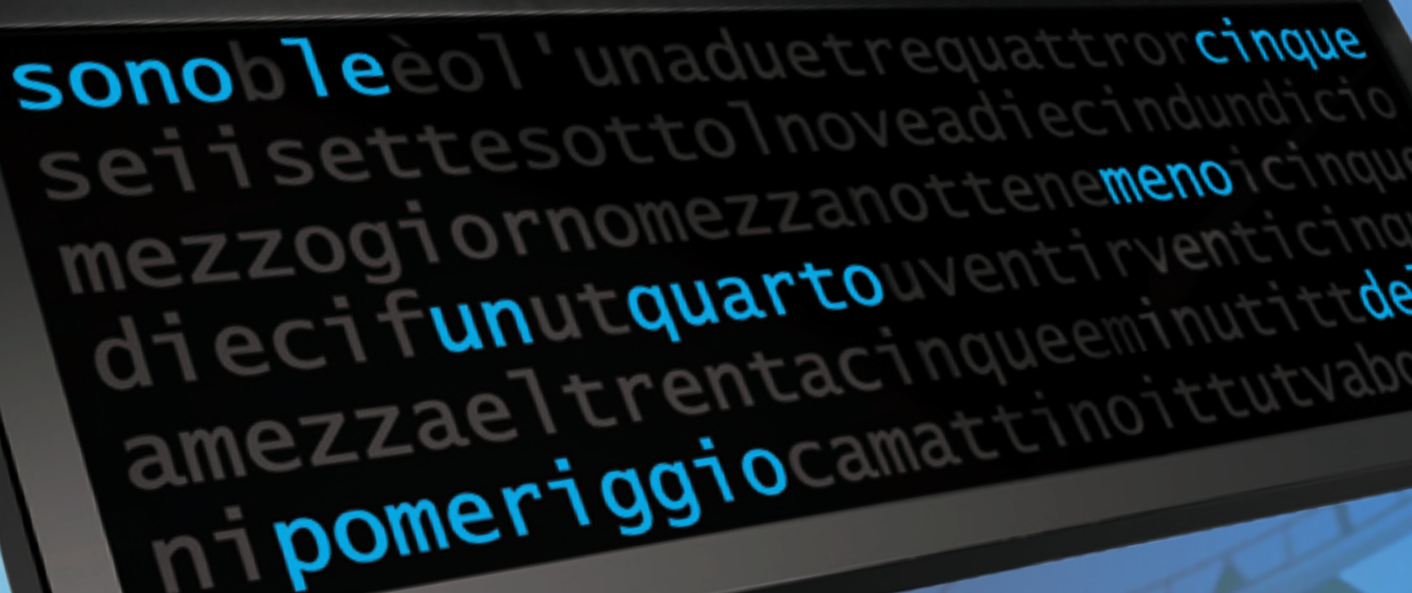


Un quadro d'arte moderna o una cornice? Qualunque cosa possa sembrare, è un orologio dal quadrante molto particolare, perché indica l'ora illuminando delle parole scritte su un pannello.



sono le è l' un a due tre quattro cinque
sei sette sotto nove a die ci undi cio
mezzogiorno mezzanotte ne meno cinque
dieci fun ut quarto venti rventi cin
a mezza e l trentacinque e minuti tte de
ni pomeriggio ca mattina o it tut va b

L' OROLOGIO CHE SCRIVE L'ORA

..... di BORIS LANDONI

Ci sono tanti modi per indicare l'ora: tralasciando l'antica meridiana (che segnava sui muri con l'ombra di un'asta l'orario) e portandoci nell'era moderna, dobbiamo dire che la tecnologia degli orologi ci dà la possibilità di scegliere tra molte forme, anche particolarmente originali. Accanto alle comuni lancette, negli anni abbiamo assistito alla nascita dei display, che ci mostrano l'ora in formato numerico e quindi estremamente

intuitivo anche per chi è poco avvezzo a leggere l'orario sommando le indicazioni delle lancette; non meno importanti sono gli orologi binari, che mostrano l'ora usando file di LED per comporre ore, minuti e secondi con l'accensione combinata di luci a formare i bit binari. Una rappresentazione molto particolare, quasi artistica di una certa avanguardia, è quella proposta nel progetto descritto in queste pagine: si tratta di un orologio digitale basato su ar-

Tabella 1 - Ecco come viene mostrata l'ora nel caso delle 12 antimeridiane e pomeridiane, per tutte le frazioni di cinque minuti.

ora	visualizzazione
00.00	è mezzanotte
00.05	è mezzanotte e cinque minuti
00.10	è mezzanotte e dieci minuti
00.15	è mezzanotte e un quarto
00.20	è mezzanotte e venti minuti
00.25	è mezzanotte e venticinque minuti
00.30	è mezzanotte e mezza
00.35	è mezzanotte e trentacinque minuti
00.40	è mezzanotte meno venti minuti
00.45	è mezzanotte meno un quarto
00.50	è mezzanotte meno dieci minuti
00.55	è l'una meno cinque minuti del mattino
12.00	è mezzogiorno
12.05	è mezzogiorno e cinque minuti
12.10	è mezzogiorno e dieci minuti
12.15	è mezzogiorno e un quarto
12.20	è mezzogiorno e venti minuti
12.25	è mezzogiorno e venticinque minuti
12.30	è mezzogiorno e mezza
12.35	è mezzogiorno e trentacinque minuti
12.40	è mezzogiorno meno venti minuti
12.45	è mezzogiorno meno un quarto
12.50	è mezzogiorno meno dieci minuti
12.55	è l'una meno cinque minuti del pomeriggio

chitettura Arduino, che permette di visualizzare l'ora illuminando le parole che la descrivono. Il visualizzatore, grande e chiaro, contiene 30 parole con cui comporre le frasi che usiamo normalmente per dire, a chi ce lo domanda, che ora è. Quindi, se sono le 11.30, il display accenderà le parole: *sono le*

undici e mezza del mattino. Se sono le 20.00, verranno accese le parole: *sono le otto del pomeriggio*. La cosa particolare è che, per semplificare il circuito e il visualizzatore, nonché per dare un tocco di originalità all'orologio, le ore vengono mostrate accendendo le parole che ne compongono la descrizione, ma non in sequenza; in pratica le frasi vengono composte accendendo le parole che le compongono, le quali sono sparse nell'area del display, mischiate con caratteri fittizi.

L'orario è composto dalle ore, dai minuti (è prevista una variazione ogni 5 minuti) e dalla scritta "del mattino/del pomeriggio".

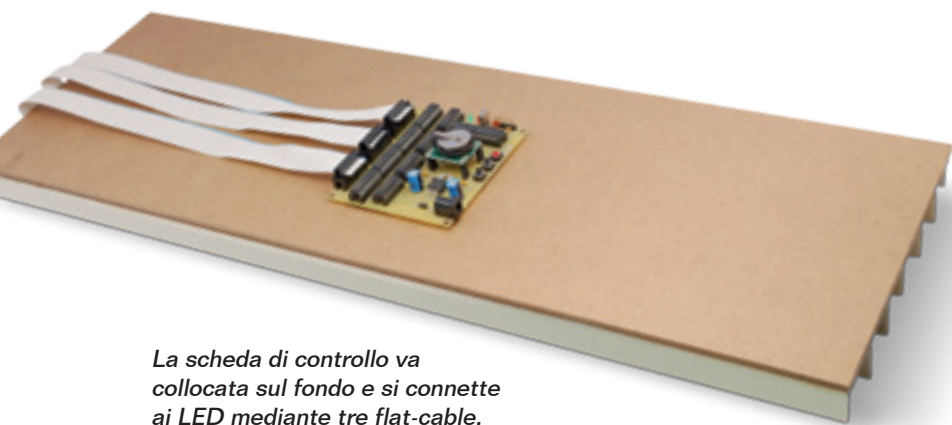
Per questa ragione, la composizione di tutti i possibili orari a frazioni di cinque minuti richiede 30 parole, cioè 13 per le ore (*l'una, due, tre, quattro, cinque, sei, sette, otto, nove, dieci, undici, mezzogiorno e mezzanotte*) 2 per mattino e pomeriggio, 7 per le frazioni di cinque minuti (*quarto, venti, venticinque, mezza, trentacinque, tre quarti*) più *sono e le* per indicare l'orario; completano il set, le parole *e, è, del, un, meno e minuti*.

Disponendo di questi vocaboli, gli orari vengono visualizzati con apposite composizioni; per averne un'idea, guardate la **Tabella 1**, dove viene mostrata la visualizzazione del mezzogiorno e della mezzanotte.

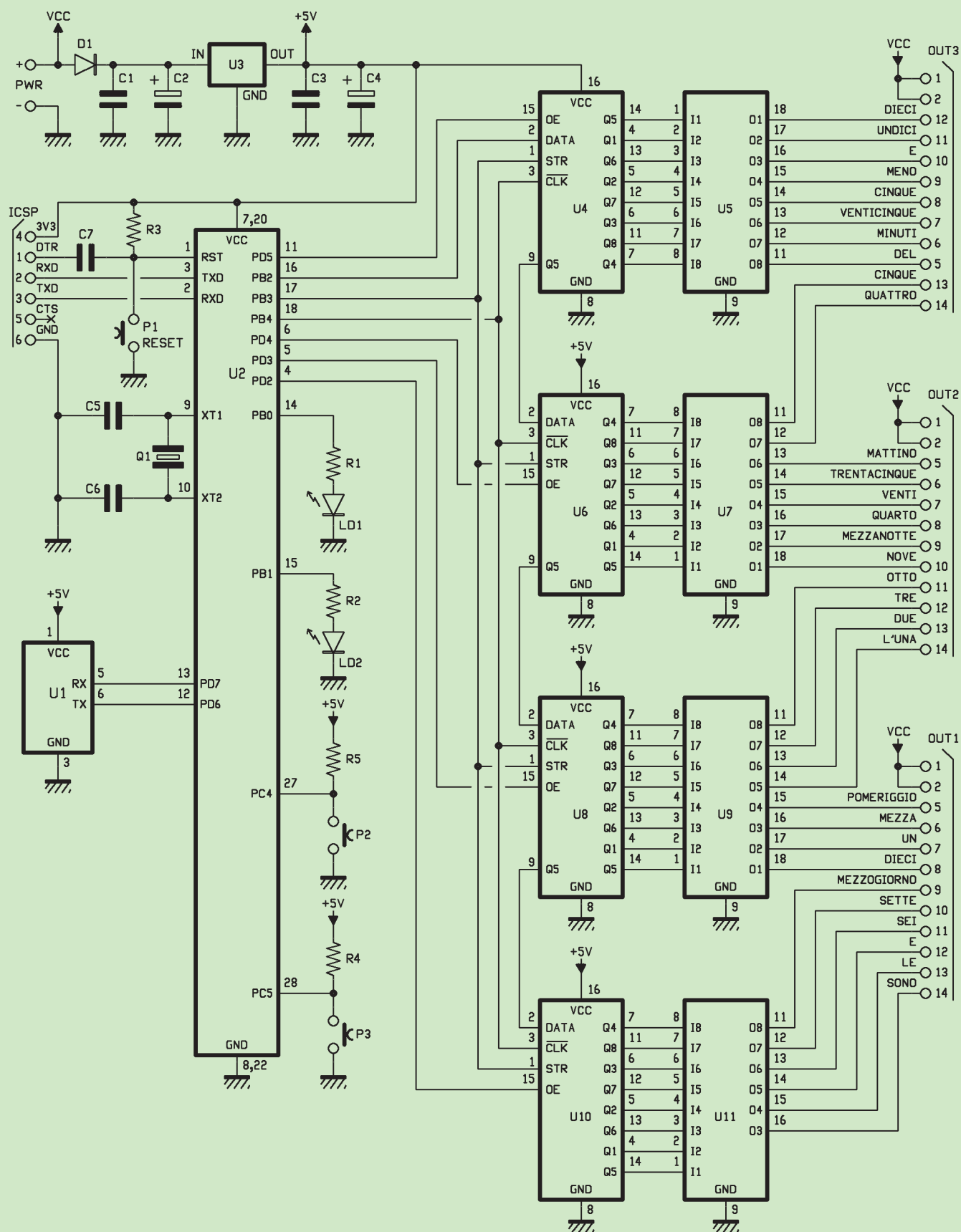
Questo ed altro vi apparirà più chiaro esaminando il circuito elettronico che permette di realizzare l'orologio ed il rispettivo visualizzatore.

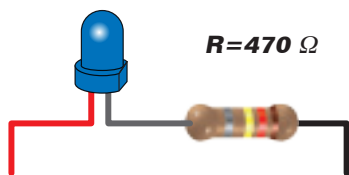
SCHEMA ELETTRICO

Il dispositivo è sostanzialmente un modulo Arduino basato su un microcontrollore ATmega328P-PU, su cui abbiamo montato anche un RTC (Real Time Clock) di tipo FT674M per mantenere l'ora e comunicarla al micro quando questo la richiede; l'RTC è un modulo già utilizzato in altre occasioni ed ampiamente descritto nel fascicolo n° 117. Di Arduino, il nostro orologio conserva l'architettura hardware ed il programma, che è un apposito sketch. Per ottenere la particolare visualizzazione, che avviene mediante 30 file di LED, ognuna destinata ad illuminare una delle 30 scritte componenti il visualizzatore, il microcontrollore si avvale di un gruppo di shift-register, ognuno dei quali pilota un array di Darlington grazie ai quali è possibile disporre della corrente necessaria ad alimentare i diodi luminosi. Siccome si tratta di pilotare 30 gruppi di diodi, per accendere simultaneamente le singole parole occorrerebbero altrettante linee del microcontrollore, anche se non usate tutte insieme; non disponendo, il nostro ATmega328, di tanti pin di I/O, abbiamo pensato di ricorrere al comando seriale, convertendo la stringa in formato parallelo e pilotando i LED con ciò che ne risulta. In pratica ripartiamo le 30 linee destinate ai LED fra quattro registri a scorrimento ad 8 bit e pilotiamo questi ultimi fornendo loro dei byte seriali rappresentanti lo stato di un gruppo di otto linee alla volta; quindi, iniziamo la sequenza generando il byte relativo al primo registro, poi

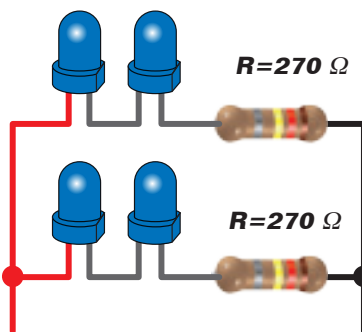
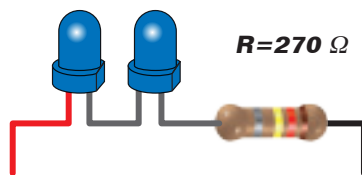
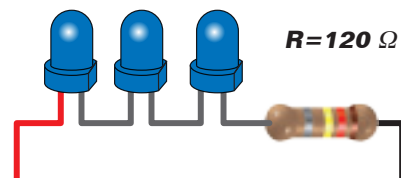


La scheda di controllo va collocata sul fondo e si connette ai LED mediante tre flat-cable.





Collegamento dei LED per le parole che ne richiedono 1, 2, 3 o 4; nel caso di parole da sei LED fate due serie da tre in parallelo, per quelle da 9 utilizzate tre serie da 3 in parallelo, per quelle da 8, due serie da tre e una da due in parallelo e così via.



quello del secondo, quindi quello del terzo e poi quello del quarto (in quest'ultimo caso vengono emessi solo 6 bit) dopodiché riprendiamo dall'inizio. Dunque, ogni visualizzazione si ottiene facendo generare al microcontrollore una stringa di 30 bit. Per capire meglio il funzionamento del circuito, descriviamo lo schema elettrico a partire dal microcontrollore Atmel, che dopo il reset al power-on inizializza le proprie linee di I/O impostando PB2, PB3, PB4, PD2, PD3, PD4 e PD5 come output destinati alla gestione della logica discreta, ossia degli shift-register; più esattamente, PB2 genera i dati in forma seriale, che ogni registro deve ricomporre sulle proprie uscite, ossia genera 30 bit per ciascuna sequenza di visualizzazione. Ora dovete notare che i registri sono collegati in cascata, il che significa che tutti sono interessati dai dati prodotti dal micro; per questa ragione, i singoli bit, corrispondenti ognuno allo stato di una delle linee, "camminano" dalla prima all'ultima uscita del primo registro, entrano nel secondo e passano dalla prima all'ultima uscita, quindi entrano nel terzo e

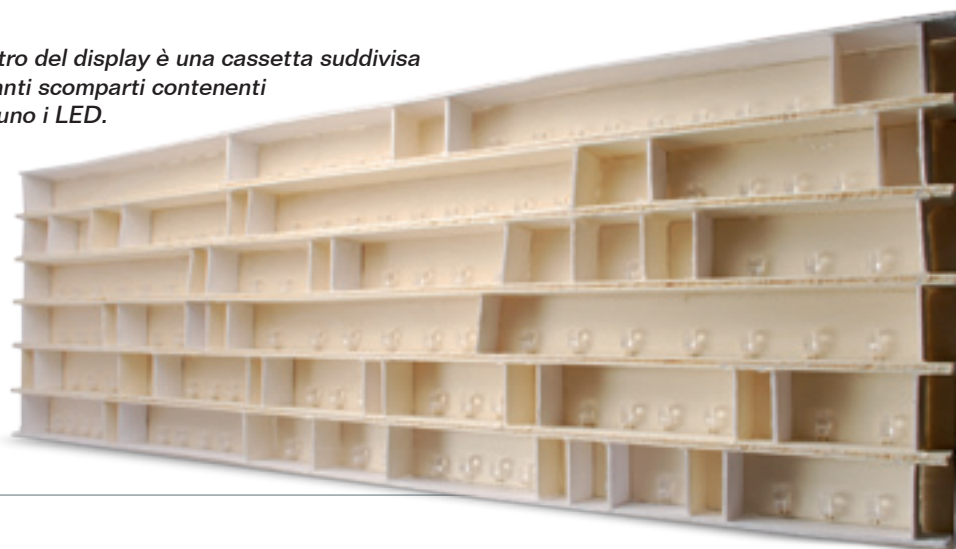
terminano nel quarto. Completato l'invio di 30 bit, viene comandato l'aggiornamento del visualizzatore mediante la linea di strobe; in pratica il microcontrollore fornisce un impulso a livello alto dal piedino 17 (PB3) triggerando le linee STR (pin 1) degli shift-register e forzando i latch di uscita dei quattro a caricare l'attuale stato dei singoli registri. Quando le STR tornano a livello logico basso, le uscite degli shift-register mantengono la condizione caricata fino al successivo aggiornamento. Ciò permette di mantenere accese le linee di LED corrispondenti alle parole desiderate anche se i dati all'ingresso degli shift-

register cambiano o vengono a mancare.

Questa tecnica, rispetto ad un multiplex, consente di lavorare a velocità relativamente basse e quindi evita di sovraccaricare di lavoro il microcontrollore; infatti basta emettere periodicamente una stringa rappresentante lo stato dei LED per accendere le parole che mostrano l'ora, poi si è a posto per i prossimi 5 minuti. L'aggiornamento serve infatti ogni 5 minuti, non più frequentemente.

Siccome per illuminare e accendere la gran parte delle parole del visualizzatore occorre usare più di un LED, si pone il problema di come accenderlo usando le linee di uscita dei registri a scorrimento, i quali sono alimentati a 5 V (per accettare i livelli logici generati dal microcontrollore, che sono TTL-compatibili) e possono erogare correnti non superiori a 5 mA. Volendo usare LED blu, che richiedono una tensione diretta di polarizzazione dell'ordine dei 3,5 volt, con ciò che possiamo prelevare dalle uscite dei 4094 potremmo al massimo accenderne uno per ciascuna uscita. Abbiamo quindi dovuto optare per l'uso di line-driver ULN2803, ognuno dei quali integra otto Darlington NPN configurati in modo open-collector, con tanto di resistenze di polarizzazione e diodi di protezione sul collettore

Il retro del display è una cassetta suddivisa in tanti scomparti contenenti ognuno i LED.



(i catodi di tutti i diodi sono collegati al positivo di alimentazione dell'integrato, ossia al pin 10, qui non collegato perché i diodi non ci servono).

Ciascun ULN2803 ha gli otto ingressi (che terminano sulle resistenze di base dei Darlington) collegati alle uscite di uno degli shift-register, ad eccezione dell'U11, del quale usiamo solo sei ingressi ed altrettante uscite; le uscite dei line-driver pilotano i LED delle scritte, ognuno dei quali deve essere collegato con il catodo ed avere l'anodo polarizzato, mediante una resistenza di caduta, dalla linea positiva comune, che fa capo al contatto +PWR del circuito.

Prevedendo di alimentare l'insieme con 12 volt in continua e usando LED blu, ogni linea di uscita potrà pilotare al massimo tre diodi in serie e più serie di due o tre LED; l'assorbimento non è un problema, perché ciascun Darlington dell'ULN2803 può assorbire tranquillamente 500 mA.

Bene, spiegato come avviene la visualizzazione, che poi è la parte cruciale del progetto, torniamo per un istante all'inizializzazione delle linee di I/O da parte del microcontrollore: PB0 e PB1 vengono settate come uscite e pilotano i LED di segnalazione del dispositivo, utili durante le impostazioni, che si fanno con i pulsanti P2 e P3, connessi rispettivamente a PC4 e PC5, linee configurate come input senza pull-up (tale configurazione viene ottenuta grazie alle resistenze R4 ed R5).

Quanto alle linee PD6 e PD7, vengono inizializzate rispettivamente come ingresso e come uscita e servono a dialogare con il modulo RTC, interfacciandosi con le sue linee TX ed RX.

Completa la panoramica sugli I/O del microcontrollore la sezione dedicata alla programma-

zione in-circuit, relativa alle linee RST, TXD ed RXD; come tutti gli Atmel destinati ad equipaggiare hardware Arduino, il nostro microcontrollore ha precaricato un bootloader che si attiva quando premiamo il pulsante P1 ed effettuiamo quindi il reset.

L'ATmega 328 da noi utilizzato

funziona con un clock di 16 MHz, assicurato dal quarzo Q1.

L'intero circuito viene alimentato a 12 Vcc tramite i punti + e - PWR; da essi viene prelevata l'alimentazione per i LED del visualizzatore. Invece, i 5 volt stabilizzati necessari alla logica vengono ricavati da un regolato-



Che ore sono? Fate un po' i conti!

Il marchio DeTomaso non significa solo automobili: è infatti impegnato anche nel design di orologi futuristici come lo Spacy Timeline, un magnifico segnatempo da polso quarzato, con quadrante rettangolare dove l'ora viene indicata mediante tre file verticali di LED. È abbastanza simile ad un orologio binario (tipo quello da noi pubblicato nel fascicolo n° 133) solo che utilizza una fila di 12 LED per le ore, 11 per i minuti (a passi di 5 alla volta) ed altri quattro per mostrare i minuti mancanti al totale; per comporre l'orario accende complessivamente i LED che appartengono alla fila delle ore, quelli della fila dei minuti (quelli da 5') e quelli relativi ai singoli minuti. Ad esempio, per indicare le 10 e 19' accende i primi dieci LED delle ore, i tre LED che indicano 15 minuti più i quattro di destra (15+4 fa 19 minuti). Premendo il pulsante MODE mostra la data, usando i LED delle ore per i mesi e quelli dei minuti per i giorni e accendendo il LED DATE. Insomma, leggere quest'orologio non è proprio immediato, però portarlo al polso attirerà l'attenzione su di voi, se non altro perché si tratta di un oggetto che al design accattivante e tecnologico aggiunge quel tocco enigmatico che non guasta mai. Magari, se qualcuno vi chiede l'ora non fategliela vedere: ditegliela voi...

Tabella 2

FRASE	NUM. LED	SUDDIVISIONE (serie e paralleli)
sono	2	2
le	1	1
è	1	1
l'una	2	2
due	2	2
tre	2	2
quattro	5	3 // 2
cinque	5	3 // 3
sei	2	2
sette	4	2 // 2
otto	3	3
nove	3	3
dieci	4	2 // 2
undici	4	2 // 2
mezzogiorno	7	2 // 2 // 3
mezzanotte	6	3 // 3
e	1	1
meno	3	3
cinque	4	2 // 2
dieci	3	3
un	1	1
quarto	4	2 // 2
venti	3	3
venticinque	7	2 // 2 // 3
mezza	4	2 // 2
trentacinque	9	3 // 3 // 3
minuti	5	3 // 2
del	2	2
pomeriggio	8	3 // 3 // 2
mattino	5	3 // 2

re lineare integrato 7805, posto a valle del diodo di protezione (D1) dall'inversione di polarità.

IL MODULO REAL TIME CLOCK

A verificare ed aggiornare l'ora provvede un'apposita routine del firmware che "gira" nel microcontrollore, il quale gestisce i dati in arrivo dal modulo RTC: quest'ultimo è un circuito basato su un chip della Maxim-Dallas siglato DS1307 e da un'interfaccia realizzata mediante un piccolo microcontrollore Microchip. Dispone di una linea dati seriale bidirezionale (TX/RX) a livello TTL, di un set di linee per la

programmazione in-circuit, di un I²C-bus e dell'ingresso V_{pp}. L'RTC è dunque l'orologio vero e proprio e l'ATmega328 lo interroga una volta al secondo (tramite l'interfaccia seriale che fa capo alle linee TX ed RX e che funziona in half-duplex ad una velocità di 9.600 bps, 8 bit di dati, nessuna parità e 1 bit di stop), ne acquisisce e memorizza i dati, quindi trasforma i byte contenenti ore, minuti e secondi in valori binari delle linee di I/O da usare per accendere i LED.

Notate che il microcontrollore dell'RTC implementa due linee di comunicazione: una I²C usata per comunicare col DS1307 ed una seriale che serve a dialogare con il sistema in cui il modulo viene montato (nel nostro caso, con l'ATmega 328).

Il DS1307 è sostanzialmente un contatore BCD a basso consumo, che conta secondi, minuti, ore, giorni, mesi e anni ed è provvisto di 56 byte di RAM statica non volatile; il componente provvede automaticamente a determinare quali sono i mesi con meno di 31 giorni e ad effettuare la correzione per l'anno bisestile, anche se questa funzione non ci interessa perché ci basta avere l'ora e non la data. L'orologio può operare nelle modalità 12 o 24 ore, con indicazione delle ore antimeridiane (AM) e di quelle pomeridiane (PM).

Le informazioni sull'ora e la data vengono collocate in un registro e trasferite dietro interrogazione al microcontrollore dell'RTC, il quale è configurato per inviarle all'esterno su specifica richiesta dell'ATmega328 del nostro orologio.

Oltre a quanto detto, il DS1307 dispone di un'uscita di clock programmabile, che restituisce un'onda quadra alla frequenza di clock dell'orologio divisa, in base

alle impostazioni, fino ad ottenere 1 Hz, 4.096 kHz, 8.192 kHz o l'intero clock. La frequenza di 1 Hz può servire a far lampeggiare i due punti o il punto dei secondi se realizziamo un orologio a display numerico; nel nostro caso tale funzione non è utilizzata.

Il DS1307 dispone anche di un circuito di controllo in grado di verificare la mancanza dell'alimentazione principale (V_{cc}) e fare in modo che la tensione occorrente venga prelevata dalla batteria di backup, la quale nel caso del nostro RTC è una pila a bottone da 3 V, che deve essere collegata tra il piedino 3 (V_{batt}) e la massa di alimentazione (4, ossia GND).

La sezione di controllo dell'alimentazione è dimensionata in modo da svolgere due compiti: preservare la memoria in cui sono contenuti i dati sull'ora e la data attuali e mantenere attivo il contatore dell'orologio.

Il primo viene svolto conservando nella RAM non volatile le informazioni corrispondenti, mentre il secondo è espletato dall'alimentazione di riserva presa dalla pila o batteria tampone collegata fra il piedino 3 e il 4. Oltre a ciò, va detto che la sezione di controllo dell'alimentazione interviene sulle altre funzioni del DS1307, bloccando la comunica-



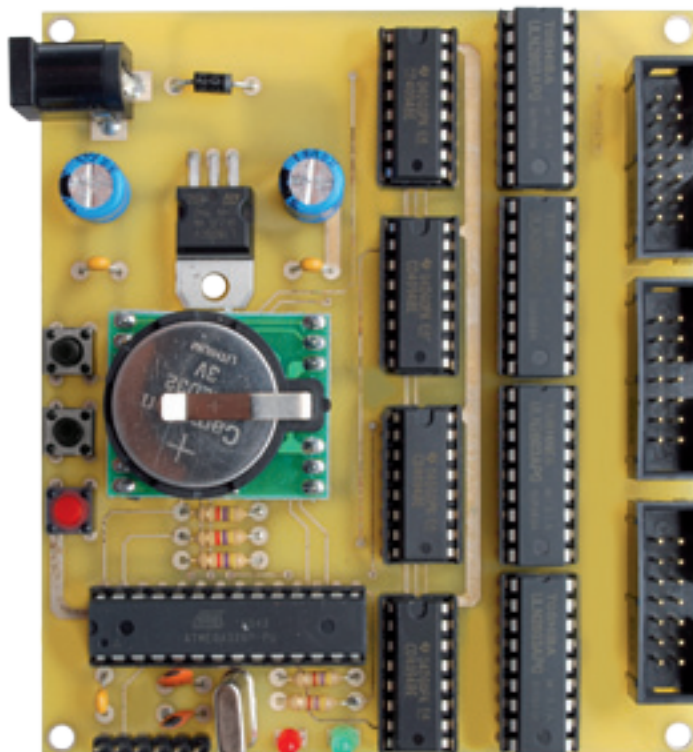
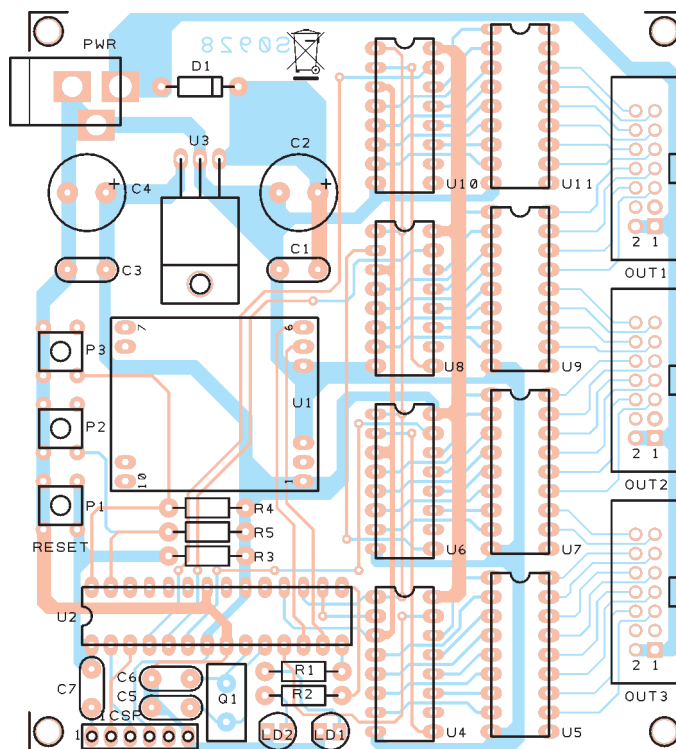
Il modulo RTC è stato pubblicato nel fascicolo n° 117.

Elenco Componenti:

R1: 470 ohm
 R2: 470 ohm
 R3: 4,7 kohm
 R4: 4,7 kohm
 R5: 4,7 kohm
 C1: 100 nF multistrato
 C2: 220 μ F 25 VL elettrolitico
 C3: 100 nF multistrato
 C4: 220 μ F 25 VL elettrolitico
 C5: 15 pF ceramico
 C6: 15 pF ceramico
 LD1: LED 3 mm verde
 LD2: LED 3 mm rosso
 D1: 1N4007
 U1: Modulo RTC (FT674M)
 U2: ATMEGA328P-PU (MF928)
 U3: 7805
 U4: CD4094BE
 U5: ULN2803
 U6: CD4094BE
 U7: ULN2803
 U8: CD4094BE
 U9: ULN2803
 U10: CD4094BE
 U11: ULN2803
 Q1: Quarzo 16 MHz
 P1: Microswitch
 P2: Microswitch
 P3: Microswitch

Varie:

- LED blu mod. L5WCN5 (45 pz.)
- Resistenza 120 ohm (17 pz.)
- Resistenza 270 ohm (25 pz.)
- Resistenza 470 ohm (3 pz.)
- Plug alimentazione
- Zoccolo 8+8 (4 pz.)
- Zoccolo 9+9 (4 pz.)
- Strip maschio 6 poli
- Strip femmina 2 poli (2 pz.)
- Strip femmina 3 poli (2 pz.)
- Connettore DIL 24 contatti da CS (3 pz.)
- Connettore DIL 24 contatti volante (6 pz.)
- Flat cable 24 vie (2 metri)
- Circuito stampato



zione con l'I²C-bus e l'emissione dell'eventuale segnale di clock ausiliario dal piedino 7; il tutto, allo scopo di minimizzare il

consumo di energia ed estendere quanto più possibile l'autonomia, la quale, comunque, può arrivare ad alcuni mesi. Il blocco

di controllo interviene quando la tensione letta tra il piedino dell'alimentazione principale (8) e massa (4) è inferiore ad 1,25

Costruiamo il visualizzatore

1) Stampate su fogli di acetato trasparente le due pagine che formano il layout del pannello del visualizzatore.

sonoble è o l' unadue	trequattro cinque
sei sette sotto l' no	vea dieci undici
mezzogiorno mezzan	ottenemeno cinque
dieci funut quartou	venti venticinque
amezza el trentacin	que eminuti ttdel ro
nipomeriggio camat	tino it tutvabo aici

Pag. 1

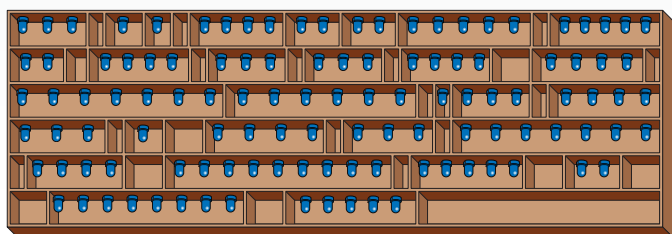
Affiancare

Pag. 2

2) I due fogli vanno affiancati senza lasciare spazio e uniti con del nastro adesivo trasparente.

sonoble è o l' unadue	trequattro cinque
sei sette sotto l' no	vea dieci undici
mezzogiorno mezzan	ottenemeno cinque
dieci funut quartou	venti venticinque
amezza el trentacin	que eminuti ttdel ro
nipomeriggio camat	tino it tutvabo aici

Nastro adesivo



Pannello di legno con gli scompartimenti per i LED.

3) Sovrapporre il layout alla base in legno stando attenti che le parole risultino centrate nei separatori.

sonoble è o l' unadue	trequattro cinque
sei sette sotto l' no	vea dieci undici
mezzogiorno mezzan	ottenemeno cinque
dieci funut quartou	venti venticinque
amezza el trentacin	que eminuti ttdel ro
nipomeriggio camat	tino it tutvabo aici

Separazione delle parole

4) Applicate il layout al contenitore e fermate con una cornice che dia all'insieme l'aspetto di un quadro.



volte la tensione della pila. Il micro presente nel modulo RTC interfaccia il DS1307 per la configurazione (effettuata dall'ATmega328 all'avvio del programma principale) e per

interrogare il chip della Maxim-Dallas usando il suo I²C-Bus, quando giunge la richiesta dell'ora dalla linea seriale TX/RX. Il DS1307 risponde inviando al microcontrollore (sempre lungo

il suo bus I²C) la risposta e le informazioni su ora e data, il quale li "gira" al dispositivo esterno che ha inviato la richiesta. Il modulo RTC deve essere alimentato con una tensione continua ben

stabilizzata del valore di 5 volt e assorbe una corrente dell'ordine di 1,5 milliampere, che scende a 500 nA nel funzionamento a pila (in caso di mancanza dei 5 V principali).

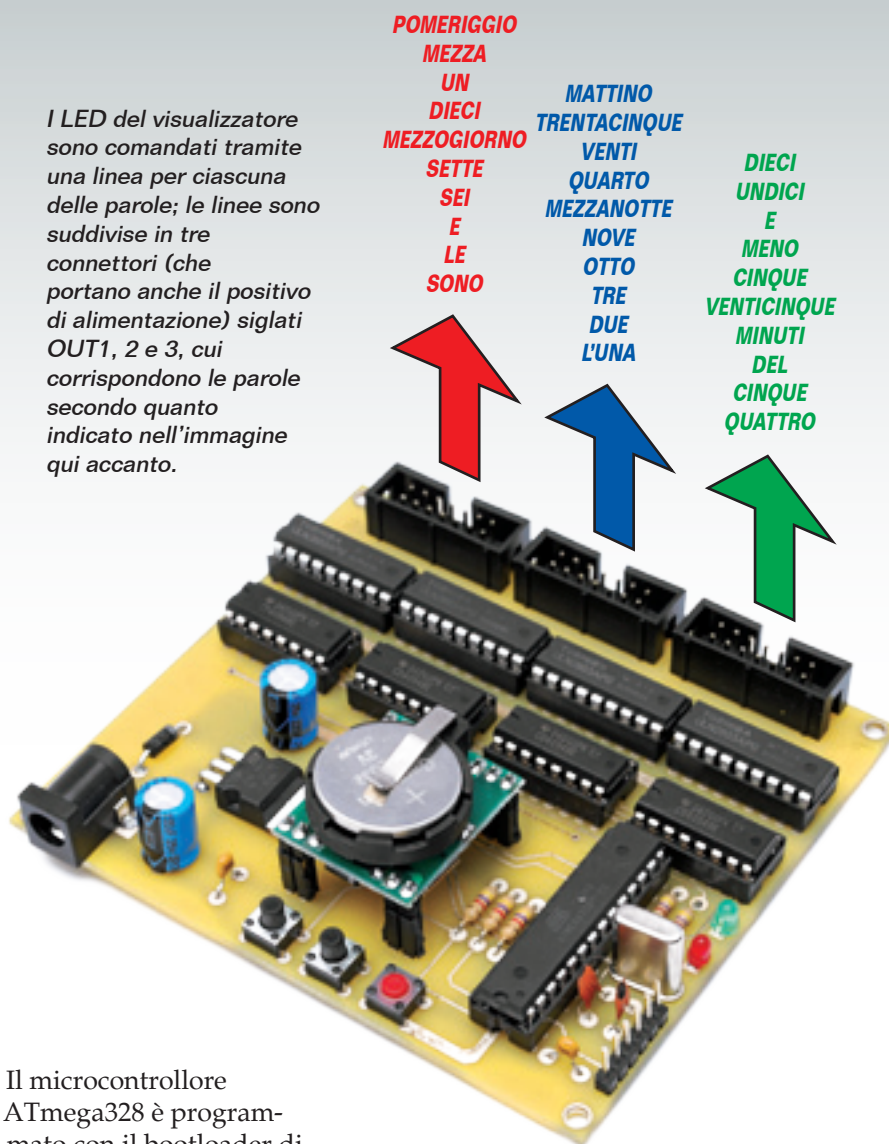
COME FUNZIONA

Bene, descritto lo schema elettrico dell'orologio vediamo brevemente come funziona l'insieme, partendo dai tre pulsanti: P1 esegue il reset del microcontrollore e predispone il circuito al caricamento dello sketch.

Il pulsante P2 (OK) collegato alla porta PC4 dell'U2 permette di accedere alla modifica dell'ora e serve per confermare ogni modifica fatta durante tale procedura, mentre l'ultimo pulsante (P3, ossia UP) collegato alla porta PC5, incrementa ore e minuti durante le impostazioni dell'orologio. Non è previsto un tasto di decremento. Quando dobbiamo impostare l'ora, dopo aver premuto P2 accediamo all'impostazione delle ore: con P3 avanziamo di un'unità alla volta e per confermare premiamo nuovamente P2; a questo punto entriamo nell'impostazione dei minuti, che definiamo premendo più volte il solito P3 e avanzando di un'unità per ogni pressione. Premendo adesso P2, abbandoniamo la procedura e salviamo le modifiche all'orologio; notate che una volta premuto il tasto OK a seguito della modifica dei minuti, tutti i LED si accenderanno per un istante, a confermare la modifica.

Premendo P2 dopo l'impostazione dell'orologio, i LED si accenderanno allo stesso modo per due volte per confermare l'operazione. A questo punto, premendo il solito tasto, dopo un secondo tutti i LED si illumineranno per circa tre secondi, dopodiché si uscirà dall'impostazione e verrà visualizzato l'orario inserito.

I LED del visualizzatore sono comandati tramite una linea per ciascuna delle parole; le linee sono suddivise in tre connettori (che portano anche il positivo di alimentazione) siglati OUT1, 2 e 3, cui corrispondono le parole secondo quanto indicato nell'immagine qui accanto.



Il microcontrollore ATmega328 è programmato con il bootloader di Arduino 2009; per la programmazione, ossia per caricare lo sketch che abbiamo previsto per il controllo dell'orologio, è necessario utilizzare un convertitore USB/TTL (va bene il modello 7300-FTDI5V Futura Elettronica, www.futurashop.it) in quanto il circuito dispone di una semplice interfaccia seriale a livello TTL. Il converter si collega al pin-strip corrispondente al connettore ICSP.

Prima di passare alla realizzazione dell'orologio, spendiamo due righe sullo sketch: si tratta di un firmware che gira in loop andando a verificare la pressione del tasto OK e leggendo continuamente l'ora.

Rilevare la pressione del pulsante P2 serve a decidere se il microcontrollore deve entrare nella procedura di impostazione dell'ora; se non viene rilevata la chiusura di tale pulsante, il firmware interroga ciclicamente il modulo RTC e ne deduce l'informazione sull'ora corrente. L'interrogazione avviene sfruttando la libreria di Arduino *NewSoftSerial*, che permette di utilizzare due pin di I/O digitale del microcontrollore come una porta seriale virtuale. Ogni cinque minuti esatti, avviene l'aggiornamento dell'ora: il micro crea una stringa di 30 bit che corrispondono all'impo-

stazione delle linee di comando dei LED all'uscita del circuito, necessaria a comporre l'ora con le scritte, quindi la trasmette serialmente alla catena di shift-register mediante la propria linea PB2. Nella visualizzazione è coinvolta la routine "controlCase", che fa un'analisi dei minuti e delle ore e in base al loro valore decide quali led e quindi quali scritte accendere.

REALIZZAZIONE PRATICA

Per costruire l'orologio dovete realizzare due unità: il circuito elettronico vero e proprio ed il quadrante o visualizzatore. Partiamo con il primo, per ottenere il quale dovete procurarvi tutti i componenti occorrenti e realizzare il circuito stampato, del quale trovate le tracce lato rame nel nostro sito www.elettronica.in.it; il montaggio dei componenti inizia da quelli a più basso profilo, ossia resistenze, diodi al silicio e zoccoli per gli integrati, quindi prosegue con i condensatori (prima quelli non polarizzati e poi gli elettrolitici, badando alla polarità di questi ultimi) i pulsanti, il quarzo e i LED.

In ultimo vanno montati i pin-strip per la connessione dei gruppi di LED del visualizzatore, il montaggio del modulo RTC (che si compera già pronto e programmato presso la ditta Futura Elettronica, www.futurashop.it) la programmazione in-circuit (ICSP) nonché la presa plug di alimentazione.

Per non sbagliare, durante tutte le fasi del montaggio non perdetevi d'occhio il disegno di disposizione dei componenti, che vi mostra l'orientamento degli elementi polarizzati.

