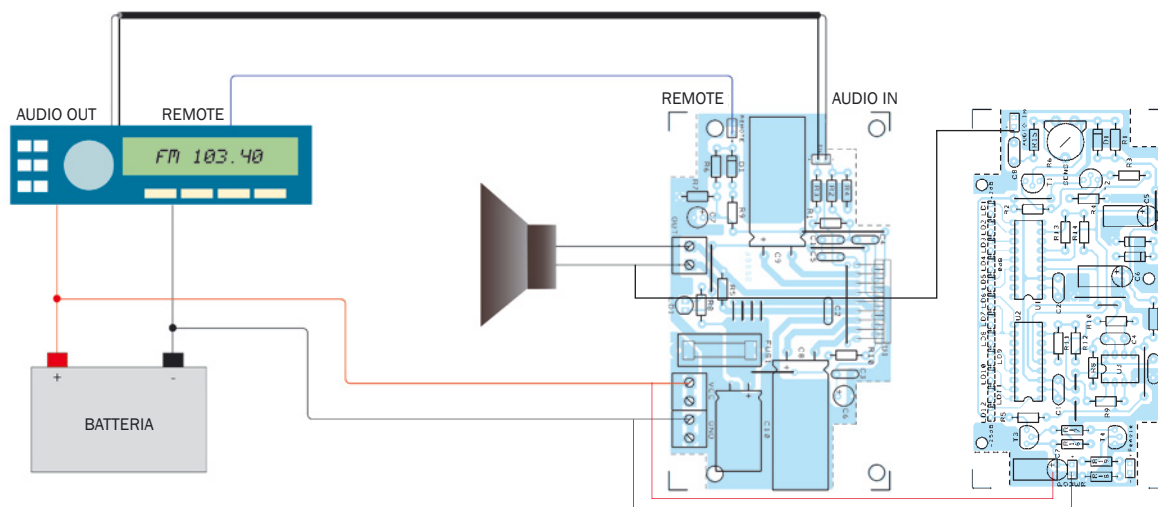


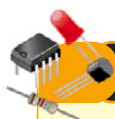
Fig. 5 - Collegamento del level-meter all'amplificatore in classe H descritto nel fascicolo n° 137. Nel caso non vogliate utilizzare la funzione remote, collegate il relativo morsetto direttamente al terminale Vcc+.



Una volta ultimato il montaggio dei led, non vi resterà altro da fare che inserire gli integrati nei rispettivi zoccoli. Prima di utilizzare il vu-meter dovrete tarare il trimmer R6. Per farlo potrete utilizzare direttamente il nostro amplificatore in classe H oppure collegare l'ingresso del meter all'uscita per altoparlante di un altro amplificatore audio di potenza. Badate che se l'amplificatore in questione ha un'uscita a ponte dovrete collegare solamente il terminale d'ingresso + del meter al rispettivo terminale d'uscita dell'amplificatore; se collegherete anche la massa del meter al terminale d'uscita negativo provocherete un cortocircuito. Realizzato il collegamento nel modo corretto secondo la Fig. 5, riproducete un brano e portate l'amplificatore al volume massimo oltre il quale si avverte della distorsione: in queste condizioni sul vu-meter deve accendersi l'ultimo led. Se così non fosse, ruotate il trimmer R6 fino ad ottenere ciò. Se utilizzate il nostro amplificatore in classe H, individuare la condizione di distorsione è molto semplice, perché

essa viene indicata dall'accensione del suo led. Per la taratura non è necessario che diventiate sordi: potete benissimo lasciare l'altoparlante sconsesso e tarare tranquillamente il circuito. Volendo montare il vu-meter nello stesso mobile in cui avrete messo l'amplificatore, dovrete innanzitutto forare il pannello frontale come indicato nella Fig. 4; a questo punto svasate i tre fori da 3 mm, quindi inserite tre viti M3x15 mm a testa svasata che bloccherete sul coperchio con un dado. Avvitare altri tre dadi sulle rispettive viti che fisseranno lo

stampato, che andrete ad inserire facendo combaciare i fori con le viti. Regolate i dadi di fermo in modo che i led risultino praticamente a filo della superficie esterna del pannello frontale del mobile, quindi bloccate lo stampato con altri tre dadi, uno per ogni vite. Realizzate quindi tutti i collegamenti necessari tra level-meter e amplificatore (e con il remote dell'autoradio) quindi chiudete il mobile plastico; nel farlo badate che le viti di sostegno del circuito non vadano a toccare alcuno dei componenti dell'amplificatore. ■