E veniamo al visualizzatore, che va realizzato approntando prima una base ed il pannello con le scritte in negativo da mettergli

davanti; la base è sostanzialmente un contenitore formato da tanti scomparti quante sono le parole da "accendere", ognuno dei quali deve combaciare con ciascuna delle scritte, delimitandola inferiormente. Lo scopo di ciascuno scomparto è contenere i LED che illuminano la parola corrispondente; affinché ci sia la massima resa luminosa, l'interno di ciascuno scomparto deve essere di colore chiaro, altrimenti assorbe la luce o parte di essa. Per questa ragione, è opportuno prevedere scomparti, privi di LED, che delimitino i caratteri fittizi presenti nel quadrante (tali caratteri servono sostanzialmente per camuffare le parole utili che non vengono accese).

La base, che nel nostro caso misura 60x20 cm (queste misure non sono particolarmente vincolanti, nel senso che potete adattare alle vostre esigenze, dimensionando in proporzione la pellicola da porre davanti alla base) può essere realizzata in legno compensato da 4 o 5 mm di spessore, materiale con cui possiamo costruire sia il fondo, sia le pareti di delimitazione degli scomparti.

Per delimitare correttamente ciascuno scomparto, stampate su acetato la pellicola con il layout delle parole del visualizzatore; questo layout è riportato nella **Fig. 1**, nella quale notate che le scritte sono in negativo, in quanto durante il funzionamento devono far passare la luce per risultare visibili.

Completata la base, dovete sistemare al suo interno i LED, che variano in base alla lunghezza della parola corrispondente; per le parole più lunghe, vanno usati LED in serie o, essendo la tensione di alimentazione limitata a 12 V e quindi capace di alimentare rami con un massimo di 3 elementi in serie.

La **Tabella 2** indica quanti diodi servono e quale configurazione adottare per accendere le singole parole del visualizzatore. A tal proposito ricordate che se usate un solo LED per ramo, dovete porgli in serie una resistenza da 470 ohm, mentre con due LED in serie bisogna usare una resistenza da 270 ohm e con tre, il resistore di limitazione deve essere da 120 ohm.

Le tre configurazioni da adottare sono visibili nelle illustrazioni.

Per fissare i LED, vi consigliamo di realizzare dei fori nel compensato con una punta da 1,5 mm e infilarvi i terminali, piegando questi ultimi nella faccia posteriore del fondo così da fermare i singoli diodi.

Quanto alle connessioni, ricordate che gli anodi dei LED vanno rivolti al positivo di alimentazione (contatti 1 e 2 dei connettori OUT1, OUT2 e OUT3) e i catodi corrispondono alle linee di uscita delle parole. ■



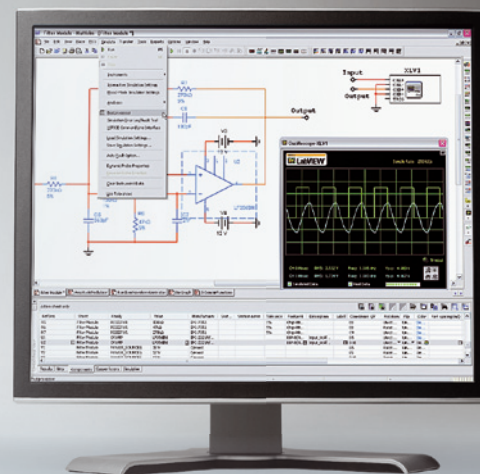
per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato e quello delle scritte possono essere scaricati dal sito dell rivista (www.elettronica.in.it) così come lo sketch utilizzato. L'integrato AVR con boot-loader (cod. 7300-DEV09217, Euro 7,00) e il Real Time Clock (cod. 7100-FT674M Euro 18,00) sono reperibili presso Futura Elettronica. I prezzi sono comprensivi di IVA.

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

Conoscere e usare NI Multisim

Progettazione più semplice,
migliore simulazione.



Studiamo nei dettagli la netlist e i metodi di sbroglio manuale che Multisim rende disponibili ad integrazione delle funzioni di autorouting, per consentire all'utente il controllo preciso del posizionamento delle tracce durante la fase di realizzazione del prototipo. Settima puntata.

di Enrico Ambrosini e Filippo Spadaro

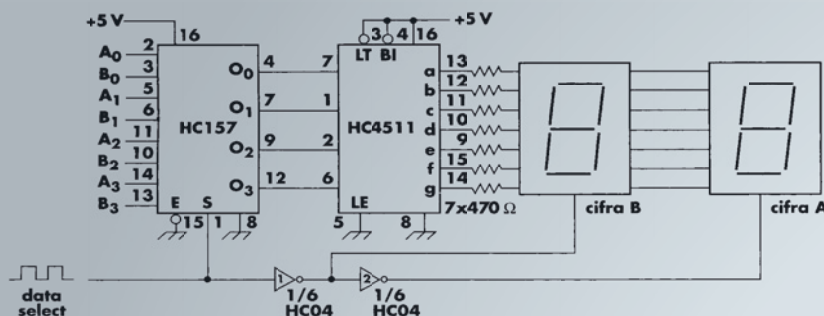
Proseguiamo lo studio di Ultiboard, l'applicazione PCB della National Instruments per la progettazione di circuiti stampati. In questa puntata vediamo la netlist e i metodi di sbroglio manuale utili per posizionare precisamente le tracce durante la realizzazione del prototipo. Oltre a ciò, prenderemo in considerazione l'integrazione tra Multisim e Ultiboard e ricorreremo al myDAQ per valutare il corretto funzionamento del circuito sotto simulazione.

Il circuito che proponiamo come esempio di progetto PCB è un dispositivo di visualizzazione a due cifre con moltiplicazione dei dati (tratto dal volume *L'Elettronica Digitale* di E. Ambrosini, citato nella prima puntata in bibliografia), riportato nella **Fig. 1**. L'integrato 74157 contiene quattro multiplexer a due ingressi, tramite i quali si inviano alternativamente al decoder per display 4511 le parole binarie in ingresso $A_3 A_2 A_1 A_0$ e $B_3 B_2 B_1 B_0$; se il segnale di selezione (data select) è L (Low: 0 V) viene inviata la parola $A_3 A_2 A_1 A_0$, mentre se

è H (High: 5 V) viene inviata la parola $B_3 B_2 B_1 B_0$. Lo stesso segnale di selezione serve per attivare, tramite i due NOT dell'integrato 7404, il display della cifra decimale corrispondente alla parola binaria scelta, decine (cifra B) o unità (cifra A). I due NOT si comportano come un semplice decoder binario a due uscite attive basse: con il data select a L, il NOT 1 ha l'uscita ad H e il display della cifra B è spento, mentre è acceso quello della cifra A; viceversa con il data select a H il NOT 1 ha l'uscita L che accende la cifra B, mentre il NOT 2 ha l'uscita a H e il display della cifra A è spento. La frequenza del segnale di selezione deve essere abbastanza elevata (almeno 50 Hz) per evitare tremolii nell'immagine visualizzata. Per fissare il numero a due cifre da rendere visibile basta porre gli ingressi $A_3 A_2 A_1 A_0$ e $B_3 B_2 B_1 B_0$ a massa 0 V o alla tensione di alimentazione +5 V in relazione al codice BCD (Binary Code Decimal) scelto.

 Per realizzare un nuovo circuito stampato in Ultiboard è necessario aprire un nuovo progetto,

Fig. 1 - Circuito di visualizzazione a due cifre con moltiplicazione dei dati.

**Elenco Componenti:**

Resistori 470 Ω	7
Display 7 segmenti catodo comune	2
4511	1
74157	1
7404	1

il quale raggruppa in un unico file disegni e modelli collegati logicamente tra di loro, per un facile accesso di tutte le risorse. Scegliendo *File>New Project* appare la finestra di dialogo *New Project* (Fig. 2a). Nella casella *Project name* inserite il nome del progetto. Come *Design type* selezionate **PCB Design**; l'altra opzione possibile è *Mechanical CAD*, che consente di utilizzare Ultiboard come CAD meccanico. Nella casella *Location* indicate la cartella di lavoro dove verrà salvato il progetto; cliccando su OK, la finestra di dialogo scompare e viene aperto uno spazio vuoto di progettazione con lo stesso nome del file di progetto. Il file appena creato è indicato nella scheda *Project* della *Design Toolbox*, insieme con eventuali altri disegni e modelli che verranno aggiunti durante la progettazione. Per dare un nuovo nome a un disegno o a un modello, fate clic su di esso con il pulsante destro del mouse e, nel menu contestuale, scegliete *Rename*; nel caso vogliate eliminarlo dal progetto, scegliete *Remove design*.

Per definire forma e misure del circuito stampato si utilizza la procedura guidata *Tools>Board Wizard*. Nella finestra che si apre è possibile cambiare la tecnologia a strati (layers) del circuito stampato, spuntando *Change the layer technology*. Per scopi didattici è preferibile usare una *Technology* di tipo *Double-sided*, ma è possibile lasciare spuntata la voce predefinita *Multi-layers constructed with double-sided boards (layer pairs)*, selezionando direttamente *Next* e nella schermata successiva scegliendo le impostazioni di laminazione con il numero di coppie di strati su cui si svilupperà il PCB. In tal caso, alla voce *Lamination settings* sceglieremo comunque *Layer pairs* pari a 1, per cui sarà possibile usare le comuni basette a singola (single sided) o a doppia faccia (double sided). Si passa quindi a definire la forma e le dimensioni della basetta (Fig. 2b). Nel campo *Units* è possibile impostare l'unità di misura per la progettazione, che di base è il mil, ma che può essere convenientemente cambiata in mm.



Fig. 2a - Nuovo progetto.

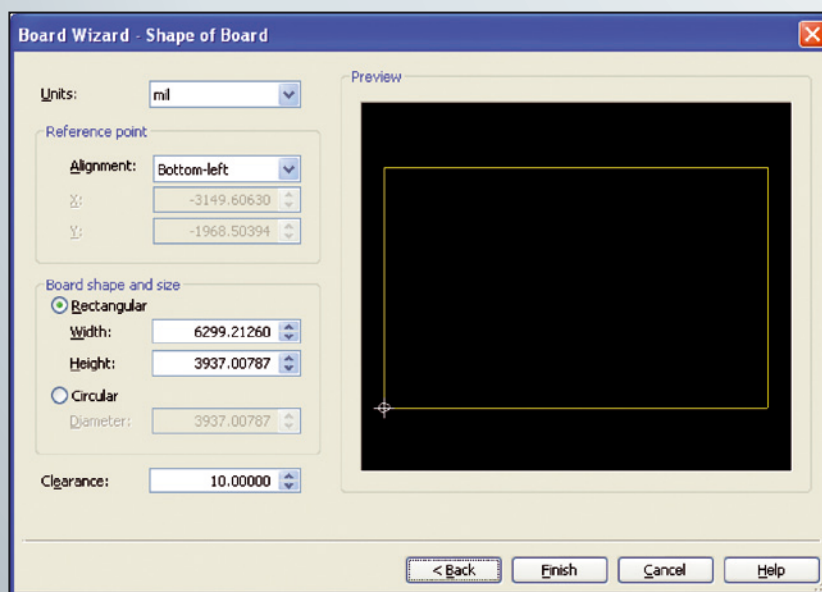


Fig. 2b
Impostazioni di forma e
dimensioni della basetta.

Per il *Reference point* (punto di riferimento) è comodo selezionare la posizione in basso a sinistra della bassetta, inserendo alla voce *Alignment* l'opzione *Bottom-left*, anche se questa scelta può essere modificata successivamente. Seguono le impostazioni *Board Shape and Size* sulla forma e le dimensioni della bassetta, che momentaneamente lasciamo nel formato rettangolare 160x100 mm (Eurocard) predefinito, modificandolo successivamente in caso il PCB dovesse richiedere una superficie maggiore o minore di quella preventivata.

In ultimo, è possibile impostare anche la **Clearance** (spazio libero) della bassetta, ossia la distanza dal bordo della scheda che deve essere mantenuta libera da altri elementi circuitali. Cliccando sul pulsante **Finish** si conclude e si visualizza la bozza della bassetta pronta per la progettazione.

La prima volta che si salva, è possibile salvare un singolo disegno su cui si sta lavorando o un intero progetto (composto da più disegni e documenti) usando lo stesso comando *File>Save As*. Successivamente si userà semplicemente *File>Save* per salvare un lavoro (anche all'interno di un progetto), oppure *File>Save all* per salvare tutti i documenti aperti in un certo momento.

Il menu *File>Open* permette l'apertura di documenti salvati in vari formati, tra i quali: Ultiboard file (*.ewprj), Orcad file (*.max, *.llb), DXF file (*.dxf), Gerber file (*.grb), net-

list file (*.ewnet). Inoltre, mediante l'opzione *File>Open Samples* è possibile aprire dei progetti di esempio per prendere dimestichezza con Ultiboard.

UTILIZZARE I COMPONENTI DEL DATABASE

È arrivato il momento di cominciare a disegnare il circuito stampato di **Fig. 1**, procedendo con l'immissione dei componenti dall'archivio di Ultiboard, scegliendo *Place>From Database* che apre la finestra *Get a Part from the Database*. La colonna *Database* permette di raggiungere tutte le librerie di componenti. Espandendo il percorso *Ultiboard Master>Through Hole Technology Parts* è possibile trovare all'interno tutti i footprint dei dispositivi che compongono il circuito elettronico proposto. Ad esempio la piedinatura per l'integrato 7404 con package Dual-In-Line da 14 pin si trova nel Database al percorso *Ultiboard Master>Through Hole Technology Parts>IC>DIP*, scegliendo *DIP14300 tra le Available parts* (**Fig. 3a**). Il footprint del dispositivo scelto appare nel *Preview*, all'interno del quale, mediante il pulsante *Show Dimension* è possibile visualizzare le dimensioni delle varie parti che lo compongono, nell'unità di misura scelta tra quelle presenti nell'elenco a discesa *Units*, oltre a poter modificare la visualizzazione, agendo sui pulsanti di zoom. Dando l'OK, appare la finestra di dialogo *Enter Reference Designation for Part*. Qui, nel cam-

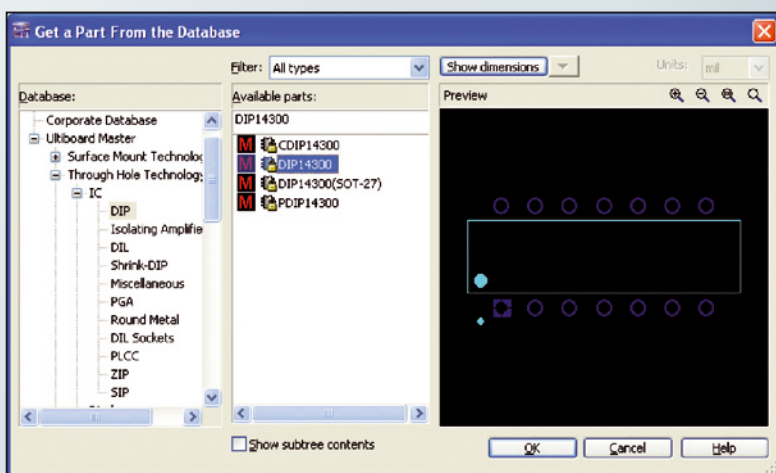


Fig. 3a - La finestra "Get a Part from the Database".

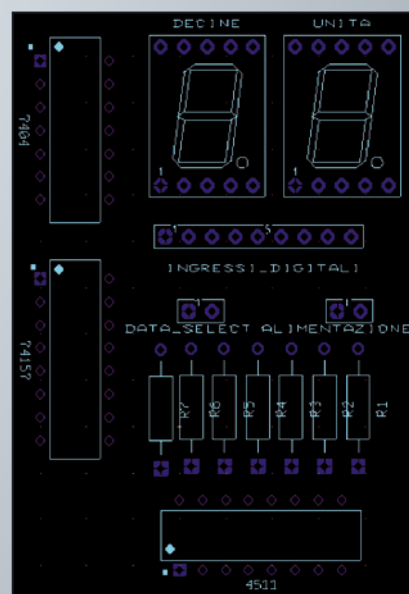


Fig. 3b - Disposizione finale dei componenti sul piano di lavoro.

Tabella 1

Dispositivo	Utilizzo	Available parts	Database	REFDES	Numero
HDR1X2	Alimentazione +5 V e massa; Data select	Connectors > SIL's & Headers	HDR1X2	ALIMENTAZIONE, DATA_SELECT	2
HDR1X9	Ingressi digitali A ₃ A ₂ A ₁ A ₀ e B ₃ B ₂ B ₁ B ₀ e massa digitale	Connectors > SIL's & Headers	HDR1X9	INGRESSI_DIGITALI	1
470Ω	Resistenze	Resistor > Various	RES1300-700X250	R1, R2, R3, R4, R5, R6, R7, R8	7
SEVEN_SEG_DECIMAL_COM_K	Display 7 segmenti catodo comune	LED Displays > Seven Segment	7SEG8DIP14B	UNITA, DECINE	2
4511BP_5V	latch/decoder/driver BCD-7 segmenti	IC > DIP	DIP16300	4511	1
74LS157N	Multiplexer/Selettore da 2 linee a 1 linea a quattro dati	IC > DIP	DIP16300	74157	1
74LS04N	6 porte logiche NOT	IC > DIP	DIP14300	7404	1

po **REFDES** (Reference Designator) va inserito il codice di designazione scelto per il dispositivo, mentre in **Value** va inserita un'indicazione sul valore del dispositivo (per esempio, 20 k per un resistore). Immettete ad esempio REFDES=7404 (oppure lasciate il REFDEF U1 proposto dal software), Value=74LS04N e date l'OK per posizionare il dispositivo sul piano di progetto.

Si procede inserendo tutti i componenti del progetto, tra cui i connettori di alimentazione del circuito con due pin: +5 V e massa (convenzionalmente a potenziale 0 V), che scegliamo di realizzare usando un connettore *Ultiboard Master>Through Hole Technology Parts>Connectors>SIL's & Headers>HDR1x2* e a seguire gli altri componenti, riportati nella **Tabella 1**. Una volta inseriti tutti i dispositivi sul piano, posizionatevi col mouse come volete (**Fig. 3b**). Si ricorda che per effettuare correttamente una selezione è comunque necessario sapere usare i filtri di selezione che consentono di distinguere i vari elementi di disegno, permettendo un controllo dettagliato delle scelte. La selezione dei filtri è raggiungibile da *Edit>Selection Filter* o sulla barra degli strumenti attraverso il menu *Select*. È possibile scegliere una o più tipologie di elementi, abilitandoli con il rispettivo pulsante (come descritto nella puntata precedente).

USARE LA NETLIST

Dopo aver scelto i componenti dal database è necessario inserirli nella netlist, utilizzando l'editor raggiungibile seguendo il percorso *Tool>Netlist Editor*, mediante il quale è possibile aggiungere o rimuovere una rete (**Net**) dalla progettazione, aggiungere o eliminare collegamenti tra le piazzole (**Pins**) o reti esistenti, regolare la larghezza delle tracce (**Width**), raggruppare le reti (**Groups**) e settare

le caratteristiche delle vie di comunicazione (**Vias**) tra i vari strati (**layers**).

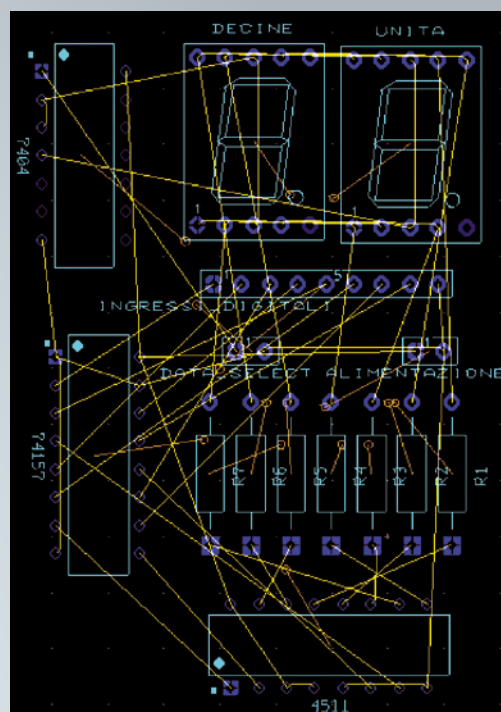
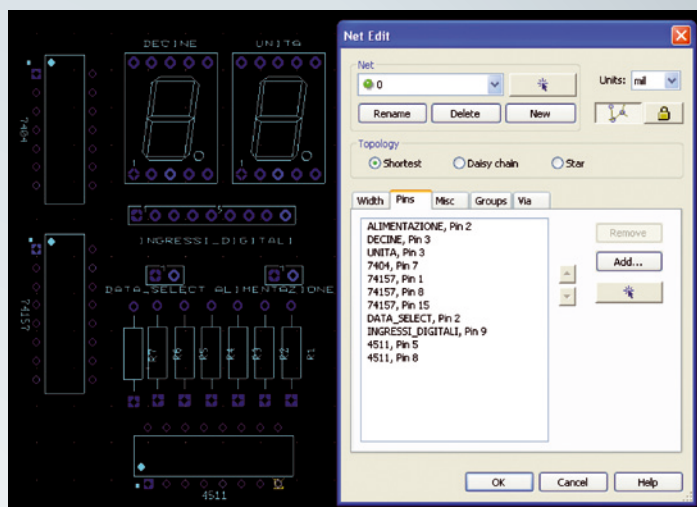
Per aggiungere una nuova rete, fate clic sulla scheda **Pins** della finestra **Net Edit** (**Fig. 4a**) e quindi sul pulsante **New** all'interno del riquadro **Net**, che apre un'ulteriore finestra di dialogo **Add Net**, all'interno della quale cominceremo ad inserire il nodo di massa, chiamandolo 0, quindi diamo OK. Il nuovo nome verrà visualizzato nell'elenco di rete.

 La luce verde luminosa accanto al nodo indica che non ci sono piazzole collegate alla rete 0; per aggiungerle occorre puntarle con il mouse dopo aver abilitato il *pulsante di puntamento* che si trova nella sezione **Pins**, oppure usare il pannello testuale *Add Pins to the Net* che riporta tutti i pin del circuito, aprendolo mediante il pulsante *Add*. Tenendo presente il circuito di **Fig. 1**, selezioniamo tutte le piazzole che vanno collegate a massa, le quali man mano assumeranno un blu più luminoso e verranno via via elencate nella sezione **Pins** (ad esempio "Alimentazione, Pin 2" indica che si è collegata la piazzola 2 del componente nominato Alimentazione). Nel circuito che si sta realizzando, alla rete 0 verranno collegate le seguenti piazzole: ALIMENTAZIONE, Pin 2; UNITA, Pin 3; DECINE, Pin 3; DATA_SELECT, Pin 2; INGRESSI_DIGITALI, Pin 9; 7404, Pin 7; 74157, Pin1; 74157, Pin 8; 74157, Pin15; 4511, Pin 5; 4511, Pin 8.

Aggiungiamo adesso la rete Vcc, usando sempre il pulsante *New* (all'interno del riquadro **Net**) e mediante essa colleghiamo le piazzole ALIMENTAZIONE, pin 1; 7404, pin 14; 7404, pin 16; 74157, pin16; 4511, pin 3; 4511, pin 4; 4511, pin 16. Proseguiamo così aggiungendo le altre reti, effettuando i collegamenti tra le piazzole dei vari dispositivi, fino a realizzare il circuito di **Fig. 1**. Confermando con OK, alla chiusura della finestra **Net Edit**, sul piano di

Fig. 4a - Circuito con linee di collegamento tra i pin dei dispositivi.

Fig. 4a - Netlist Editore.



lavoro appaiono linee di raccordo che indicano i collegamenti tra le varie piazzole inseriti in netlist. Il circuito completo e pronto per lo sbroglio è riportato in **Fig. 4b**.

Aperto poi la finestra *Net Edit*, ci accorgiamo che la luce verde accanto alle reti indicate nel menu a discesa rimane luminosa solo per quelle che sono rimaste non collegate, mentre diviene verde opaco per le reti che sono state utilizzate per collegare almeno due piazzole. Allo stesso modo, volendo inserire un nuovo componente dal database, è possibile collegare le relative piazzole alle reti già presenti nella Netlist. Operando nella finestra *Net Edit* si sceglie di volta in volta (dall'elenco a discesa all'interno del riquadro *Net*) la rete cui collegare una o più piazzole del nuovo componente e poi si associano le piazzole usando il sistema di selezione della sezione **Pins**. È buona regola usare la Netlist per tenere aggiornati i collegamenti tra i vari componenti del PCB; ciò tornerà utile al momento del tracciamento delle piste.

È importante considerare le altre sezioni presenti nella finestra *Net Edit*: in *Width* è possibile settare per ogni rete (net) l'ampiezza delle relative piste da tracciare su PCB, nei campi *Minimum trace width*, *Default trace width* e *Maximum trace width*, esprimendole nell'unità di misura prescelta. Può essere comodo lasciare i valori predefiniti dal software (riportati in **Fig. 5a**) oppure scegliere *Minimum trace width*=10

mil, *Default trace width*=20 mil e *Maximum trace width* pari al valore predefinito. Tali scelte saranno poi utilizzate dall'autoroute, quindi è bene non eccedere con l'ampiezza delle piste, dato che con valori alti la procedura automatica presenta più difficoltà nello sbroglio.

Da considerare anche lo spazio libero o **Clearance**, ossia la distanza minima che va tenuta vuota attorno alle piazzole, alle tracce e a qualsiasi altro elemento in rame segnato su PCB, come pure dal bordo della basetta stessa. Il tentativo di eseguire una traccia o di mettere un componente all'interno della distanza minima consentita è segnalato con un errore, cerchiato di rosso ed evidenziato nella relativa sezione *DRC* dello *Spreadsheet View*. Ultiboard setta una *Clearance* predefinita di 10 mil.

Analizziamo la sezione *Misc* della *Net Edit* (**Fig. 5b**): nell'area *Routing Layers* è possibile indicare, per ciascuna rete (net), uno o più strati dove si potrà sviluppare. Ciò in funzione del successivo inserimento delle piste sia in modalità automatica (sfruttando gli algoritmi di routing di Ultiboard) che in modalità manuale. Se si vuole un master su **singola faccia**, va lasciata per ogni rete la spunta solo su *Copper Bottom*, forzando quindi lo sbroglio su tale layer: questo costringerà l'algoritmo di routing a sviluppare le reti solo sul lato rame (copper bottom), opposto al lato componenti (copper top). Allo stesso modo, se si procedesse allo sbroglio in modalità manuale, non

Fig. 5a - Net Edit: Width.

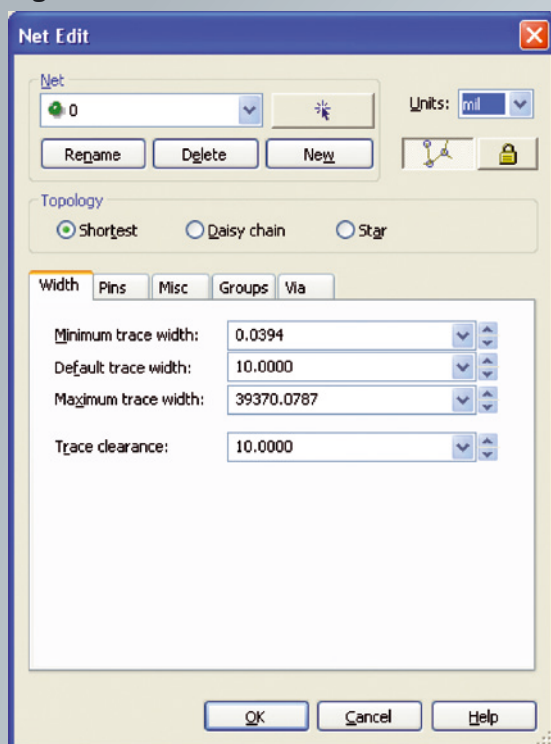
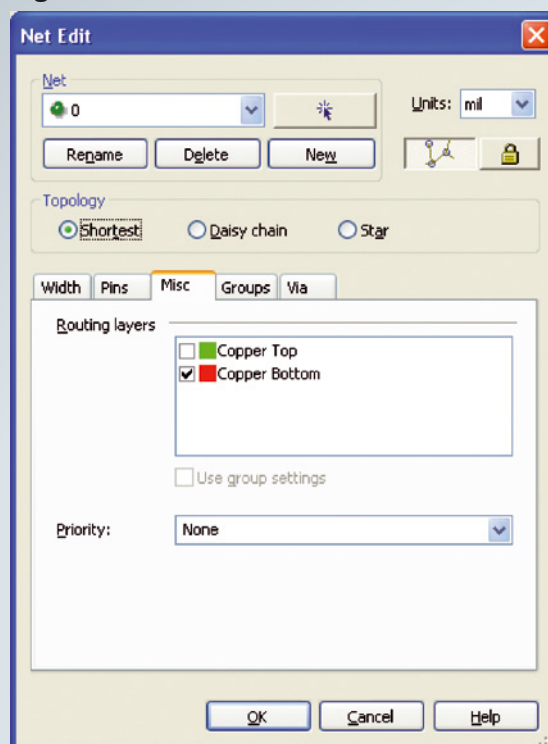


Fig. 5b - Net Edit: Misc.



sarebbe possibile realizzare una certa pista tra tali piazzole, facendola scorrere su un layer diverso dal Copper Bottom, perché verrebbe visualizzato un errore nella sezione *DRC della Spreadsheet View*. In ogni caso, per circuiti singola faccia è indifferente eseguire lo sbroglio con Ultiboard lato rame o lato componenti, pur di sistemare bene le cose all'atto di inviare il master ad una fresa, mentre se si sceglie di riprodurre il master su basette presensibilizzate, prima di stampare su carta lucida va considerata l'opzione di stampa *Print Reflection* (per quanto riguarda la stampa si rimanda alla puntata precedente di questo corso). Nel caso, invece, si voglia realizzare un circuito stampato doppia faccia, è conveniente settare per ciascuna rete ambedue i layer **Copper Top** e **Copper Bottom** (lasciando all'algoritmo di routing di Ultiboard un grado di libertà sulla scelta di posizionamento della pista), oppure è possibile forzare per ogni rete una tra le due facce, secondo le proprie necessità progettuali. Come visto nella puntata precedente di questo corso, è possibile abilitare per ogni rete, all'interno della sezione *Misc* della netlist, tutti i layer disponibili, salvo poi, durante lo sbroglio, indirizzare in altri modi le piste sui layer voluti. In tal caso, si selezionano tutte le piazzole (pads) che si vogliono modificare, scegliendo le relative *Properties* (raggiungibili

mediante il menu contestuale delle piazzole o cliccandoci sopra due volte con il pulsante sinistro del mouse) cui segue l'apertura della finestra *Through Hole Pin Properties*. Qui, nella sezione *Layer settings* è possibile rimuovere la spunta nei checkbox degli strati sui quali non si vuole fare scorrere piste di collegamento tra le piazzole selezionate; questa scelta non modifica la netlist ma forza l'autorouting e lo sbroglio manuale. In alternativa è possibile effettuare la stessa scelta nella scheda *Copper layers* dello *Spreadsheet View*.

Prima dello sbroglio è bene controllare e modificare le caratteristiche delle piazzole di tutti i componenti che potrebbero avere dimensioni predefinite diverse da quelle reali. Si procede dunque impostando ad esempio le seguenti caratteristiche per tutte le piazzole (per selezionarle tutte abilitare il pulsante *Enable Selecting Pads* e usare la combinazione dei tasti CTRL+A): Pad Diameter di 75 mil (circa 1,9 mm) e Drill Diameter di 40 mil (circa 1 mm); nessuna indicazione nel campo *Hole*, lasciando le forme delle piazzole come da footprint originali.

INTEGRARE ULTIBOARD CON MULTISIM

Sebbene il circuito preso in considerazione sia abbastanza semplice, la sua implementazione in NI Ultiboard, sia per la scelta del footprint

adatto a ciascun componente, sia per la successiva compilazione della netlist, può essere un'operazione lunga e poco agevole. Pertanto è comodo sfruttare l'integrazione di Ultiboard con Multisim, consentendo di ottimizzare il processo di progettazione del circuito stampato dallo schema elettrico alla produzione. Operando come descritto nella puntata precedente, dopo aver realizzato lo schema di **Fig. 6** (file *Display con dati multiplati.ms11*) e aver dato il comando *Transfer>Transfer to Ultiboard>Transfer to Ultiboard 11*, per trasferirlo in Ultiboard, si giunge comunque (e probabilmente con meno fatica) allo stesso circuito riportato in **Fig. 4b**, pronto per lo sbroglio. In questo caso Multisim è usato solo per creare lo schema e non sarà invece possibile usarlo per la simulazione, in quanto non potendo operare in tempo reale risulta inutile. È arrivato il momento di tracciare le piste di connessione tra le piazzole, scegliendo di procedere in modalità manuale oppure automatica. In genere è conveniente procedere avviando l'autoroute fino a sbroglio ultimato o quantomeno fino a portare il lavoro a un buon punto, risolvendo il massimo numero di connessioni possibili, poi ultimare lo sbroglio in modalità manuale. Vediamo dunque come procedere in questa seconda modalità.

LO SBROGLIO IN MODALITÀ MANUALE

In questa modalità, gioca un ruolo importante la presenza di una netlist aggiornata, che evidenzia a colpo d'occhio le piazzole da connettere insieme mediante piste, grazie alle linee

di collegamento visibili sul piano di lavoro (vedi **Fig. 4b**).

Il metodo *Manual Trace* consente il massimo controllo nel collocamento delle tracce tra piazzole e vias; si abilita scegliendo il comando *Place>Line*. Dopo aver selezionato una piazzola, muovendosi con il mouse si traccia il percorso fino a raggiungere la successiva piazzola da collegare o fino a una pista già tracciata. Torna utile avere una netlist aggiornata con le reti esistenti, perché Ultiboard, durante il tracciamento, indica con una X le piazzole che andranno collegate a quella di partenza. Agendo sulla barra *Draw Settings* (come si vede in **Fig. 7a**) è possibile scegliere il valore desiderato per lo spessore (width) delle piste, in luogo di quello automatico. Nel caso del PCB proposto si è scelto 40 mil (pari a 1 mm circa). Allo stesso modo, durante la tracciatura di una pista, è possibile variare il suo spessore, cliccando con il pulsante destro del mouse in un punto qualsiasi e scegliendo (nel relativo menu contestuale) se aumentarlo usando il comando *Widen* (CTRL+Maiusc+W) oppure se ridurlo, con *Narrow* (CTRL+Maiusc+N). La stessa barra permette di selezionare lo strato (layer) su cui tracciare una pista (in **Fig. 7a** si sta usando il copper top). Questo metodo supporta un tracciamento continuo delle piste: la pista successiva comincerà dall'ultima piazzola collegata. Per interrompere il tracciamento, cliccare con il tasto destro del mouse, scegliendo **Cancel**, oppure premendo ESC da tastiera: l'ultimo tratto di pista verrà rimosso dal PCB.

Follow Me Router è il metodo che permette

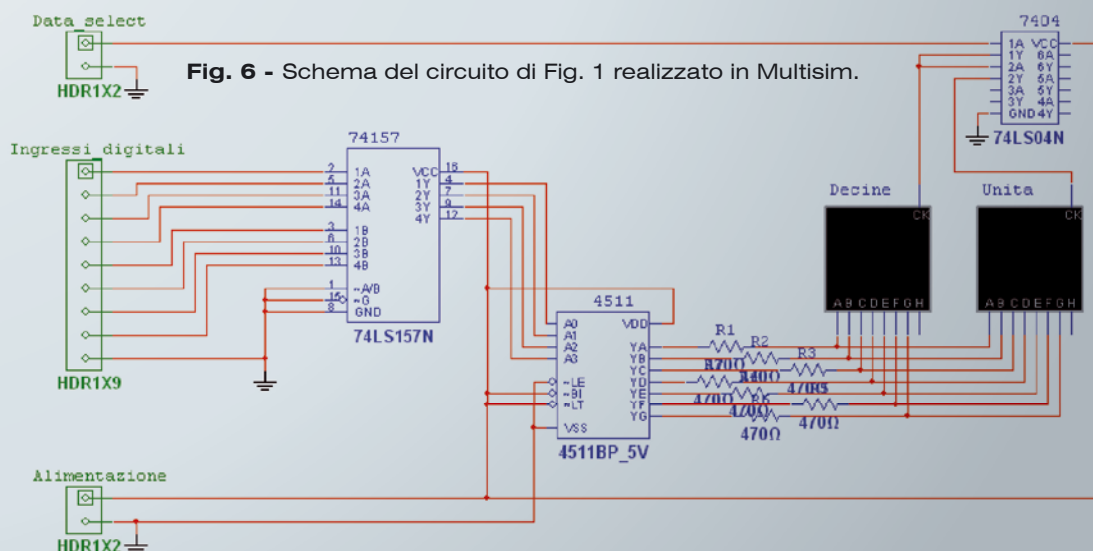
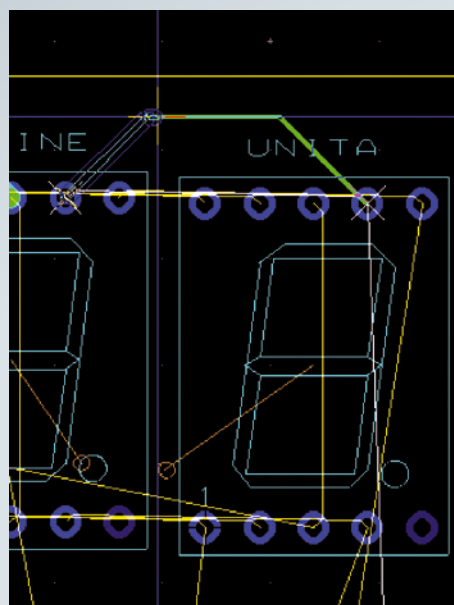


Fig. 6 - Schema del circuito di Fig. 1 realizzato in Multisim.

Fig. 7b - Tracciamento piste in modalità: Follow Me Router.

Fig. 7a - Tracciamento piste in modalità: Manual Trace con evidenziata la barra Draw Settings.



di tracciare le piste selezionando una prima piazzola o una via e poi le successive piazzole da collegare a questa, lasciando ad Ultiboard il compito di seguire il nostro movimento del mouse e di trovare il percorso migliore tra i vari ostacoli (**Fig. 7b**). Si abilita selezionando *Place>Follow-me*. Questo metodo supporta un tracciamento continuo delle piste: la successiva pista comincia dall'ultima piazzola collegata. Anche in questa modalità, durante la tracciatura di una pista, è possibile variare la sua ampiezza, agendo sulla barra *Draw Settings* per impostare il valore desiderato, oppure cliccando con il destro in un punto qualsiasi e scegliendo se aumentarla con *Widen* oppure se ridurla con *Narrow*. Dopo aver tracciato un gruppo di piste, per concludere e passare ad un'altra parte del circuito, è necessario cliccare con il pulsante destro del mouse, scegliendo **Cancel** nel menu contestuale, oppure premendo ESC da tastiera: rimarranno sul piano di lavoro solo le piste fin qui disegnate, l'ultimo tratto di pista verrà rimosso dal PCB.

 Per uscire da ciascuna delle modalità di tracciamento descritte (Manual trace e Follow Me Router) e tornare alla modalità di puntamento del mouse è necessario abilitare l'icona *Place>Select*.

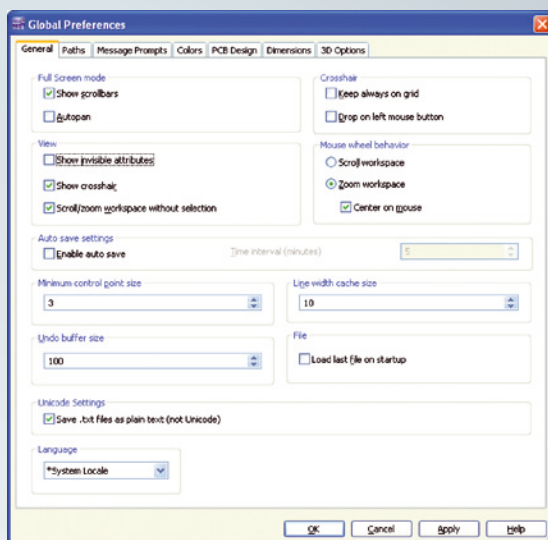
Per cancellare un'intera pista o parti di essa, è necessario effettuare una selezione con il mouse, dopo avere abilitato *Edit>Selection Filter>Enable Selecting Traces*, quindi procedere alla cancellazione facendo uso dell'opzione *Delete* contenuta nel relativo menu contestuale, oppure mediante il tasto CANC.

Per spostare da un layer ad un altro una pista appena tracciata è necessario tornare alla modalità di puntamento del mouse, selezionando la pista in oggetto, dopo avere abilitato *Enable Selecting Traces* (oppure usando il menu *Edit>Selection Filter>Enable Selecting Traces*), fare doppio clic su di essa e nella successiva finestra di dialogo **Trace Properties**, nella sezione **Position**, scegliere il layer voluto. Ultiboard inserirà in automatico la via per passare da un layer all'altro.

 Il comando *View>Clearances* visualizza lo spazio che va tenuto libero (clearance) intorno ad ogni elemento circuitale, il cui limite è rappresentato da sottili linee blu che circondano tracce, piazzole e vias.

 In alcuni casi può essere necessario inserire un ponticello in luogo di una pista impossibile da tracciare sul PCB a meno di non dovere usare un ulteriore layer. Per fare ciò, assicurarsi di avere selezionato uno strato di rame (ad esempio di stare lavorando sul Copper Bottom). Scegliere *Place>Jumper* e cliccare sulla prima piazzola da collegare con il ponticello, quindi spostare il puntatore sul disegno per trasportare l'asse del ponticello verso la seconda piazzola. Se non si riesce a raggiungerla, fare clic in un punto qualunque (non occupato da elementi circuitali) per confermare il primo tratto, selezionare il punto di arrivo e tirare un altro tratto di ponticello verso la piazzola voluta, modificandolo in lunghezza e direzione, e così via fino alla fine. Per concludere o annullare l'inserimento di un ponticello, cliccate con il tasto destro. È possibile anche inserire

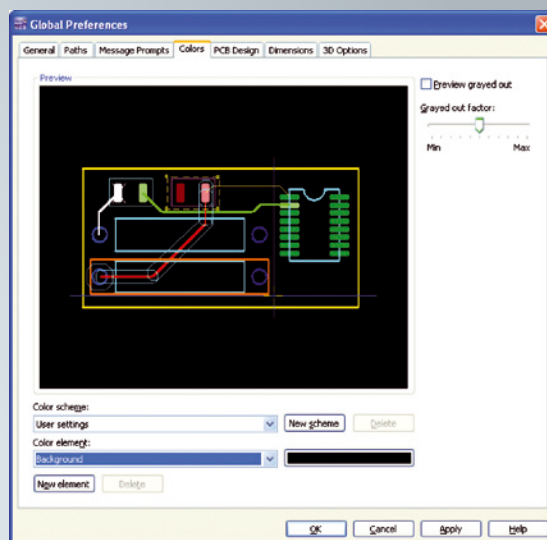
Fig. 8a - Preferences: General Settings.



dei ponticelli in successione, proseguendo con il successivo a partire dalla piazzola finale del precedente. Per modificare le proprietà di un ponticello, dopo avere abilitato *Edit>Selection Filter>Enable Selecting other objects* (o usando la relativa icona di comando), selezionarlo mediante doppio clic oppure mediante il pulsante destro del mouse, scegliendo *Properties* nel menu contestuale. Si aprirà la finestra di dialogo *Jumper Properties* dove sarà possibile intervenire su vari suoi attributi.

 A conclusione dello sbroglio automatico, prima di passare alla produzione del PCB (ma anche prima di salvare, stampare o esportare un disegno), oppure durante lo sbroglio manuale, è importante fare uso del comando *Design>DRC and Netlist Check*, che forza un controllo mirato a scovare errori rispetto alle regole di disegno impostate (DRC=Design Rule Errors) o in merito alla netlist compilata. Gli errori vengono cerchiati in rosso sul foglio di progettazione ed elencati all'interno della sezione DRC dello *Spreadsheet View*. In genere gli errori sono dovuti alla sovrapposizione di oggetti circuitali (piazzole, collegamenti tra layer, piste) all'occupazione dello spazio da lasciare libero (clearance) tra gli oggetti, oppure a collegamenti tra piazzole che non andavano fatti. Durante l'autorouting il controllo degli errori viene eseguito e risolto in automatico, a meno del caso in cui il software non riesca a completare lo sbroglio, segnalando nella scheda *Results* dello *Spreadsheet View* la percentuale di lavoro portato a termine e il numero di reti rimaste non collegate e riportando in *DRC* tutti gli erro-

Fig. 8b - Preferences: Colors.



ri dovuti al non completamento dell'operazione (con doppio clic sulla voce è possibile andare al punto cui l'errore si riferisce).

Un altro importante comando, utile quando si lavora in modalità manuale e in ogni caso da provare almeno una volta prima di passare alla produzione del PCB, è *Design>Connectivity Check*, che esegue un controllo sulle connessioni. Una volta selezionato, si apre la finestra di dialogo *Select a Net*, dove inserire la rete su cui eseguire il controllo di connessione, oppure predefinito un controllo generale su tutte le reti # **All Nets** #. Al solito, i risultati vengono visualizzati nella scheda *Results* dello *Spreadsheet View*, compresi gli errori che indicano i collegamenti dichiarati in netlist ma non realizzati nello sbroglio (visibili con doppio clic sulla relativa voce).

PREFERENZE GENERALI DEL PROGRAMMA

Il menu *Option* permette di impostare le preferenze generali del programma. Scegliendo *Option>Global preferences* si apre la finestra di dialogo *Preferences*. Per accedere a tutte le sezioni di impostazione è comunque necessario che si stia lavorando ad un progetto e che quindi sia aperto un file in Ultiboard. La finestra *Global Preferences* offre le seguenti schede:

- **General (Fig. 8a)** regola le impostazioni generali del software e tra queste potrebbe tornare utile abilitare l'*Auto save* ogni 5 minuti, impostare la preferenza di utilizzo della rotellina del mouse (nella *Mouse wheel behavior*) e settare un valore per l'*Undo buffer size* (che determina il numero massimo

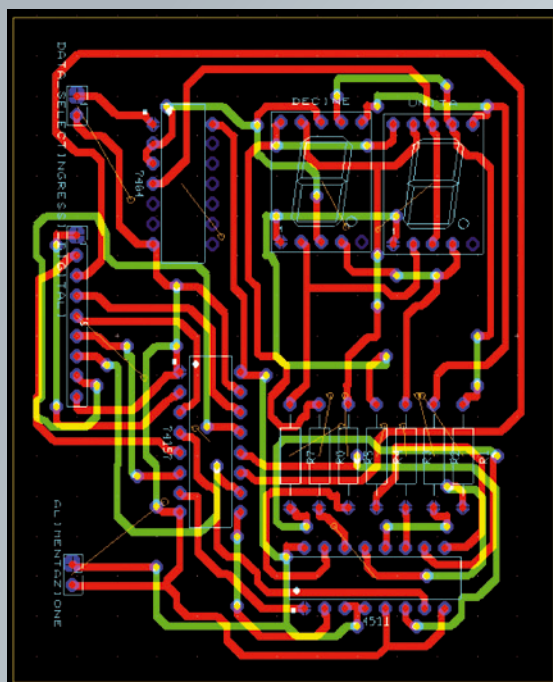
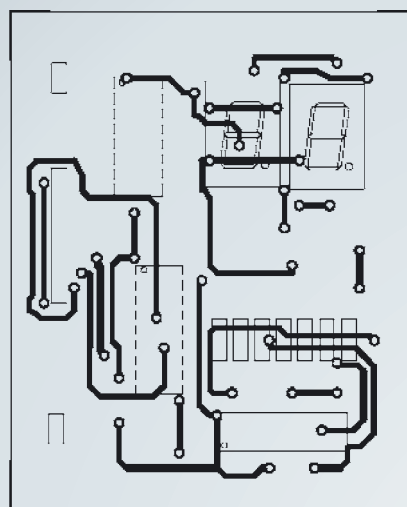


Fig. 9a - Sbroglio circuitale in Ultiboard.

Fig. 9b - Copper top con disposizione dei componenti, ridotto al 50 % dell'originale.



di modifiche fatte che è possibile annullare in caso di errore) compatibilmente con le proprie esigenze e la memoria del proprio PC (100 è una buona scelta).

- *Paths* consente sotto *General > Circuit default path* di impostare il percorso di salvataggio dei documenti sul proprio PC, nella sezione *User settings* di assegnare e configurare il template desiderato per i propri file e in *Database Files* di indicare gli archivi che contengono le impronte (footprints) dei vari componenti elettronici, rispettivamente Master, Corporate e User Database;
- *Colors* (Fig. 8b) permette di scegliere i colori degli elementi presenti sul foglio di lavoro;
- *PCB Design* include numerose opzioni di scelta, tra cui consideriamo:
 - *View>Show pin 1 mark* permette di contrassegnare il piedino 1 di un dispositivo con un marchio unico;
 - *View>Show pin info in pin* consente di visualizzare il numero di pin e l'ampiezza della piazzola quando è ingrandita;
 - *Part drag>Re-route after move* abilita il routing automatico delle piste collegate ad un componente dopo che è stato spostato con il mouse;
 - *Action on DRC error*; in quest'area va impostata l'azione da intraprendere quando Ultiboard incontra un errore nel disegno; è possibile settare *Cancel the current action* per l'annullamento automatico dell'even-
- to che ha causato l'errore, oppure *Ask for confirmation* per chiedere conferma se annullarlo o completarlo, oppure *Overrule* per consentire il completamento dell'evento (opzione consigliata);
- *DRC & Net check*; in quest'area si può scegliere *No Realtime Check* per impedire il controllo degli errori di disegno in tempo reale già al collocamento degli oggetti sul piano di lavoro, oppure *Check on action end* per effettuare un controllo solo al termine di ogni azione (ad esempio l'inserimento o lo spostamento di un componente), oppure *Full Realtime Check* per il pieno controllo in tempo reale (opzione consigliata) degli errori;
- *Follow me router*, settando l'opzione *Continuous place* si dice al software di iniziare una nuova pista nel punto finale di quella precedente, quando si lavora in modalità manuale di tipo *Follow Me Router*; in caso contrario sarà necessario fare clic per iniziare una nuova traccia;
- *Dimensions* permette di configurare le caratteristiche delle linee di quota: frecce, testo, distanza dagli oggetti quotati;
- *3D Options* permette di scegliere il *Background color* (colore di sfondo) del visualizzatore 3D e le *Internal View Options* nel caso si lavori su più layer.

CONCLUSIONE DEL PROGETTO

Scegliamo di realizzare per il nostro circuito

un master su doppia faccia, abilitando per ogni rete sia *Copper Top* che *Copper Bottom*. Per le piazzole lato copper bottom si è impostato il diametro (*Pad Diameter*) a 75 mil (circa 1,9 mm), e il diametro del foro (*Drill Diameter*) a 40 mil (circa 1 mm). Si sono tracciate piste di spessore di 40 mil (circa 1 mm) e clearance minime di 15 mil, distanziando ulteriormente le tracce tra di loro, ove possibile, a sbroglio automatico ultimato. Si è scelta quindi la modalità di sbroglio automatico, perfezionando successivamente lo sbroglio in maniera manuale, ottenendo il risultato riportato in **Fig. 9a**. Il master è invece riportato nelle **Fig. 10a** e **Fig. 10b** (sono stampati i layer *Copper Bottom* e *Copper Top*, spuntando le opzioni *Print reflection*, *Leave drill holes open* e *Print layers on separate paper*), mentre la stampa di **Fig. 9b** è comoda come riferimento per l'inserimento dei componenti lato copper top (è stata ottenuta stampando i layer *3D-Info Top* e *Copper Top*, spuntando solo l'opzione *Leave drill holes open*). L'opzione *Print header* in stampa inserisce o meno l'intestazione della pagina che indica cosa si stia stampando: è consigliabile spuntarla sempre.

BACKWARD ANNOTATION DI ULTIBOARD

L'integrazione tra Ultiboard e Multisim assicura un facile trasferimento delle informazioni nelle varie fasi di messa a punto del progetto, grazie anche alla *Backward Annotation* (annotazione all'indietro), un processo altamente automatizzato che assicura che le modifiche apportate durante una progettazione Ultiboard siano trasferite nello schematico di origine di Multisim. La backward annotation è una

caratteristica importante dei software CAD. Un componente rinominato o eliminato può provocare incongruenze tra lo schematico e la progettazione del PCB. La backward annotation permette di superare queste incongruenze, tenendo traccia delle modifiche, aiutando a mantenere schema e layout del PCB coerenti l'uno con l'altro. Il processo di backward annotation avviene in Ultiboard selezionando *Transfer>Back annotate to Multisim>Back annotate to Multisim 11* che apre una finestra di dialogo *Save As* per salvare il file nel formato *.ewnet (volendo è possibile sovrascrivere il file creato da Multisim durante il processo di forward annotation, quando si è trasferito lo schematico originario da Multisim verso Ultiboard).

Multisim cerca il file (nel formato *.ms11) da associare a quello appena salvato in Ultiboard con gli aggiornamenti CAD e se non è già aperto, chiede di aprirlo (nel nostro caso si tratta del file *Display con dati multiplati.ms11*). Una volta aperto il relativo file di disegno di Multisim (nel formato *.ms11), appare una finestra di dialogo *Backward Annotation* (**Fig. 11**) che mostra i cambiamenti annotati, confermati una volta dato l'OK e riportati nello schematico di Multisim. All'interno di questa finestra è comunque possibile scegliere quei cambiamenti cui dare corso e quelli da evitare o da modificare ulteriormente. Durante un'annotazione in avanti o indietro (forward o backward annotation) è possibile aggiornare:

- la rimozione di un componente, la ridenominazione di un componente, di una rete, di un footprint e di un sottocircuito;

Fig. 10a
- Circuito stampato copper bottom, lato rame, ridotto al 50%.

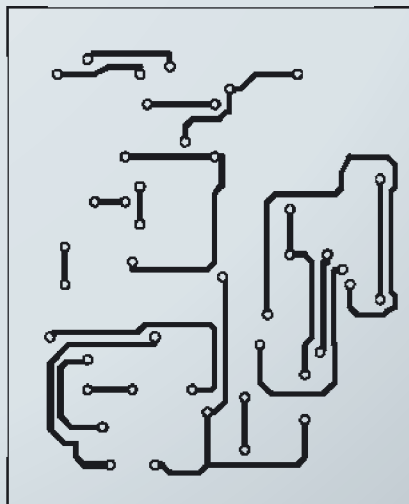


Fig. 10b
- Circuito stampato copper top, lato componenti, ridotto al 50%.

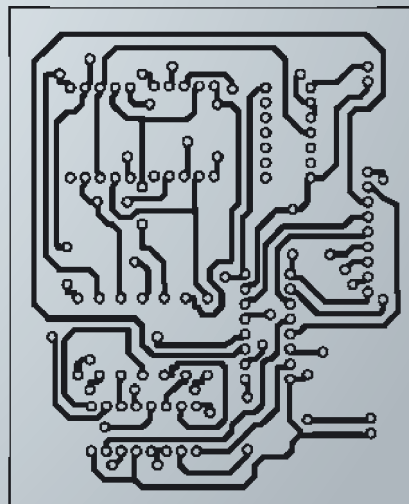
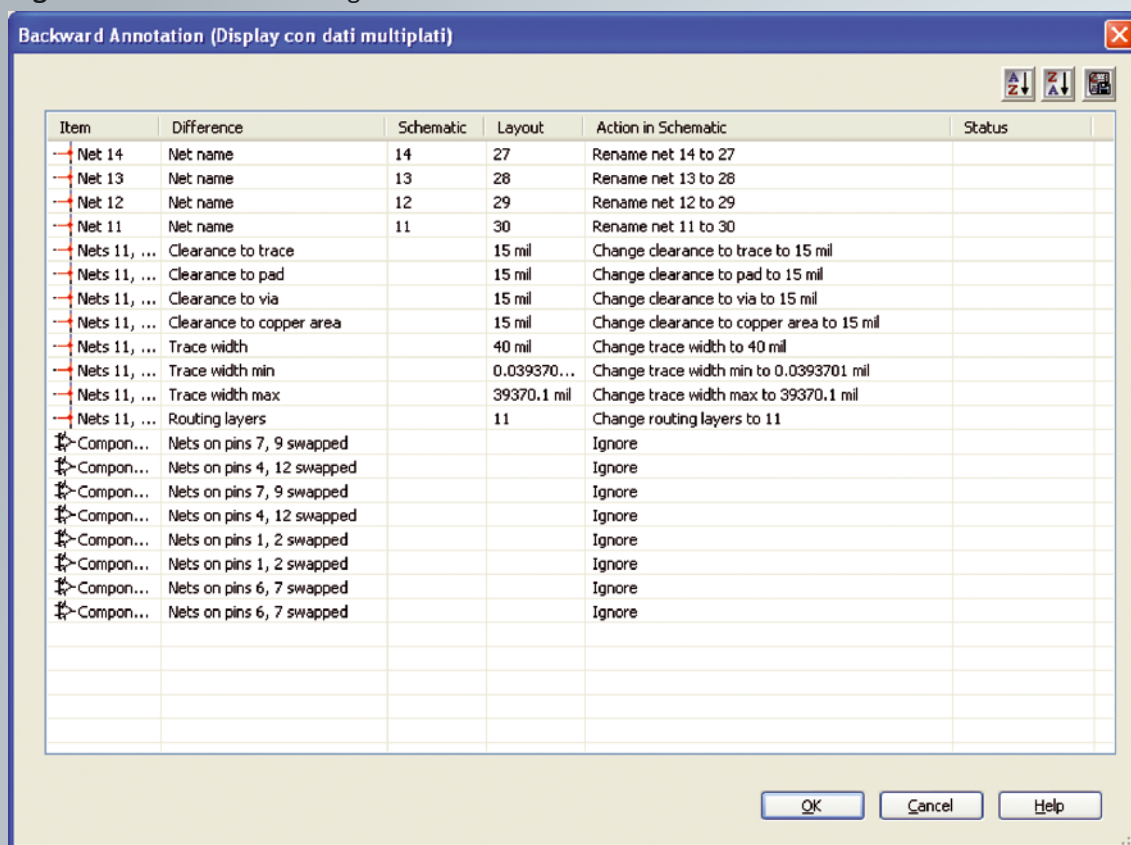


Fig. 10 - Una finestra di dialogo Backward Annotation.



- lo scambio (swapping) di pin e porte: al fine di consentire l'instradamento migliore, Ultiboard nell'acquisizione di uno schematico da Multisim potrebbe effettuare uno swap tra i pin di uno stesso componente (ad esempio, scambiando tra loro i due ingressi di una porta AND), o effettuare uno swap tra le sezioni di un componente (ad esempio, utilizzando una diversa porta logica in un integrato QUAD AND);
- la massima e la minima ampiezza di traccia, la massima e la minima lunghezza di traccia, la clearance di una piste, di una piazzola, di un foro passante tra gli strati e di un'area di rame.

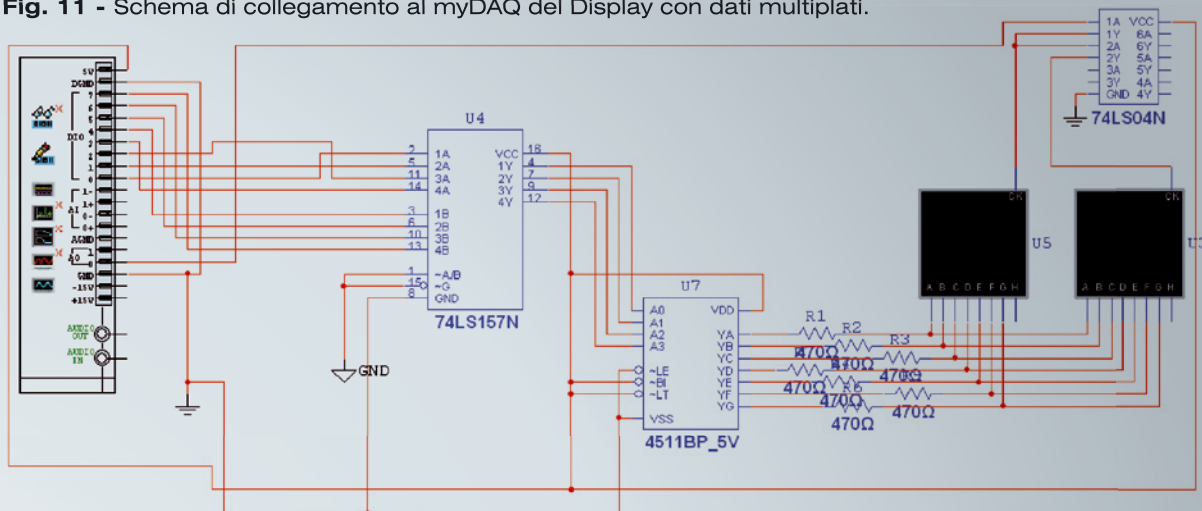
Allo stesso modo, le successive modifiche di schema in Multisim potranno essere annotate in avanti verso Ultiboard, in modo che i progressi di progettazione PCB compiuti fino al momento non vadano persi se viene apportata una piccola variazione nello schema elettrico. In tal caso selezionate *Transfer>Forward Annotate to Ultiboard>Forward Annotate to Ultiboard 11*, salvare il file con il nome desiderato e aprirlo in Ultiboard, usandolo come aggiornamento di un circuito già disegnato.

ANALISI CIRCUITALE DEL PROGETTO MEDIANTE MYDAQ

Il laboratorio elettronico offerto dal myDAQ (spiegato nella quinta puntata di questo corso) è utile per analizzare il funzionamento del Display con dati multiplati, dopo averlo montato su breadboard, ma anche per testarlo dopo averlo realizzato normalmente. Lo schema di collegamento è riportato nella Fig. 12, dove notiamo che:

- l'alimentazione a 5 V è fornita dal myDAQ;
- le parole binarie $A_3 A_2 A_1 A_0$ e $B_3 B_2 B_1 B_0$ in ingresso al 74157 vengono prodotte mediante le otto linee digitali DIO, considerando separatamente il gruppo dei pin $0 \div 3$ e $4 \div 7$, pilotandoli mediante il NI ELVISmx Digital Writer;
- il segnale data select da collegare al 74157 e al 7404 è ottenuto sfruttando il generatore di funzioni NI ELVISmx FGEN impostato su un'onda quadra a frequenza maggiore di 50Hz in uscita al canale AO 0;
- la massa digitale DGND e quella analogica GND vengono collegate insieme.

Dopo avere avviato il generatore **FGEN** per la produzione del data select, usando il **Digital**

Fig. 11 - Schema di collegamento al myDAQ del Display con dati multipliati.

Writer è possibile visualizzare sui display un numero tra $0 \div 99$, ottenuto impostando manualmente due codici binari, ciascuno nell'intervallo $0000 \div 1001$, che agiscono sulle otto linee digitali **DIO** raggruppate in scrittura a gruppi di quattro pin. La configurazione in uscita del Digital Writer rimane presente sui pin **DIO** fino a quando non viene generato un nuovo codice.

I file delle simulazioni presentati in questo corso sono scaricabili gratuitamente dal sito www.elettronica.in.

Il software Multisim fa parte del pacchetto NI Circuit Design Suite Education (cod. 8100-NICDSE) disponibile presso www.futurashop.it. La versione per utenti privati e studenti costa 99,95 Euro.



Un approccio integrato alla progettazione di circuiti

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

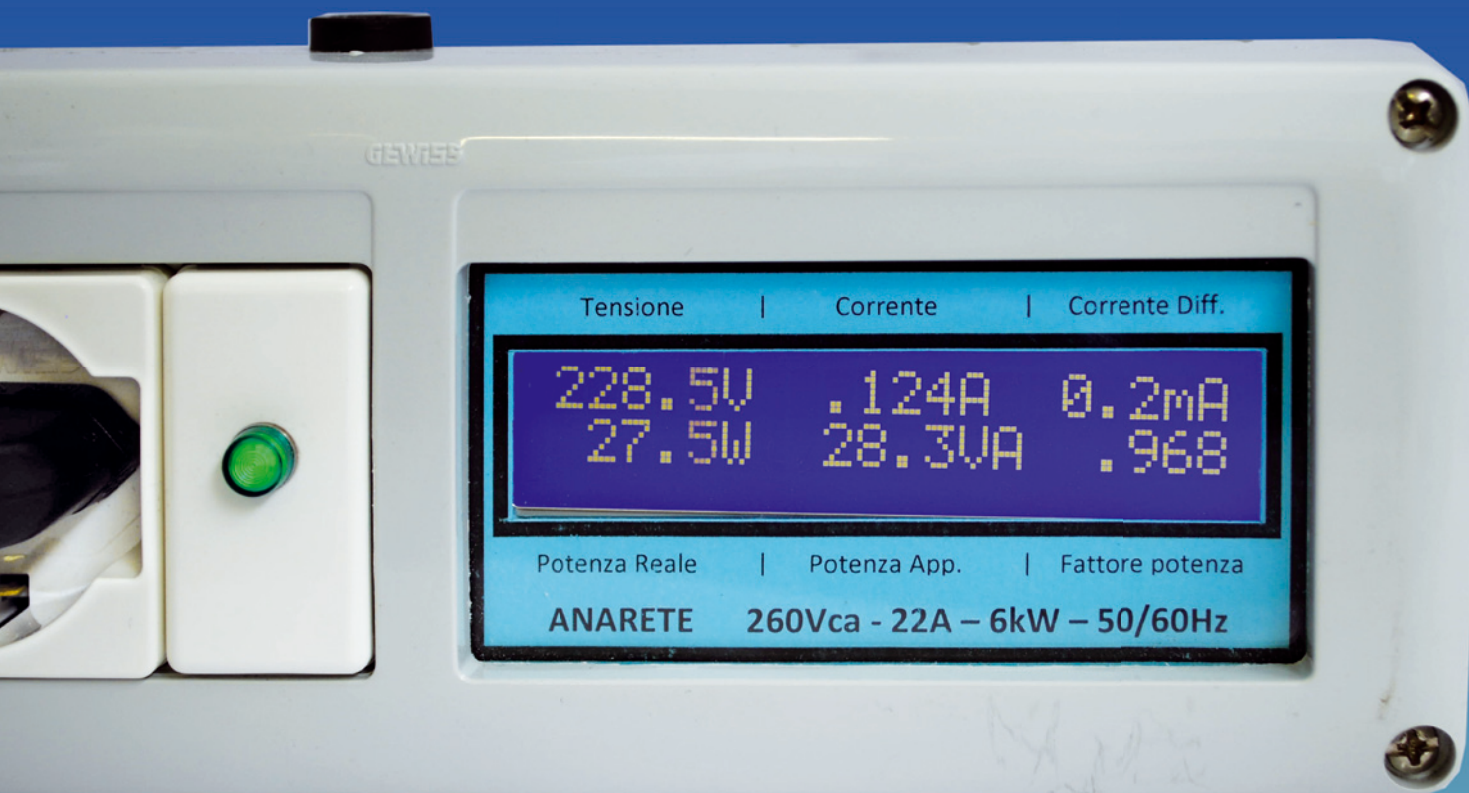
>> Scarica le risorse gratuite su ni.com/academic/i/eeep

02 41.309.1

NATIONAL INSTRUMENTS (ITALY S.p.A.) • Via Anna Kulscioff, 22 • 20132 Milano • Tel.: 02-413091 • Fax: 02-41309215
PIÙ/V.C.F. - 0077218151 • C.C.A.A. Milano 1317483 • Iscritt. Trib. Milano 266039/236023 • S.p.A.
©2010 National Instruments. Tutti i diritti riservati. LabVIEW, Multisim, National Instruments, NI e ni.com sono marchi registrati di National Instruments. Altri prodotti e nomi aziendali citati sono marchi commerciali delle rispettive aziende. 1008

IRS Ingegneria
Ricerca
Sistemi
Distributore unico per l'Italia
Tel.: +39 049 8705156 • info@irsweb.it

**NATIONAL
INSTRUMENTS**



Analizzatore di tensione, corrente, potenza, fattore di potenza, ma soprattutto prezioso strumento per scoprire a quanto ammontano le dispersioni, spesso responsabili del distacco del salvavita. Seconda puntata.

COSTRUIAMO UN ANALIZZATORE DI RETE

..... dell'ing. SALVATORE COSENTINO

Dopo aver descritto il funzionamento del sistema, la filosofia progettuale, il firmware che gira nel microcontrollore ed aver spiegato nel dettaglio come si costruisce l'analizzatore, è giunto il momento di passare alla programmazione ed alla messa in opera. Ricordiamo che il progetto qui proposto riguarda uno strumento di misura capace di rilevare i principali parametri della rete elettrica domestica e quantificare le fughe verso terra, causa

dei repentini quanto dannosi distacchi dell'interruttore differenziale che ci protegge dalla folgorazione.

PREDISPOSIZIONE ALLA PROGRAMMAZIONE E ALLA TARATURA

Per programmare il microcontrollore "in-circuit" e per fare qualche prova e taratura senza pericolo, è indispensabile alimentare il nostro circuito con un alimentatore esterno da laboratorio o con un

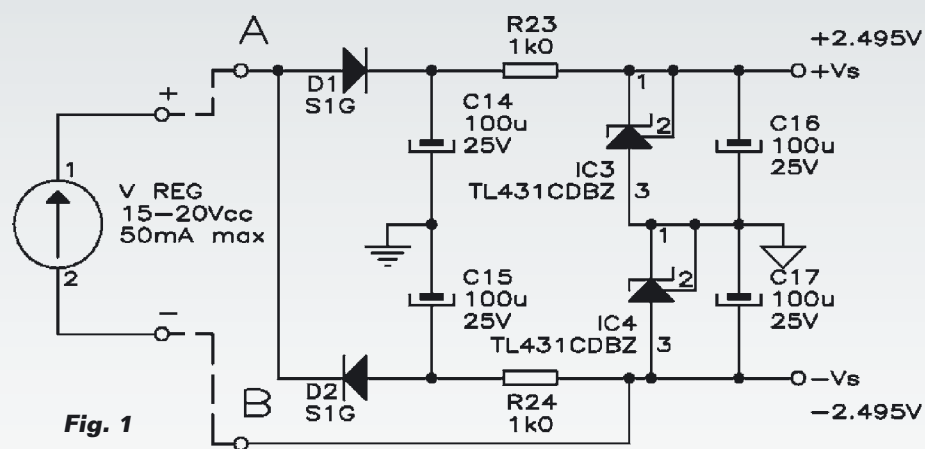


Fig. 1

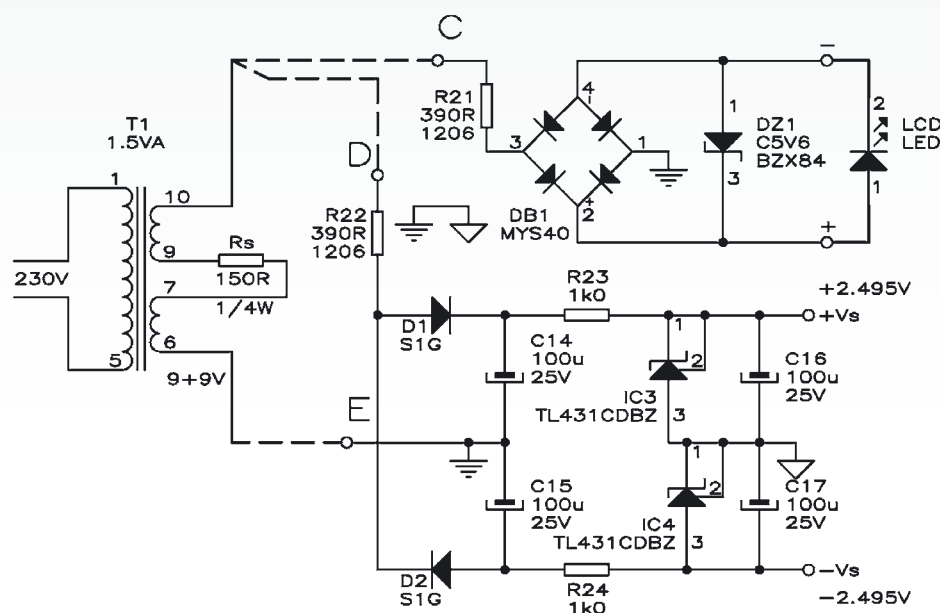


Fig. 2 - Alimentazione provvisoria in corrente alternata tramite trasformatore (15÷18 V). La resistenza da 150 Ω in serie al secondario limita leggermente la corrente nel caso venga usato un trasformatore da 18 volt.

piccolo trasformatore. La prima soluzione è forse più semplice, ma in genere non basta ad alimentare la retroilluminazione del display, il quale resta poco leggibile; la seconda richiede un piccolo trasformatore che abbia un secondario a tensione compresa tra 15 e 18 V e potenza di 1 W o poco più. Per ognuna delle soluzioni occorre collegare due fili ai componenti del circuito cui applicare

la tensione, seguendo lo schema di Fig. 1 se usate l'alimentatore in cc e quello di Fig. 2 se, invece, ricorrete al trasformatore. Una volta applicata la tensione di alimentazione provvisoria (Fig. 3), se il microcontrollore è già programmato ed il circuito collegato al suo LCD, questo si accenderà e sarà possibile provvedere alla taratura. Altrimenti si potrà procedere alla programmazione del microcontrollore tra-

mite l'interfaccia BDC, procedura che descriveremo nella prossima puntata.

TARATURA

Per completare il nostro apparecchio dovete sovrapporre e montare una sull'altro la basetta con il circuito di misura finito ed il display. I due connettori si sovrapporranno perfettamente e tre colonnine filettate di altezza pari a 10 mm provvederanno a irrigidire l'assieme e assicurarne la planarità. Attenzione a che le viti applicate sul lato del display e le stesse colonnine non cortocircuitino nelle vicinanze.

A questo punto il circuito dovrebbe essere perfettamente funzionante e potrà essere tarato.

Per la taratura occorrerà accedere ai trimmer presenti sulla scheda. Se il montaggio nel contenitore isolato esterno lo permette, conviene fissare prima l'assieme al pannello o al guscio frontale, altrimenti esso dovrà essere tarato fuori dal mobile.

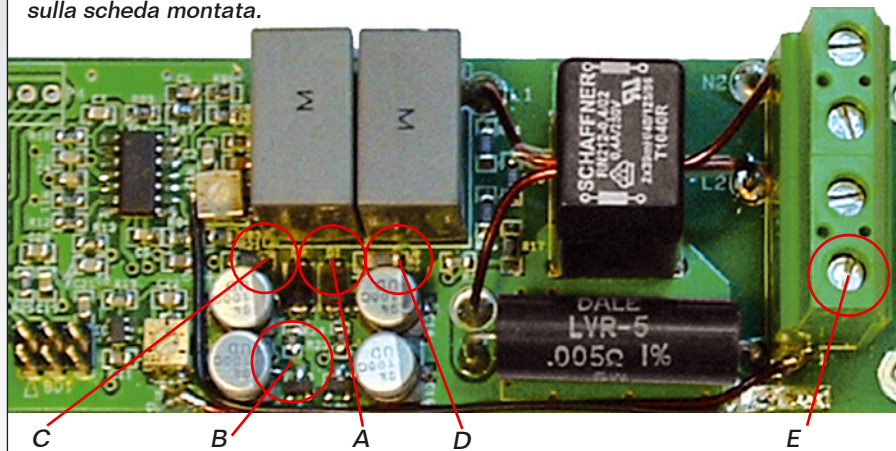
In ambedue i casi, ma specie nel secondo, occorre porre grandissima attenzione alla sicurezza personale e predisporre i circuiti, gli apparati di misura e tutte le parti in modo da evitare, anche per sbaglio, di toccare elementi sotto tensione di rete. Si ricorda che, durante il funzionamento, il circuito è sempre collegato alla rete e non c'è certezza di quale capo sia effettivamente collegato al "neutro" e quale alla "fase": quindi comportatevi come se qualsiasi punto del circuito sia sottoposto a tensione pericolosa. Indossate guanti di gomma mentre maneggiate il circuito sotto tensione: guanti del tipo utilizzato in cucina sono adeguati, purché si presti la massima attenzione a non forarli con elementi appuntiti del circuito (ad esempio i terminali sporgenti dei componenti).

Altra precauzione utile è quella di indossare scarpe dalla suola di gomma ed evitare di toccare, durante la taratura, qualsiasi strumento o mobile metallico collegato alla rete e/o a terra. Tutti i componenti coinvolti nel processo di taratura ed in particolare il circuito da tarare, devono essere appoggiati su un tavolo con la superficie in legno o in plastica. Alimentate il circuito con un trasformatore di isolamento, ovvero con primario e secondario isolato a 230 V, renderebbe più sicuro il lavoro, tuttavia a causa della corrente a cui converrà effettuare la taratura, questo elemento dovrebbe avere una potenza di circa 1 kW ed è difficile pensare che un hobbista abbia in casa un trasformatore del genere. Un'altra possibilità è effettuare la taratura del circuito in corrente a bassa tensione, per esempio a 12 Vca. Ciò è possibile utilizzando come carico, ad esempio, delle lampadine ad incandescenza alogene di tipo faretto da 25/40 W alimentate da un trasformatore. La cosa tuttavia è piuttosto macchinosa, perché i 12 Vca non bastano ad alimentare il nostro strumento, quindi dovrete disporre di un'alimentazione supplementare come descritto per la fase di programmazione del micro. D'altra parte, con un po' di attenzione e procedendo con prudenza è facile e veloce tarare il circuito collegato alla rete.

Per la taratura occorrono:

1. un carico di tipo resistivo in grado di assorbire da 2,5 a 4 Aca a 230 V; quello più indicato è un insieme di lampade alogene per complessivi $600 \div 1.000$ W o, in alternativa, una stufetta possibilmente priva di ventilatore (che altrimenti introdurrebbe una componente reattiva) o un fornello elettrico di pari potenza;

Fig. 3 - Queste sono le posizioni dove saldare i fili di alimentazione provvisoria sulla scheda montata.



2. un carico resistivo di piccola potenza (20 W) capace di assorbire non più di 100 mA a 230 V; l'ideale è una lampadina ad incandescenza, ma, dato che non è facile averne a disposizione una con questa potenza, è possibile adoperarne due da 40 o 60 W in serie; altro carico ideale è una resistenza da $2.700 \Omega / 25$ W;
3. un amperometro in c.a. di

buona precisione ($>0,5\%$) possibilmente del tipo "a vero valore efficace", anche se, essendo resistivo il carico di prova, ciò non è indispensabile in quanto la corrente sarà ragionevolmente sinusoidale; ricordate che dalla precisione di questo strumento, utilizzato come "campione primario", dipenderà l'accuratezza del nostro strumento nella misura

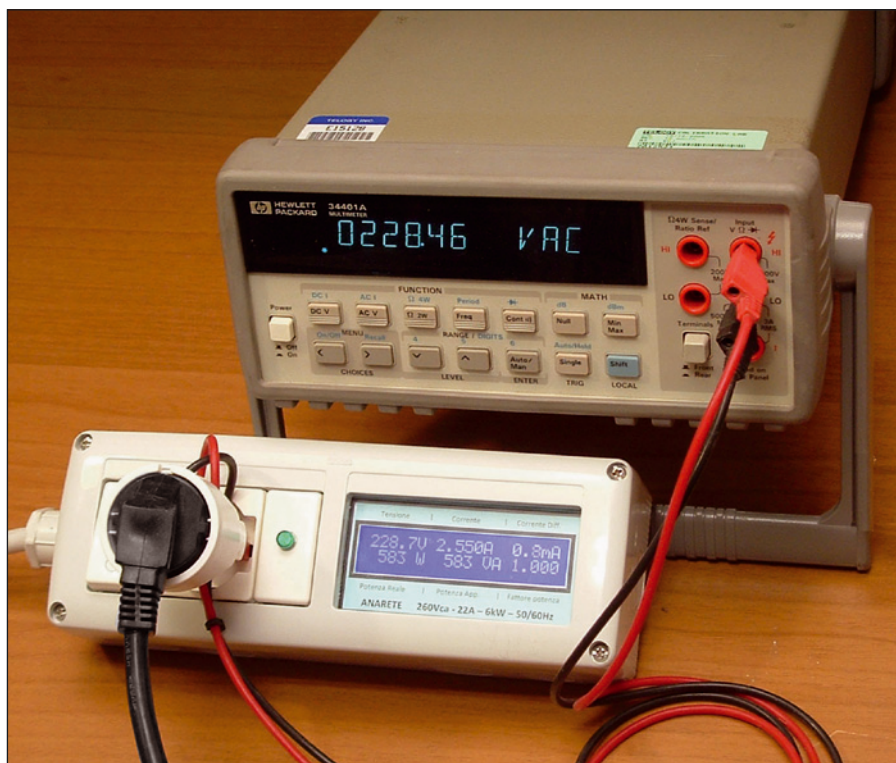


Fig. 4 - Fase della taratura con carico resistivo e il voltmetro/amperometro campione.



Escono rifiuti, rientra elettricità

Presso la base militare dell'aeronautica militare di Edwards, in California (U.S.A.) è stato installato un generatore sperimentale che permette di ricavare energia dai rifiuti di vario genere normalmente prodotti nei vari settori dell'insediamento militare. Si tratta del GEM (Green Energy Machine) ed è prodotto dalla IST Energy e risparmierà i costi di trasporto e smaltimento dei rifiuti, i quali, anzi, frutteranno energia sotto varie forme. Ma attenzione: non si tratta di un inceneritore, bensì di un gassificatore che mediante pirolisi (forte riscaldamento in assenza o scarsità di ossigeno) trasforma la spazzatura -compattata e poi ridotta in pellet- in syngas, ossia un gas di sintesi composto per la maggior parte di monossido di carbonio, idrogeno e metano. Il syngas può essere poi utilizzato per alimentare motori endotermici a gas (con cui azionare generatori elettrici) o bollitori o essere mescolato ad altri combustibili convenzionali; lo scarto può diventare nuovo combustibile e comunque non influisce significativamente sul processo, che produce energia da ben il 95 % del materiale trattato.

in corrente ed in potenza (un tester economico ben difficilmente avrà questo tipo di precisione nella misura della corrente alternata);

4. un voltmetro a tensione alternata a valore efficace di buona precisione; servirà solamente a verificare la precisione della

misura di tensione fatta dal nostro analizzatore.

REGOLAZIONE DEI MISURATORI DI TENSIONE E CORRENTE

Collegare il circuito alla rete secondo lo schema di Fig. 5 e regolare il trimmer RT1 per il miglior contrasto di lettura del display.

Si noti che se il cursore del trimmer è a centro scala, il display potrebbe risultare illeggibile; se vi trovate in questa condizione, prima di arrivare alla conclusione che il circuito non funziona, provate a ruotare il cursore del trimmer quasi completamente in senso orario.

Leggete la tensione rilevata dal nostro misuratore e comparatela con quella del voltmetro di riferimento: le due letture dovranno differire per meno dello 0,5 % (meno di 1 V su 230). Scostamenti maggiori, salvo grossolani problemi circuitali o componenti di bassa qualità, potrebbero far dubitare della precisione del voltmetro campione. Si noti che buona parte della precisione delle misure deriva dalla precisione della tensione di riferimento, che nel nostro caso coincide con quella di alimentazione ed è quella che si misura tra l'anodo di IC3 ed il catodo di IC4: questa tensione deve essere pari a $4,990 \pm 0,020$ V.

Bene, fatto ciò abbiamo tarato, così, il misuratore di tensione; ora passiamo a regolare il circuito per la misura della corrente. Con la stessa configurazione di Fig. 5, con il carico collegato, tarate il trimmer RT2 per ottenere la stessa misura del misuratore di corrente campione. Si può notare che, siccome il carico è di tipo resistivo, la potenza reale indicata in W e quella apparente in VA (che è pari al prodotto tensione per corrente) sono coincidenti.

REGOLAZIONE DELLA SENSIBILITÀ DELLA MISURA DI CORRENTE DIFFERENZIALE

Collegate il sistema come mostrato nella Fig. 6, in cui il ritorno della corrente di carico non si chiude attraverso la linea di uscita; ora tarate il trimmer RT3 per ottenere la stessa misura del misu-

Fig. 5 - Setup taratura corrente.

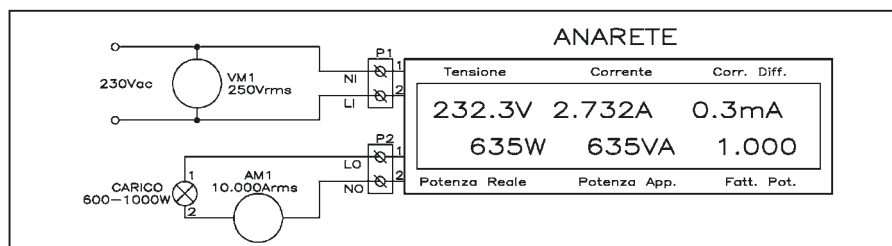
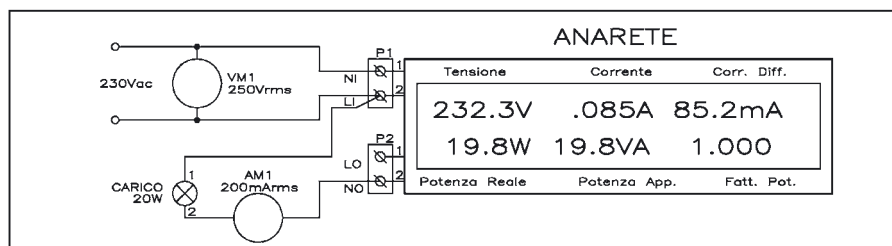


Fig. 6 - Setup taratura corrente differenziale.



ratore di corrente campione. Sia la corrente assorbita che quella differenziale letta dal nostro strumento, avranno valori pressoché identici a parte la migliore risoluzione della seconda misura.

Con questa regolazione si conclude la descrizione della taratura del misuratore di grandezze di rete, che è così pronto a funzionare con qualsiasi tensione e corrente previste nei limiti di funzionamento.

ASSEMBLAGGIO FINALE

Il circuito finito e tarato deve essere montato in un contenitore ben isolato, possibilmente realizzato completamente in plastica, oppure metallico dotato di colonnine distanziali isolanti e messa a terra di sicurezza. In ogni caso la soluzione scelta deve minimizzare il rischio di contatti accidentali con tensioni elevate o con la rete. Data la presenza della tensione di rete su tutto il circuito, la scelta migliore è quella di un mobiletto completamente in plastica provvisto di una finestra trasparente per rendere visibile il display, una o più prese per il carico di misura, un eventuale interruttore e una spia di accensione.