per il MATERIALE

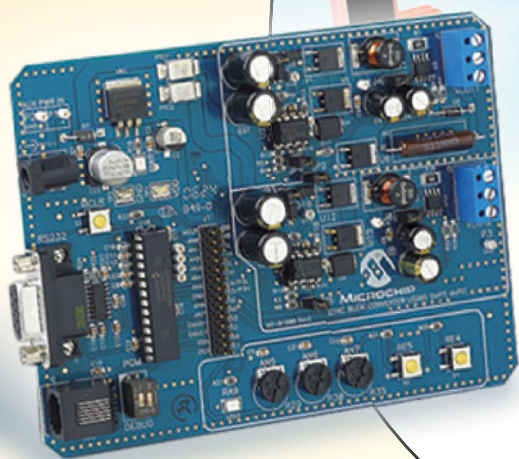
Tutti i componenti utilizzati in questo circuito sono facilmente reperibili in commercio mentre il master dello stampato può essere scaricato gratuitamente dal sito della rivista. Ricordiamo che questo Vu-meter è stato progettato per lavorare unitamente al booster in classe H presentato il mese scorso; il booster è disponibile in scatola di montaggio (cod. FT806K) al prezzo di 29,00 Euro mentre il costo del solo integrato TDA1562Q è di 14,90 Euro.

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

Introduzione ai dsPIC

Proseguiamo lo studio dei DSC di casa Microchip con un'applicazione consistente nella lettura di un encoder in quadratura con un dsPIC.

di CALOGERO LOMBARDO



Con lo stesso spirito con cui, nella prima puntata, abbiamo iniziato ad affrontare l'argomento, continueremo a descrivere i DSC dando, in queste pagine, un altro esempio tangibile delle caratteristiche di verticalità dei dsPIC. Qui impareremo a leggere un encoder incrementale (il sensore di posizione angolare per eccellenza) sfruttando una periferica appositamente progettata dalla Microchip e integrata nel dsPIC: la *Quadrature Encoder Interface*. Ci renderemo così conto che il lavoro che

tocca allo sviluppatore è notevolmente semplificato rispetto al caso in cui si effettui la decodifica mediante le tecniche tradizionali. Peraltro, usando i dsPIC si ottengono risultati più affidabili con minor dispendio di risorse. Nell'esercitazione qui descritta continueremo a usare la dsPICdem2, che risulta molto comoda per visualizzare sul display le grandezze lette dall'encoder. Possibili applica-



zioni di ciò che impareremo, si trovano nelle macchine di lavorazione industriali, nei manipolatori antropomorfi presenti nelle catene di montaggio robotizzate, oppure nella realizzazione di un contagiri o di manopole di regolazione di alta precisione, senza dimenticare i robot mobili e tutte quelle applicazioni dove è necessario rilevare con elevata accuratezza la posizione angolare di un'asse in rotazione.

LA PERIFERICA QEI

Entriamo subito nel vivo, svelando una delle caratteristiche che vi faranno sicuramente preferire un dsPIC a molte altre MCU.

In questa sezione sarà descritto come utilizzare la periferica *Quadrature Encoder Interface* (QEI) presente nel dsPIC30F4012, che ci permette di contare gli impulsi provenienti da un encoder incrementale.

La QEI consente d'immagazzinare direttamente in un registro a 16 bit il numero d'impulsi dell'encoder (fino a 65.535) che potremo andare a leggere ogni volta che ne avremo la necessità; tutto questo senza bisogno di dover scrivere righe di programma.

Immaginate che lavoro è necessario per fare una decodifica 4x con riconoscimento del verso di rotazione; con un dsPIC, pensa a tutto la QEI, lavorando direttamente in hardware: ciò significa che oltre a non scrivere codice, non sono impegnate risorse della CPU, che rimangono a disposizione per altre operazioni. Inoltre nella decodifica fatta in maniera classica avremmo dovuto lavorare con interrupt

che, nel caso di encoder ad alta risoluzione, sono generati a frequenze elevate; va poi detto che se i MIPS a disposizione non sono molti, diviene problematico riuscire a gestire la decodifica assieme alle altre operazioni che la MCU deve compiere. Questo vuol dire che se non si ha a disposizione la QEI, a parità di applicazione è necessaria una MCU più potente e una programmazione condotta da un esperto. Una bella invenzione, non credete? Inoltre, utilizzare la QEI è molto affidabile ed evita i falsi conteggi che spesso si verificano utilizzando gli interrupt, perché al suo interno ha delle funzioni filtranti che saranno descritte più avanti.