Per il prototipo è stato utilizzato un mobiletto modulare commerciale della Gewiss, facilmente

reperibile presso un negozio di articoli elettrici, destinato all'alloggiamento di prese ed interruttori. Delle due aperture presenti a questo scopo, una è stata modificata internamente

fino a ricavarne la finestra per il display del circuito, protetta da una lamina trasparente di polycarbonato, mentre l'altra è stata utilizzata per montarvi dentro una presa polivalente da 16A, un interruttore bipolare anch'esso da 16A e un tasto falso a supporto di una spia al neon di piccolissima potenza, che segnala la presenza di rete sulla presa. Lateralmente al mobiletto sono stati collocati un portafusibile di sicurezza da 20 A del tipo 6,3x32mm e il cavo di alimentazione tripolare (2 fasi+terra) con sezione 3x1,5mmq, attraverso un passacavo.

Nella **Fig. 7** vedete l'interno dello strumento cablato; nella **Fig. 8** è invece mostrato lo schema di collegamento adottato. Nel mettere insieme il dispositivo è importante mantenere un buon isolamento

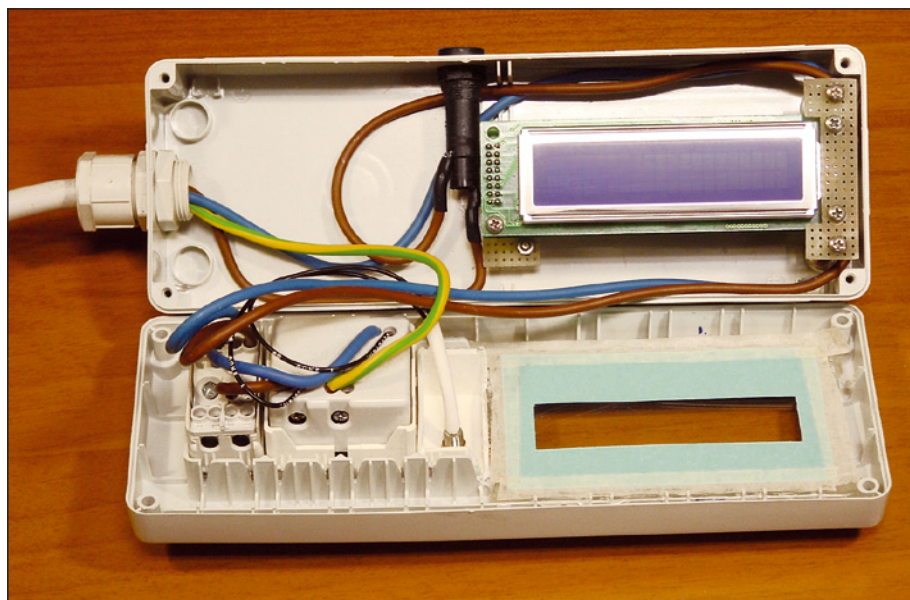


Fig. 7 - Assemblaggio interno del misuratore dentro il mobiletto modificato.

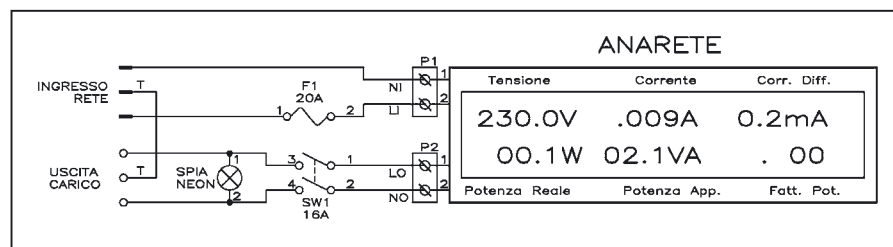


Fig. 8 - Cablaggio definitivo dell'analizzatore di rete.

Fattore di potenza e normativa EN61000-3-2

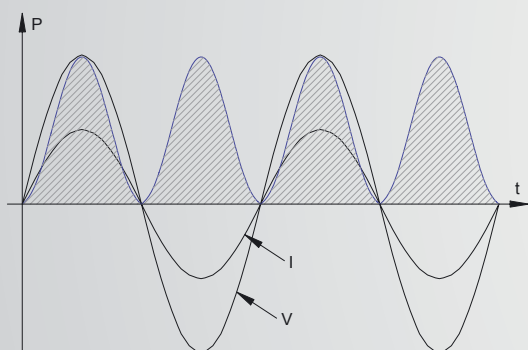
La definizione di valore efficace per la corrente è precisamente: "il valore di corrente efficace è quel valore in corrente continua che produce lo stesso effetto riscaldante in un carico puramente resistivo di quello della corrente alternata di cui si vuole misurare l'intensità, mediato nel tempo". In regime perfettamente sinusoidale c'è una precisa relazione tra corrente efficace, corrente di picco e corrente media. In particolare, la corrente efficace (I_{eff}) è uguale alla corrente di picco (I_{pk}) diviso $\sqrt{2}$. La corrente media (I_m) è invece inferiore alla corrente efficace, infatti $I_{eff} = I_m \times 1,11$. I normali tester per corrente alternata in genere rilevano il valore

medio che è molto più facile da misurare e da questo ricavano il valore efficace moltiplicandolo per la costante 1,11. Se la corrente però non è sinusoidale ma ha un'altra forma (p.e. onda quadra) o è variamente distorta, il valore così ricavato può essere fortemente errato. Da ciò deriva l'importanza di misurare il vero valore efficace quando l'effetto termico è quello più rilevante, come nel caso del riscaldamento indotto dalla corrente nei fili elettrici dell'impianto di casa oppure negli avvolgimenti di un trasformatore che devono essere dimensionati per la massima corrente efficace. Com'è noto, un carico resistivo assorbe corrente in maniera perfettamente

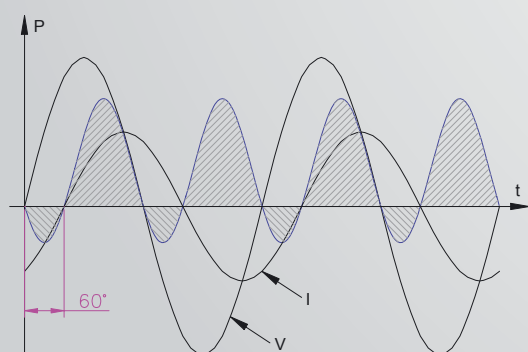
proporzionale alla tensione ad esso applicata. In questo caso si dice che la corrente è "in fase" con la tensione e la sua forma d'onda è una replica di quest'ultima. In questo caso la potenza assorbita dal carico è massima ed è pari a $P(t) = v(t) \times i(t)$, ove a $v(t)$ ed $i(t)$ possiamo sostituire rispettivamente il valore della tensione e della corrente efficace. La potenza istantanea ha anch'essa forma sinusoidale con frequenza pari al doppio di quella della tensione e della corrente, ed è sempre positiva (potenza assorbita).

Non è così per carichi di tipo reattivo, induttivo o capacitivo. In questo caso la corrente, pur rimanendo sinusoidale, non è più in fase con la tensione ma è in ritardo (carico induttivo) o in anticipo (carico capacitivo). Nella pratica è più facile trovare in giro carichi di tipo induttivo come motori, trasformatori o induttori in serie a lampade a scarica di gas (neon o equivalenti). I carichi capacitivi sono in genere utilizzati in parallelo a quelli induttivi per compensare ("rifasare") l'assorbimento non in fase. Sviluppando il prodotto $P(t) = v(t) \times i(t)$, il risultato è ancora sinusoidale con frequenza doppia di quella delle componenti tensione e corrente, ma la sinusoide risultante non è più sempre nel quadrante positivo: quando la sinusoide della potenza istantanea è nel quadrante negativo, il carico cede energia alla rete. Senza entrare di più nel teorico, il valore della potenza media assorbita è proporzionale al coseno dell'angolo di sfasamento tra tensione e corrente ($\cos \varphi$). La potenza è massima quando l'angolo è 0° e diminuisce progressivamente man mano che l'angolo aumenta, fino ad annullarsi del tutto quando l'angolo diventa 90° . Nei tre esempi, la stessa corrente con sfasamento 0° , 60° e 90° produce potenza pari a $P_o = V_{eff} I_{eff}$, $P_o/2$ e 0.

Correnti fortemente sfasate rispetto alla tensione aumentano inutilmente la circolazione di corrente nelle linee elettriche senza cedere effettivamente potenza agli utilizzatori e conseguentemente causano perdite di conduzione. Per questo motivo le società elettriche pretendono in genere che l'assorbimento sia sostanzialmente in fase $\cos \varphi \geq 0,95$ ovvero applicano una penale quando così non è. Il nostro misuratore di rete legge senza difficoltà il $\cos \varphi$ di un carico reattivo (per esempio una lampada a neon con reattore non rifasata) e permette di valutare facilmente per tentativi il condensatore da inserire in parallelo per rifasarla. I carichi costituiti dagli apparati elettronici moderni (PC, Televisori, Alta fedeltà, Strumentazione elettronica generica, ecc.) non si comportano nel modo appena descritto. A causa della presenza praticamente costante nell'alimentatore collegato alla rete di un rettificatore che carica un condensatore per produrre corrente continua, assorbono corrente solo in prossimità del picco positivo e negativo della tensione di ingresso, per un tempo relativamente piccolo rispetto al periodo della sinusoide (in genere il 15-30 %). Sebbene la corrente sia sostanzialmente in fase con la tensione, l'assorbimento è di tipo impulsivo e il valore di picco della corrente è di gran lunga superiore al suo valore medio: in genere è compresa tra 2 e 5 volte. Ricordiamo che nel caso di corrente perfettamente sinusoidale, il valore di picco è pari a $I_{pk} = 1,11 \times \sqrt{2} \times I_m$, ovvero $I_{pk} = 1,57 \times I_m$. Da questo consegue che anche la corrente efficace è di parecchio maggiore della corrente media assorbita ed effettivamente utilizzata e poiché le perdite nelle linee elettriche sono proporzionali alla corrente efficace, queste ultime sono parecchio maggiori che nel caso



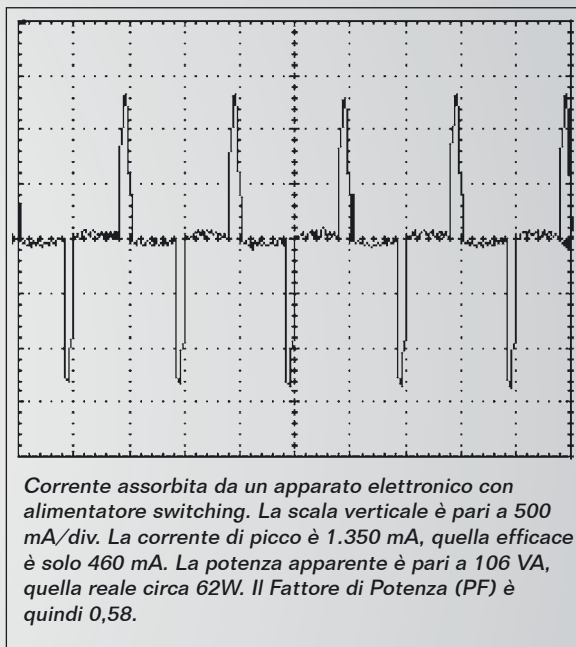
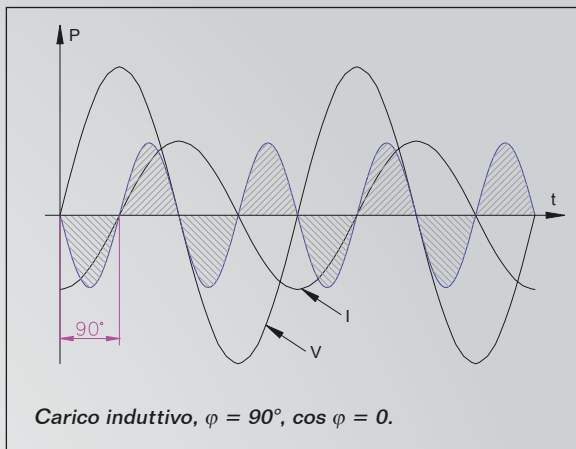
Carico resistivo con corrente "in fase",
 $\varphi = 0^\circ$, $\cos \varphi = 1$.



Carico induttivo, $\varphi = 60^\circ$, $\cos \varphi = 0,5$.

la corrente assorbita fosse di tipo sinusoidale. La corrente in questo caso è fortemente distorta ed accanto alla componente a frequenza fondamentale pari a quella di rete, che sopporta tutta la potenza reale assorbita dal circuito, sono presenti innumerevoli armoniche di ordine dispari (III, V, VII, ecc) che non veicolano potenza reale ma sono causa di perdite sulla rete.

Anche in questo caso le Compagnie Elettriche vorrebbero limitare questo tipo di assorbimento, per gli stessi motivi di prima. A tal fine sono state emanate nel corso degli ultimi anni delle norme che limitano per legge questo comportamento nella maggior parte degli apparati di tipo domestico, consumer e parzialmente di tipo professionale sottoposti alle normative CE, vigenti all'interno della Comunità Europea. La principale norma in materia è la EN61000-3-2, che si applica alla maggior parte degli apparati elettrici/elettronici di tipo comune con assorbimento dalla rete monofase a 220 V minore di 16 A e potenza utilizzata compresa tra 75 e 1.000 W. Gli apparati sono divisi in quattro classi (A, B, C, D) e la maggior parte di quelli il cui comportamento è simile a quanto rilevato nell'oscillogramma di cui sopra è collocata in categoria D (PC, Televisori, Alta Fedeltà, ecc.). La norma limita pesantemente e progressivamente la quantità di corrente "distorta" che l'apparato può assorbire e lista il valore massimo di corrente assorbita alle varie armoniche della frequenza di rete fino alla 40^a (2.000Hz). L'abbattimento del contenuto armonico nella corrente assorbita si può ottenere con semplici



filtri passa-basso di tipo passivo (induttori in serie) nei casi meno impegnativi, che però diventano pesanti e grossi all'aumentare della potenza e comunque non sopprimono efficacemente le armoniche (che sono quelle che generano le perdite di potenza inutili nella rete). Quasi tutti gli alimentatori elettronici switching di ultima generazione a norma, di potenza superiore a 75 W, impiegano invece oramai dei circuiti elettronici di front-end che correggono il fattore di potenza ovvero un circuito PFC (Power Factor Corrector). Sebbene

questo tipo di circuito complichino ed aumentino il costo dell'alimentatore, grazie ad esso l'assorbimento torna ad essere sostanzialmente di tipo sinusoidale e il PF è generalmente superiore a 0,95÷0,98 %.

Il Misuratore di Grandezze di Rete non fa distinzioni tra misura di $\cos \varphi$ e di PF, infatti ambedue questi valori, sempre inferiori o pari ad 1, sono misurati allo stesso modo, facendo il rapporto tra Potenza Reale [W]/Potenza Apparente [VA] che il Misuratore rileva contemporaneamente e separatamente in ogni istante.

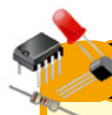
tra tutti gli elementi metallici collegati al circuito ed in particolare con le colonnine utilizzate per il fissaggio reciproco tra il display e la scheda di misura.

Per questo motivo non è possibile prolungare le colonnine metalliche distanziali fino al fondo o comunque alle pareti del mobile per fissarle con viti esterne: l'isolamento finale ne sarebbe certamente compromesso.

Occorre invece utilizzare delle colonnine di prolunga isolate oppure degli elementi isolanti tra i fissaggi sul circuito e quelli esterni. Nella Fig. 7 potete notare l'uso di alcuni spezzoni di basetta isolante in vetroresina di tipo "millefori", utilizzati per il collegamento isolato tra i fissaggi interni e quelli esterni.

Affinché sia garantito l'isolamento, dovete asportare tutte le piazzole di rame comprese tra le viti di fissaggio nel raggio di almeno 4 mm, allo scopo di ottenere l'isolamento imposto dalle normative di sicurezza; in alternativa, usate pezzi di vetroresina non ramata.

Il mese prossimo approfondiremo le problematiche relative alla programmazione del micro e proporremo alcune possibili elaborazioni circuitali e sviluppi hardware/software.



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato ed il firmware per programmare il microcontrollore Freescale utilizzato possono essere scaricati gratuitamente dal sito della rivista (www.elettronica-in.it).

PUNTATA 4

Corso di programmazione iPhone



In questa quarta puntata del nostro corso vogliamo spendere qualche parola sull'interfaccia utente dei dispositivi Apple: faremo quindi una veloce carrellata su tutti i controlli grafici di cui essa dispone e descriveremo un primo esempio di interazione utente-dispositivo. Successivamente parleremo di *autorotating* ed *autosizing*, due interessanti proprietà che permettono di adattare la visualizzazione sullo schermo quando si gira il dispositivo, analiz-

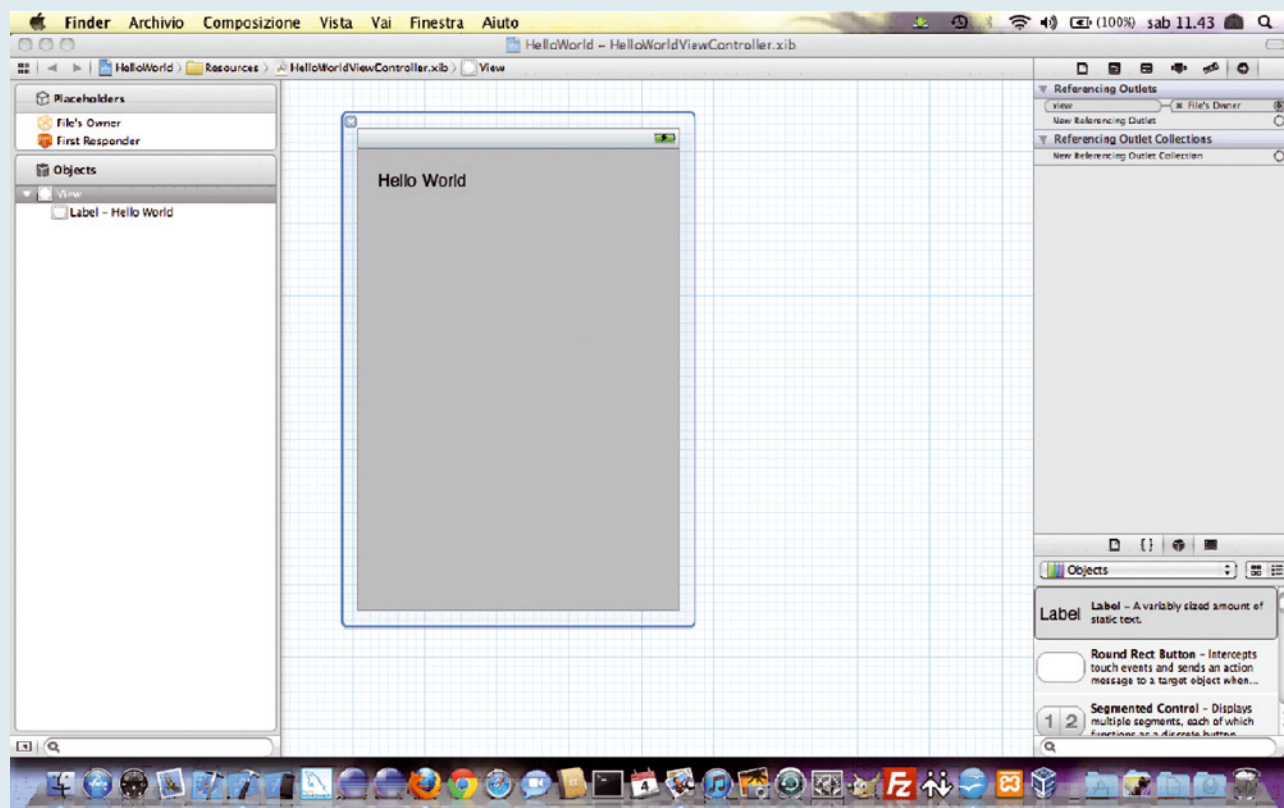
zando i problemi legati a rotazione e dimensionamento. In ultima analisi vedremo un nuovo progetto pratico, fatto utilizzando la Table View, forse il tipo di vista più usato nel mondo iPhone e legata alle App.

Interfaccia utente

L'iPhone, così come iPod Touch, ha un display fulltouch; questo rappresenta l'unica interfaccia verso l'utente che possiamo utilizzare, dal punto di vista programmatico, per generare

Conosciamo l'interfaccia utente ed i controlli grafici, approfondendo il discorso su autorotating e autosizing, concludendo con un progetto pratico realizzato mediante la Table View.

Fig. 1 - Aggiungere il pulsante alla nostra interfaccia.



azioni che lavorino con il controller, nel modo che abbiamo visto nel modello MVC.

Tutti i “design” grafici che possiamo fare su iPhone, estendono la classe base di tipo UIView; questo concetto è fondamentale, infatti le etichette, i pulsanti, i picker e tutti gli altri componenti visuali, sono estensioni di tale classe base.

I componenti grafici che ci vengono forniti dalla Apple sono molteplici e possono essere divisi in quattro categorie:

- Controllers;
- Data Views;
- Input & Values;
- Windows, Views & Bars.

I Controllers sono un metodo per attaccare alla nostra vista dei controlli già realizzati e pronti per l’uso; tra questi spiccano i “Tab Bar Controller”, i “Navigation Controller”, i “TableView

Controller” ed il “View Controller” che abbiamo già visto ed utilizzato nella scorsa lezione. Altri Controllers li analizzeremo già da questa puntata, iniziando con il “TableViewController”; nelle prossime puntate utilizzeremo i restanti.

Il secondo gruppo, “Data Views”, riguarda viste dedicate a risolvere problemi specifici, come la “Table View” per le tabelle, la “Image View” per visualizzare immagini, la “Web View” (che utilizzeremo nelle prossime pun-

Listato 1 - Aggiungere il prototipo nell’installazione.

```
#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController {
    UILabel *etichetta;
}

- (IBAction) cliccaQui:(id)sender;

@property (nonatomic, retain) IBOutlet UILabel *etichetta;

@end
```


tate per creare un nostro browser web), la "Map View" per vedere le mappe di Google (e molte altre) e i pickers per selezionare date oppure valori a nostra scelta.

Le "Input & Views" sono le più utilizzate nel disegno personale delle interfacce; tra esse troviamo le "Label", i "Button" per far interagire l'utente, le "Text Field" per raccogliere il testo che un utente inserisce, gli "Switch" e così via.

L'ultimo gruppo riguarda le "Windows, View & Bars", dedicate alla navigazione; qui troviamo la "Search Bar" per fare ricerche personalizzate, le "ToolBar" per la creazione di una propria barra personalizzata, per aggiungere contenuti e molto altro.

Potete navigare in queste categorie direttamente da Interface Builder e approfondirne la conoscenza tramite la documentazione in linea. In questo corso faremo uso dei controlli più famosi ed utilizzati.

Lavoriamo sull'Hello World

Riprendiamo in mano il codice che abbiamo scritto nella precedente lezione, cui vogliamo aggiungere l'interazione con l'utente; per fare questo dobbiamo in primo luogo aggiungere un Button nella nostra vista, come abbiamo fatto nella scorsa puntata tramite Interface Builder (Fig. 1). Il componente grafico che vogliamo aggiungere è il "Round Rect Button", per utilizzare il quale lo selezioniamo facendo clic su di esso e lo trasciniamo all'interno della vista. Una volta che l'abbiamo posizionato, vi facciamo doppio clic per inserire il testo al suo interno, mettiamo il testo "Clicca Qui" e premiamo Invio per confermare. Per inviare le azione al controller dobbiamo creare il nostro primo metodo, che poi collegheremo alla vista tramite Interface Builder; allo scopo, all'interno del file *HelloWorldViewController.h* aggiungiamo una action (metodo), come possiamo vedere nel **Listato 1**.

Abbiamo già visto che il simbolo "-" dichiara un metodo di istanza e non di classe. Tra le parentesi tonde troviamo il corrispettivo di "IBOutlet" per quello che riguarda le azioni, ossia "IBAction". Anche questo non ha alcun senso programmatico, ma serve solo per

Listato 2 - Implementazione del metodo "cliccaQui".

```
-(void)cliccaQui: (id)sender{
    self.etichetta.text = @"L'utente ha fatto clic";
}
```

generare le dovute informazioni all'interno di Interface Builder. Come vedremo tra poco, nell'implementazione della classe il tipo di ritorno sarà "void" e non IBAction, come ci aspetteremmo.

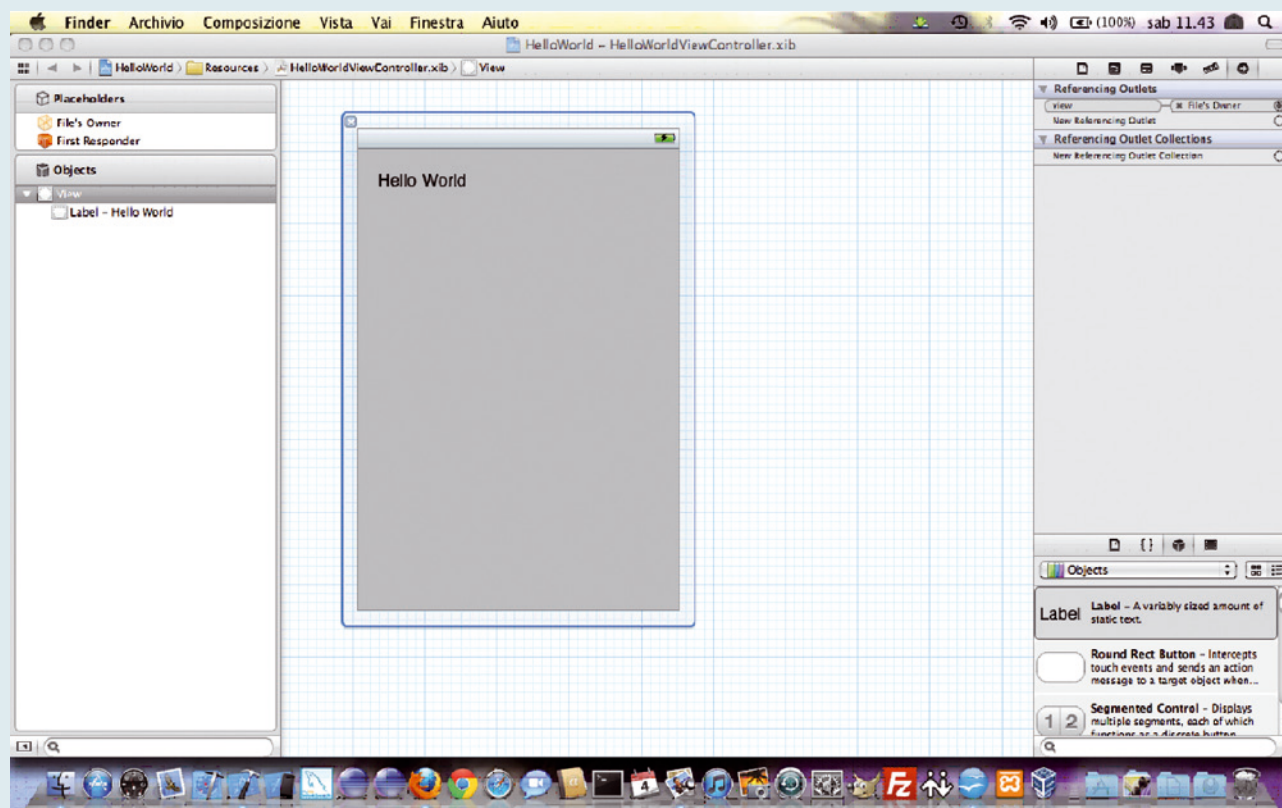
Dopo questo abbiamo il nome del metodo, "cliccaQui" e poi i parametri, come lasciano intendere i due punti. Per i parametri dobbiamo sottolineare la presenza della parola chiave "id", in quanto il tipo non è definito, ma viene stabilito a run-time; abbiamo già detto che ObjectiveC è debolmente tipizzato e quindi possiamo, in certe circostanze, non avere un tipo definito a priori. Notate che "id" è seguita dal nome della variabile, che in questo caso è "sender".

Sino ad ora abbiamo aggiunto la nostra "firma di metodo" o "prototipo", se preferite; ora ne dobbiamo scrivere l'implementazione, quindi apriamo il relativo file ".m" ed aggiungiamo la nostra implementazione, come mostrato nel **Listato 2**. Qui abbiamo dichiarato il tipo di ritorno come "void"; il placeholder "IBAction" introdotto nel file ".h" non esiste e serve solo per l'utilizzo di Interface Builder.

Ora dobbiamo collegare il componente grafico introdotto in Interface Builder con il codice; allo scopo, apriamo la vista "HelloWorld-ViewController.xib". Qui, se facciamo clic su "File's Owner" e guardiamo il box "Connections", notiamo che si è creato un box "Received Actions" con il nome del metodo da noi sviluppato (Fig. 2).

Come abbiamo fatto per collegare la label al codice, nello stesso modo facciamo clic e trasciniamo la "cordina" azzurra sul pulsante; sempre nella Fig. 2, vediamo che invece di eseguire subito il collegamento, come era successo precedentemente, appare un box grigio con diverse possibilità. Esistono molti tipi di interazioni disponibili, ma quella che ci interessa per ora è il "Touch Up Inside"; in breve, ciò significa che andremo a fare il touch su di esso.

Fig. 2 - Selezione del tipo di azione.



Una volta selezionata questa informazione, si è formato il link nel box "Connections".

A questo punto possiamo avviare la nostra App e fare clic sul pulsante: vedremo che la nostra stringa non è completa, dato che nel po-

sizionare l'etichetta nella vista, abbiamo lasciato la dimensione predefinita, ma adesso, con un testo lungo quale "L'utente ha fatto clic", il campo viene tagliato e in esso vengono inseriti dei puntini per indicare che il contenuto continua. Per aumentare la dimensione del campo possiamo riaprire IB e fare clic sulla label: viene quindi mostrato un contorno con diversi piccoli quadrati, dove selezioniamo quello di destra; facendo clic e tenendo premuto il

tasto del mouse, trasciniamolo fino al bordo destro (Fig. 3). A questo punto lo spazio a disposizione per la nostra etichetta è grande a sufficienza. Di tutti i controlli grafici è possibile modificare le dimensioni, perciò possiamo fare lo stesso lavoro anche con il pulsante e tutte le altre viste che utilizzeremo. Se proviamo a riavviare la nostra applicazione e facciamo clic, notiamo che ora la stringa non è più troncata, ma viene visualizzata per intero. Ricordate che dobbiamo sempre liberare la memoria, quindi inserite la chiamata al metodo "release" nel metodo "dealloc", come mostrato nel **Listato 3**.

Autorotating e Autosizing

Autorotating ed autosizing sono due proprietà fornite dalla Apple: la prima riguarda la capa-



Fig. 3 - Dimensionamento dell'etichetta.

Listato 3 - Liberare la memoria dalla nostra etichetta.

```
- (void)dealloc {
    [etichetta release];
    [super dealloc];
}
```

Listato 4 - Metodo per il controllo dell'orientazione del dispositivo.

```
// Override to allow orientations other than the default portrait orientation.
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {
    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}
```

Listato 5 - Abilitare l'autorotating.

```
// Override to allow orientations other than the default portrait orientation.
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {
    // Return YES for supported orientations
    return YES;
}
```

cità del dispositivo di capire se è stato ruotato dall'utente e quindi riadattare la vista che si sta visualizzando per adeguarsi al cambio di dimensione dello schermo; questo perché, se pensiamo all'iPhone 3GS, si passa da 320x480 pixel in posizione verticale (formato *portrait*) a 480x320 quando si è in modo *landscape*, ovvero orizzontale.

Intercettando *autorotating* possiamo scegliere se adattare semplicemente la vista eseguendo l'*autosizing*, oppure spostare i componenti programmaticamente o, ancora, cambiare vista. Per attivare l'*autorotating* del dispositivo, dobbiamo aprire il file "HelloWorldViewController.m" e trovare il metodo "shouldAutorotateToInterfaceOrientation". Nel **Listato 4**, questo metodo ritorna un boolean e nella configurazione decommentata viene ritornata la vista "Portrait"; se vogliamo gestire tutte le orientazioni disponibili possiamo semplicemente eseguire il "return" di "YES", come si vede nel **Listato 5**.

Se ora avviamo la nostra applicazione, tramite il menu del simulatore "Hardware" e poi "Ruota a sinistra", vediamo che automaticamente la nostra vista si è riconfigurata per la nuova orientazione (**Fig. 4**).

Se volessimo gestire solo le viste orizzontali, dovremmo utilizzare il metodo "shouldAutorotateToInterfaceOrientation" e utilizzare la costante "UIInterfaceOrientationLandscapeLeft" al posto di "Portrait" (**Listato 6**).

Quando orientiamo il dispositivo dobbiamo fare attenzione alle viste, perché le dimensioni

dello schermo si scambiano e quindi le viste progettate per una visualizzazione "Portrait" non saranno adatte a quella "Landscape". La gestione dell'*autosizing* è invece possibile tramite il menu "View Size", direttamente da Interface Builder (**Fig. 5**) tramite il box a destra, chiamato proprio "Autosizing".

L'*autosizing* è molto comodo, in quanto permette lo stretching automatico dei componenti; però fate attenzione che non tutte le risorse possono essere scalate con semplicità. Lasciamo alla vostra curiosità l'approfondimento di questo componente; per i nostri scopi ci dedicheremo principalmente alla gestione di una singola orientazione.

MultiViews & Controllers

Come abbiamo già detto, tutti i componenti che possono essere posizionati sullo schermo sono delle View ed estendono la classe base "UIView"; per questo motivo una "UILabel" è

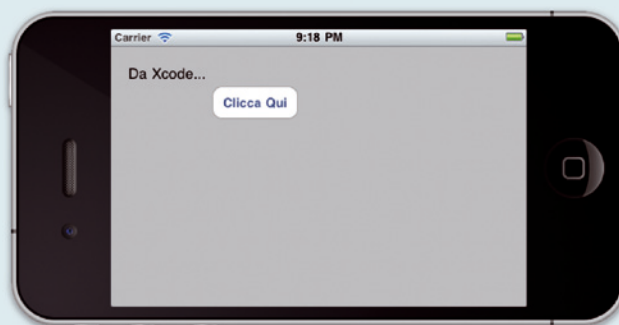


Fig. 4 - Rotazione della vista.

Listato 6 - Orientare il dispositivo in orizzontale.

```
// Override to allow orientations other than the default portrait orientation.
- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation {
    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationLandscapeLeft); //orizzontale-sinistra
}
```

concettualmente identica alla vista che abbiamo progettato nella terza puntata. Siccome tutte estendono la classe base `UIView`, possiamo gestirle allo stesso modo ed avranno tutte un preset di metodi e proprietà di base.

Con `MultiViews` si vuole indicare che è possibile mettere in una vista tante altre viste, come abbiamo già fatto con la label dentro la nostra view.

Creare nuovi Controllers e legare le Viste

Come per le viste, è possibile generare e gestire più controller sia singolarmente che innestati uno dentro l'altro. Per farlo, non dobbiamo fare altro che creare un nuovo controller con associata una vista: facendo `MELA+N` sul progetto, aggiungiamo un file di tipo "`UIViewController subclass`" (Fig. 6). Attenzione a spuntare la selezione "`With XIB for user interface`".

Facendo clic su `next` in basso a destra, aggiungiamo il nome alla classe "`SecondViewController`" e proseguiamo.

Nel nostro progetto, oltre alla vista "`HelloWorldViewController.xib`" ne è disponibile

ora una seconda, ed abbiamo anche il codice di un altro controller.

Per aggiungere la vista del controller appena creato alla vista del componente base, non dovremo fare altro che creare il controller con la vista associata in via programmatica ed aggiungere la sua vista a quella del primo. Il **Listato 7** riguarda proprio questa casistica. Nella prima riga stiamo creando un nuovo controller di tipo "`SecondViewController`" collegando la vista tramite il suo file "`XIB`"; eseguiamo il metodo "`alloc`" e poi "`initWithNibName`" come costruttore dell'oggetto (Controller, in questo caso).

Alla seconda riga, invece, stiamo chiamando il metodo "`addSubview`" sulla proprietà "`view`" del controller su cui stiamo lavorando: in pratica stiamo aggiungendo una sotto-vista a quella corrente. Come potete immaginare, le dimensioni sono importanti e dovete lavorare molto con IB per costruire la seconda interfaccia ed attaccarla bene alla vista.

Prima di passare alle `TableView` dobbiamo sottolineare che allocando il controller in quel

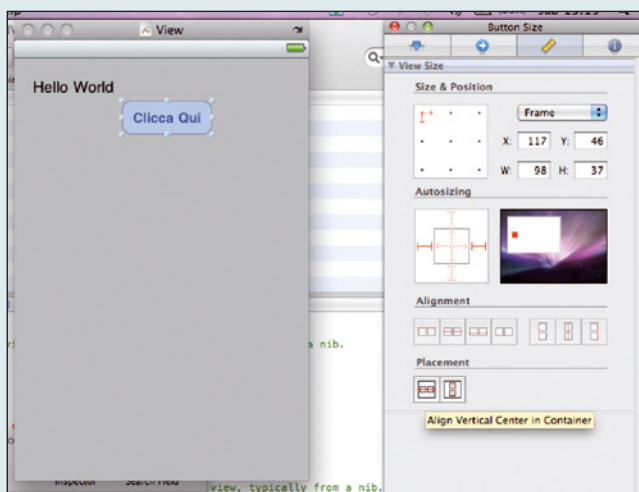


Fig. 5 - Gestione dell'autosizing.

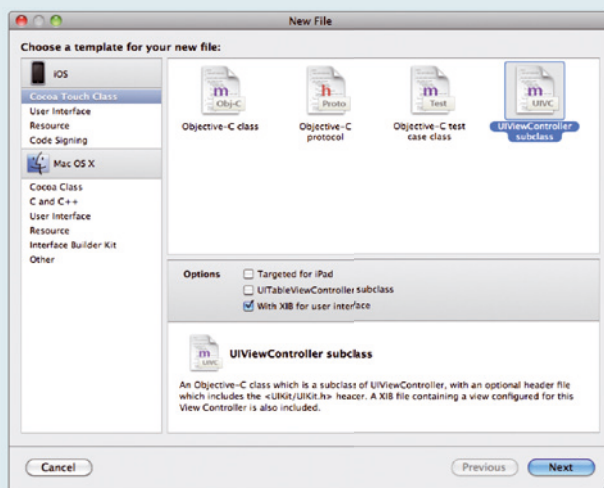


Fig. 6 - Creare un nuovo controller.

Listato 7 - Aggiungere una vista ad un'altra.

```
SecondViewController *controller = [[Second-
ViewController alloc] initWithNibName:@"Sec-
ondViewController" bundle:nil];
[self.view addSubview:controller.
view];
```

modo, non possiamo poi liberarlo nel metodo "dealloc", in quanto non è nello "scope" del nostro oggetto, quindi le possibilità sono due: lo rilasciamo subito oppure dobbiamo introdurlo come proprietà di istanza per poi averlo visibile e poterlo rilasciare in modo consono. Abbiamo fatto questa breve analisi del Multiviews perchè lo utilizzeremo subito per le TableView e per i progetti che andremo a realizzare nello sviluppo del corso.

Progetto Pratico con Table View

Le Table View sono veramente utili ed abbastanza semplici da utilizzare; sono forse il componente grafico maggiormente utilizzato e personalizzato. La Fig. 7 mostra la struttura tipica di una tabella per iPhone.

Tralasciamo la parte di colore azzurro in alto, che rappresenta la navigation bar; le Table View in iOS sono semplicemente delle righe, ognuna delle quali corrisponde ad una UI-

View. Sta a noi utilizzare quella predefinita oppure crearne alcune a piacimento.

Per creare il nostro progetto chiudiamo il precedente e ne iniziamo uno nuovo; indichiamo che vogliamo crearne uno di tipo "Navigation Based Application" e chiamiamolo "TableViewExample". Una volta creato il progetto, avviamo il simulatore per vedere il risultato nella Fig. 8. Iniziamo subito a "colorare" il codice per renderlo utile; il file che ci interessa si chiama "RootViewController".

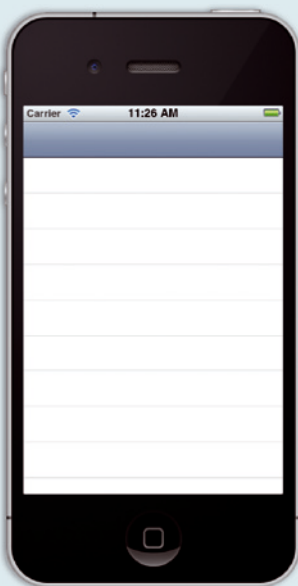


Fig. 7 - Esempio di Table View.

Come indica il nome, è il controller radice, ovvero il primo. Vediamo che questo particolare oggetto estende la classe base "UITableViewController", la quale è un'estensione di quella utilizzata nei precedenti progetti di tipo "UIViewController". Ciò perché, tramite questa classe base, vengono introdotte le funzionalità che estendono un normale controller. Prima di introdurre dei dati all'interno della vista, dobbiamo analizzare due metodi che vediamo nel Listato 8.

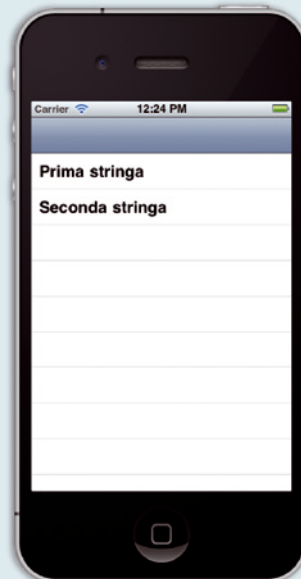
Il primo metodo, "numberOfSectionsInTableView", deve ritornare un numero intero che ci dice in quante sezioni è divisa la nostra tabella; questo perché possiamo dividere la tabella in sezioni e per ognuna avere dei dati specifici. In questo esempio ci sarà sempre una sola sezione, ma sappiate che potete dividere la tabella in più parti.

Il secondo metodo, "numberOfRowsInSection", si occupa di dire quante righe sono presenti in una sezione specifica. Il parametro section, di tipo intero, indica che il core di iOS sta elaborando la sezione specificata e che dobbiamo ritornare il numero di righe che vogliamo stampare per questa sezione.

Per introdurre dei dati ci baseremo su un modello che, per semplificare, sarà il tipo "NSString"; se vogliamo dei tipi più complessi possiamo creare un'estensione di "NSObject" ed utilizzarla come per il tipo base stringa. Per prima cosa dobbiamo creare l'array che andrà a contenere i nostri modelli di dato, quindi nel nostro file ".h" andiamo a fare la dichiarazione di un "NSMutableArray" e sintetizziamo quest'ultimo nell'implementazione della classe. Il Listato 9 contiene entrambi i file da modificare allo scopo.

Ricordate di andare subito a liberare la memoria; non abbiamo bisogno di liberare ogni

Fig. 8 - La nostra App sul modello Table View.



Listato 8 - Metodi base delle Table View.

```
// Customize the number of sections in the table view.
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return 1;
}

// Customize the number of rows in the table view.
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSectionSection:(NSInteger)section {
    return 0;
}
```

Listato 9 - Dichiarare l'array nell'implementazione e nell'intestazione.

```
//RootViewController.h
#import <UIKit/UIKit.h>

@interface RootViewController : UITableViewController {
    NSMutableArray *dati;
}

@property (nonatomic, retain) NSMutableArray *dati;

@end

-----

//RootViewController.m
@implementation RootViewController

@synthesize dati;

//Resto dell'implementazione

- (void)dealloc {
    [dati release];

    [super dealloc];
}
```

singolo oggetto nell'array, in quanto sarà "NSMutableArray" a farlo per noi.