In realtà non è proprio esatto affermare che non bisogna scrivere una riga di codice; però basta soltanto configurare dei registri che a breve andremo ad illustrare.

Innanzitutto, dal datasheet del DSC o dal manuale della dsPICdem2 (vedere DS51558A sul sito internet www.microchip.com), possiamo osservare gli ingressi del modulo QEI, ai quali collegare i terminali dell'encoder, che sono i pin 5 (RB3), 6 (RB4), 7 (RB5), rispettivamente per l'index (INDX), il canale A (QEA) e in canale B (QEB). Questi pin sono multiplexati con altre funzionalità del DSC ed in particolare con i canali ADC AN3, AN4 e AN5, quindi sarà necessario ricorrere al registro ADPCFG per impostarli come I/O digitali ed al TRISB per configurarli come ingressi.

Nella Fig. 1 possiamo osservare uno schema a blocchi che ci illustra l'architettura del modulo: ritroviamo i tre ingressi INDX, QEA e QEB ai quali è possibile applicare un filtraggio effettuato sempre in automatico dal modulo stesso (bisogna solo configurare alcuni parametri); poi i tre segnali arrivano ad un blocco che effettua la decodifica, dando in uscita un segnale (CLOCK) che va ad incrementare/decrementare il registro a 16 bit denominato POSCNT, nel quale è immagazzinato il numero d'impulsi dell'encoder. Il blocco di decodifica restituisce anche il verso di rotazione dell'encoder (segnale DIR) che, se necessario, può essere mandato

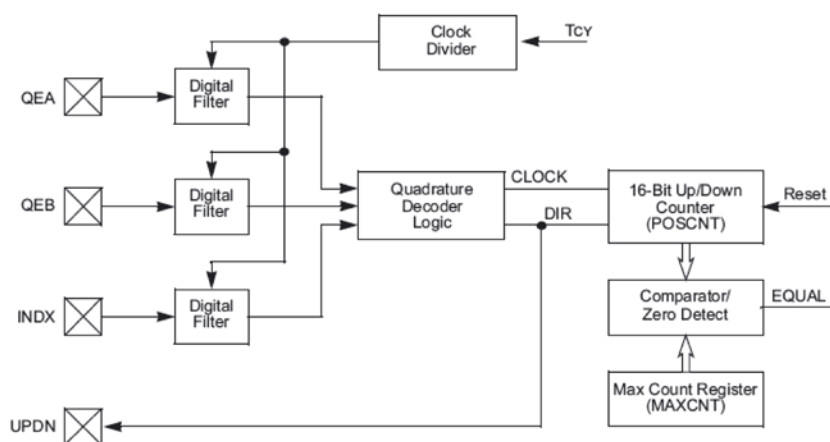


Fig. 1 - Schema blocchi della periferica QEI.



a un pin d'uscita del dsPIC. In figura notiamo anche il registro MAXCNT: questo è settato da software e serve per impostare il numero massimo d'impulsi, raggiunto il quale si vuole ricominciare a contare da zero. Notate che è possibile usufruire di un interrupt scatenato da questo evento.

Oltre ad essere molto comodo in certe situazioni, il registro MAXCNT è utile nel caso in cui l'encoder non dispone del pin index (è il caso pratico che andremo ad illustrare).

Per chiarire meglio il funzionamento di questi registri osserviamo l'esempio nella **Fig. 2**: in alto abbiamo i segnali provenienti dai due canali dell'encoder; il segnale *count_clock* (nella **Fig. 1** viene chiamato *clock*) è quello che esce dal blocco di decodifica, che in questo caso, com'è facile intuire, sta effettuando una decodifica 4x. Ad ogni impulso del *count_clock*, il POSCNT è incrementato, fino al valore 6, per poi ripartire da zero; basta guardare più in basso per vedere che 6 è il valore assegnato al MAXCNT. La figura ci mostra anche che alla presenza di un'inversione del senso di rotazione, il POSCNT è decrementato e non appena raggiunge lo 0, il conteggio riparte dal valore impostato su MAXCNT. Come preannunciato, ad ogni evento in cui POSCNT passa da MAXCNT a zero e viceversa, si può far scatenare un interrupt.