Ora che abbiamo dichiarato i dati possiamo allocare la memoria necessaria all'array e introdurre alcune stringhe, per poi visualizzarle dall'app. Nel metodo "viewDidLoad" aggiungiamo il codice necessario, come possiamo vedere nel **Listato 10**.

A questo punto abbiamo aggiunto i dati; non ci resta che informare il controller che vogliamo visualizzarli nella tabella. Nel metodo "numberOfRowsInSection" chiediamo al componente "NSMutableArray" il numero di elementi e restituiamo questo conteggio

Listato 10 - Aggiungere elementi all'array.

```
- (void)viewDidLoad {
    [super viewDidLoad];

    // Uncomment the following line to display an Edit button in the navigation bar for this view controller.
    // self.navigationItem.rightBarButtonItem = self.editButtonItem;

    //Allochiamo l'array
    dati = [[NSMutableArray alloc] init];
    //Creazione della stringa
    NSString *stringa = @"Prima stringa";
    //Aggiungiamo la stringa all'array
    [dati addObject: stringa];

    //Creazione della stringa
    NSString *stringa2 = @"Seconda stringa";
    //Aggiungiamo la stringa all'array
    [dati addObject: stringa2];
}
```


nel metodo “numberOfRowsInSection”, come vediamo nel **Listato 11**.

Ora ci manca solo un altro passo: informare la vista che vi sono dei dati da visualizzare nelle celle. Per fare questo esiste un altro metodo fondamentale, che si chiama “cellForRowAtIndexPath”. Prima di scrivere il codice dobbiamo sapere che le TableView sono divise in celle, ossia “Cells”, che contengono i dati da visualizzare; per ragioni di riutilizzo e gestione della memoria, non sono sempre tutte disponibili. Il controller “UITableViewController” le inserisce in una coda, evitando di caricarle tutte in memoria, riutilizzandole e rendendo disponibili solo quelle che sono realmente visualizzate dall’utente; perciò è possibile avere un elenco molto lungo senza incorrere in problemi di memoria.

Analizziamo ora il metodo che serve a rendere disponibili i dati (**Listato 12**). Come possiamo vedere, questo metodo ha due parametri: il primo è la vista associata al controller UITableView ed il secondo è un oggetto chiamato “NSIndexPath”. Questa struttura ingloba tutte le informazioni che riguardano la cella che si sta configurando, quindi la sezione di cui fa parte, la riga che si deve configurare, più altre informazioni.

Nelle prime righe si configura la parte base della cella e la sua estrazione dalla coda. Se questa esiste, viene direttamente utilizzata, altrimenti viene creata. Noi andremo a lavorare più in basso, dove è stato posizionato il commento “Configure the cell”. Come mostra il **Listato 13**, dobbiamo inserire solo tre righe, al fine di rendere visibile il dato nella cella indicata dalla proprietà “row” dell’oggetto “IndexPath”. Per costruire la cella stiamo ricevendo l’oggetto stringa dal container di tipo “NSMutableArray” tramite il metodo “objectAtIndex” e lo assegniamo alla variabile locale chiamata strin-

ga; fatto ciò, configuriamo la cella assegnando questo riferimento alla proprietà text della cella stessa. A questo punto avviamo la nostra App e le celle ora visualizzeranno il testo da noi richiesto, come si vede nella **Fig. 7**.

Prima di terminare, sottolineiamo che quando estraiamo l’oggetto dall’array dobbiamo eseguire il “cast” del tipo, che in questo caso è “NSString”. Tale esecuzione è descritta nel **Listato 14**.

Nella prossima puntata riutilizzeremo questa applicazione integrando la “Navigation Bar”, la quale ci permette di navigare nei contenuti della nostra applicazione ogni qual volta facciamo clic su una singola cella di essa. ■

Listato 11 - Informare il controller dei dati disponibili.

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSectionSection:(NSInteger)section {
    return [dati count];
}
```

Listato 12 - Il metodo di visualizzazione delle celle.

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {

    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (cell == nil) {
        cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier] autorelease];
    }

    // Configure the cell.

    return cell;
}
```

Listato 13 - Configurare la cella per la visualizzazione.

```
// Configure the cell.
NSString *stringa = [dati objectAtIndex:indexPath.row];
cell.textLabel.text = stringa;
return cell;
```

Listato 14 - Cast esplicito di una istanza di tipo NSString.

```
NSString *stringa = (NSString *)[dati objectAtIndex:indexPath.row];
cell.textLabel.text = stringa;
return cell;
```



Osservano, spiano ed in alcuni casi colpiscono con grande precisione: sono i velivoli senza pilota a bordo, meglio noti come UAV (Unmanned Aerial Vehicle) il cui ruolo, specie in campo militare, sta diventando sempre più importante.

L'INVASIONE DEI DRONI

di ARSENIO SPADONI

È una delle poche armi che abbiamo contro il terrorismo:

è quanto ripetono, concordi per una volta, politici e vertici militari delle principali nazioni. In realtà gli UAV (Unmanned Aerial Vehicle) o UAS (Unmanned Aerial Systems) stanno diventando delle vere e proprie macchine da guerra e nei prossimi decenni - anche in questo caso le previsioni sono concordi - sostituiranno gli attuali aerei da combattimento con pilota a bordo tanto che molti considerano l'F22 e l'F35, i più recenti caccia prodotti (l'F35 è ancora in fase di sviluppo) come gli ultimi velivoli da combattimento costruiti prevedendo la presenza di un pilota a bordo. Ovviamente questa tecnologia avrà importanti risvolti anche in campo civile tanto che già si parla di aerei cargo completamente unmanned.

Ma come funziona un UAV, qual è stata l'evoluzione della tecnologia e quali gli sviluppi futuri? È quanto cerchiamo di spiegare in questo articolo.

Le tecnologie che

hanno consentito lo sviluppo di questi velivoli sono principalmente tre:

- a) i sistemi di posizionamento e navigazione satellitare che consentono di conoscere, in qualsiasi luogo e con grande precisione, la posizione e l'altezza del velivolo con una approssimazione che in alcuni casi è inferiore al metro e che in futuro sarà addirittura di qualche centimetro. Attualmente questa informazione è garantita dal sistema americano GPS e, in parte, anche dal Glonass russo, mentre in futuro saranno disponibili l'europeo Galileo ed il cinese Beidou;
 - b) i sistemi di trasmissione dati satellitari a larga banda in grado di coprire tutto il globo;
 - c) le piattaforme inerziali e i sistemi di elaborazioni embedded, sempre più performanti.
- A tutto ciò bisogna aggiungere l'evoluzione della sensoristica, dai sistemi di ripresa ottici a quelli radar ad apertura sintetica in grado di operare anche con nuvole e pioggia. Questi mezzi possono essere completamente autonomi (ovvero eseguire una missione di volo pre-impostata) o essere controllati a

Dal più piccolo al più grande

Gli UAV hanno dimensioni, pesi e funzionalità molto diverse a seconda dell'impiego cui sono destinati. La storia di questi velivoli inizia con i cosiddetti aerei bersaglio e con i primi aerei radiocomandati utilizzati dagli americani in Vietnam per scattare foto delle postazioni avversarie. La maggior parte degli UAV viene attualmente utilizzata per la sorveglianza dall'alto con dispositivi ottici, ad infrarossi o radar; solo recentemente alcuni

velivoli sono stati dotati della capacità di colpire (generalmente con missili) le postazioni nemiche. I modelli più piccoli vengono utilizzati sui campi di battaglia per controllare le linee nemiche o aree nascoste (tipicamente per vedere cosa c'è "oltre la collina"); i più diffusi sono quelli dell'americana AeroVironment che attualmente dispone di tre modelli lanciabili a mano: Wasp, Raven e Puma. Il raggio di azione è compreso tra 5 e

15 km e l'autonomia varia tra mezz'ora e un paio d'ore. I velivoli vengono controllati mediante una Ground Station ma sono in grado di compiere anche delle missioni autonome. Le immagini riprese dalla telecamera di bordo (funzionante anche al buio) vengono visualizzate sullo schermo della Ground Station. Esistono tuttavia degli UAV ancora più piccoli come il Nano Hummingbird, realizzato dalla stessa AeroVironment, che ha la forma di un colibrì e che pesa appena 19 grammi.

Sviluppato per l'agenzia DARPA (Defense Advanced Research Projects Agency) dell'esercito americano, questo uccellino-robot presenta la particolarità di sfruttare per il volo un sistema ad ali battenti, del tutto simile a quello del vero colibrì, che gli conferisce una grande agilità con la possibilità di passare attraverso porte e finestre e posarsi ovunque, anche sui fili della corrente. Il minuscolo UAV è comandato a distanza via radio ed invia all'operatore le immagini riprese da una minuscola telecamera; le



immagini servono sia per sorvegliare l'ambiente in cui opera l'uccello-robot che per consentire all'operatore di controllarne il volo.

La società californiana realizza anche il più grande drone oggi operativo, il Global Observer GO-1 un velivolo con un'apertura alare di 53 metri in grado di volare ad un'altezza di 55.000-65.000 piedi per circa una settimana trasportando un payload (carico utile di sensori) del peso di 400 libbre (181 kg). In questo caso la tecnologia utilizzata è abbastanza inusuale: un motore endotermico brucia idrogeno liquido facendo girare una dinamo che produce elettricità, utilizzata per alimentare i quattro motori elettrici che fanno volare l'aereo. Secondo AeroVironment (che sta progettando la versione GO-2 con un'apertura alare di 77 metri ed un payload di 1.000 libbre), le fuel-cell attualmente disponibili non sono ancora affidabili per questo tipo di applicazione.

Ma, in questo campo, i veri padroni dei cieli restano il Global Hawk della Northrop Grumman ed il Predator della General Atomics che da soli coprono il 60% del mercato mondiale di UAV. Il primo, un velivolo a getto, è in grado di volare ad un'altezza di 60.000 piedi ad una velocità di 320 nodi per oltre 24 ore. Il velivolo pesa a pieno carico oltre 10 tonnellate con un payload di 1.360 kg ed un raggio di azione di oltre 22.000 km.

Il Predator (MQ-9 Reaper), nella versione più recente, è in grado di trasportare un payload di 340 kg, vola ad una quota di 50.000 piedi ed ha un'autonomia massima di 30 ore. Il carico può anche comprendere tre missili Hellfire a guida laser.

Tra le caratteristiche più importanti di un velivolo per la sorveglianza e la ricognizione c'è sicuramente l'autonomia che, con i sistemi tradizionali (motori alimentati con carburante), non può mai superare qualche giorno o arrivando al massimo ad una settimana. Per questo motivo sono allo studio dei velivoli con motore elettrico alimentati da pannelli fotovoltaici. In questo caso, infatti, se il consumo giornaliero di energia dei motori è inferiore a quello catturato dai pannelli durante le ore di luce, il velivolo può volare per un tempo illimitato. È quanto sta sperimentando la Qinetiq col suo Zephyr che ha volato l'anno scorso per ben 336 ore.



distanza da una stazione fissa o mobile.

Tutti i sistemi UAV sono composti da quattro componenti principali: il velivolo pilotato a distanza, i sensori presenti a bordo, il link radio di comunicazione con la stazione a terra e la Ground Station dove convergono le informazioni e da dove si controlla la missione del velivolo.

Il link di comunicazione può essere realizzato con sistemi radio basati a terra o con sistemi che sfruttano ponti radio satellitari, oppure con entrambe le tipologie. Allo stesso modo le stazioni terrestri (le cosiddette Ground Station) possono disporre di entrambe le funzionalità. Tipicamente un UAV posizionato in zona operativa viene fatto decollare ed atterrare da personale in loco mediante l'utilizzo di una Ground Station Mobile; successivamente il velivolo viene preso "in carico" da una stazione remota (distante anche migliaia di chilometri) mediante un link radio satellitare. Ad esempio, gli UAV italiani di stanza in Afghanistan vengono fatti decollare da personale presente in loco per poi essere seguiti e controllati durante la missione dal personale della base aerea di Amendola in Puglia.

Attualmente gli UAV svolgono prevalentemente compiti di sorveglianza del territorio; in campo militare questa sorveglianza è ovviamente destinata a controllare i movimenti di uomini e mezzi del nemico in modo da allertare le truppe che operano sul terreno. In campo civile gli UAV vengono utilizzati per il controllo dei confini terrestri e marittimi, per la lotta al contrabbando, al narcotraffico e all'immigrazione clandestina, in funzione antincendio ed anche per il monitoraggio meteo e



Etica e droni

Restare comodamente seduti sulla poltrona della propria postazione nella base vicino a casa, osservare cosa avviene a migliaia di chilometri di distanza, lanciare un missile ed uccidere decine di persone per poi la sera tornare a casa e andare a mangiare una pizza con i propri figli: è quanto accade ai militari americani addetti al controllo dei droni USA armati con missili Hellfire che sorvolano l'Iraq e l'Afghanistan. Questo nuovo scenario sta sollevando moltissimi interrogativi sulla legittimità dell'impiego di questi sistemi d'arma che, in futuro, potrebbero anche agire in maniera autonoma, scegliendo l'obiettivo da colpire in base a particolari algoritmi. Da molte parti si sostiene che tutto ciò violi le regole della guerra e che questo genere di operazioni

siano controproducenti dal momento che, come è successo più volte, le persone uccise non erano né militari avversari né terroristi. I fautori di queste tecnologie sostengono invece che questa è l'unica arma che gli USA hanno per combattere una guerra non dichiarata come quella dei terroristi islamici e che mai, in un confronto tra nazioni, questi mezzi verrebbero utilizzati al di fuori di casi di guerra dichiarata (ed ovviamente contro obiettivi militari). D'altra parte anche gli interventi con aerei tradizionali - in quelle regioni ma anche in altre zone - causano spesso dei "danni collaterali". Per quanto riguarda la possibilità che anche i droni italiani vengano dotati di missili, a livello delle gerarchie militari è in corso da tempo un dibattito, tutto sommato piuttosto inutile dal momento che mai la politica italiana accetterebbe di utilizzare mezzi di questo tipo.

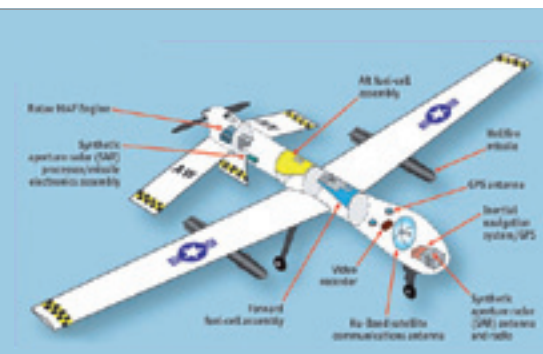
ambientale (recentemente hanno volato all'interno di un uragano per studiarne le caratteristiche e sopra la centrale nucleare di Fukushima per valutare i danni e misurare il livello di radioattività). Alcuni modelli di UAV per impiego militare sono stati dotati di armi (tipicamente missili a guida laser) in grado di colpire dall'alto il nemico.

Questi mezzi possono volare silenziosi e pazienti per ore e giorni sopra la zona sospetta e colpire fulmineamente il nemico non appena questo compare. A differenza dei missili da crociera che sono adatti a colpire bersagli fissi, gli UAV sono l'ideale per colpire bersagli in movimento o che si muovono in aree urbane, come i terroristi islamici, o in zone sperdute, come i guerriglieri e i narcos.

Gli UAV hanno dimensioni, pesi e funzionalità molto diverse a seconda dell'impiego cui sono destinati. I modelli più piccoli vengono utilizzati sui campi di battaglia per controllare le linee nemiche o zone nascoste (tipicamente per vedere cosa c'è "oltre la collina"). I più diffu-

si sono quelli dell'americana AeroVironment che attualmente dispone di tre modelli lanciabili a mano: Wasp, Raven e Puma. Il raggio di azione è compreso tra 5 e 15 km e l'autonomia varia tra mezz'ora e un paio d'ore. I velivoli vengono controllati con una Ground Station portatile e sono in grado di compiere anche delle missioni autonome. Le immagini riprese dalla telecamera di bordo (funzionante anche al buio) vengono visualizzate sullo schermo della Ground Station. Alla stessa categoria appartengono anche alcuni sistemi sviluppati nel nostro paese che però non hanno avuto lo stesso successo commerciale. Ci riferiamo allo Strix sviluppato e costruito da Alpi Aviation e che viene lanciato mediante una piccola catapulta. A questo proposito dobbiamo osservare come il nostro paese sia sempre stato all'avanguardia in questo settore anche se, come nel caso dello Strix, gli sviluppi commerciali non sono mai stati così eclatanti. Per quanto riguarda i velivoli più grandi, attualmente l'unico UAV operativo è il Falco, prodotto da Selex Galileo, che è stato venduto in circa 25 esemplari al Pakistan mentre Alenia Aeronautica sta sviluppando le piattaforme Sky-X, Sky-Y e Molynx. Le nostre aziende, ed in particolare Alenia, dopo aver visto sfumare l'accordo con il consorzio anglo-francese per lo sviluppo di un nuovo UAV di tipo MALE (medium-altitude, long-endurance) hanno stretto un accordo con gli israeliani della IAI e probabilmente entreranno nel consorzio guidato da EADS per lo sviluppo del Talarion.

Per quanto riguarda le nostre Forze Armate, è in corso un rafforzamento della struttura esistente: recentemente sono sta-



Anche ad **ala rotante**

Sotto vari aspetti, lo sviluppo di un velivolo autonomo ad ala rotante è più semplice di un velivolo ad ala fissa. Infatti un elicottero (manned o unmanned che sia) può restare fermo in aria, può muoversi orizzontalmente e non ha bisogno di una pista d'atterraggio. D'altra parte pilotarli e stabilizzarne il volo è molto più complesso, ma a questo sopperisce l'elettronica che da questo punto di vista ha fatto passi da gigante. Anche in questo settore, dunque, sono numerosi i progetti in corso. Si tratta di velivoli completamente nuovi ma anche di versioni unmanned di elicotteri già esistenti. Tra l'altro l'assenza dei piloti consente di aumentare il carico utile (l'elettronica di controllo pesa meno dei piloti). I tre progetti più avanzati in questo campo sono il Fire Scout della Northrop Grumman, l'A160T della Boeing ed il K-Max della Lockheed Martin: i primi due sono velivoli completamente nuovi mentre il terzo è un'elaborazione di un velivolo commerciale della Kaman.

Il Fire Scout nella versione MQ-8B è attualmente operativo sia nell'esercito che nella marina USA dove

viene impiegato in missioni anti-sommergibile, per il puntamento di precisione e per lo sminamento. Il velivolo è in grado di atterrare in maniera completamente autonoma anche sui ponti delle navi in movimento e

può essere dotato di missili (i soliti, micidiali, Hellfire).

Il Boeing A160 Hummingbird è un elicottero ancora in fase di sviluppo col supporto del DARPA americano. L'obiettivo finale è quello di realizzare un velivolo autonomo con un raggio di azione di 4.000 km, un'autonomia di 24 ore, la possibilità di raggiungere una quota di 30.000 piedi e una capacità di carico di quasi 3 tonnellate.

Caratteristiche simili ha anche il K-Max della Lockheed Martin/Kaman, destinato prevalentemente al trasporto di rifornimenti in zone di guerra.

In questo settore sono numerosi i progetti, sia militari che civili. Anche nel nostro paese l'Augusta Westland ha recentemente annunciato l'intenzione di realizzare un RUAV (Rotorcraft Unmanned Aerial Vehicle) partendo dalla struttura del monomotore a turbina PZL SW 4.



ti ordinati sei Predator A+ e sei MQ-9 Reaper dalla General Atomics unitamente a tre Ground Station, oltre ai quattro Shadow 200 dalla AAI. L'Italia ha iniziato ad utilizzare i Predator RQ-1A in Iraq nel 2004 e successivamente in Afghanistan dove sono stati utilizzati in supporto alle truppe di terra essenzialmente in funzione anti IED (Improvised Explosive Device), controllando

dall'alto le strade percorse dai convogli militari. Recentemente i velivoli italiani sono stati utilizzati anche in Libia, unitamente a quelli di altri paesi della Nato tra cui i Global Hawk americani di base a Sigonella, in Sicilia. Sono proprio il Global Hawk della Northrop Grumman ed il Predator della General Atomics i veri padroni dei cieli: da soli questi due modelli coprono quasi il 60% del mercato mondiale. Il primo, un velivolo a getto, è in grado di volare ad un'altezza di 60.000 piedi e ad una velocità di 320 nodi per oltre 24 ore. Il velivolo pesa a pieno carico oltre 10 tonnellate, con un payload di 1.360 kg ed un raggio di azione di oltre 22.000 km.

Il Predator (MQ-9 Reaper), nella versione più recente, è in grado di trasportare un payload di 340 kg, vola ad una quota di 50.000 piedi ed ha un'autonomia massima di 30 ore. Il payload può anche essere rappresentato da tre missili Hellfire a guida laser. Per quanto riguarda i costi, siamo su livelli decisamente importanti: un set di quattro Predator con Ground Station e link satellitare costa oltre 150 milioni di dollari; di poco inferiore è il costo del set di quattro Shadow



200 e delle relative Ground Station: appena (si fa per dire) 68 milioni di dollari.

Tra i velivoli con maggior autonomia, indicati per le lunghe missioni di sorveglianza dall'alto, troviamo gli UAV con le dimensioni maggiori,

simili a quelle di un piccolo aereo di linea.

Attualmente il drone più grande è il Global Observer GO-1 della Aeri-Viromental, un velivolo con un'apertura alare di 53 metri in grado di volare ad un'altezza di 55.000-65.000 piedi per circa una settimana trasportando un payload (carico utile di sensori) del peso di 400 libbre (181 kg). In questo caso la tecnologia utilizzata è abbastanza inusuale: un motore endotermico brucia idrogeno liquido facendo girare una dinamo che produce elettricità utilizzata per alimentare quattro motori elettrici che fanno volare l'aereo. Secondo AeroViromental (che sta progettando la versione GO-2 con un'apertura alare di 77 metri ed un payload di 1.000 libbre), le fuel-cell attualmente disponibili non sono ancora affidabili per questo tipo di applicazione. Un altro velivolo della stessa categoria è l'Orion dell'Aurora Flight Sciences, un UAV in grado di volare per 120 ore ad un'altitudine di 20.000 piedi trasportando un payload di 1.000 libbre.

Tra le caratteristiche più importanti di un velivolo per la sorveglianza e la ricognizione c'è sicuramente l'autonomia che,

L'Italia è stato uno dei primi paesi occidentali a dotarsi di velivoli unmanned in grado di operare a distanza con link satellitari ed anche uno dei primi paesi a costruire degli UAV con questa capacità, purtroppo con scarso successo dal punto di vista commerciale. Attualmente l'unico velivolo operativo è il Falco, prodotto da Selex Galileo, che

2004 e successivamente in Afghanistan dove sono stati utilizzati in supporto alle truppe di terra essenzialmente in funzione anti IED (Improvised Explosive Device), controllando dall'alto le strade percorse dai convogli militari. Recentemente i velivoli italiani sono stati uti-

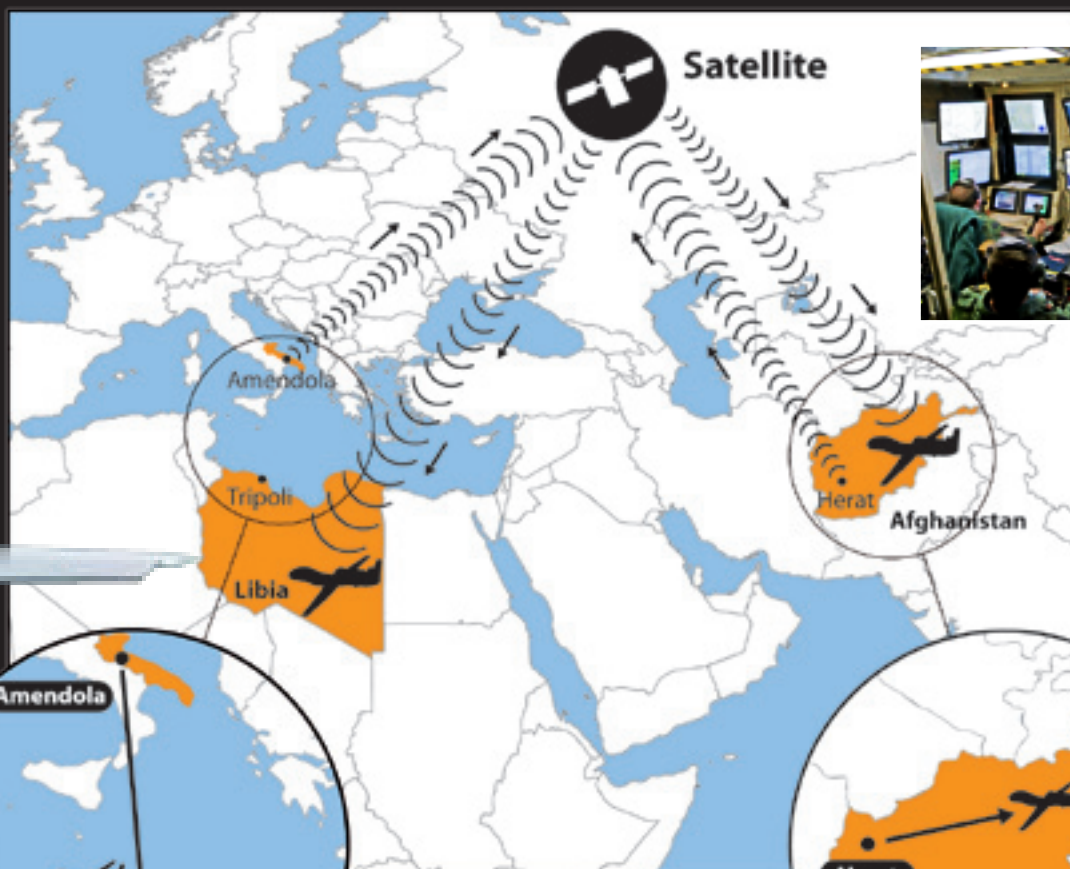
due Predator mentre tutti gli altri velivoli sono dislocati nella base di Amendola. Tipicamente i velivoli presenti in Afghanistan vengono controllati durante il decollo e l'atterraggio dalla Ground Station locale per poi essere "presi in carico" dalla base

è stato venduto in circa 25 esemplari al Pakistan mentre Alenia Aeronautica sta sviluppando le piattaforme Sky-X, Sky-Y e Molynx. Per quanto riguarda le nostre Forze Armate, è in corso un rafforzamento della struttura esistente: recentemente sono stati ordinati sei Predator A+ e sei MQ-9 Reaper dalla General Atomics unitamente a tre Ground Station oltre a quattro Shadow 200 dalla AAI. Attualmente la piattaforma della General Atomics è la più diffusa al mondo e quella che ha accumulato il maggior numero di ore di volo. L'Italia ha iniziato ad utilizzare i Predator RQ-1A in Iraq nel

lizzati anche in Libia, unitamente a quelli di altri paesi della Nato tra cui i Global Hawk americani di base a Sigonella, in Sicilia. I nostri UAV non sono armati ed hanno esclusivamente compiti di sorveglianza. Inizialmente la Ground Station di controllo si trovava solo in loco (Herat) mentre attualmente la base che coordina le attività di tutti i nostri velivoli unmanned si trova ad Amendola, in Puglia, e fa capo al 28° Gruppo, 32° Stormo. In Afghanistan sono schierati

di Amendola che, tramite link satellitare, gestisce le missioni. Per la Libia, invece, i nostri UAV partono direttamente da Amendola. Attualmente i nostri velivoli sono dotati di sensori ottici e ad infrarossi ma sui nuovi Predator verranno installati sistemi Lynx1 mentre i Reaper verranno dotati di sensori Lynx2. Si tratta di radar ad apertura sintetica o SAR (Synthetic Aperture Radar) in grado di fornire immagini in alta risoluzione anche attraverso le nuvole e le perturba-





zioni. Insomma, quello che si definisce un sistema all-weather, in grado di funzionare in ogni situazione ambientale. L'arrivo dei nuovi mezzi ha reso necessario il potenziamento del personale della base; infatti per condurre una missione sono necessarie 4 persone: un pilota, un addetto ai sensori che aziona le telecamere ed i sensori, un addetto al controllo ed allo smistamento delle immagini ed un ingegnere di volo che controlla, tramite la telemetria, tutti i parametri del velivolo. Un quinto addetto sarà necessario quando gli UAV verranno dotati dei sistemi

radar ad apertura sintetica. Una missione di 20 ore richiede 4-5 equipaggi: in totale fanno almeno 20 persone. Qualcuno potrebbe dire: alla faccia dell'unmanned! Recentemente il nostro paese ha anche acquistato dalla AAI per 68 milioni di dollari quattro Shadow 200 con sensore elettro-ottico a raggi infrarossi e funzionalità laser, un sistema di lancio, una stazione di controllo a terra ed un'altra portatile, un terminale video remoto e un datalink per video e dati (con due terminali). Le nostre Forze Armate dispongono anche di altri UAV da utilizzare sul campo di battaglia per la sorveglianza delle linee nemiche o di zone nascoste: si tratta di piccoli velivoli lanciati a mano,

come i Raven o i Maveric, oppure mediante una piccola catapulta come gli Strix. Tutti mezzi che sono ancora in fase di valutazione e disponibili in pochi esemplari.





con i sistemi tradizionali (motori alimentati con carburante), non può mai superare qualche giorno o al massimo una settimana. Per questo motivo sono allo

studio dei velivoli con motore elettrico alimentati da pannelli fotovoltaici. In questo caso, infatti, se il consumo giornaliero di energia dei motori è inferiore a quello catturato dai pannelli durante le ore di luce, il velivolo può volare per un tempo illimitato. È quanto sta sperimentando la Qinetiq col suo Zephyr che ha volato l'anno scorso per ben 336 ore. Si tratta ancora di velivoli molto fragili, in grado di trasportare un payload di qualche chilogrammo, ma questa è una delle strade per disporre di sistemi di sorveglianza dall'autonomia illimitata.

La vera novità nel campo dei sistemi unmanned è rappresentata dagli UCAV (Unmanned Combat Air Vehicle), velivoli da combattimento senza pilota a bordo che prenderanno il posto degli attuali cacciabombardieri. E come gli attuali caccia potranno essere dislocati sulle

portaerei ed avere capacità Stealth, ovvero essere invisibili ai radar. Attualmente sono in fase di sviluppo numerosi progetti tra i quali il Phantom Ray della Boeing e l'X-47 della Northrop Grumman. Entrambi questi UAV hanno iniziato i test di volo i primi mesi di quest'anno; in particolare l'X-47B, dopo le prove alla base di Edwards che continueranno sino alla fine di quest'anno, inizierà una seconda campagna di test presso la Naval Air Station Patuxent River finalizzata ad accertarne il grado di prontezza e la resistenza in ambiente marittimo, in vista del suo schieramento a bordo di una portaerei, previsto per il 2013. In Europa è stato presentato recentemente dalla RAF inglese il Taranis che dovrebbe effettuare le prove di volo entro la fine del 2011. Anche in questo caso si tratta di un velivolo Stealth che dovrebbe entrare in servizio tra il 2018 e il 2025 andando a sostituire velivoli quali l'Harrier e il Tornado.

Un altro UCAV in fase di svi-

luppo è il Neuron della Dassault alla cui realizzazione collabora anche l'italiana Alenia Aeronautica.

Da tutti questi velivoli (in gran parte sperimentali) dovrebbero derivare i caccia senza pilota della sesta generazione che, a partire dal 2030-2035, prenderanno *in toto* il posto degli attuali velivoli con pilota a bordo. ■



REGOLATORE DI CARICA PER AUSSIE ZERO

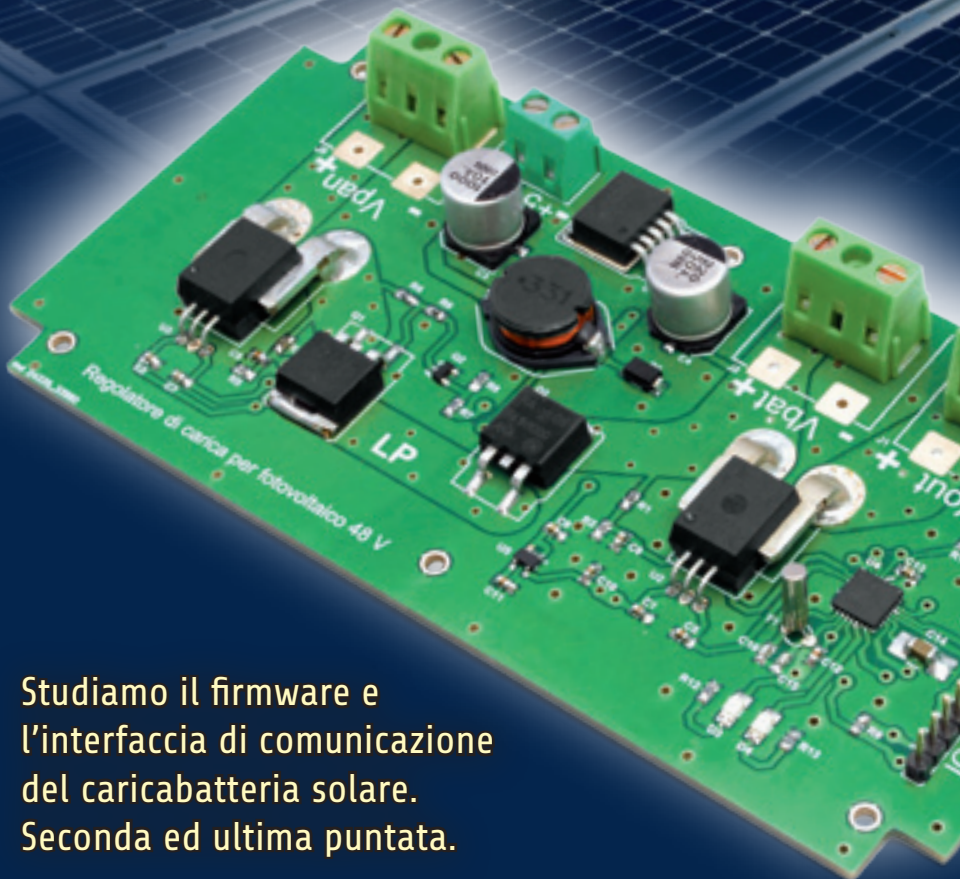
..... • dell'Ing. LUCA PASCARELLA



Siamo giunti alla conclusione della descrizione del caricabatteria ad alimentazione solare che abbiamo realizzato e pubblicato a partire dal fascicolo di Luglio/Agosto scorsi; in queste pagine descriveremo sostanzialmente il firmware che gira nel microcontrollore PIC24FJ64GA102.

Ma prima di farlo è opportuno riepilogare le caratteristiche e le finalità del progetto: si tratta di un caricabatteria basato su un regolatore di carica per pannelli fotovoltaici (quattro, per l'esattezza...) in grado di caricare quattro accumulatori da 12 volt/60 Ah in serie, utilizzando una tecnica innovativa basata

Studiamo il firmware e l'interfaccia di comunicazione del caricabatteria solare. Seconda ed ultima puntata.



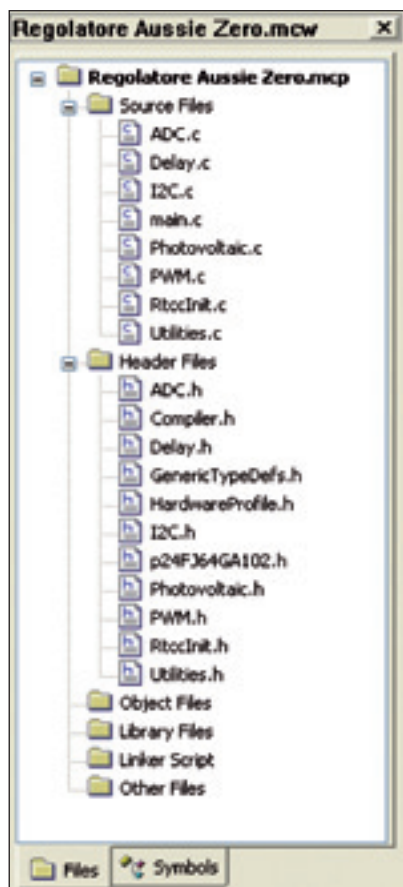


Fig. 1 - Lista dei file impiegati per la realizzazione del firmware del regolatore di carica per il veicolo Aussie Zero, in ambiente MPLAB-IDE.

sull'MPPT. Questa metodica si basa sulla ricerca del punto di lavoro ottimale istantaneo dei pannelli solari, ovvero la condizione per la quale questi ultimi cedono agli accumulatori la massima potenza possibile; la ricerca viene effettuata rilevando la tensione erogata e valutando la corrente uscente dai pannelli e inviata alle batterie. In ogni istante e per ogni condizione di insolazione, si leggono tensione e corrente e si fa in modo che il loro prodotto sia sempre il massimo; ciò è possibile perché il regolatore di carica limita la corrente erogata se si accorge che determina un eccessivo calo della tensione di uscita dei pannelli solari, ovvero la lascia crescere

se da ciò deriva un incremento della potenza. Tutto ciò, al fine di ottenere il miglior rendimento. In pratica è come se il nostro regolatore adattasse la resistenza di uscita del generatore elettrico equivalente alla serie di pannelli, a quella del carico (le batterie). La modulazione della corrente e quindi della potenza ceduta in fase di carica si ottiene adottando la tecnica PWM, ossia alimentando il gruppo di accumulatori con una forma d'onda rettangolare della quale si varia la larghezza degli impulsi. Ma vediamo come questa metodica viene espletata dal microcontrollore che governa il circuito, analizzando il programma che gira nel PIC; infatti un progetto nel quale si fa uso di un microcontrollore programmabile non si può considerare totalmente descritto senza spiegare il funzionamento del firmware fornito a corredo.

IL FIRMWARE

Avendo impiegato un microcontrollore PIC, abbiamo scelto come

ambiente di sviluppo il noto IDE della Microchip MPLAB IDE e come compilatore il C30 in versione Lite, normalmente reperibile dal sito www.microchip.com. I PIC 24F sono compatibili con la maggior parte dei programmatori e debugger disponibili sul mercato; nel nostro caso è stato impiegato l'ICD3, che è molto pratico e veloce.

Il software sviluppato non è particolarmente complicato, infatti il PIC non esegue le istruzioni alla massima velocità, ma impiega una frequenza di clock di soli 8 MHz, quindi avanzerà a 4 MIPS, riducendo i consumi energetici a soli 4 mA; ciò centra il primo obiettivo di un circuito alimentato a batterie: ridurre al massimo il consumo.

Ora passiamo alla descrizione del firmware ottenuto dalla compilazione dei file rappresentati nella Fig. 1. Come di consueto, le prime operazioni da effettuare allo start-up del PIC sono l'inizializzazione della frequenza di clock, dei timer di conteggio, delle linee di I/O e delle periferiche har-

Output Function Number ⁽¹⁾	Function	Output Name
0	NULL ⁽²⁾	Null
1	C1OUT	Comparator 1 Output
2	C2OUT	Comparator 2 Output
3	U1TX	UART1 Transmit
4	U1RTS ⁽³⁾	UART1 Request To Send
5	U2TX	UART2 Transmit
6	U2RTS ⁽³⁾	UART2 Request To Send
7	SDO1	SPI1 Data Output
8	SCK1OUT	SPI1 Clock Output
9	SS1OUT	SPI1 Slave Select Output
10	SDO2	SPI2 Data Output
11	SCK2OUT	SPI2 Clock Output
12	SS2OUT	SPI2 Slave Select Output
18	OC1	Output Compare 1
19	OC2	Output Compare 2
20	OC3	Output Compare 3
21	OC4	Output Compare 4
22	OC5	Output Compare 5
23-28	(unused)	NC
29	CTPLS	CTMU Output Pulse
30	C3OUT	Comparator 3 Output
31	(unused)	NC

Fig. 5 - Schema riassuntivo dei valori associabili ai registri RPNR_x per assegnarli ad uno specifico pin d'uscita una data periferica hardware.

dware. Le prime tre operazioni vengono effettuate dalla funzione **PicInit()**. Terminata questa fase preliminare, si passa all'inizializzazione delle singole periferiche; in particolare, il PIC invocherà in sequenza le seguenti chiamate di funzione: **InitADC()**, **InitPWM()**, **EnableSecOsc()**, **RtccInit()** e **InitValues()**. Per ognuna di queste chiamate, l'obiettivo sarà quello di inizializzare la periferica associata; i nomi dati alle funzioni lasciano intuire facilmente a quale periferica si fa riferimento.

Nel **Listato 1**, è riportata una porzione del codice impiegato nella funzione **InitPWM()**, che consente di assegnare l'output del PWM ad un generico pin rimappabile. Come si può notare nel listato, si è fatto uso di due sezioni di codice scritte in assembler che vengono impiegate, rispettivamente, per sbloccare e bloccare il bit di **IOLOCK**, utilizzato per evitare accidentali modifiche della rimappatura software. L'assegnazione dell'output del PWM avviene in un'unica istruzione, ovvero in **RPOR6bits.RP12R = 18**; l'obiettivo di questa istruzione è assegnare al pin rimappabile individuato dalla sigla **RP12** la periferica d'uscita numero 18, che corrisponde ad **OC1** (Output Compar 1, ovvero il PWM 1). L'associazione della periferica con un specifico valore decimale avviene seguendo lo schema riportato nella **Fig. 5**. Una situazione analoga si verifica nel caso in cui il pin della periferica è un pin d'ingresso; in tale evenienza si fa riferimento ai registri denominati **RPINRx**. Terminata l'inizializzazione, il programma entra in un banale loop infinito (presente nel file *main.c*) nel quale vengono monitorati ad intervalli scanditi dal contatore 1 (impostato a 25 Hz) i valori della tensione del pan-

nello e della batteria, allo scopo di poter intraprendere la giusta azione di controllo. Tale azione consiste nel regolare il duty-cycle del PWM in modo da seguire il profilo di carica ottimale per la ricarica della batteria, evitando sovratensioni e sovraccarichi, ma anche la solfatazione della batteria.

Il programma presenta una struttura molto snella e compatta, grazie all'uso intensivo degli

interrupt, che consentono una programmazione multitasking tale da desincronizzare le diverse operazioni di monitoraggio. Infatti nel software sono previste ben cinque routine di gestione dell'interrupt, di cui tre sono dedicate alla gestione dei Timer (impiegati per la segnalazione dei LED, dei 25 Hz precedentemente nominati e del calcolo della potenza media) mentre i restanti due servono per gestire la con-



300 Smart elettriche per la California

Gli americani si sono sempre distinti per fare le cose in grande; non si smentiscono in occasione del lancio di un'iniziativa, a San Diego (California) di car sharing con auto elettriche. L'operazione, realizzata in cooperazione con car2go (www.car2go.com, società sussidiaria del gruppo tedesco Daimler) vedrà in campo entro la fine dell'anno 300 vetture Smart fortwo ED a batteria, con motore da 30 kW (41 CV) e autonomia di 135 km, estesa dalla presenza di pannelli fotovoltaici sul tetto che danno anche un supplemento di corrente per azionare il climatizzatore. Ritiro e riconsegna dell'auto si possono fare in una qualsiasi delle piazzole di carica sparse per le città, grazie anche al fatto che tutte le Smart usate sono geolocalizzate mediante GPS e quindi si può conoscere l'ubicazione di quella più vicina anche via cellulare. car2go non è nuova a queste iniziative: ha esordito ad Ulm (Germania) nel 2008, poi ad Amburgo, Vancouver (Canada) ed Austin (Texas). Entro fine 2011, il progetto sbarcherà ad Amsterdam (Olanda). Il noleggio si paga a minuti: in Germania costa da 24 a 29 centesimi di euro, ad Austin e Vancouver 0,35 dollari (rispettivamente, americano e canadese).

versione analogico/numerica e le chiamate dalla periferica I²C per gli eventuali comandi impartiti da remoto.