Il blocco di decodifica restituisce anche il verso di rotazione dell'encoder (nella **Fig. 1** è denominato DIR) mediante il segnale UPDN (**Fig. 2**) che (come vedremo) è il valore di un particolare bit, il quale, se necessario, può essere mandato ad un pin d'uscita del dsPIC. Per chi non lo sapesse, il senso di rotazione di un encoder incrementale con due canali in quadratura (sfasati di 90° gradi di periodo d'onda) si ricava dalla sequenza delle coppie di bit provenienti dai due canali; potete osservare le due sequenze, per i due sensi di rotazione, nella parte inferiore della **Fig.2**.

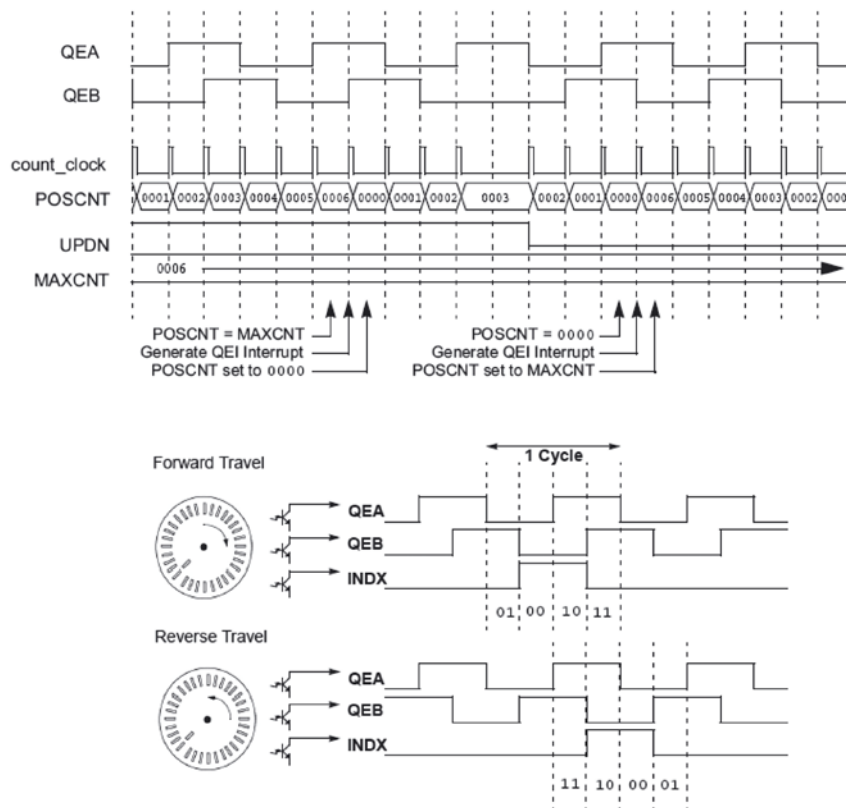


Fig. 2 - Andamento dei segnali provenienti dai due canali dell'encoder e funzionamento dei registri.

LA CONFIGURAZIONE DELLA QEI

Adesso, per capire realmente come utilizzare il modulo QEI, affrontiamo un semplice caso pratico che non richiede altre conoscenze, se non quelle che state assimilando da queste pagine. Naturalmente, per l'esperimento ci serve un encoder; il modello scelto è l'E4P da 300 cpr (*clic per revolution*) della USDigital (<http://www.usdigital.com>). Potete osservare il componente nella **Fig. 3**.

L'encoder va applicato ad un'asse rotante: noi utilizzeremo un motoriduttore e l'asse è quello prima della riduzione meccanica. Possiamo facilmente muovere il sistema con una mano,



**Fig. 3 - Encoder
USDigital E4P 300 cpr.**



Fig. 4 - Ruota, motoriduttore ed encoder.

grazie ad ruota montata sull'albero dopo la riduzione meccanica. Nella Fig. 4 potete osservare il sistema assemblato comprendente ruota, motoriduttore, encoder. Il motoriduttore in questione, della HSINAG NENG (<http://www.hsiangnengmotors.com.tw>) ha un rapporto di riduzione 30:1, cioè, per ogni giro della ruota, l'asse del motore su cui è montato l'encoder compie 30 giri. Facciamo un rapido calcolo: l'encoder è da 300 cpr e in decodifica 4x fornirà 1.200 impulsi per giro dell'asse del motore, che diventano 36.000 per ogni giro della ruota; quindi possiamo apprezzare il centesimo di grado.

Nella Fig. 5 trovate la pinout dell'encoder, con l'indicazione dei colori dei fili del connettore.

Adesso passiamo alla configurazione degli appositi registri della QEI: il primo è QEICON, che è il vero e proprio registro con il quale configurare tutte le opzioni. Il secondo è il DFLTCON, che configura i filtri e gli interrupt del modulo. In Fig. 6 potete osservare tutti i singoli bit dei vari registri.

Precisiamo che la QEI può essere utilizzata anche come timer, quindi alcuni di questi bit sono dedicati a tale modalità, che in questa sede non sarà analizzata.

Iniziamo a descrivere i bit del registro QEICON, partendo per comodità dai bit 10, 9, 8.

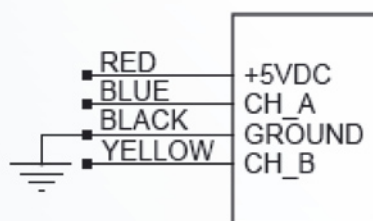


Fig. 5 Connettore dell'encoder.

Bit10, 9, 8 QEIM: permettono di impostare il tipo di decodifica (2X o 4X), l'evento che azzerà POSCNT (mediante il MAXCNT o impulso dell'INDX). I bit QEIM permettono anche di impostare la funzionalità timer, ma ciò non è oggetto di questa trattazione. Le possibili opzioni per QEIM sono:

- 111 = 4x, azz. da MAXCNT;
- 110 = 4x, azz. da INDX;
- 101 = 2x, azz. da MAXCNT;
- 100 = 2x, azz. da INDX.

Per il nostro esperimento sceglieremo QEIM = 7 (111). Le opzioni 0xx servono per la modalità timer o disabilitano il modulo.

Bit15 CNTERR: si utilizza solo nei casi in cui QEIM vale 110 o 100. Ci avvisa se è avvenuto un azzeramento di POSCNT e permette di disabilitare l'interrupt corrispondente. Nel nostro caso, oltre a non essere interessati all'interrupt, non è in ogni caso applicabile, quindi non lo utilizzeremo.

Bit14: non viene usato.

Bit13 QEISIDL: serve per impostare l'utilizzo della QEI, quando utilizziamo il dsPIC in modalità idle (risparmio energetico). Non stiamo usando tale modalità, quindi impostiamo questo bit a zero.

Bit12 INDX: è in sola lettura e indica se l'eventuale pin INDX è alto o basso. Il nostro encoder non ha index, quindi questo bit non ci interessa.

Bit11 UPDN: nel caso in cui QEIM vale 1xx, vale a dire quando la periferica QEI è usata come decoder, è un bit in sola lettura. Noi andremo a leggere questo bit, che se vale uno indica che la rotazione sta avvenendo nel verso positivo (incrementa POSCNT) mentre se vale zero la rotazione è nel verso opposto. In pratica non è altro che il segnale DIR che



Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
QEICON	QNTERR	Unused	QEISIDL	INDX	UPDN	QEIM2	QEIM1	QEIM0	SWPAB	PCDOUT	TQGATE	TQCKPS1	TQCKPS0	POSRES	TQCS	UDSRC
DLFTCON	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
POSCNT	Position Count Register															
MAXCNT	Maximum Count Register															

Fig. 6 - I registri associati alla periferica QEI.

abbiamo visto in figura 1 o l'omonimo segnale UPDN dell'immagine nella Fig. 2.

Bit7 SWPAB: solitamente il canale A dell'encoder è connesso al pin QEA del dsPIC e il canale B al pin QEB. Se ad esempio stiamo lavorando con un robot mobile, che ha due ruote contrapposte, per farlo andare avanti è necessario (data la contrapposizione delle ruote) che i motori girino l'uno nel verso opposto all'altro. Per far ciò basta invertire le connessioni dei canali di uno degli encoder, oppure agire sul bit in questione, mettendolo ad 1. Nel nostro caso, lavorando con un solo motore lo impostiamo a zero.

Bit6 PCDOUT: abilita la possibilità di mandare ad un pin esterno il valore del bit11. Diciamo che nel caso del funzionamento da decoder ciò è uno spreco, ma può tornare utile, quando si usa il QEI come timer. Questo bit lo impostiamo a zero.