CONTROLLO REMOTO

Il nostro regolatore di carica non è solo un modulo stand alone, ma può essere interfacciato a sistemi più complessi, funzionando da periferica intelligente e controllabile tramite apposito set di comandi da impartire mediante interfaccia I²C. L'interfaccia Inter Integrated Circuit è stata scelta per la maggiore flessibilità ottenibile rispetto ad altri link di comunicazione standard, come possono essere l'SPI o l'USART. Infatti l'interfaccia di comunicazione I²C consente al Master di selezionare il dispositivo Slave tramite indirizzamento software; impiegando due sole linee di comunicazione è possibile comunicare con più moduli periferici, le cui limitazioni sono imposte solo dall'indirizzamento software (7 o 10 bit). Nel caso dell'SPI ed ancor peggio nell'USART, il Master deve riservare uno o più pin di I/O per ogni dispositivo Slave indirizzabile. Questo inconveniente impone una limitazione

Listato 1

```
// Assignment of the PWM output to the specified pin
// Disable lock
asm volatile ( "MOV #OSCCON, w1 \n"
               "MOV #0x46, w2 \n"
               "MOV #0x57, w3 \n"
               "MOV.b w2, [w1] \n"
               "MOV.b w3, [w1] \n"
               "BCLR OSCCON,#6");

RPOR6bits.RP12R = 18;

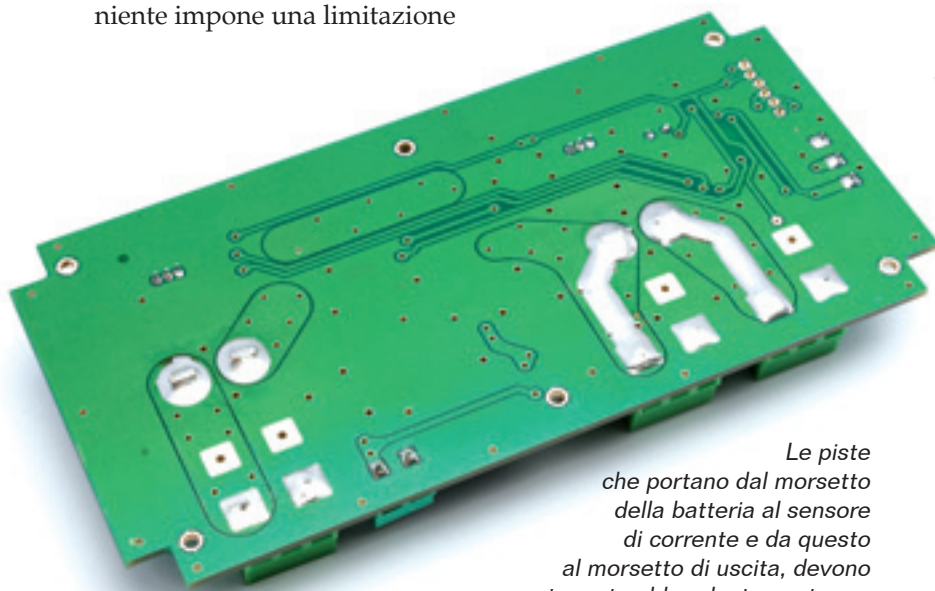
// Enable lock
asm volatile ( "MOV #OSCCON, w1 \n"
               "MOV #0x46, w2 \n"
               "MOV #0x57, w3 \n"
               "MOV.b w2, [w1] \n"
               "MOV.b w3, [w1] \n"
               "BSET OSCCON, #6");
```

pratica nel caso in cui bisogna cablare delle linee dati ed implica di conoscere il numero di dispositivi Slave indirizzabili in fase di progettazione, limitando future espandibilità.

A parte queste considerazioni, diciamo che lo scopo della connessione dati è permettere ad un computer di bordo, che svilupperemo prossimamente, di interrogare il regolatore di carica per dedurne informazioni sul tempo occorrente alla piena ricarica delle batterie, sulla corrente prelevata da queste ultime durante la trazione (grazie al fatto che sul circuito descritto in

questo articolo si trova un rilevatore di corrente posto sul ramo che collega le batterie ai motori) ed altro ancora.

Per il momento si tratta di un hardware in fase di progetto, ma che lavorerà a stretto contatto con il regolatore di carica qui descritto. Apportando modifiche al firmware originale, il nostro caricabatterie per pannelli fotovoltaici potrà anche essere comandato via I²C Bus in modo da variare determinati parametri di carica ed eseguire altri compiti: avete l'interfaccia, quindi il limite è dato solo dalla vostra inventiva e capacità di programmare. ■



Le piste che portano dal morsetto della batteria al sensore di corrente e da questo al morsetto di uscita, devono essere stagnate abbondantemente per ridurre la resistenza.



per il MATERIALE

Tutti i componenti utilizzati in questo circuito sono facilmente reperibili in commercio. Il master del circuito stampato così come il firmware utilizzato per programmare il microcontrollore PIC24FJ64GA102 sono scaricabili gratuitamente dal sito della rivista (www.elettronica.in.it). Ricordiamo che nei fascicoli 156 e 157 è stato presentato il circuito del ricaricatore da rete per le batterie del nostro progetto Aussie Zero.

Per fare un po' di pratica realizziamo un Data Logger su SD-Card usando l'ambiente di sviluppo Libero IDE. Seconda parte.



FPGA SMART FUSION

di MARIANO SEVERI

Nella precedente puntata abbiamo introdotto le caratteristiche generali dei dispositivi SmartFusion, esponendo funzioni ed architettura delle tre diverse sezioni: quella basata sul microcontrollore Cortex-M3, quella analogica e la FPGA Fabric. In questa seconda parte, quale esempio di un'applicazione reale, vedremo come realizzare un semplice data logger interamente basato sui dispositivi SmartFusion. Il sistema sarà in grado di

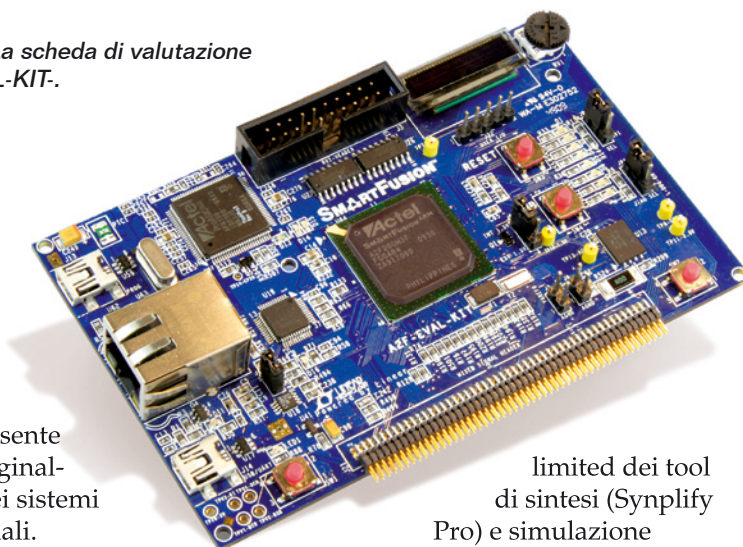
acquisire periodicamente un canale di corrente, uno di tensione ed uno di temperatura e di salvare i dati su di una memoria SD utilizzando un file system standard. Come dimostrazione, useremo la scheda di valutazione per FPGA SmartFusion A2F-EVAL-KIT (Fig. 1).

La Fig. 2 mostra lo schema di principio di un tale data logger, i cui moduli principali sono il processore Cortex-M3, il circuito di condizionamento ed ac-

quisizione dei canali analogici di ingresso, il controller di memoria SD. Il processore incorpora le seguenti periferiche: il controller di memoria embedded non volatile e SRAM per la memorizzazione e l'esecuzione dell'applicativo, il timer per la temporizzazione dell'acquisizione dei canali, il controller UART per la comunicazione su linea seriale a scopo di debug. Il primo modulo sarà implementato configurando opportunamente la sezione MSS della FPGA SmartFusion, mentre il secondo si basa sulle funzionalità dei circuiti AFE ed ACE della stessa. Il controller di memoria SD sarà invece realizzato come periferica custom all'interno della FPGA Fabric e sarà accessibile dal processore Cortex-M3 mediante la porta APB3 master della matrice di interconnessione AMBA. Lo sviluppo del nostro esempio copre quindi, come si può vedere, tutti gli aspetti relativi alla progettazione con FPGA SmartFusion.

Prima di iniziare con gli aspetti pratici, vogliamo chiarire un concetto: come molti di voi sapranno, esistono in commercio diversi microprocessori embedded, a costi anche inferiori a quelli delle FPGA SmartFusion, che dispongono di periferiche del tipo indicato e che quindi possono allo stesso modo essere utilizzati per applicazioni simili. Il vantaggio reale delle FPGA SmartFusion rispetto a queste soluzioni sta nella possibilità di utilizzare la FPGA Fabric per personalizzare alcune interfacce e funzioni o, anche, per accelerare queste ultime in hardware. In questi termini, le FPGA Smart Fusion rappresentano una piattaforma a microprocessore completamente riprogrammabile, non soltanto dal punto di vista software ma anche da quello hardware, caratteristica, quest'ul-

Fig. 1 - La scheda di valutazione A2F-EVAL-KIT.



tima, presente solo marginalmente nei sistemi tradizionali.

INSTALLAZIONE DELL'AMBIENTE DI SVILUPPO

Lo sviluppo di un progetto con FPGA SmartFusion è supportato dai due seguenti ambienti:

- Libero IDE, per la realizzazione dell'architettura hardware del sistema e la programmazione del dispositivo;
- SoftConsole, per la compilazione, il download ed il debug dell'applicativo software.

Le versioni più recenti dei due software possono essere scaricati gratuitamente dal sito web, a partire dall'indirizzo riportato nei **Riferimenti** al punto [1]. Il presente esempio è stato sviluppato con le versioni 9.0 SP3A di Libero e 3.2 di SoftConsole.

Libero richiede l'installazione dapprima della distribuzione base e, successivamente, dell'ultimo aggiornamento disponibile. Durante l'installazione della distribuzione base, ci verrà richiesto di selezionare i componenti che intendiamo installare: confermiamo la configurazione predefinita, che installerà l'IDE unitamente alle versioni vendor-

limited dei tool di sintesi (Synplify Pro) e simulazione (Modelsim); poi sotto la voce *Modelsim* selezioniamo almeno la famiglia SmartFusion per consentire l'installazione della relativa libreria. Eventualmente possiamo aggiungere alla configurazione predefinita per l'installazione il tool Identify (spuntando la relativa casella), un ambiente di debug in-circuit del codice HDL; nel nostro esempio non faremo comunque uso di tale tool. Ci verrà successivamente richiesto se intendiamo usare il programmatore Flash Pro o Flash Pro Lite su porta parallela: rispondiamo *no*; conatteremo la scheda di sviluppo A2F-EVAL-KIT mediante cavo USB ed useremo il programmatore low-cost disponibile a bordo.

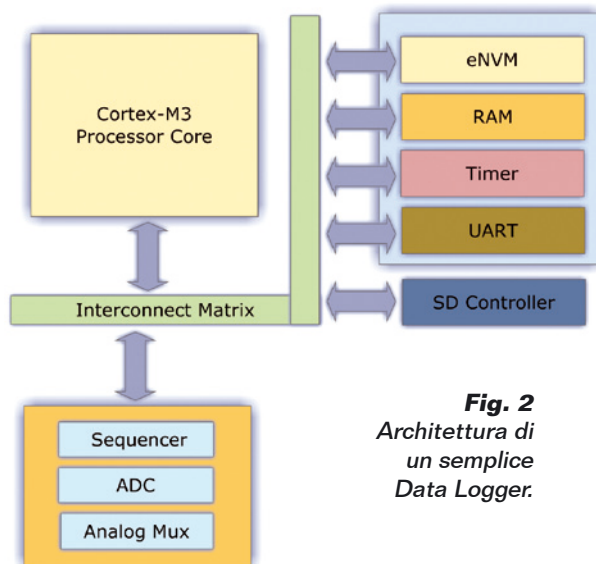
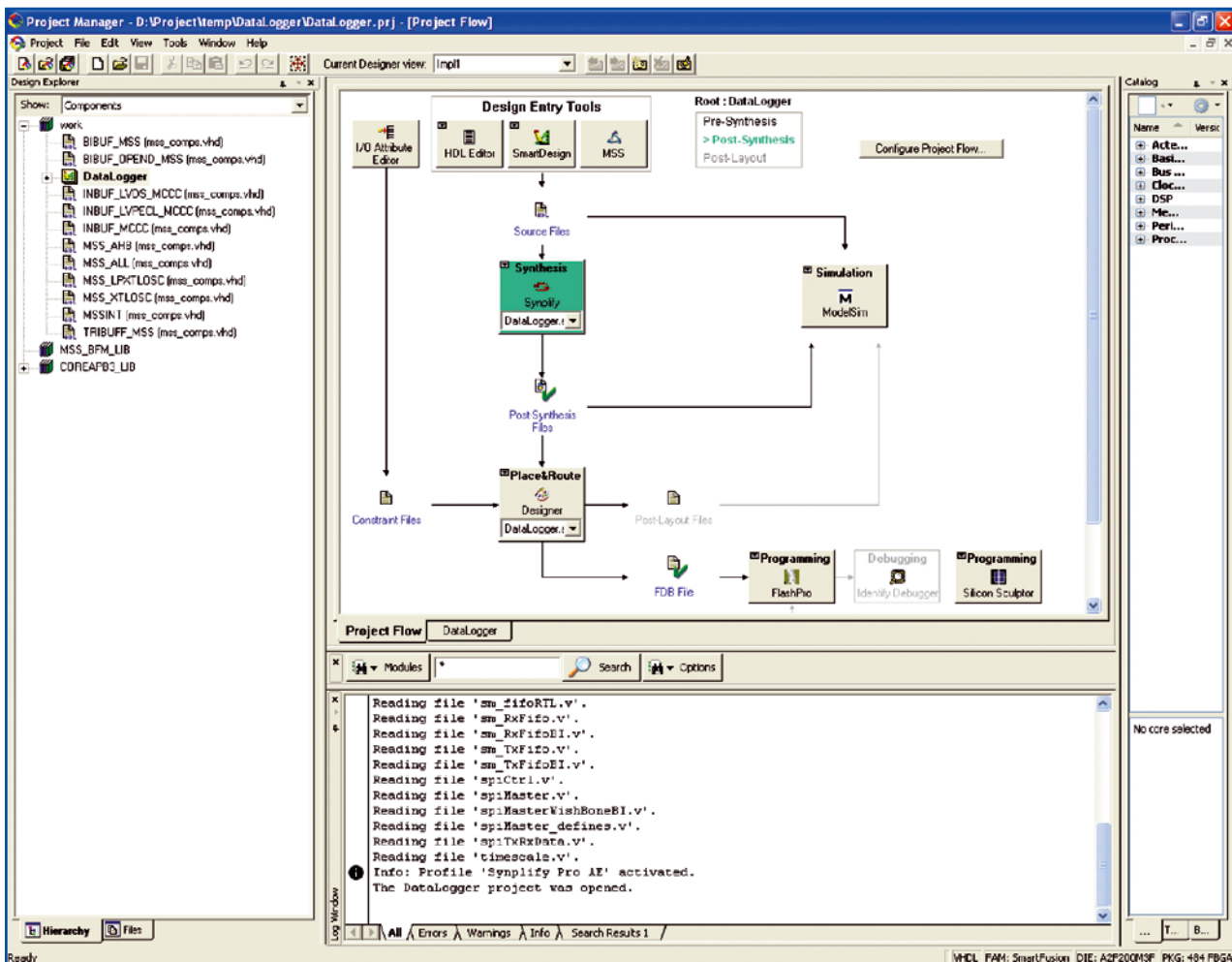


Fig. 2 Architettura di un semplice Data Logger.

Fig. 3 - L'ambiente di progetto Libero IDE.



Al termine dell'installazione ci verrà richiesto se abbiamo una licenza valida; se non è questo il caso, selezioniamo la voce che rimanda automaticamente al portale web Actel [3]. Registriamoci come nuovo utente e quindi richiediamo una licenza Libero di tipo Gold; tale licenza è gratuita, ha validità di un anno ed è bloccata sull' ID del nostro PC, ovvero sul volume dell'hard-disk o sull'indirizzo fisico della scheda di rete. La licenza ci sarà spedita via e-mail all'indirizzo indicato in fase di registrazione. Salviamo il file di licenza sul disco rigido, ad esempio nel percorso `c:\flexlm\license.dat`. Infine, definiamo la variabile di ambiente `LM_LICENSE_FILE` assegnandole tale percorso; qualora la variabile sia già definita, aggiungiamo

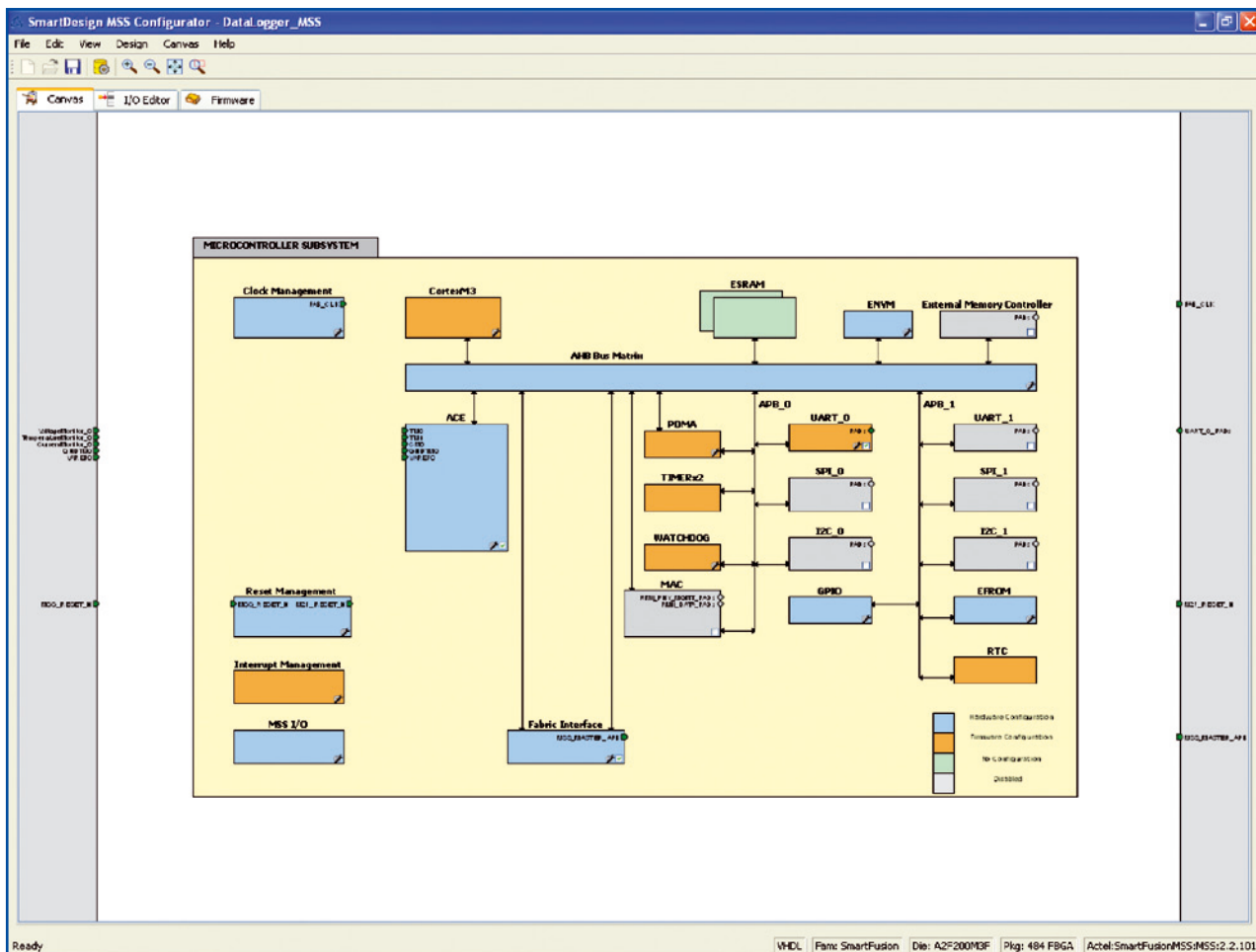
il percorso suddetto alla sua definizione. Se dovete incontrare problemi durante l'installazione della licenza, potete consultare le informazioni disponibili direttamente sul portale Actel. Durante l'installazione di Soft Console, invece, vi verrà richiesto quali tool-chain intendete installare; tra esse è disponibile quella per il processore ARM Cortex-M3, necessaria a sviluppare per FPGA Smart Fusion. Oltre ad essa, sono incluse tool-chain per implementare il tutto in dispositivi FPGA Actel dei processori Cortex-M1, ARM-7 ed 8051.

CREAZIONE DEL PROGETTO HARDWARE IN LIBERO IDE

Dopo aver correttamente installato l'ambiente di sviluppo, creiamo il progetto hardware del

nostro sistema. Lanciamo quindi l'ambiente Libero IDE (Fig. 3). A tale proposito segnaliamo che in alcune installazioni abbiamo notato apparire ripetutamente una finestra di errore "SmartHep Library : MEM_BAD_POINTER" durante l'apertura del software o l'esecuzione di certe funzioni. L'origine dell'errore non è ben chiara ma comunque non ha effetti sul funzionamento dell'ambiente di sviluppo. Selezioniamo quindi il pulsante OK ogni volta che apparirà la finestra. Dalla barra dei menu di Libero, selezioniamo Project->New Project; facendo ciò si aprirà una finestra per la creazione guidata del progetto. Inseriamo dapprima il nome del progetto (nel nostro caso *DataLogger*) nel campo

Fig. 4 - L'ambiente grafico MSS Configurator per la configurazione della sezione analogica e di quella a micro-controllore dei dispositivi SmartFusion.



Project Name e quindi selezioniamo, nella casella *Project Location*, la cartella di destinazione. Confermiamo la selezione e passiamo alla finestra successiva, che consente di selezionare il dispositivo FPGA da utilizzare. Nel nostro caso, volendo impiegare la scheda di sviluppo A2F-EVAL-KIT, selezioniamo il die A2F200M3F con package 484 FBGA. Scorriamo, quindi, senza apportare modifiche, i due successivi passi del wizard (*Select Tools* e *Add Files*) ed infine selezioniamo *Finish*. Si aprirà automaticamente una finestra dal titolo *Add Microcontroller Subsystem*. Nel campo *Select Core*, in alto, selezioniamo la più recente versione dell'MSS (nel caso della release 9.0 SP3A di Libero si tratta della 2.2.101) e quindi scriviamo il nome 'Data-

Logger_MSS' nel campo *Design Name* in basso. Confermiamo la selezione; così il wizard aprirà la finestra SmartDesign MSS Configurator (Fig. 4), che permette di configurare, mediante interfaccia grafica, la sezione a microcontrollore e quella analogica della FPGA SmartFusion. La finestra riporta una sorta di diagramma a blocchi di tali sezioni, dove i blocchi in azzurro rappresentano sezioni configurabili in hardware, ovvero in maniera statica in fase di configurazione del dispositivo. Per definirne la configurazione è sufficiente fare doppio clic sulla relativa icona; si aprirà lo specifico pannello di configurazione. I blocchi in arancione si riferiscono, invece, a sezioni configurabili (anche dinamicamente) mediante istruzioni software dedicate;

per questi è possibile definire, mediante lo stesso pannello, una configurazione hardware predefinita impostata all'atto della programmazione. Il colore verde rappresenta, infine, moduli non configurabili. Alcune periferiche, caratterizzate dalla presenza di una casella all'interno del relativo blocco, possono essere disabilitate, qualora le relative funzionalità non siano necessarie per la specifica applicazione. Questo consente, evidentemente, di ridurre la dissipazione di potenza statica del dispositivo. Periferiche disabilitate appaiono, all'interno dello schema a blocchi, di colore grigio. Per la nostra applicazione, disabilitiamo, come mostrato in Fig. 4, le seguenti periferiche: MAC, SPI_0, SPI_1, I2C_0, I2C_1, UART_1 ed Exter-

nal Memory Controller. Passiamo quindi a configurare le altre.

CONFIGURAZIONE DELLA SEZIONE A MICROCONTROLLORE

Il modulo ClockManager (in alto a sinistra nello schema a blocchi nell'MSS Configurator), in particolare, gestisce il condizionamento e la generazione dei segnali di clock all'interno del dispositivo. La Fig. 5 mostra il relativo pannello di configurazione.

ACLK, ad esempio, è il clock utilizzato dalla sezione analogica; PCLK0 e PCLK1 sono invece i clock dei due bus APB accessibili dalla matrice di interconnessione AMBA e su cui risiedono le periferiche low-speed del sistema. FAB_CLK, infine, è il segnale di clock per la logica utente realizzata all'interno della FPGA Fabric. La configurazione da adottare per il nostro progetto è mostrata nella Fig. 5.

Il modulo Fabric_Interface (si veda in basso nello schema a blocchi nell'MSS Configurator) implementa la porta di accesso della FPGA_Fabric alla matrice di interconnessione AHB. Consente, così, di rendere accessibili al processore eventuali periferiche custom create dall'utente. Nel nostro caso, ad esempio, dovremo realizzare una periferica per la gestione della memoria SD su cui memorizzare i dati acquisiti. Configuriamo quindi il modulo Fabric Interface come mostrato in Fig. 6, abilitando l'interfaccia APB3 master. Il segnale di reset verso l'FPGA Fabric può essere abilitato all'interno del modulo Reset Manager, come mostrato nella Fig. 7. Si noti che, per semplicità, lo schema a blocchi all'interno dell'MSS Configurator non visualizza (per impostazione predefinita) le connessioni dei segnali di I/O dell'intera sezione

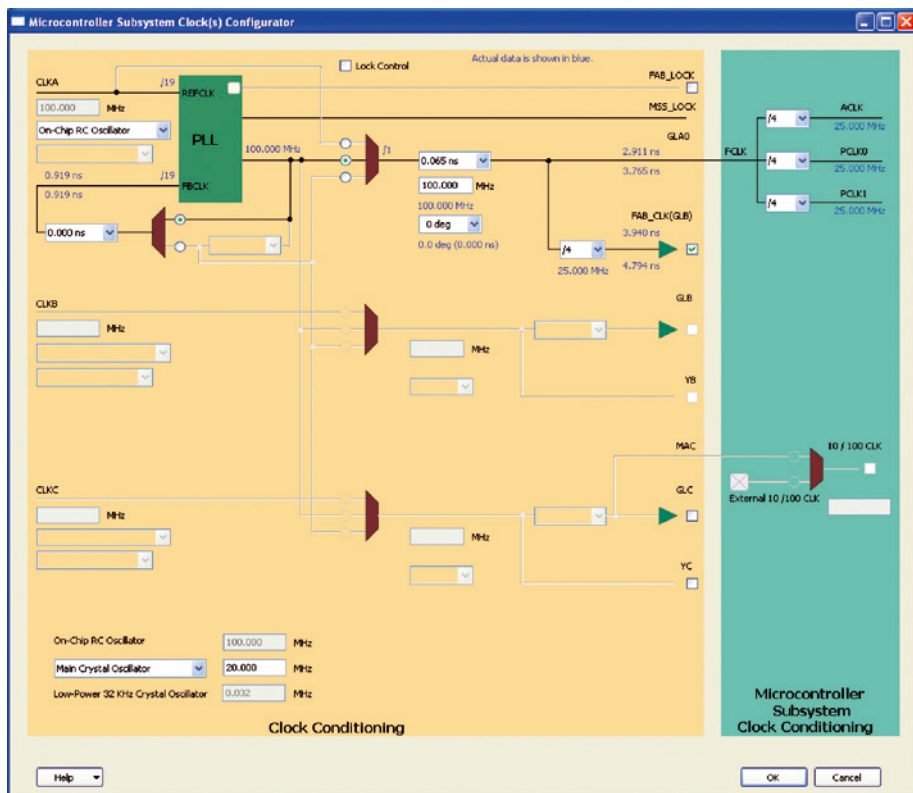


Fig. 5 - Configurazione del modulo Clock Manager.

Fig. 6 - Configurazione del modulo Fabric Interface.

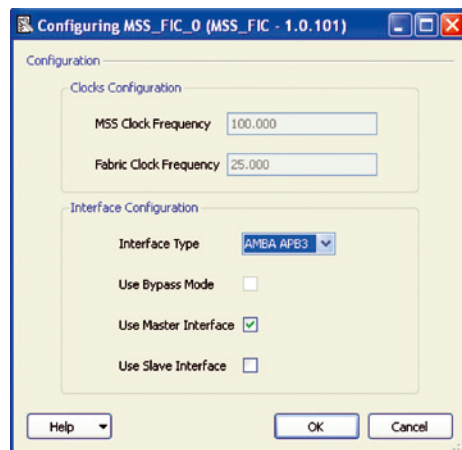
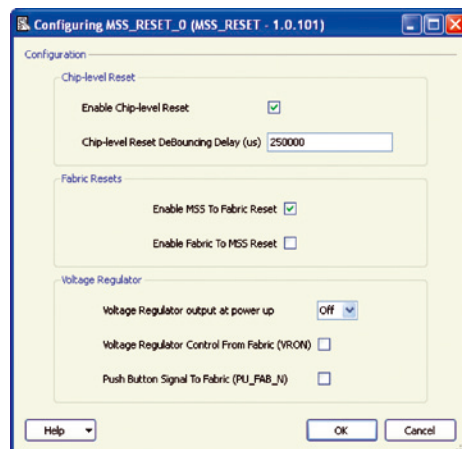


Fig. 7 - Configurazione del modulo Reset Manager.

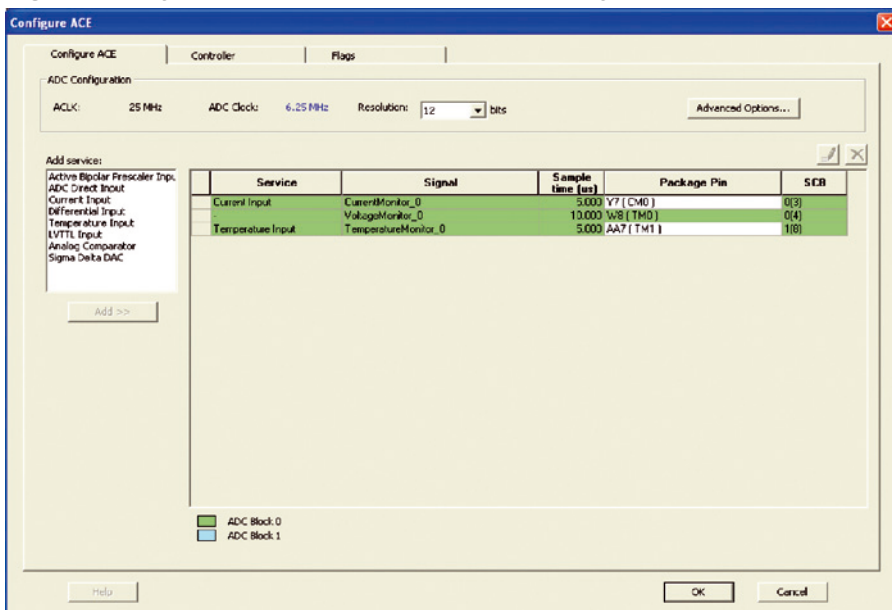


alle relative periferiche; essi sono rappresentati semplicemente come porte di colore verde sul bordo dello schema. Le relative connessioni appaiono solo quando la porta è selezionata; per renderle sempre visibili, è sufficiente selezionare la porta e, dal menu di pop-up che si apre facendo clic con il pulsante destro del mouse, selezionare l'opzione *show net*.

CONFIGURAZIONE DELLA SEZIONE ANALOGICA

Passiamo ora a configurare la sezione analogica del nostro sistema; facendo doppio click sul modulo ACE, si apre la finestra mostrata in Fig. 8, sulla sinistra della quale, nel riquadro *Add Service*, sono elencati i servizi che la periferica rende disponibili. Nel

Fig. 8 - Configurazione della sezione ACE: il tab Configure ACE.



nostro caso vogliamo monitorare un ingresso di corrente, uno di tensione ed uno di temperatura. La scheda di sviluppo A2F-EVAL-KIT, in particolare, dispone, come mostrato in Fig. 9, di un transistor MMBT3904LT1 in configurazione a diodo (Fig. 9b) utilizzabile come sensore di temperatura (il principio di funzionamento di tale misura è spiegato chiaramente nella guida [4] della sezione analogica dei dispositivi SmartFusion) e di un generatore di corrente (Fig. 9a) realizzato mediante un potenziometro da 50 kohm in serie ad una resistenza di misura di 100 ohm;

oltre alla corrente in quest'ultima, è possibile acquisire la tensione ai capi del potenziometro. Dalla lista dei servizi selezioniamo quindi dapprima la funzione *Current Input*. Facendo clic sul pulsante *Add >>* il servizio viene aggiunto e si apre la finestra di configurazione del canale, all'interno della quale possiamo modificare una serie di valori, come ad esempio, il tempo di acquisizione. Nel nostro caso, modifichiamo il valore della resistenza shunt in 100 Ω e lasciamo inalterati gli altri parametri; spuntiamo inoltre la casella *Use voltage monitor*, che consentirà di configurare un

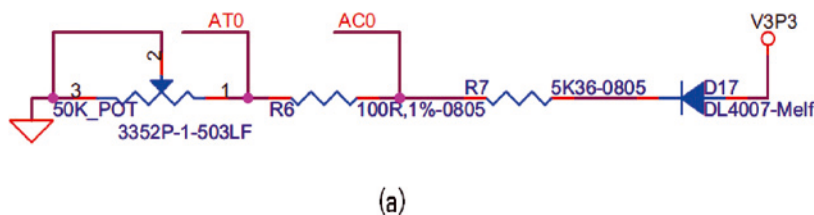
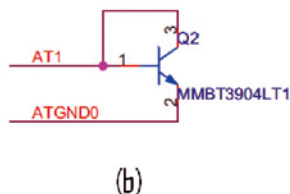


Fig. 9
Sensing
di corrente (a)
e misura di
temperatura (b)
sulla scheda
A2F-EVAL-KIT.



canale d'ingresso per la misura della tensione ad uno dei capi del resistore di shunt. Aggiungiamo infine, allo stesso modo, il servizio *Temperature_Monitor* per la misura di temperatura, lasciando inalterate, anche in questo caso, le relative configurazioni predefinite. Infine, nella colonna *Package Pin* del pannello di configurazione dei servizi, associamo ogni canale al pin del dispositivo SmartFusion. Nel caso della scheda di sviluppo A2F-EVAL-KIT, il canale di corrente è connesso al pin Y7 e quello di tensione al pin W8; per il canale di temperatura è usato invece il pin AA7.

Restando sempre all'interno della finestra di configurazione dell'ACE, passiamo al tab 'Controller' (vedi Fig. 10). Selezioniamo quindi l'opzione *Manual* per il campo *Operating sequence entry*. Ciò permetterà di definire una sequenza utente per la scansione dei canali; tale sequenza sarà eseguita automaticamente dall'ACE senza necessità di intervento del processore. Nel nostro caso applichiamo le definizioni mostrate nella Fig. 10. In questo modo, l'ACE eseguirà ripetutamente la scansione dei canali di ingresso; i valori acquisiti potranno essere successivamente letti dal processore mediante opportune istruzioni. Terminata la configurazione dell'ACE, ritorniamo alla finestra principale dell'MSS Configurator. Il passo successivo è la generazione di un BSP minimale per il nostro sistema, ovvero dell'insieme dei driver delle periferiche presenti nella configurazione hardware che abbiamo creato.

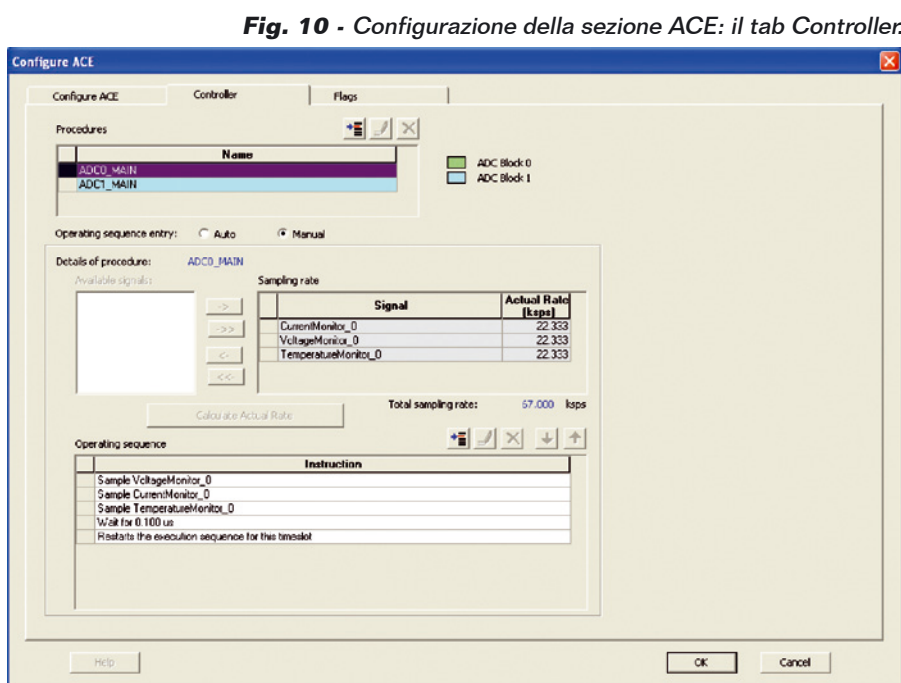
GENERARE IL FIRMWARE PER L'ACCESSO ALLE PERIFERICHE

Salviamo il progetto e selezioniamo, quindi, il tab *Firmware* (Fig. 11). Selezioniamo i driver

che ci interessano, i quali, in linea di principio, dovrebbero essere quelli delle periferiche che abbiamo istanziato e selezionato, supposto di non voler sviluppare driver dedicati per alcune di esse. Se il driver di libreria non è disponibile già nel repository di Libero, verrà richiesta l'autorizzazione a scaricarlo dal web. Salviamo ancora il progetto e selezioniamo infine la voce *Generate* dal menu *Design* presente nella barra superiore. Il software crea automaticamente una cartella firmware nella root del nostro progetto, contenente l'insieme dei driver richiesti, opportunamente configurati in accordo all'architettura hardware impostata. Se proviamo a guardare nella cartella, non ci sarà difficile riconoscere le sottocartelle con i driver per ogni periferica, oltre al layer CMSIS HAL su cui essi poggiano. Informazioni dettagliate sui driver di libreria e sulle modalità di utilizzo si trovano alla pagina web indicata nei riferimenti [2]. Chiudiamo l'MMS Configurator.

UN CONTROLLER DI MEMORIA SD

Nei due precedenti paragrafi abbiamo mostrato come configurare opportunamente le sezioni a micro-controllore ed analogica del nostro sistema. Di seguito mostreremo quindi come istanziare una periferica custom all'interno della FPGA Fabric e in che modo renderla accessibile al processore. Nel nostro progetto, in particolare, abbiamo bisogno di un controller di memoria SD per la memorizzazione su di un supporto non volatile dei dati acquisiti. È solo il caso di osservare che, in generale, una memoria SD può essere gestita direttamente mediante interfaccia SPI, implementando in software la gestione



del protocollo di scambio dei messaggi per gli accessi in scrittura e lettura. In molti sistemi embedded, questa è la strada seguita più spesso. Nel nostro esempio, invece, abbiamo scelto di adottare un controller dedicato che implementi in hardware l'intero protocollo di accesso alla memoria, sia per ridurre il carico del processore, sia per mostrare, appunto, come realizzare una periferica custom all'interno della FPGA Fabric di un dispositivo SmartFusion.

Un IP core (ovvero una descrizione in linguaggio HDL sintetizzabile), denominato spiMaster e che implementa un controller di memoria SD/MMC, è disponibile per il download gratuitamente sul sito OpenCores [5], una delle principali fonti di distribuzione di core IP open-source. Il core si presenta con una semplice interfaccia software che include un registro di comando che consente di iniziare le procedure per l'inizializzazione della memoria e la lettura o scrittura di

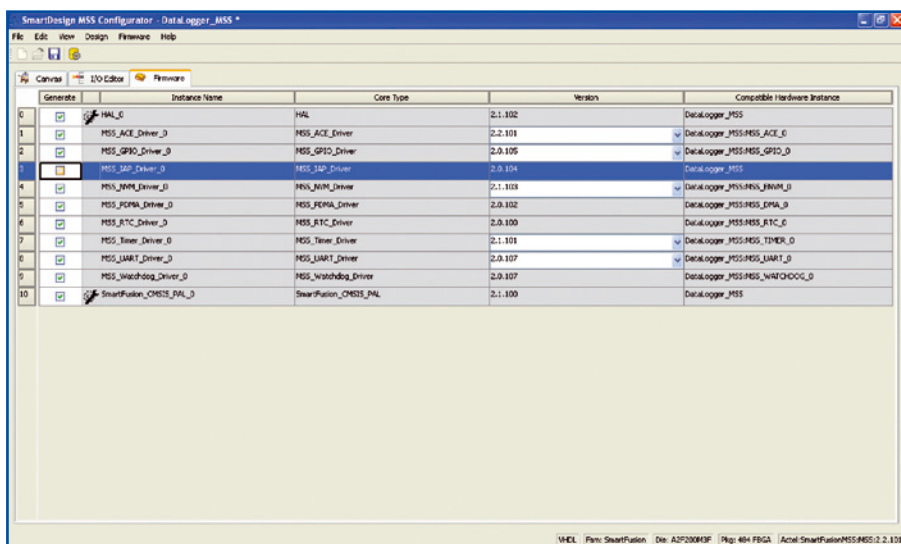


Fig. 11 - Generazione del firmware da MSS Configurator.