Bit5, 4, 3: servono nella modalità *timer*.

Bit2 POSRES: si usa nei casi in cui QEIM vale 110 o 100 e serve per abilitare il reset di POSCNT al passaggio dall'index. Nel nostro caso non è applicabile perché QEIM=111 e il nostro encoder non ha index. Possiamo ignorare questo bit o in ogni caso settarlo a zero.

Bit1-0: servono nella modalità *timer*.

Adesso, prima di passare alla configurazione del registro DFLTCON dedichiamo qualche riga a parlare dei filtri che si possono applicare ai pin d'ingresso del modulo QEI. Questi permettono di eliminare eventuali disturbi che possono sovrapporsi al segnale utile, creando falsi conteggi; sono realizzati in hardware all'interno del modulo stesso. Non ci soffermeremo sul loro funzionamento. Per i più curiosi, nella Fig. 7 è riportato lo schema a blocchi semplificato di uno di essi. Questi filtri, per il loro funzionamento, necessitano del clock di sistema; infatti, per regolare la banda

passante del segnale, applicano una divisione al clock (vedi blocco *clock divider* nel circuito illustrato nella Fig. 7).

Il valore che divide il clock varia a seconda del numero di MIPS a cui si opera e dipende anche da alcune caratteristiche meccaniche del sistema. Riportiamo di seguito le formule che ci permettono di calcolare tale divisore:

$$DIV = MIPS \times T / 3$$

dove il valore T si calcola come:

$$T = 30 / MAX_RMP \times ENC_CPR \times 10^6$$

Calcoliamo adesso il valore del divisore per il nostro sistema. Il motoriduttore impiegato può arrivare a 200 rpm alla ruota, quindi a 6.000 rpm all'asse su cui è montato l'encoder, che, ricordiamo, è da 300 cpr. Dato che stiamo lavorando a circa 30 MIPS, otteniamo:

$$T = 16,6 \quad DIV = 166$$

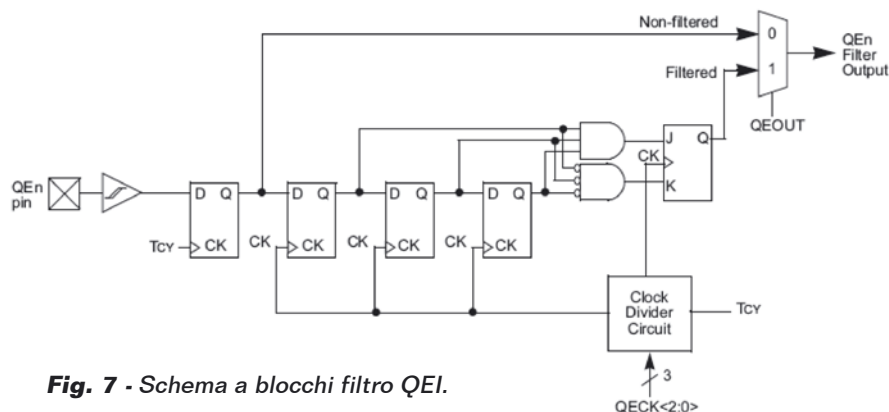
Adesso finalmente possiamo andare a configurare il registro DFLTCON; ci occupiamo solo dei bit riguardanti i filtri.

Bit7 QEOUT: abilita il filtro sugli ingressi QEA e QEB. Lo impostiamo a uno.

Bit6-5-4 QECK: questi tre bit ci permettono di impostare il fattore di divisione che abbiamo calcolato (valido per tutti e tre gli ingressi del modulo). I valori possibili sono 000=1, 001=2, 010=4, 011=16, 100=32, 101=64, 110=128, 111=256. Nel nostro caso scegliamo il più vicino a 166, quindi QECK = 6 (110)

Bisogna precisare che per alcuni modelli di dsPIC è possibile impostare in maniera separata un filtro per i pin QEA-QEB e uno per il pin INDX, quindi vengono messi a disposizione altri 4 bit che solitamente sono il 3 e i 2-1-0.





In ogni caso, ciò nella nostra applicazione risulta indifferente, perchè non abbiamo index. Quanto detto fino ad ora su QEICON e DLFTCON, si trova all'interno della funzione *Qei_Init()* del file *Qei.c* del progetto (scaricabile dal sito www.elettronica.in.it). Nella *Qei_Init()* noterete anche che il registro MAXCNT: è stato impostato al numero massimo di clic che si possono ottenere in un giro della ruota (visto che non abbiamo index) e che prima per sicurezza è effettuato un clean di POSCNT. Precisiamo che come per diverse periferiche, anche per la QEI o per la MCPWM vista nella prima puntata di questo corso, esistono apposite funzioni di libreria del C30 che permettono di configurare e utilizzare la periferica. Il problema è che per poterle impiegare, oltre a studiarne il manuale bisogna in ogni caso consultarne il datasheet, per capire con certezza a cosa si riferiscono i parametri d'impostazione da passare alle funzioni.

Questo, è solo uno degli inconvenienti, perchè trattandosi di una libreria, esiste la possibilità di bug o potenzialità della periferica non sfruttate a pieno, quindi è notevolmente più veloce e sicuro agire direttamente sui registri e capire con certezza quello che si sta facendo.

IL CALCOLO DELLA VELOCITÀ

Come promesso, abbiamo già imparato ad effettuare la decodifica 4x per la posizione. Adesso passiamo al calcolo della velocità, che realizzeremo derivando la posizione. Per fare ciò è necessario leggere a cadenza costante POSCNT, quindi ci serviremo del TIMER1, che abbiamo imparato ad usare nel precedente articolo. Faremo anche uso di una ISR (interrupt service routine) ad esso collegata, scatenata ogni volta che TMR1=PR1. Scegliamo come periodo di campionamento 0,005 s. Ricordando che FCY=29491200 (invariato rispetto agli esempi fatti nella prima

puntata di questo corso), per ottenere un campionamento di 5 ms è facile calcolare che occorre un prescaler del TIMER1 a 1:64 e $PIR1=2304$. Ad ogni istante di campionamento, la ISR permette la lettura di POSCNT e quindi il calcolo velocità. Prima di andare avanti, fac-

ciamo alcuni semplici, ma importanti calcoli, per capire se il tempo di campionamento scelto ci permetterà di avere buoni risultati. In particolare, ci interessa capire qual è la più piccola velocità che riusciremo a rilevare. Il motore può fare al massimo 200 rpm alla ruota, quindi 3,333 giri in un secondo. In questo intervallo (ricordando che in un giro della ruota rileviamo 36.000 impulsi) alla massima velocità abbiamo $36.000 \times 3,333 = 120.000$ impulsi dall'encoder.

Misurando lo spazio percorso ogni 5 ms, alla massima velocità otteniamo 600 impulsi di encoder, vale a dire che POSCNT si incrementa al massimo di 600.

In 5 ms o in qualsiasi altro intervallo scelto, è ovvio che il valore minimo di impulsi rilevabili è uno, quindi se nei 5ms non abbiamo nessun incremento di POSCNT, significa che siamo fermi o sotto la velocità minima rilevabile; in tal caso dovremmo utilizzare un tempo di campionamento più grande di 5 ms. Se alla massima velocità, in 5 ms contiamo 600 impulsi, quando ne contiamo uno è facile calcolare che il motore gira a 0,333 rpm cioè 1/3 di giro al minuto; questa è quindi la minima velocità rilevabile.

Abbiamo anche dedotto un importante dato: quale sarà la precisione massima di un nostro eventuale sistema di controllo; alla luce di ciò, ad esempio, potremo imporre una velocità di 134,666 rprm perché siamo in grado di rilevare 0,333 rpm.

Ricordiamo che in un sistema di controllo, se non possiamo misurare (o stimare, o ricostruire) una grandezza con una data precisione, non possiamo neanche imporre quella grandezza con quella precisione, perchè non riusciamo a sapere se poi effettivamente il sistema sta seguendo il riferimento, e quindi se è il caso o meno di effettuare un'azione correttiva. Ad esempio, non possiamo mai

imporre 100,5 rpm, ma al limite imponiamo 100,333 rpm oppure 100,666 rpm. Per quanto riguarda l'algoritmo di calcolo, che è molto semplice, lo trovate implementato nella funzione *SpeedCalc()* nel file *Qei.c*. In teoria, basta solamente sottrarre al valore attuale di POSCNT, quello all'istante del campionamento precedente e dividere proprio per il tempo di campionamento. Nella pratica bisogna anche gestire la situazione in cui sistematicamente POSCNT passa per il valore zero, sia in caso di conteggio ascendente che discendente. Per fare ciò si esegue un controllo sulla differenza effettuata fra i valori di POSCNT (variabile dx) verificando se assume valori anormali, che vanno ben oltre il massimo teorico.