Listato 1

-- APB to Wishbone Bridge Interface

```
pready    <= WbAck_0;
prdata    <= x"00" & x"00" & x"00" & WbRdData;

WbClk     <= pclk;
WbReset    <= not preseln;
WbAddr    <= paddr(9 downto 2);
WbWrData   <= pwdata(7 downto 0);
WbWe      <= pwrite;
WbStrobe   <= psel and penable;
```

un registro; l'indirizzo di pagina della memoria cui accedere è specificato in un registro dedicato. Due memorie FIFO permettono di memorizzare i dati letti dalla memoria o da scrivere in questa. Per i dettagli è possibile consultare il datasheet del core scaricabile unitamente al codice sorgente. Per accedere al codice sorgente è richiesta la registrazione sul sito OpenCores; facciamola e scarichiamo i file. Notate che il linguaggio adottato per la descrizione hardware del core è il Verilog; per continuare nello sviluppo del nostro progetto non serve conoscerlo, in quanto non occorrerà modificare i file sorgente. Torniamo alla finestra principale

di Libero ed eseguiamo il comando *File -> Import Files...* dalla barra dei menu. Selezioniamo quindi i file sorgente dalla cartella trunk\RTL all'interno della distribuzione del core scaricata dal sito. Confermiamo la selezione e ritorniamo alla finestra principale di Libero: se non ci sono problemi nell'import della

periferica, vedremo apparire il componente spiMaster all'interno della lista dei file di progetto riportata nel tab Hierarchy della sezione Design Explorer (presente sulla sinistra dell'ambiente di sviluppo di Libero). Espandendo l'albero associato, è possibile riconoscere la struttura del core con i diversi moduli

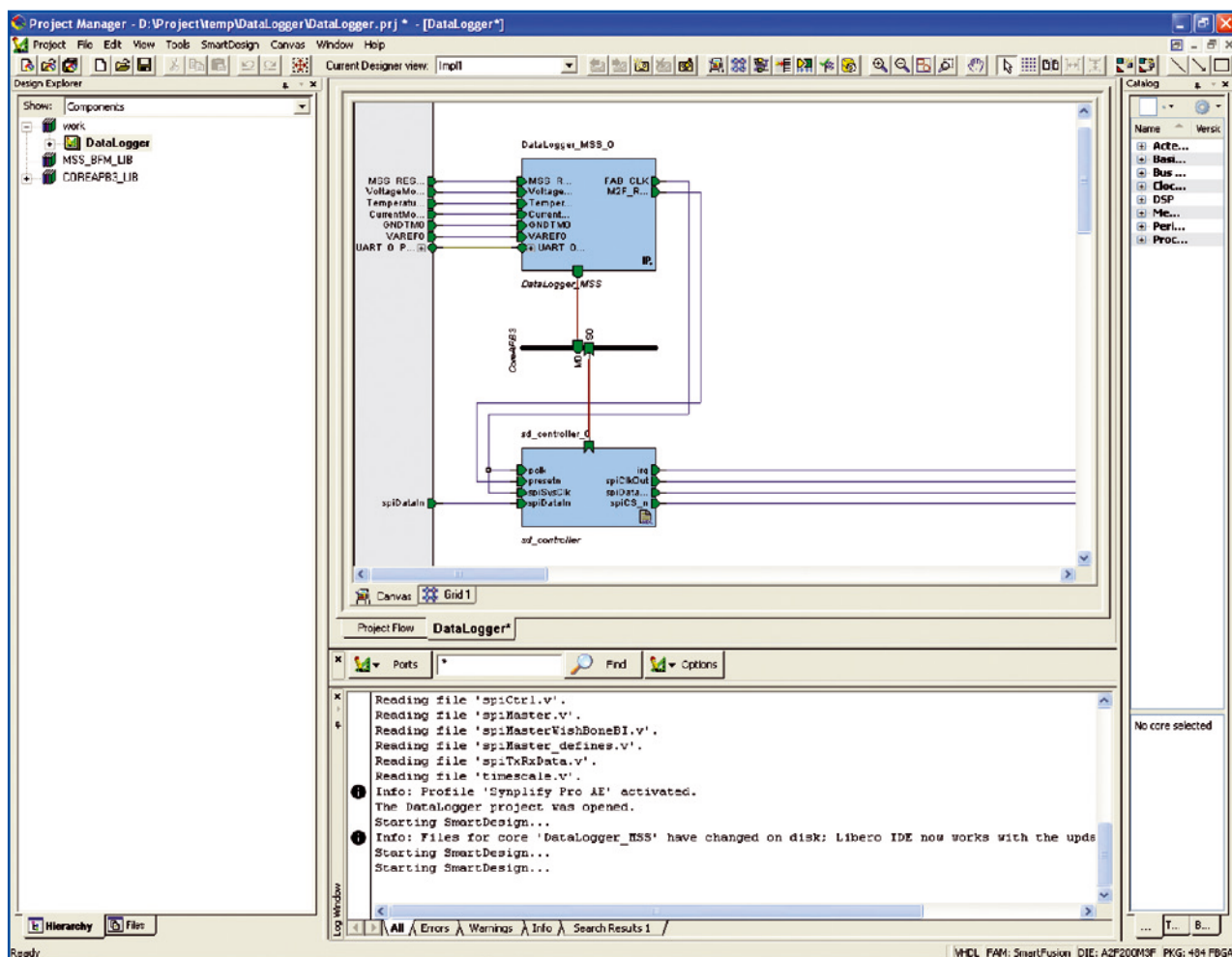


Fig. 12 - La finestra di lavoro di SmartDesign all'interno dell'ambiente di Libero.

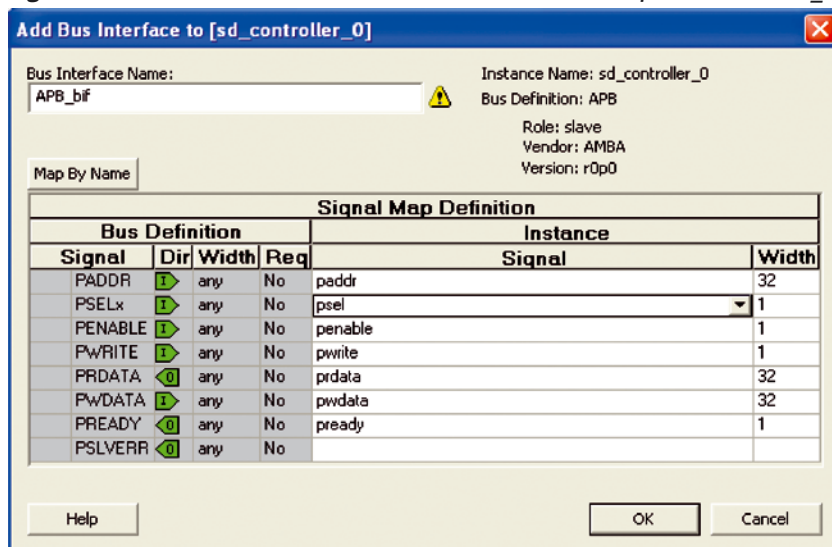
istanziati al suo interno.

Il core spiMaster dispone di interfaccia Wishbone verso il processore host; Wishbone è lo standard, piuttosto elementare, supportato da OpenCores per l'integrazione di periferiche in applicazioni System-On-Chip. Nel progetto del nostro DataLogger, abbiamo invece bisogno di interfaccia APB3.

Per quanti fossero interessati all'argomento, gli standard Wishbone ed APB3 possono essere scaricati dal web agli indirizzi riportati nei riferimenti [6] e [7]. Nel nostro progetto dobbiamo quindi creare un wrapper per il core spiMaster, ovvero un modulo di più alto livello che istanzi il core al suo interno insieme alla logica per tradurre le transazioni del bus APB3 in accordo al protocollo Wishbone. Il codice sorgente di tale wrapper, scritto in linguaggio VHDL, può essere scaricato dal nostro sito web nell'allegato a questo articolo (file *DataLogger/hw/hdl/spiMaster_APBWrapper.vhd*). Un estratto del codice che mostra la semplice traduzione di controlli, indirizzi e dati tra i due bus Wishbone ed APB3 è mostrato nel **Listato 1**.

Per quanti conoscono il VHDL, non sarà difficile riconoscere le poche istruzioni che implementano la logica di bridge tra i due bus Wishbone ed APB3. Eseguiamo il comando *File -> New* dalla barra dei menu di Libero; selezioniamo *VHDL Source File* e inseriamo *sd_controller.vhd* nel campo *Name*. Si aprirà una finestra dell'editor di testo. Copiamo il codice dal file che abbiamo scaricato in precedenza e salviamo il file dall'editor. Se non ci sono errori, vedremo apparire nel Design Explorer il nuovo componente *sd_controller*, che integra il controller spiMaster precedentemente importato.

Fig. 13 - Finestra di definizione dell'interfaccia APB slave per il modulo *sd_controller*.



L'ARCHITETTURA DI ALTO LIVELLO

L'ultimo passo che ci resta da fare è definire un componente di più alto livello (e che rappresenta l'architettura interna della nostra FPGA) che istanzi il controller di memoria SD appena creato unitamente alla sezione MSS, la

quale già include la parte analogica. Dalla finestra principale di Libero eseguiamo quindi il comando *File -> New*; selezioniamo il tipo *SmartDesign Component* ed inseriamo un nome (nel nostro caso *DataLogger*). SmartDesign è un tool di design entry di tipo grafico; la finestra

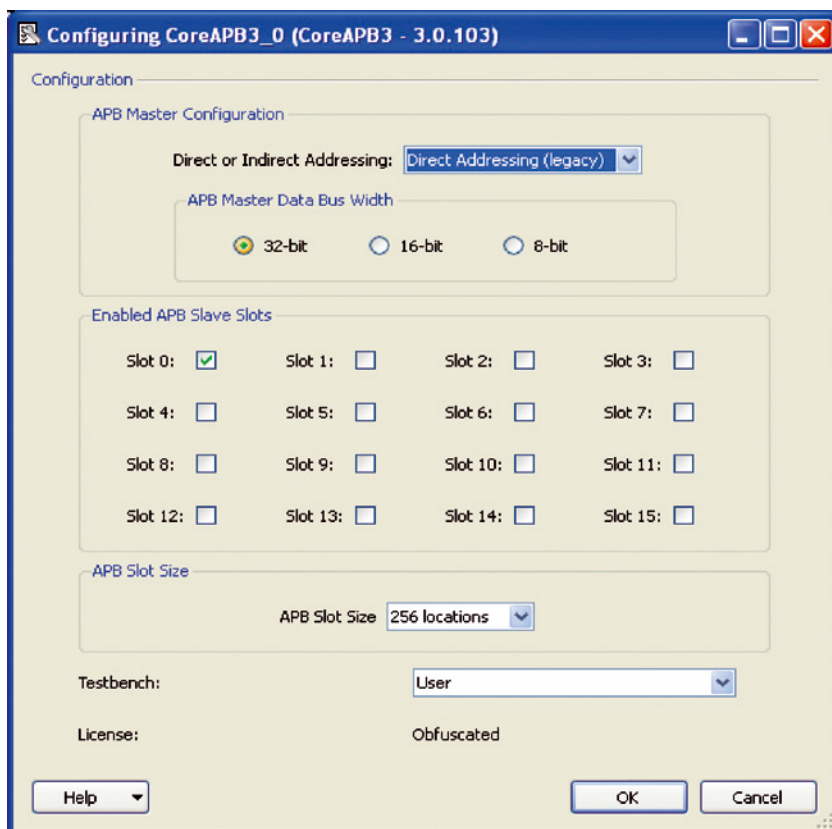


Fig. 14 - Finestra di configurazione del bus APB.

corrispondente si aprirà automaticamente all'interno dell'area di lavoro di Libero (Fig. 12) in un nuovo tab disposto accanto a *Project Flow* ed avente il nome del componente creato. La finestra di lavoro di SmartDesign consiste a sua volta di due tab: *canvas* è un editor grafico che consente di inserire i componenti e connetterli; *grid* mostra, invece, gli stessi e le relative connessioni in forma tabellare. Inizialmente il tab *canvas* risulterà interamente bianco. Prima di creare il diagramma a blocchi del nostro sistema, passiamo alla finestra *Design Explorer* di Libero e verificiamo che il nuovo componente *DataLogger* sia configurato come quello di più alto livello; all'interno del tab *Hierarchy*, sotto la voce *work*, dovrebbe infatti apparire evidenziato in grassetto. In caso contrario, selezioniamo il componente, apriamo il menu di pop-up facendo clic con il pulsante destro del mouse e scegliamo la voce *Set as Root*.

Verificato questo, selezioniamo il componente *DataLogger_MSS* da *Design Explorer* (nel tab *Hierarchy*) e trasciniamolo all'interno della finestra di SmartDesign (tab *canvas*).

Vedremo comparire un blocco logico corrispondente alla sezione a micro-controllore che avevamo creato in precedenza; facendo doppio clic sul blocco, si aprirà l'*MSS Configurator* che abbiamo imparato a conoscere in precedenza. È solo il caso di osservare che nella finestra *Design Explorer* il componente *DataLogger_MSS* apparirà ora sotto la voce *DataLogger*, coerentemente con la gerarchia di file di progetto che stiamo creando. Allo stesso modo, selezioniamo da *Design Explorer* il componente *sd_controller* e trasciniamolo nell'area di lavoro di SmartDesign; comparirà il blocco corrispondente. Quindi andiamo nella finestra *Catalog*, che si trova sulla destra dell'area di lavoro di Libero, e, all'interno

del tab *Bus Definition* (si veda in basso), selezioniamo la voce APB AMBA2 slave. Quindi trasciniamo la selezione direttamente sul modulo *sd_controller* nel *canvas* di SmartDesign. Vedremo apparire una finestra (Fig. 13) che consente di definire per il nostro modulo un'interfaccia APB3 slave (come richiesto per poterlo connettere alla porta APB3 del processore) e di associare i pin dello stesso a tale interfaccia. Selezioniamo *Map by name* per l'associazione automatica; quindi, in corrispondenza del segnale PSELx della colonna *Bus Definition* (per il quale non viene riconosciuta tale associazione automatica) digitiamo PSEL nel corrispondente campo *Signal* sotto *Instance*. Confermiamo per ritornare in SmartDesign. Vedremo che i segnali dell'interfaccia APB sono ora stati raggruppati in un'unica porta. Ritorniamo alla sezione *Catalog* e, all'interno del tab *Core*, sotto la voce *Bus Interface*, selezioniamo il componente Core APB3, che descrive il controller di un bus APB3. Trasciniamo la selezione nel *canvas* di SmartDesign. Nella finestra di configurazione (Fig. 14) del componente che si aprirà automaticamente, selezioniamo la modalità *Direct Addressing* nel riquadro *APB Master Configuration* e spuntiamo in *Enabled APB Slave Slot* la sola casella corrispondente a *Slot 0*. Lasciamo invariate le restanti opzioni e confermiamo la nostra configurazione per ritornare al *canvas* di SmartDesign.

A questo punto, facendo clic con il pulsante destro del mouse in una zona libera, selezioniamo *Auto connect* dal menu di pop-up. Vedremo create automaticamente (come mostrato nella Fig. 12) le connessioni dei diversi componenti che avevamo precedentemente istanziato. Infine selezioniamo le porte *spiDataIn*,

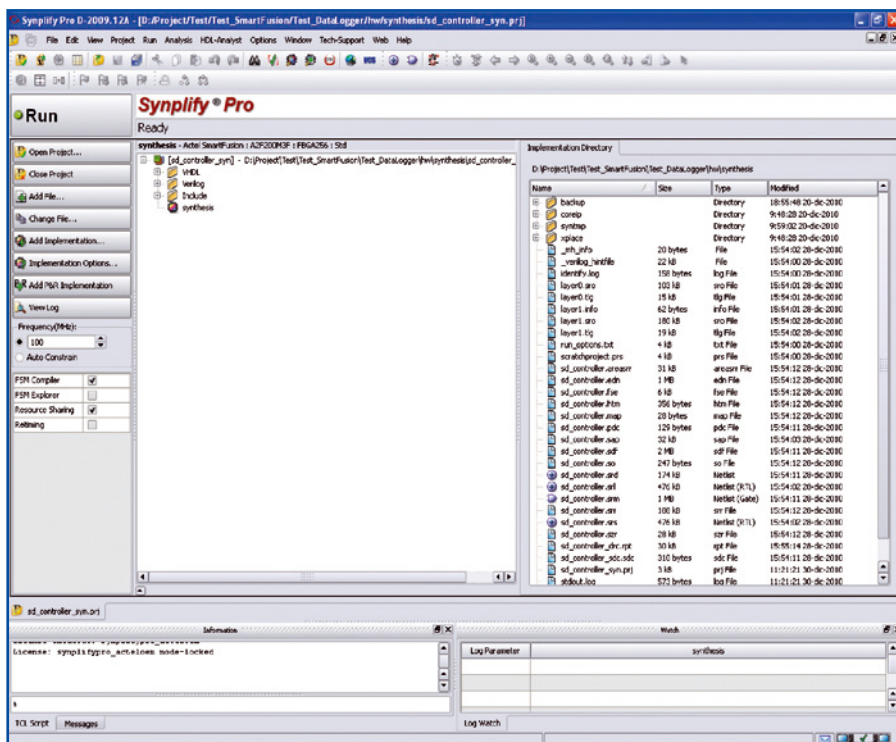


Fig. 15 - La finestra principale del tool di sintesi Synplify.

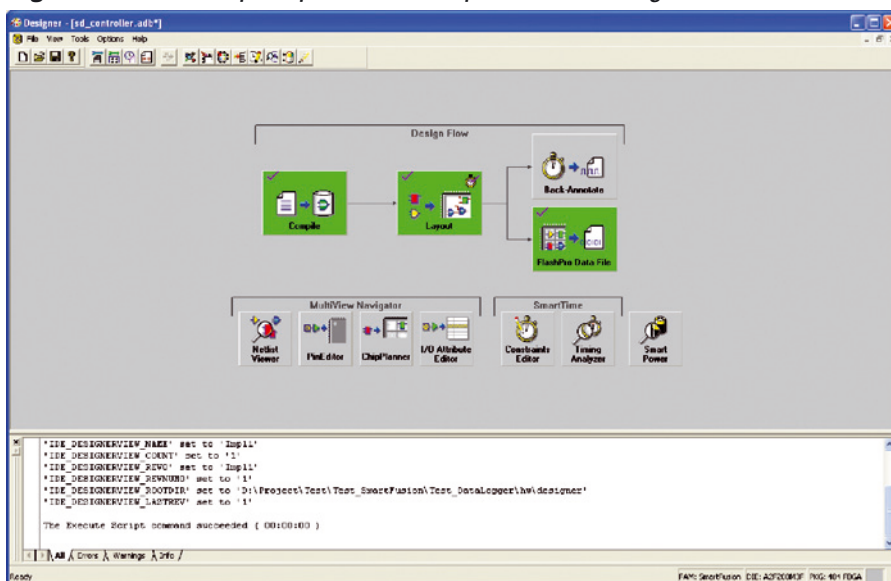
spiClkOut, *SpiDataOut* e *SpiCS_n* del modulo *sd_controller_0* e, dal menu di pop-up (che si apre, al solito, facendo clic con il pulsante destro del mouse) scegliamo la voce *Promote top level*: vedremo i suddetti segnali del componente connessi a porte di I/O della FPGA. Infine dalla barra dei menu impartiamo dapprima il comando *Tools -> Check Design Rules* e quindi, in assenza di errori o warnings significativi, la voce *Tools -> Generate Design*....

SINTESI DEL CIRCUITO E GENERAZIONE

DEL FILE DI PROGRAMMAZIONE

Ritorniamo a questo punto al tab *Design Flow* della finestra centrale di Libero e, facendo clic sulla corrispondente icona, lanciamo il tool di sintesi Synplify. Si aprirà la schermata principale del software, come mostrato in Fig. 15. Lanciamo Run ed attendiamo che il processo termini. Verifichiamo, anche in questo caso, nella finestra in basso, che non vi siano errori: vedremo diversi warning; in molti casi, purtroppo, interpretare il significato di questi richiede dimestichezza con l'ambiente di sviluppo e con l'architettura della FPGA. Nel nostro caso, in linea di principio, se abbiamo seguito correttamente i passi descritti, possiamo ignorarli tranquillamente. Chiudiamo Synplify e ritorniamo

Fig. 16 - La finestra principale del tool di place&route Designer.

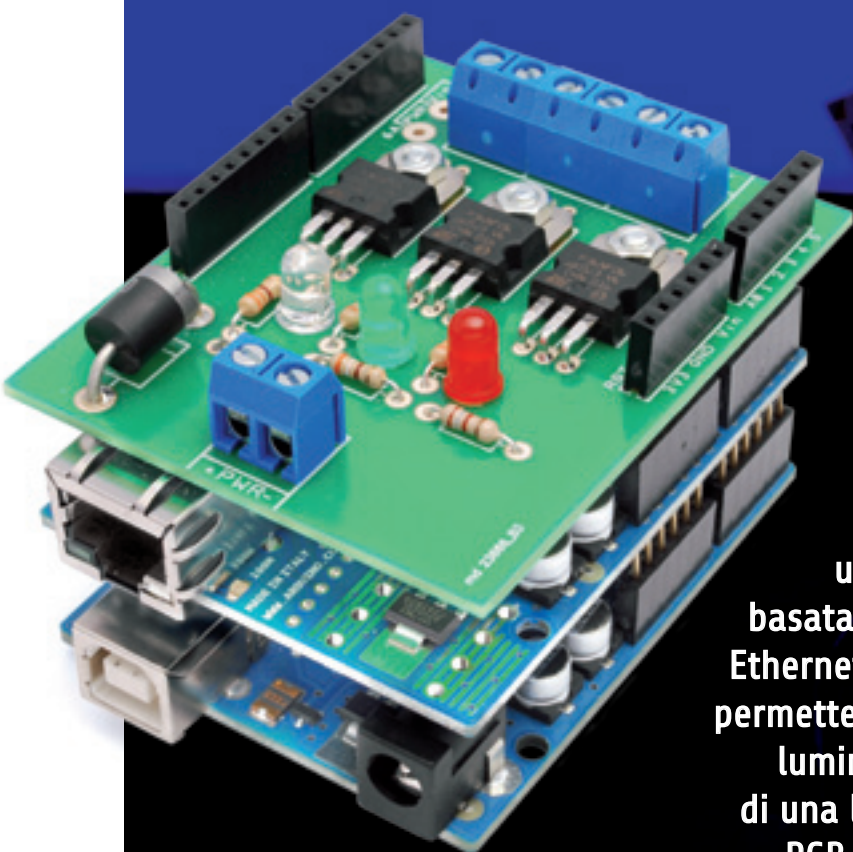


alla vista Design Flow di Libero. Selezioniamo l'icona Place&Route e, nella finestra che si apre automaticamente, rimuoviamo il file *DataLogger.pdc* dal riquadro *Constraint files for Designer*. Confermando la selezione, si aprirà la schermata principale di Designer, come mostrato in Fig. 16. Confermiamo le configurazioni che ci vengono proposte e quindi impartiamo il comando *Compile* facendo clic sulla corrispondente icona. Verifichiamo, al solito, che non ci siano errori e lanciamo il comando *I/O Attribute Editor*. Associamo quindi i pin del dispositivo SmartFusion alle porte *spiDataIn*, *spiClkOut*, *SpiDataOut* e *SpiCS_n*, che rappresentano i segnali dell'interfaccia SPI di comunicazione con la memoria SD. Nel nostro caso, inten-

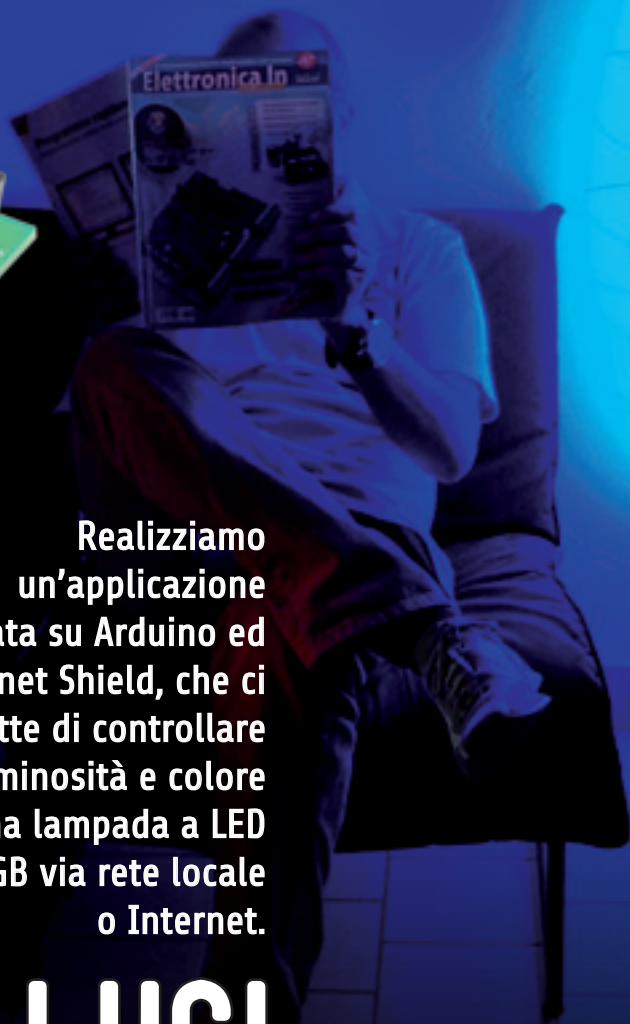
dendo utilizzare la scheda A2F-EVAL-KIT come banco di prova, selezioneremo i pin K19, J22, J20 e J21 che corrispondono ai pin 2, 3, 4 e 5 del connettore J22 riservato per I/O generici. Nella terza ed ultima parte di questo tutorial, descriveremo come connettere la memoria SD alla scheda di sviluppo. Confermiamo la selezione mediante il comando *File -> Commit* dalla barra dei menu e chiudiamo la finestra per ritornare alla schermata principale di Designer. Lanciamo il comando *Generate FlashPro Data File* (facendo clic sull'icona corrispondente) e verifichiamo che il processo termini correttamente; salviamo il progetto e chiudiamo la finestra di Designer per ritornare in Libero. Per ora ci fermiamo qui. ■

Riferimenti

- [1] <http://www.actel.com/products/software/default.aspx>
- [2] http://www.actel.com/products/software/smartdesign/docs_mss.aspx
- [3] <http://register.actel.com>
- [4] http://www.actel.com/documents/SmartFusion_Analog_UG.pdf
- [5] <http://opencores.org/project/spimaster>
- [6] http://opencores.org/cdn/downloads/wbspec_b3.pdf
- [7] <http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ki1hk8p7dqPG1c.html>



Realizziamo un'applicazione basata su Arduino ed Ethernet Shield, che ci permette di controllare luminosità e colore di una lampada a LED RGB via rete locale o Internet.



CONTROLLO LUCI CON ARDUINO VIA ETHERNET

di BORIS LANDONI

L'illuminazione è uno dei sistemi elettrici domestici che per primi si è pensato di comandare a distanza, non a caso esistono da anni dimmer radio-comandati o controllati da telecomandi ad infrarossi; la comodità di variare o accendere e spegnere le luci di casa comodamente seduti in poltrona è innegabile e diventa una cosa indispensabile per le persone disabili o per chi, a seguito di una momentanea permanenza forzata a letto o limitata mobilità, non può muoversi agevolmente. Poter controllare l'illuminazione via rete locale (LAN, se preferite) è qualcosa

di ancor più comodo, dato che le luci possono essere gestite con sistemi domotici governati da un controller stand-alone o da un computer; ancor meglio, da telecomandi che sfruttano la rete locale o wireless. Il progetto che descriviamo in queste pagine realizza proprio quest'ultimo tipo di controllo: si basa su Arduino dotato della Ethernet Shield (quindi fornito di connettività ethernet) e permette di controllare una lampada illuminata da strip di LED rossi, verdi e blu, capace perciò di assumere diverse tonalità e variare la propria luminosità in ciascun colore.

Avendo l'interfaccia LAN, l'unità può essere comandata a distanza tramite rete locale, ovvero da cellulari come l'iPhone, sfruttando la connessione Internet (in questo caso la LAN cui è collegato il dispositivo deve essere connessa ad Internet tramite un router) o direttamente l'interfaccia Wi-Fi (in tale evenienza la LAN deve essere provvista di un access-point wireless). Proprio tale possibilità consente di utilizzare un iPhone o altro dispositivo dotato di browser web e interfaccia wireless (qualsiasi PC, notebook, tablet o netbook) come telecomando, senza spendere soldi per connettersi ad Internet: una comodità indiscutibile.

COME FUNZIONA

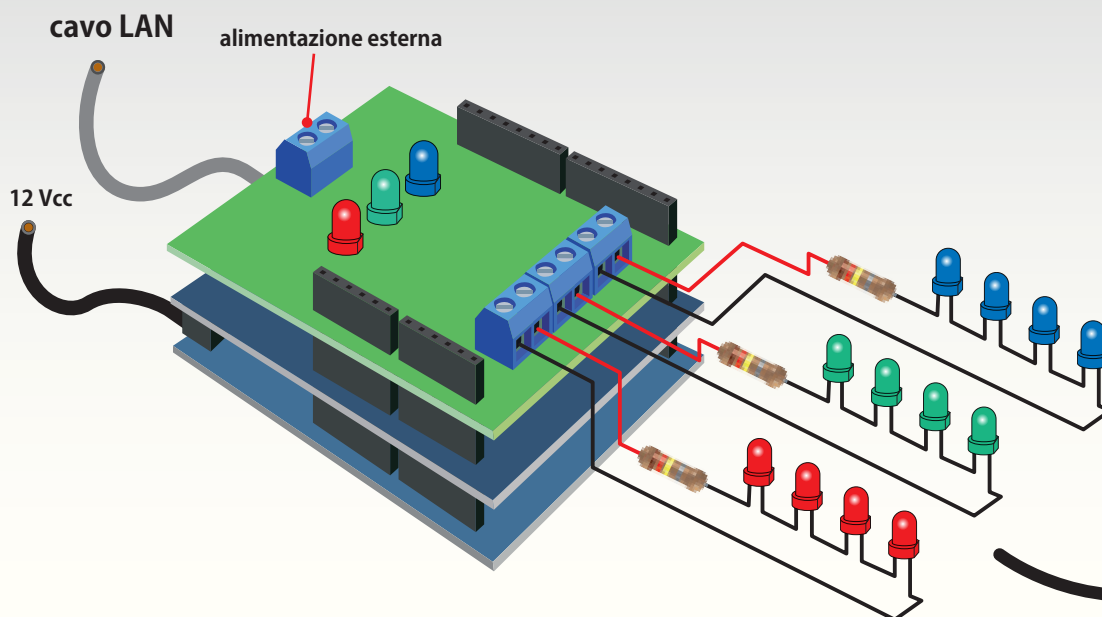
Il sistema che vi proponiamo di realizzare si basa sul modulo Arduino Uno, sul quale sono montati a sandwich due shield: l'Ethernet Shield, che realizza la connessione LAN e dispone di

tutto ciò che occorre a livello hardware, e lo shield da noi realizzato (che chiamiamo *shield RGB*) il quale monta sostanzialmente tre driver di potenza per alimentare dei LED dietro comando di livelli logici in arrivo da Arduino. Lo shield ethernet è un prodotto finito della serie Arduino e si basa sul chip WS5100 della Wiznet; è pin-to-pin compatibile con Arduino Uno ed integra, oltre al chip Wiznet (che da solo svolge tutte le funzioni necessarie ad instaurare e mantenere una connessione LAN a 10/100 Mbps) e ad un pulsante per resettarlo (il pulsante di reset sullo shield resetta sia il chip W5100 che la scheda Arduino), anche un lettore di microSD, che in questo progetto non viene utilizzato. Lo shield è completato dal connettore RJ45, al cui interno si trovano i LED di segnalazione (Link e Speed) e i trasformatori di accoppiamento. Lo shield supporta fino a quattro connessioni socket simultanee.

Le linee che Arduino usa per comunicare con il chip W5100 sono quelle digitali relative ai contatti 10, 11, 12 e 13 (porta SPI) che quindi non possono essere utilizzati come I/O per altre applicazioni. Il chip ethernet Wiznet W5100 dello shield implementa a livello hardware uno stack IP completo di protocollo di comunicazione TCP ed UDP; nella nostra applicazione utilizziamo il primo, che è quello impiegato anche da Internet, quindi lavoriamo in TCP IP.

Dobbiamo ricordare che il protocollo TCP si basa sulla creazione di una comunicazione tra un *Client* ed un *Server*, che deve essere iniziata dal *Client* inviando una richiesta al *Server*, il quale è sempre in attesa su una porta ben precisa; se il *Server* accetta, viene aperta una comunicazione a questo punto sarà possibile il trasferimento di dati. Una peculiarità del TCP è che conosce lo stato della connessione.

Realizzare la lampada



ne e verifica se i pacchetti sono stati ricevuti correttamente dal ricevente.

Lo shield ethernet da noi usato dispone di sei LED montati sul circuito stampato, due dei quali sono accanto alla presa RJ45; i quattro vicini al quarzo hanno le seguenti funzioni:

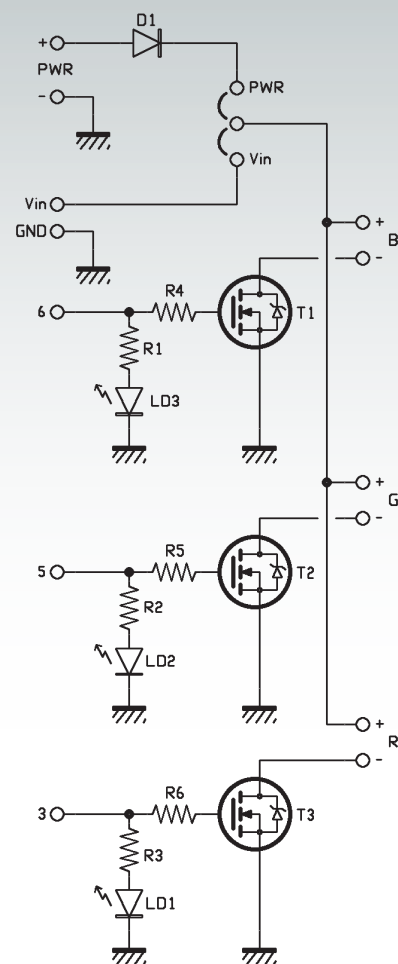
- TX: lampeggia quando la shield trasmette dati (indica cosa accade sul pin TX di Arduino);
- RX: lampeggia quando la shield riceve dati (indica cosa avviene sull'RX di Arduino);
- L: è collegato al piedino 13 (linea per comando di un LED generico o SCK) di Arduino;
- ON: indica che lo shield è sotto tensione.

Quanto ai due diodi posti accanto alla RJ45, sono LINK (indica la presenza di una rete e lampeggia quando lo shield trasmette o riceve dati) e 100 M (segnala quando la connessione sta avvenendo a

100 Mbps); i due ripetono quelli sul connettore.

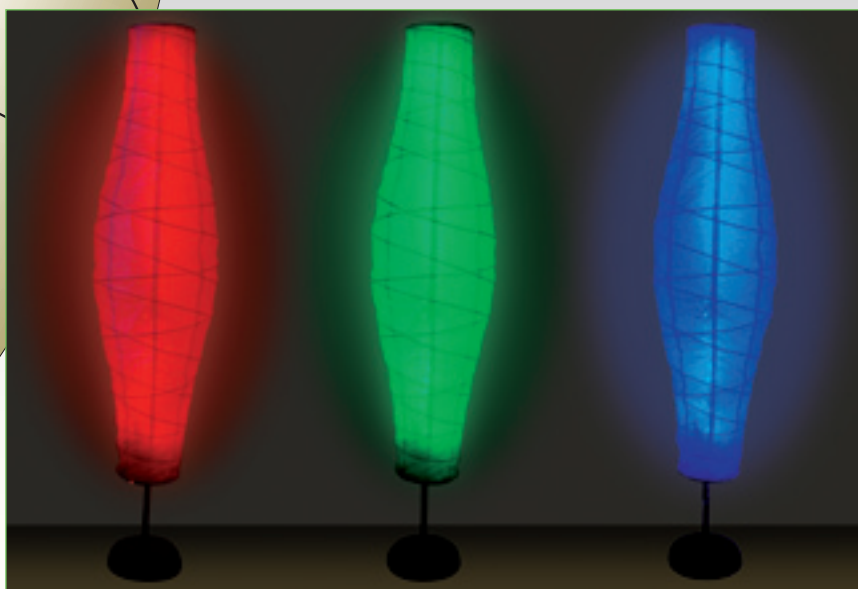
Realizzare una connessione LAN è abbastanza facile, dato che l'ambiente di lavoro di Arduino contiene già la libreria e gli esempi per la gestione della ethernet; a tal proposito, rimandiamo chi volesse approfondire l'argomento alla puntata 10 (Elettronica In n° 156) del corso su Arduino che abbiamo pubblicato a partire dal fascicolo n° 147.

In Arduino è caricato un apposito sketch che consente di gestire la comunicazione via LAN, ovvero l'Ethernet Shield, creare e far funzionare un'interfaccia di tipo web (che si presenterà a chi tenterà di accedere tramite rete locale) ed eseguire i comandi ricevuti, operando programmi di funzionamento come la variazione progressiva (Fader) del colore. Nello specifico, Arduino si comporta come un web server, mostrando una pagina html a chi accede tramite un browser,



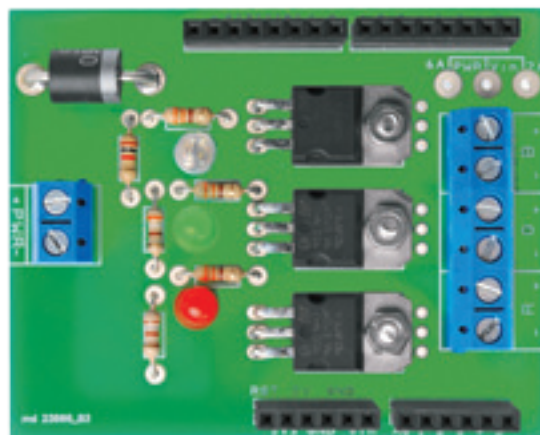
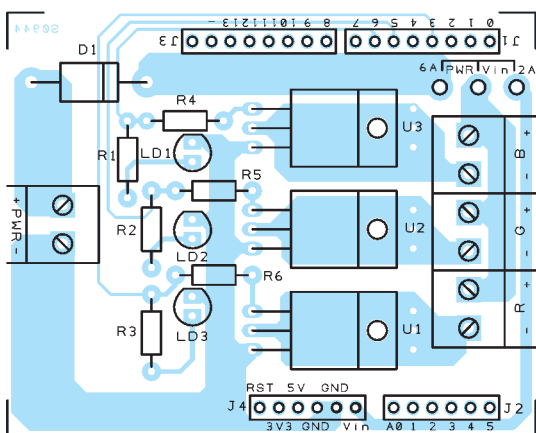
[schema elettrico SHIELD RGB]

introducendo nella barra degli indirizzi l'IP address corrispondente allo shield ethernet; questo indirizzo, per impostazione predefinita è 192.168.0.88. Ciò significa che la lampada può anche essere accesa e controllata tramite qualsiasi dispositivo in rete o da remoto via Internet. La pagina web mostra l'attuale impostazione dei canali R, G, B della lampada e permette di modificarle. Volendo, è possibile cambiare l'indirizzo IP assegnato di base dallo sketch; allo scopo, dovete editare la libreria che realizza la funzionalità ethernet e definire il nuovo indirizzo. Ricordate, però, che l'IP va definito compatibilmente con le impostazioni della rete locale in cui avete inserito Arduino, nel senso che se la vostra rete inizia con 192.168.0. dovete rispettare questo formato,



Alle uscite dello shield potete collegare delle strisce di LED già pronte o auto-costruite, che avrete cura di inserire all'interno dell'involucro della lampada; qui sopra vedete la nostra realizzazione, con accese le sole strisce rossa, verde o blu.

[piano di montaggio **SHIELD RGB**]



Elenco Componenti:

R1: 1 kohm
R2: 390 ohm
R3: 180 ohm
R4÷R6: 330 ohm
T1÷T3: STP36NE06

LD1: LED 5 mm blu (510LB7C)
LD2: LED 5 mm verde
LD3: LED 5 mm rosso
D1: 6A606

Varie:
- Morsetto a 2 poli (4 pz.)
- Strip M/F 6 poli (2 pz.)
- Strip M/F 8 poli (2 pz.)
- Circuito stampato

quindi anche l'IP address dello shield ethernet deve iniziare con 192.168.0.

Per rendere accessibile la pagina web dalla rete cui il sistema viene collegato, dobbiamo impostare correttamente, oltre all'indirizzo IP, il MAC. Quest'ultimo (MAC address) è un codice di 12 caratteri che identifica ogni singolo apparato in grado di connettersi a una rete ethernet; nel nostro caso, sia la scheda di rete

che il router (o access-point) ne avranno uno univoco.

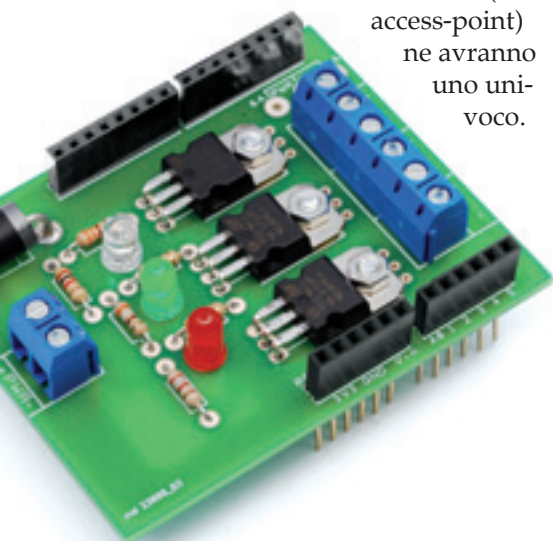
Per semplicità, lasciate invariato quello proposto nello sketch; invece, se volete accedere da Internet, dovrete preoccuparvi di recuperare un indirizzo MAC libero. L'indirizzo IP (IP Address) è una sequenza di 4 numeri compresi tra 0 e 255 e, similmente al MAC, identifica ogni dispositivo all'interno di una rete. Anche questo deve essere univoco per ogni dispositivo sulla LAN e deve essere assegnato in congiunzione con il Subnet Mask (sequenza di quattro numeri che permettono di specificare delle sottoreti).

Nel nostro caso, avendo una subnet mask pari a 255.255.255.0, significa che potranno colloquiare tra di loro tutti i dispositivi facenti parte alla sottorete che ha indirizzo IP con i primi tre numeri uguali ed il terzo scelto a piacere e compreso tra 0 e 255. L'indirizzo IP di Arduino dovrà quindi essere del tipo 192.168.0.xxx dove xxx sarà un numero compreso tra 0 e 255 non utilizzato da altri dispositivi nella

rete. In una connessione TCP, oltre all'indirizzo IP della postazione remota cui devono giungere le informazioni, bisogna conoscere anche a quale applicazione devono essere passate; a tale scopo si usano le porte, le quali sono una specie di allocazione della memoria del computer che esiste quando due computer sono in comunicazione tra loro e forniscono un punto terminale per l'applicazione remota. Le porte vengono sempre identificate con un numero compreso tra 0 e 65.535 (i valori 0 e 1.023 sono già assegnati per specifiche funzioni del sistema operativo).

LO SHIELD RGB

Arduino comanda l'accensione dei LED appartenenti ai canali rosso, verde e blu mediante uno shield molto semplice, contenente tre MOSFET enhancement-mode a canale N, di tipo P36NF06; trovate in queste pagine lo schema elettrico dello shield, dal quale potete vedere che ciascun MOSFET viene pilotato sul gate,



tramite un resistore, con il segnale logico che Arduino invia. Per l'esattezza, la linea 3 comanda T3 (rosso) la 5 pilota T2 (verde) e la 6 agisce su T1 (blu); ciascuna linea di comando dispone di un LED di stato, polarizzato mediante una resistenza di limitazione (i LED indicano come si sta comportando il rispettivo canale). Notate che Arduino comanda i singoli transistor mediante segnali PWM, il cui duty-cycle determina la presenza e l'intensità di un certo colore; più precisamente, la larghezza degli impulsi può variare da un minimo ad un massimo per decidere quanto luminoso debba essere il gruppo di LED del rispettivo colore. Il Drain di ciascun MOSFET alimenta il proprio carico, ossia i LED di colore corrispondente, che devono essere collegati con l'anodo alla linea positiva comune di alimentazione (+); per ciascun canale è prevista una morsettiera con un contatto positivo (va sulla linea comune) ed un negativo (corrisponde al drain del rispettivo MOSFET).