Nel sorgente, noterete anche che la velocità è lasciata proprio nell'intervallo di variazione massimo di dx (-600, +600) senza riportarla in rpm. In pratica abbiamo la velocità in rpm×3; questo per un preciso scopo: avere a disposizione la massima risoluzione utilizzando numeri interi. Questa caratteristica ci tornerà utile in futuro, perché prossimamente pubblicheremo un articolo sul controllo di velocità di un motore in corrente continua sfruttando quello che stiamo imparando adesso. L'ISR, nella quale è richiamata la funzione *SpeedCalc()*, si trova nel file *Timers.c*.

LA VISUALIZZAZIONE NELL'LCD

Anche se non è obiettivo di questa trattazione, comunichiamo le informazioni fondamentali riguardanti la visualizzazione delle grandezze ricavate dall'encoder.

Faremo uso del display della dsPICdem2, un LCD 16×2 sul quale è possibile scrivere mediante il bus seriale SPI. In realtà il nostro dsPIC comunica mediante SPI con un chip che a sua volta pilota il display. Questo chip non è altro che un dsPIC30F2011 in SMD, saldato sulla board e già programmato. Non dobbiamo quindi preoccuparci di come avviene la comunicazione, ma solo di conoscere le funzioni per inviare i caratteri.

I file del progetto riguardanti la visualizzazione nell'LCD sono due: il primo, *Spi_for_Lcd.c* mette a disposizione la funzione *SPI_Init()* che inizializza il bus (non necessita di parametri) e la funzione *WriteSPI_to_LCD()* che non fa altro che prendere una stringa di testo opportunamente formattata e spedirla sul bus. Queste due funzioni sono usate nel *main()*, la prima solo una volta, la seconda ciclicamente dentro il main loop.

Il secondo file è *DisplayRoutines.c*, che implementa le funzioni per creare e formattare la stringa di testo da visualizzare: una prima funzione converte tipi di dati numerici (posizione e velocità) in stringa e una successiva funzione crea la variabile buffer usata da *WriteSPI_to_LCD()*.

Implementando il sistema vi accorgete dell'elevata sensibilità dell'encoder che fa variare il valore visualizzato su LCD, semplicemente se la ruota si muove sull'escursione del gioco meccanico dell'asse o addirittura mentre con le mani si fa girare la ruota in un senso, ma nell'impugnarla, la si muove impercettibilmente nel senso opposto.

In **Fig. 8** potete osservare il sistema completo in tutte le sue parti. Per vederlo in azione, visitate la pagina web www.calogerolombardo.altervista.org.

Per far ruotare il motore, basterà comandarlo in PWM come vi abbiamo insegnato a fare nella prima puntata del nostro corso sui dsPIC.

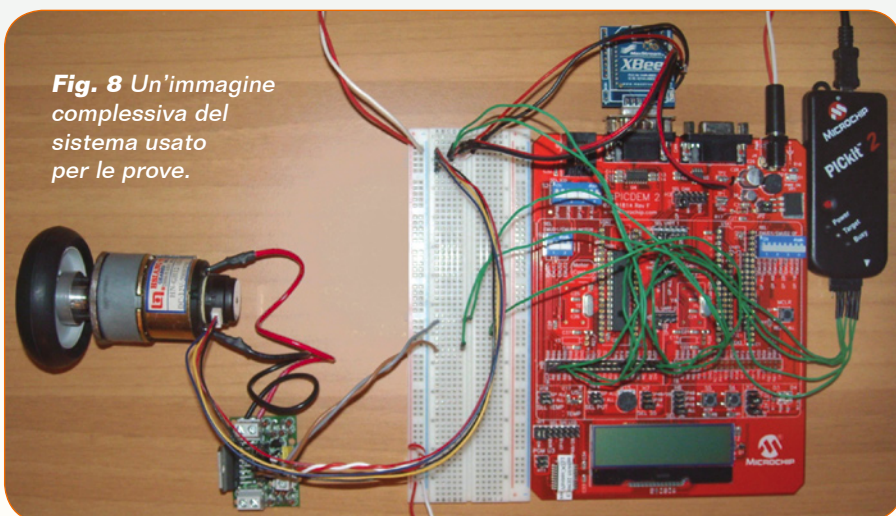


Fig. 8 Un'immagine complessiva del sistema usato per le prove.