Notate che abbiamo previsto la possibilità di alimentare i LED in due modi: con l'alimentazione prelevata dal contatto Vin di Arduino (in tal caso bisogna chiudere il jumper su Vin) o con una tensione fornita alla morsettiera PWR (jumper chiuso su PWR); potete optare per la prima soluzione se pensate che le vostre lampade assorbano meno di 1,5 ampere, mentre se avete bisogno di più corrente dovete alimentare a parte lo shield, con un alimentatore adeguato.

Notate che chiudendo il jumper su Vin, il modulo Arduino deve essere alimentato a 12 V con un alimentatore in grado di erogare tutta la corrente occorrente, ossia quella che serve ad esso e quella richiesta dai LED.

LO SKETCH PER ARDUINO

La pagina html risiede nella memoria di programma di Arduino; infatti, come potete vedere dallo sketch (**Listato 1**) abbiamo previsto una serie di istruzioni (*prog_char string_X[] PROGMEM*) che "scrivono" la memoria di programma in fase di programmazione. In questo modo viene caricata l'intera pagina HTML (con tanto di funzioni javascript) che è poi restituita nel browser. All'accensione viene richiamata la routine SETUP, che inizializza lo shield ethernet e legge dalla memoria i valori R, G, B, che devono essere impostati sulle uscite. Se la funzione fader è stata attivata (si può attivare e disattivare direttamente dalla pagina web) il programma inizia a incrementare e decrementare i valori delle uscite RGB portando queste alternativamente da 0 a 255 e viceversa. In caso di interrogazione dalla pagina web, Arduino carica la pagina che ha in memoria e cliccando su un pulsante piuttosto che impostando un valore da 0 a 255 per i campi RGB, vengono

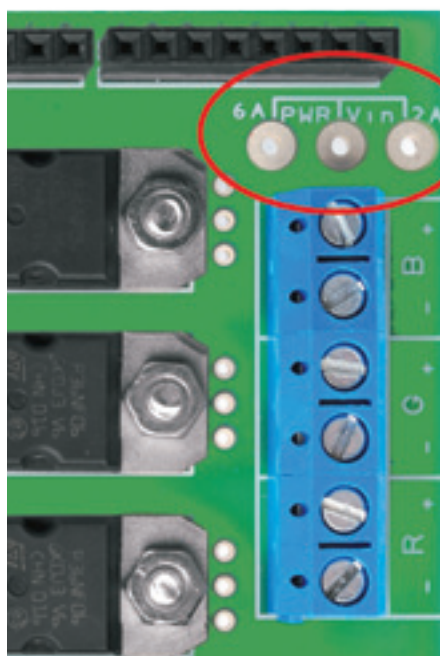
Listato 1

```
void
(int r, int g, int b)
{
    Serial.print("RGB: ");
    Serial.print(r);
    Serial.print(" ");
    Serial.print(g);
    Serial.print(" ");
    Serial.print(b);
    if (r>255) r=255;
    if (g>255) g=255;
    if (b>255) b=255;
    if (r<0) r=0;
    if (g<0) g=0;
    if (b<0) b=0;

    analogWrite(red, r);
    analogWrite(green, g);
    analogWrite(blue, b);
}
```

inviati alla stessa pagina, tramite il metodo GET, i valori impostati. Arduino legge dall'URL questi valori e agisce di conseguenza, impostando le relative uscite. La funzione che si occupa di gestire le uscite si chiama **RGB()**. Ad essa devono essere passati semplicemente i valori delle tre componenti cromatiche; Arduino imporrà di conseguenza le uscite analogiche.

Come è possibile notare dallo sketch, la routine acquisisce i tre dati e dopo un debug seriale controlla se i valori passati sono



**ALIMENTAZIONE
ESTERNA**



**ALIMENTAZIONE
DA ARDUINO**

*Il ponticello sullo shield
permette di prendere
l'alimentazione da Arduino se
la corrente non supera 2 A o
dalla morsettiera (fino a 6 A).*

100 W di luce a LED

La principale critica mossa alle lampadine a LED bianchi è che, tonalità della luce a parte, difficilmente possono sostituire le lampadine a incandescenza ed alogene nell'illuminazione degli ambienti; in pratica, ciò significa che per avere la stessa luce occorre montare più lampadine, rivoluzionando l'impianto d'illuminazione. Oggi, però, questo limite sembra sia stato superato, perché la Osram Sylvania (la consociata americana della storica casa OSRAM tedesca) ha presentato un prototipo di lampadina a LED ad alta efficienza in grado di emettere un flusso luminoso di ben 1.500 lumen, cioè l'equivalente di luce emessa da una lampada a incandescenza da 100 W di buona qualità, consumando, però, appena 14 Watt. Ciò significa aver raggiunto un'efficienza complessiva (cioè comprendendo il regolatore di corrente) di 107 lm/W. Secondo il costruttore, la durata di questa nuova lampadina è dell'ordine delle 25.000 ore e l'indice di resa cromatica è superiore all'80 %.



compresi tra 0 e 255; successivamente imposta le uscite PWM con la funzione **analogWrite**.

REALIZZAZIONE PRATICA

Bene, spiegato l'hardware ed il software, non dobbiamo fare altro che spendere due parole sulla costruzione dello shield RGB, ossia del circuito che dovete realizzare per alimentare i LED; Arduino e l'Ethernet Shield si comperano già fatti. Per realizzare lo shield RGB basta scaricare le tracce lato rame dal nostro sito internet www.elettronica.in.it, stamparle per ricavare le pellicole e procedere con la fotoincisione; una volta in possesso del c.s., unite le piazzole di comunicazione tra le due facce dello stampato

("vie") e disponete i componenti partendo dalle resistenze e dai MOSFET, che dovete coricare sulla basetta dopo averne piegato i terminali ad angolo retto,

stagnando le rispettive piazzole solo dopo averne stretto le alette metalliche con bulloni da 3 MA e dadi. Procedete con il diodo 6A60 (il cui catodo deve "guardare" verso l'interno dello stampato) e i tre LED rosso, verde e blu, ricordando che il catodo di tali diodi è l'elettrodo che sta dal lato smussato del corpo. Completate lo shield montandovi quattro morsettiere da c.s. a passo 5 mm ed altrettante file di contatti a passo 2,54 mm con terminali lunghi 20 mm: ne vanno due da 8 poli sul lato delle linee di I/O e due da 6 poli sul lato opposto. Terminato lo shield RGB, passate all'Ethernet Shield, sul quale dovete saldare analoghe file di contatti a passo 2,54 mm per il montaggio su Arduino e per accogliere e connettere lo shield RGB. Ora assemblate l'insieme, montando lo shield RGB sull'Ethernet Shield e quest'ultimo su Arduino, nel quale avrete già caricato lo sketch servendovi dell'apposito bootloader e della connessione USB; se il fondo, ovvero le saldature dello shield RGB fosse troppo vicino al connettore RJ45 dell'Ethernet Shield, consigliamo di interporre un pezzo di cartoncino o di acetato o polietilene. In alternativa, interporre delle file di contatti



Fig. 1

femmina a passo 2,54 mm tra i pin dello shield RGB e i contatti dell'Ethernet Shield.

LA PAGINA WEB

Concludiamo vedendo i comandi e le segnalazioni disponibili nella pagina web, illustrata nella Fig. 1; in essa troviamo un certo numero di pulsanti per impartire comandi, caselle d'opzione per scegliere alcune modalità di funzionamento e delle caselle dove scrivere i parametri. Partiamo dall'alto, dove troviamo i sette pulsanti *Invia query*, ognuno relativo ad un colore: ciascuno di essi invia al sistema la richiesta di impostare nella lampada la combinazione dell'accensione dei LED R, G e B per ottenere il colore corrispondente; ricevendo il comando,



Arduino imposta la relativa tinta. Sotto questi pulsanti troviamo tre caselle, con a destra il pulsante *Calculate*: da sinistra a destra, esse rappresentano i colori rosso, verde e blu. In ogni casella possiamo scrivere, con un numero compreso tra 0 e 255, l'intensità che vogliamo far assumere alla luce corrispondente; ad esempio, scrivendo 255 nella casella centrale facciamo illuminare alla massima intensità la luce verde. Il valore in ogni casella può essere fatto variare con i pulsanti sottostanti Red+, Green+ e Blue+ (che aumentano l'intensità luminosa di, rispettivamente, rosso, verde e blu) ovvero con Red-, Green- e Blue-, che riducono l'intensità. Facendo clic sul pulsante *Calculate*, vediamo di che colore apparirà la combinazione impostata, mentre con *Set colors* inviamo ad Arduino la richiesta di impostazione del colore corrispondente alla combinazione di intensità delle tre caselle. Va detto che il colore risultante è quello ottenibile da una lampada formata da LED rossi, verdi e blu montati in un involucro bianco che permetta di amalgamare le tre tinte; montando LED da soli vedremo i diodi emettere ognuno la propria luce. L'ultima sezione della pagina

per il MATERIALE

Tutto il firmware necessario al funzionamento di questo progetto è scaricabile dal sito della rivista all'indirizzo www.elettronica.in

I componenti utilizzati in questo circuito sono tutti facilmente reperibili.

L'RGB Shield è anche disponibile in scatola di montaggio al prezzo di 12,00 Euro IVA compresa; il kit comprende tutti i componenti, le minuterie ed il CS (codice prodotto RGB_Shield). Per completare il dispositivo è necessario uno Shield Ethernet, una scheda Arduino e le strip di led.

Il materiale va richiesto a:

Futura Elettronica, Via Adige 11,
21013 Gallarate (VA)

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

web che ci resta da descrivere è quella relativa al fader: si tratta dell'effetto di variazione di colore continua, attivabile facendo clic nella casella d'opzione *On* accanto alla voce omonima (*Fader*); per disattivare questa funzione bisogna fare clic su *Off*. Attivando il fader, vedremo la luce della lampada cambiare ciclicamente di tonalità a partire dalla tinta attualmente impostata; è anche possibile scegliere tra più velocità di esecuzione del ciclo, nel menu a tendina cui si accede facendo clic nella casella centrale *inc/speed* posta sopra la sezione *Fader*.

Elettronica In
Rivista mensile di elettronica applicata

Sei un appassionato di elettronica? Ti piace progettare, costruire, modificare qualsiasi cosa purché abbia a che fare con transistor, display o microcontrollori? Vorresti condividere le tue conoscenze con gli altri lettori?

Cosa aspetti? ... Entra subito a far parte dei

COLLABORATORI HARDWARE della nostra Rivista

Realizzare un progetto per Elettronica In è molto semplice!

Dopo aver concordato con la redazione le linee guida del progetto, ti invieremo il materiale necessario con quale farai le prove e realizzerai un primo prototipo, anche su breadboard. Successivamente ci manderai il disegno dello stampato; alla sua realizzazione pratica provvederemo noi che ti rimanderemo la basetta ed i componenti per realizzare il prototipo finale, quello che poi andremo a fotografare e che presenteremo sulle pagine della rivista. Agli schemi elettrici definitivi provvederemo sempre noi mentre a te non rimarrà altro che descrivere il progetto nello stile della rivista.

Manda una mail a: redazione.hardware@elettronica.in

Non perdere questa opportunità!
Contattaci oggi stesso!

Perché autonomi?

Dopo aver letto delle incredibili capacità che hanno raggiunto i velivoli senza pilota e degli sviluppi futuri di questa tecnologia anche in campo civile, ecco la ciliegina sulla torta: il nostro primo progetto di quadricottero. Ma non di un normale quadricottero radiocomandato, di un giocattolo o poco più, bensì di un vero e proprio UAV in grado di compiere delle missioni autonome, come il bimotore presentato nei precedenti numeri della rivista. Il massiccio impiego delle nuove tecnologie in questo campo, dai sensori di vario tipo, ai sistemi intelligenti di controllo dei motori e dei servomeccanismi, alla accresciuta potenza dei sistemi embedded, ha consentito in primis di realizzare dei velivoli controllati a distanza molto più stabili e più facilmente pilotabili. L'utilizzo dei sistemi di rilevamento della posizione (GPS), la disponibilità di mappe digitali e la programmabilità del tutto ha, successivamente, reso possibile trasformare questi modelli in veri e propri velivoli in grado di compiere delle missioni autonome. Per realizzare tutto ciò è necessario integrare il top dell'hardware e software: è proprio per questo motivo che le gare tra studenti più impegnative hanno al centro i veicoli autonomi, come quelle organizzate dall'AUVSI di cui ci occupiamo più avanti. Un modo di mettere in pratica le competenze acquisite in vari settori, dall'informatica, all'elettronica, alla meccanica.

È questo, in gran parte, anche lo spirito della RoboCup Junior la cui stagione 2012 sta per riprendere dopo la finale nazionale di Catania e quella mondiale di Istanbul. Di questo e di molto altro abbiamo parlato col Prof. Giovanni Marciànò coordinatore della Rete di scuole per la RoboCup Junior Italia che organizza le manifestazioni nazionali.



IN QUESTO NUMERO

108 ■ NEWS ED EVENTI

In volo o a terra, sopra o sott'acqua: purché in grado di compiere delle missioni autonome e ritornare alla base: sono le gare organizzate ogni anno dall'AUVSI, l'organizzazione che ha come obiettivo la diffusione delle tecnologie legate ai veicoli autonomi.

111 ■ A TU PER TU ROBOCUP JUNIOR ITALIA, AL VIA LA QUARTA STAGIONE

A poche settimane dalla ripresa delle attività legate alla stagione 2012, abbiamo incontrato il Prof. Giovanni Marciànò coordinatore della Rete di scuole per la RoboCup Junior Italia.

114 ■ DRONE ZONE REALIZZIAMO UN QUADRICOTTERO CON CAPACITÀ DI VOLO ASSISTITA

Non un semplice quadricottero radiocomandato ma un vero e proprio UAV in grado di atterrare e decollare autonomamente ed effettuare delle missioni. Telemetria bidirezionale con possibilità di modificare le configurazioni anche in volo.



AUVSI, veicoli autonomi in gara tra loro



di Arsenio Spadoni



In volo o a terra, sopra o sott'acqua: purché in grado di compiere delle missioni autonome e ritornare alla base.

L' AUVSI (Association for Unmanned Vehicle Systems International) è un'associazione internazionale senza fini di lucro che ha come obiettivo la diffusione della robotica - ed in particolare delle tecnologie legate ai veicoli autonomi - tra gli studenti delle scuole superiori e dell'Università. La progettazione e la realizzazione di questi mezzi fornisce agli studenti l'opportunità di esperienze non solo divertenti, ma anche di notevole valenza didattica nel campo delle cosiddette STEM (Science, Technology, Engineering and Math). L'AUVSI organizza conferenze e seminari, pubblica riviste specializzate e, soprattutto, organizza e supporta gare nazionali e internazionali tra veicoli autonomi. Per la loro particolarità e per l'altissimo contenuto tecnologico, queste manifestazioni attirano l'attenzione dei media e del grande pubblico e sono sponsorizzate dalle più importanti aziende hi-tech del mondo. Le gare vengono supportate anche dagli enti governativi che operano nel campo della innovazione e della ricerca, sempre alla ricerca di menti brillanti da



coinvolgere nelle loro attività. Anche se i veicoli autonomi coinvolti in queste sfide operano in campi diversi – acqua, terra e aria – le problematiche sono molto simili, dagli algoritmi che ne stabiliscono il comportamento, ai sistemi di gestione dei sensori e dei servocomandi.

Quest'anno l'AUVSI ha organizzato cinque gare tra veicoli autonomi:

- *Intelligent Ground Vehicle Competition (IGVC)*, gara tra veicoli terrestri;
- *International RoboBoat Competition*, gara tra imbarcazioni autonome;
- *Student Unmanned Air Systems (SUAS) Competition*, gara tra velivoli unmanned;
- *International RoboSub Competition*, gara tra mezzi sottomarini
- *International Aerial Robotics Competition (IARC)*, gara tra velivoli unmanned in grado di compiere delle missioni autonome.

Intelligent Ground Vehicle Competition (IGVC)

Si tratta di una gara tra rover autonomi che quest'anno si è svolta dal 3 al 6 di giugno a Rochester nel Michigan.

La competizione consiste nel fare compiere al veicolo una serie di missioni sempre più complesse che richiedono il superamento di ostacoli e difficoltà di varia natura. Le tecnologie coinvolte nell'IGVC



riguardano una vasta gamma di discipline, tutte di grande attualità sia sul piano industriale che su quello dell'istruzione. I premi dell'IGVC sono numerosi e piuttosto consistenti, così come quelli di tutte le altre manifestazioni. Quest'anno il premio finale è stato vinto dall'Università dello Stato della California di Northridge.

International RoboBoat Competition

La competizione, che quest'anno si è svolta a Virginia Beach dal 9 al 12 giugno, vede in gara modelli di natanti "intelligenti" in grado di compiere missioni autonome, pattugliare dei corsi d'acqua, rilevare ed evitare ostacoli nonché attraccare da soli. Anche in questo caso le tecnologie messe in campo sono varie e piuttosto sofisticate compresa la visione artificiale.

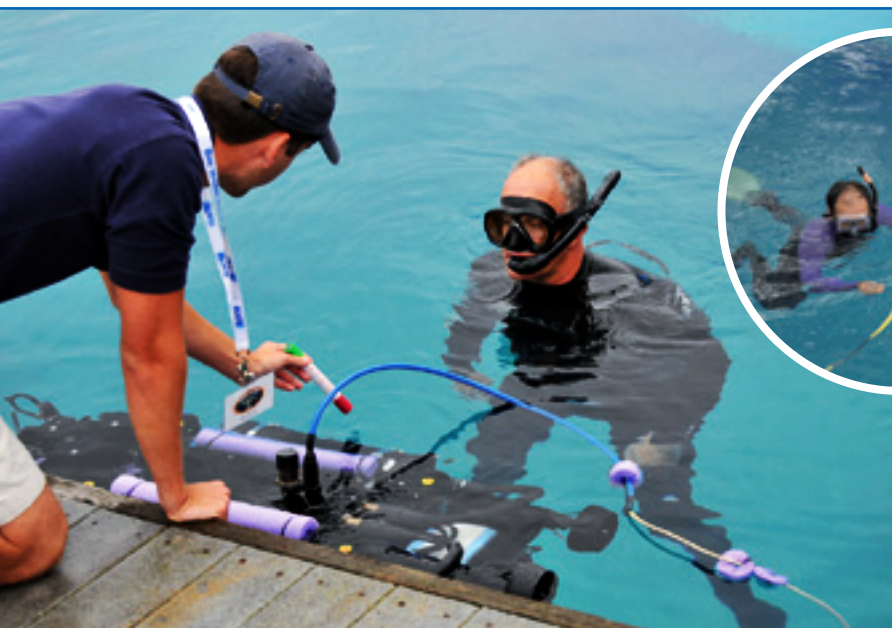
Quest'anno il primo premio di 6.000 dollari è stato vinto dalla Università di Rhode Island.



Student Unmanned Air Systems (SUAS) Competition

È una delle competizioni più affascinanti in quanto vede in gara velivoli senza pilota in grado di compiere delle missioni autonome. Quest'anno la manifestazione si è tenuta dal 15 al 19 giugno a Webster Field nel Maryland. La gara, oltre a stimolare e promuovere interesse nei confronti dei velivoli unmanned, richiede ai vari team di progettare un





Ecole de technologie supérieure di Montreal che si è aggiudicato i 7.000 dollari del primo premio.

International Aerial Robotics Competition (IARC)

È una sfida tra robot volanti in grado di svolgere delle missioni completamente autonome in ambienti ostili, anche di guerriglia urbana. La gara di quest'anno si è svolta dall'8 al 12 di agosto in North Dakota, a Grand Forks, che è anche la sede del Centro di Eccellenza per la Ricerca, Istruzione e Formazione sui velivoli UAS delle forze armate statunitensi. Proprio per questo motivo e per la sua peculiarità, la manifestazione è stata seguita con particolare interesse dai rappresentanti del Dipartimento della Difesa USA.

La missione che debbono compiere i velivoli è molto complessa: infiltrarsi all'interno di una base nemica, recuperare un sistema che alimenta un flash, sostituirlo con uno analogo e rientrare alla base. Il tutto senza farsi scoprire.

Ovviamente vengono utilizzati velivoli ad ala rotante, tipicamente quadricotteri. Quest'anno il primo premio era di ben 30.000 dollari con un monte premi complessivo di 80.000 dollari. □

sistema in grado di condurre missioni aeree con capacità di volo autonomo nonché di trasportare payload di vario genere, dai sensori ambientali ai sistemi di ripresa. Inoltre i vari team sono tenuti a presentare tutti i dettagli tecnici dei loro progetti. Complessivamente la manifestazione mette in palio oltre 70.000 dollari. Quest'anno il primo posto se lo è aggiudicato l'Università dello Stato dello Utah, mentre al secondo posto è arrivata la North Carolina State University.

I team che si sono classificati ai primi posti utilizzavano come autopilota piattaforme "Paparazzi" e "Piccolo" presenti sul mercato ormai da diversi anni; da segnalare anche numerosi team che hanno iniziato ad utilizzare la piattaforma APM (ArduPilot Mega), la stessa che stiamo utilizzando nei nostri progetti di velivoli UAV. Il migliore risultato con questa piattaforma è stato quello dell'indiana Delhi Technical University.



International RoboSub Competition

Svoltosi dal 12 al 17 luglio a San Diego, in California, questa manifestazione, organizzata insieme al Naval Research americano (ONR), ha l'obiettivo di promuovere lo sviluppo di veicoli sottomarini autonomi (AUVs) che debbono eseguire missioni di vario tipo in un ambiente subacqueo. La gara di quest'anno è stata vinta dal team canadese SONIA della



RoboCup JUNIOR ITALIA, AL VIA LA QUARTA STAGIONE

A poche settimane dalla ripresa delle attività legate alla stagione 2012, abbiamo incontrato il Prof. Giovanni Marcianò coordinatore della Rete di scuole per la RoboCup Junior Italia.

di ARSENIO SPADONI

Come e quando è nata la Rete di scuole per la RoboCup Junior Italia?

È nata a Taranto, precisamente a Didamatica 2008, il Convegno promosso annualmente dall'AICA (Associazione Italiana per l'Informatica ed il Calcolo Automatico) che si propone di fornire un quadro delle ricerche, degli sviluppi innovativi e delle esperienze in atto nel settore dell'Informatica applicata alla Didattica, nei diversi domini e nei molteplici contesti di apprendimento. Quell'anno io coordinavo una sessione pomeridiana che si proponeva di fare il punto sul futuro della robotica nelle scuole italiane. Fin dal 2000 avevamo visto e documentato che i bambini e i ragazzi apprendevano in maniera più positiva quando avevano a che fare con i kit robotici anziché col PC e con altri strumenti multimediali, quando li costruivano e li facevano gareggiare tra loro.

In quel contesto nacque l'idea di organizzare ogni anno una manifestazione nazionale per diffondere l'utilizzo didattico della robotica nelle scuole, con un bando attorno al quale si sarebbero potute aggregare scuole di tutte le regioni italiane. In quegli anni nel nostro paese si organizzavano numerose gare di robotica, tante belle esperienze (Catania, Roma, Pisa, ecc.), ma che avevano tutte il difetto di svolgersi in ambito locale. Da questo pun-



to di vista ci è stata d'aiuto l'autonomia scolastica che consente alle varie scuole di mettersi in rete e come rete abbiamo autonomia di ricerca e sperimentazione. L'obiettivo era quello di creare quel quadro nazionale in cui tutte le esperienze, nate e vissute a livello locale, potessero portare i loro frutti in un contesto nazionale condiviso da tutti. E se la Rete è nazionale, anche la manifestazione sarà nazionale!

Tra i vari modelli di gara, la scelta cadde sulla RoboCup che proprio quell'anno era in fase di evoluzione con nuove regole che, eliminando quelle che erano diventate soltanto delle prove di forza (quindi con molta meccanica e motori di potenza), riportava l'attenzione sulla sfida tra intelligenze, limitando anche i costi dei robot. Da qui nacque il manifesto che fu mandato alle scuole che sapevamo interessate.

Tra i vari modelli di gara, la scelta cadde sulla RoboCup che proprio quell'anno era in fase di evoluzione con nuove regole che, eliminando quelle che erano diventate soltanto delle prove di forza (quindi con molta meccanica e motori di potenza), riportava l'attenzione sulla sfida tra intelligenze, limitando anche i costi dei robot. Da qui nacque il manifesto che fu mandato alle scuole che sapevamo interessate.

Dopo tre anni, qual è il bilancio di questa esperienza?

La manifestazione è cresciuta molto più di quanto immaginassimo. In questi tre anni abbiamo organizzato una manifestazione nazionale a Torino, una a Vicenza e una a Catania. Abbiamo avuto a che fare con tre contesti logistici, abbiamo girato l'Italia. Abbiamo mantenuto aperto il discorso anche con chi all'inizio speravamo condi-



videsse il progetto nazionale e che ancora non ne fa parte.

Quest'anno ho seguito personalmente tutte le gare ed ho notato come ci sia ancora questa separazione tra struttura nazionale e realtà locali che ancora non ne fanno parte. Pensa che anche per il 2012 continuerà questa rivalità?

Per quanto riguarda questo aspetto abbiamo firmato un protocollo d'intesa nazionale con tutti i soggetti interessati. Però alle firme debbono seguire i fatti. In ogni caso abbiamo demandato ai nostri superiori la risoluzione di questo genere di problemi.

Per quest'anno cosa si auspica?

Noi miriamo alla diffusione, non ci interessa che un nostro team vinca il campionato mondiale ma piuttosto che sempre più scuole partecipino a questa iniziativa.

Da questo punto di vista che tipo di azioni state facendo? Nella nostra zona, ad esempio, dove ci sono molti ITIS e IPSIA ad indirizzo elettronico, questo genere di manifestazioni sono decisamente poco conosciute...

Negli anni passati la robotica educativa è stata spesso al centro della convegnistica e di articoli presentati sulle riviste specia-

listiche, pubblicazioni che arrivano a tutte le scuole insieme alle comunicazioni del Ministero sulla RoboCup Junior. Mi ricordo almeno quattro dossier sull'argomento presentati sulle riviste del gruppo *Le Monnier*. Se poi la segreteria non la passa al professore competente è un altro discorso, purtroppo noi non possiamo sanare i problemi della scuola italiana. In ogni caso le comunicazioni istituzionali ci sono. Da questo punto di vista è anche importante l'attenzione dei media, sia dei quotidiani nazionali che in occasione delle manifestazioni pubblicano sempre un articolo sull'evento, sia di riviste più specialistiche come la vostra che, con più costanza, si occupano di tutti gli eventi ma anche delle tecnologie. È assurdo che scuole come quelle da lei citate non conoscano o non prendano parte a queste iniziative.

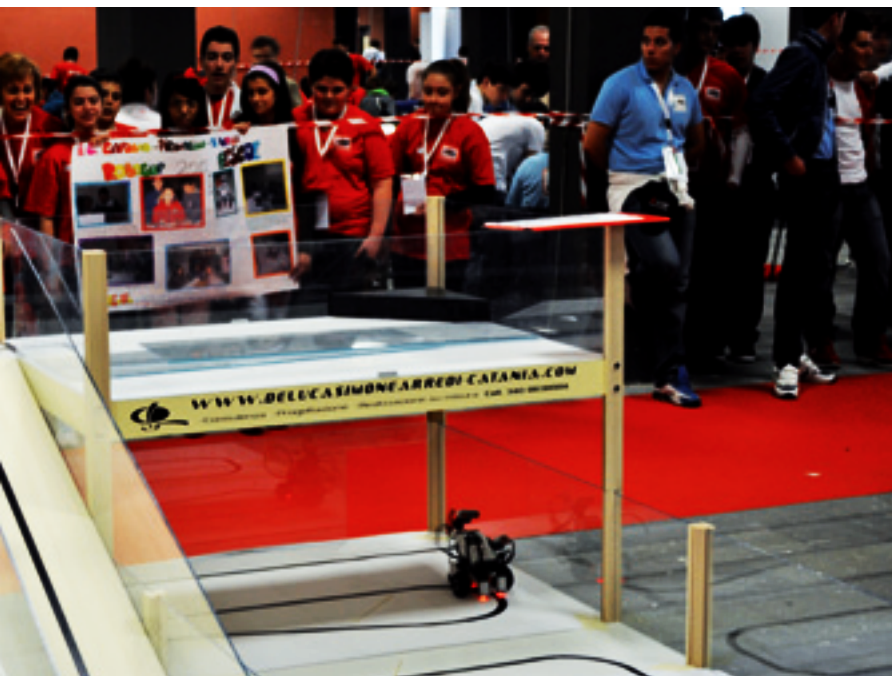
Effettivamente la nostra rivista viene inviata a tutti gli ITIS e gli IPSIA italiani; che poi vada all'insegnante giusto è un altro discorso. A proposito, quante sono le scuole nuove quest'anno?

Al momento abbiamo la scuola di Pescara e un paio di circoli didattici. Noi manderemo un invito a tutti quelli, non della Rete, che hanno partecipato alla gara di Catania proponendogli di entrare nella Rete.

Quali vantaggi ha una scuola che aderisce alla Rete?

Quest'anno stiamo inserendo dei meccanismi per cui, iscrivendosi alla Rete, la spesa complessiva sostenuta sarà, alla fine, inferiore a quella di chi partecipa solo alla gara nazionale.

Stiamo anche tentando di dare altri vantaggi alle scuole della Rete, per entrare nella quale è sufficiente la delibera del Consiglio di Istituto ed il versamento di una quota molto bassa. A quel punto, però, il dirigente entra automaticamente nel comitato direttivo mentre il docente entra nel comitato tecnico e può immediatamente usufruire delle comunicazioni tecniche che sviluppiamo all'interno della Rete, entra in un link di comunicazione con i suoi colleghi e, dovendo avviare un discorso di robotica nella sua scuola, riceve le informazioni tecniche per essere subito operativo. E nel percorso di pre-



parazione della gara ha tutto il supporto logistico.

Abbiamo fatto di tutto affinché una scuola nuova che voglia entrare, non debba perdere - come gli altri - almeno 2 o 3 anni per acquisire l'esperienza necessaria. Anche la documentazione che renderemo disponibile per chi si iscriverà permetterà a tutti di partire al meglio: ci saranno i report tecnici di tutti i kit che hanno partecipato alle precedenti edizioni, le motivazioni che hanno portato a scegliere questo o quell'altro robot, i listati di programmazione, gli algoritmi e tanto altro ancora.

Questa struttura che vi siete dati quale tipo di evoluzione potrà avere in futuro?

Quando a Taranto ragionavamo sul futuro della robotica nella scuola italiana, la risposta era sempre la stessa: "Ci vuole un momento in cui le scuole che si stanno impegnando in questa attività possano confrontarsi, stimolarsi nel confronto, misurarsi, per andare avanti."

La struttura nazionale poteva essere privatistica, come negli Stati Uniti, oppure universitaria, come per altre attività. Oppure poteva essere ministeriale: il Ministero già organizza gare di elettronica, informatica, gare delle varie specializzazioni degli istituti professionali.

Ma anche un'azienda privata che avesse promosso a livello nazionale una gara di robotica, poteva essere l'innesco che portava a quell'obiettivo, per noi importante, della diffusione della robotica nella scuola italiana. È un settore nuovo in cui la ricerca non finisce mai.

Attualmente nessuno di noi lavora a tempo pieno per la RoboCup, ognuno ha il suo lavoro e dà il suo contributo per passione perché crede in questo percorso. In tre anni questo meccanismo è diventato molto ampio e la tensione che c'è intorno al lavoro di preparazione è sempre maggiore. Con realtà differenti ed esempi molto positivi (come le scuole del Triveneto che hanno fatto le loro gare di selezione).

Ad oggi abbiamo maturato la validità della formula e tutte le scuole della Rete se sono ancora nella Rete è perché quello che hanno ricevuto è di gran lunga superiore a quel minimo finanziamento che hanno dato. Il problema è ancora quello



della dimensione, vorremmo che il modello che alla buona hanno fatto in Veneto venisse regolamentato, fermo restando che la finale sappiamo già dov'è ma aspettiamo di capire quanti posti Trento è in grado di supportare logisticamente. In tutti questi casi ci occupiamo di tematiche di cui noi non siamo esperti. Ormai è arrivato il momento di dare all'organizzazione una struttura più seria sotto tutti i punti di vista.

Noi possiamo garantire le qualità di contenuto. Non solo i ragazzi ma anche i professori hanno imparato tantissimo dall'esperienza. Abbiamo riconfermato con la RoboCup che la scuola c'è, basta creare l'occasione! La potenzialità che i ragazzi hanno *dentro* riusciamo a farla esprimere.

Molti dei problemi che abbiamo affrontato sul campo non sono scolastici, ma logistici, organizzativi e finanziari e appartengono ad altre professionalità: se ci fosse l'azienda interessata ad occuparsi della parte logistica e organizzativa noi saremmo ben felici.

In occasione della finale mondiale, la formula RoboCup ha acceso le fantasie di molti, ma noi attualmente non siamo neppure in grado di supportare logisticamente una squadra che volesse andare all'estero. □

*L'intervista completa al Prof. Giovanni Marcianò è disponibile sul nostro blog
<http://blog.elettronica.in.it>*

COME ADERIRE ALLA RETE NAZIONALE DI SCUOLE

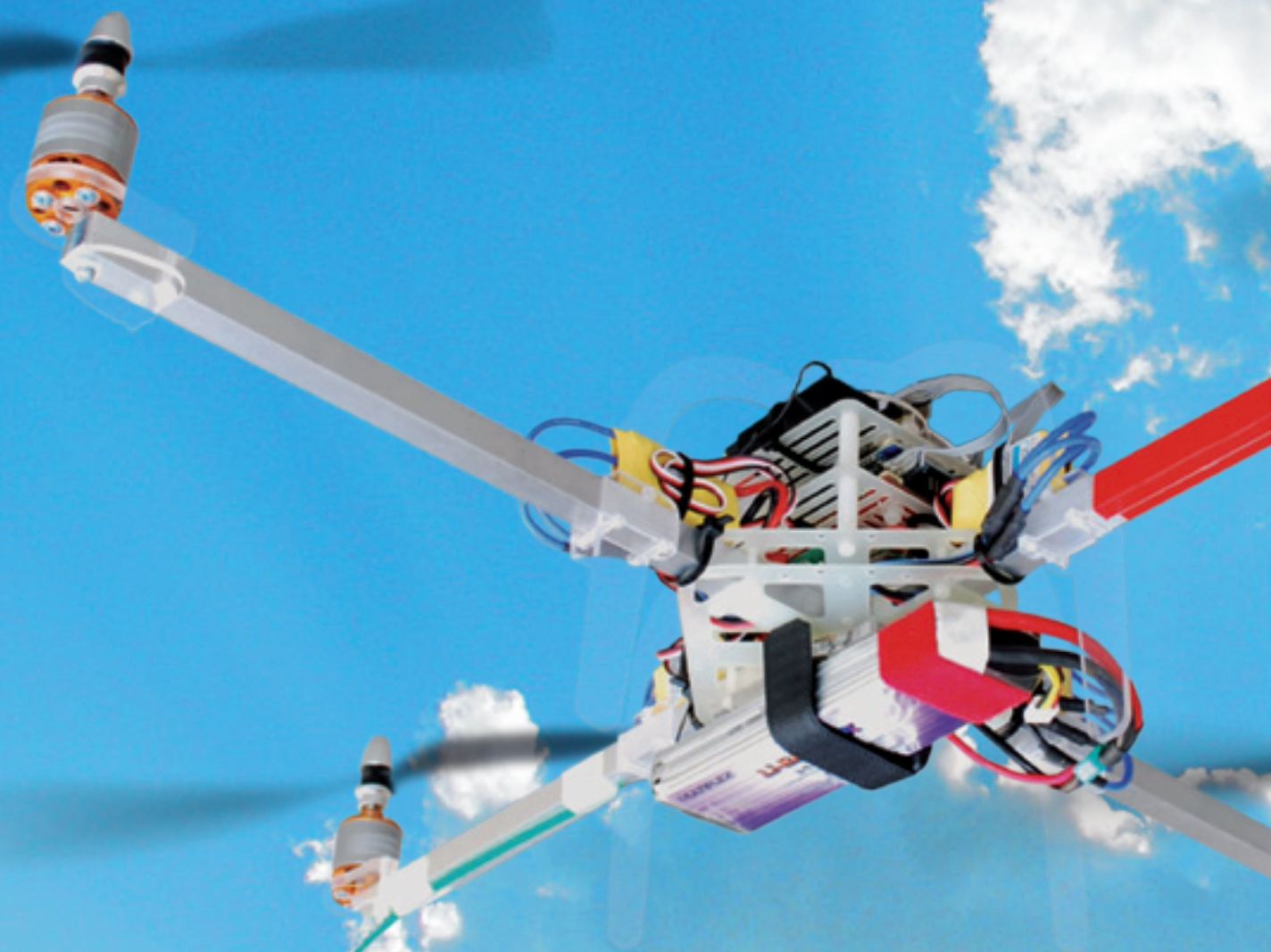
ENTRO IL 30 SETTEMBRE 2011 inviare l'estratto della deliberazione del Consiglio di Istituto relativo all'adesione alla Rete RCJ Italia - ai sensi art. 7 DPR 275/99 a:

**Dirigente scolastico - IIS
Marconi Galletti
Via Oliva 15- 28845
DOMODOSSOLA (VB)**

Quindi dare comunicazione via email a dirigente@robocupjr.it specificando:

1. Nome, recapito telefonico fisso e mobile, e-mail del Dirigente scolastico (Comitato Gestione)
2. Nome, recapito telefonico fisso e mobile, e-mail del Docente referente (Gruppo Tecnico)

Non un semplice quadricottero radiocomandato ma un vero e proprio UAV in grado di atterrare e decollare autonomamente ed effettuare delle missioni. Telemetria bidirezionale con possibilità di modificare le configurazioni anche in volo.



REALIZZIAMO UN QUADRICOTTERO CON CAPACITÀ DI VOLO ASSISTITA

di MARCO MAGAGNIN

Nei numeri precedenti vi abbiamo presentato due progetti di sistemi adatti a controllare il volo di un modello di aereo. In particolare il secondo progetto, Ardupilot Mega, basato sulla piattaforma Arduino Mega e corredato di una notevole serie di accessori, si avvicina molto allo stato dell'arte per sistemi di questo tipo non professionali.

In questo numero vi proponiamo un ulteriore progetto basato sul sistema Ardupilot Mega (APM) destinato al controllo di elicotteri e multicotteri, una particolare configurazione di mezzi volanti ad ala rotante che somiglia ad un ragno, con un corpo centrale ed un certo numero di bracci, alle estremità di ciascuno dei quali sono montati uno o due motori con le relative eliche.

Il vantaggio operativo della configurazione multicottero sta nella capacità di operare in spazi ristretti e nella possibilità di eseguire manovre impossibili per un aereo come, per esempio:

- Decollo e atterraggio in verticale in spazi ristretti,
- Holding ad altezza, posizione ed orientamento fissi,
- Possibilità di volo traslato in tutte le direzioni, mantenendo un orientamento qualsiasi,
- Individuazione e mantenimento

dell'orientamento verso un POI (Point Of Interest); è la funzione che permette di individuare, fissare, girare attorno o inseguire uno specifico target.

La semplicità meccanica e costruttiva lo fanno preferire ad un elicottero per taluni impieghi come l'utilizzo in spazi ristretti o all'interno di edifici od aree dove un malfunzionamento di un elicottero potrebbe arrecare danni eccessivi.

I principali svantaggi della configurazione multicottero stanno nella difficoltà di stabilizzazione, che richiede abilità di pilotaggio maggiori rispetto ad un elicottero, e la incapacità "genetica" della architettura di sopravvivere al malfunzionamento di uno dei motori.

Il primo svantaggio viene generalmente attenuato adottando sistemi di stabilizzazione e navigazione hardware/software come il nostro Ardupilot Mega, che nella configurazione per le ali rotanti prende il nome di ArduCopterMega.

La seconda caratteristica negativa non ha soluzione e restringe drasticamente l'uso dei multicotteri a luoghi e missioni dove non sia possibile procurare danni a cose o persone in caso di malfunzionamenti.

Escluso totalmente l'utilizzo nelle vicinanze di persone, animali, veicoli, luoghi sensibili dal punto di vista della sicurezza, ecc.

Quando operiamo con modelli volanti teniamo sempre presenti e rispettiamo tutte le norme che regolano il settore oltre che il buon senso comune:

- Volare in luoghi adatti come i campi volo per modelli,
- Dotarsi di una assicurazione Responsabilità Civile adatta alle attività di aeromodellismo
- Operare sempre con il modello in vista,
- Il sistema di pilotaggio deve comprendere un set Trasmittente e Ricevente RC omologati,
- Il sistema di controllo deve essere SEMPRE a priorità di comando del pilota, in qualunque modalità un intervento sugli stick deve sempre essere recepito dal modello sovrapponendosi a qualsiasi altro input,
- Ricordiamoci che in aviazione la prima responsabilità in caso di incidente è sempre personale e sempre del pilota.

ARCHITETTURA MULTICOTTERO

Nel corso del tempo sono state

sviluppate diverse configurazioni di multicotteri, per risolvere i problemi di stabilità, di carico utile, di possibilità di recupero in caso di guasto ad un motore o semplicemente per estetica.

Tutte hanno in comune la struttura a "ragno" costituito da un corpo centrale dove trovano alloggio l'elettronica ed il carico utile, dal quale si dipartono un numero variabile di bracci, all'estremità di ciascuno dei quali sono installati uno o due motori con le relative eliche.

Le diverse configurazioni prendono il nome dal numero dei bracci e/o dei motori, tricottero un modello con tre bracci e tre motori, quadricottero un modello con quattro bracci e quattro motori, esacottero con sei bracci e sei motori oppure con tre bracci e due motori per braccio.

Le configurazioni con più di quattro bracci sono state sviluppate sia per aumentare il carico utile trasportabile, sia con l'intento di risolvere il problema del guasto ad un motore, ripartendo il carico in volo sui motori rimasti, riconfigurati automatica-

mente in modo opportuno.

Noi ci siamo orientati su una configurazione tradizionale a quattro bracci e quattro motori sulla quale installare il sistema di controllo APM con il software Arducopter-Mega.

In un quadricottero sono possibili due configurazioni, denominate a "+" e a "X".

Nella configurazione a "+" si considera uno dei bracci come "frontale" e tutti gli orientamenti ed i controlli sono riferiti a quest'ultimo.

Nella configurazione a "X" la posizione frontale è definita da due bracci ciascuno a 45 gradi rispetto alla direzione di movimento in avanti; questa configurazione è utile per avere libero il "muso" del quadricottero nel caso si vogliano montare frontalmente telecamere e macchine fotografiche.

Nella configurazione a "+" il motore frontale ed il motore posteriore girano in senso orario e montano eliche destrorse mentre i motori destro e sinistro girano in senso antiorario e montano eliche sinistrorse. Questo per compensare le coppie generate dal senso di rotazione dei motori e semplificare il controllo della stabilità e della navigazione.

Combinando le velocità di rotazione dei vari motori è possibile ottenere tutte le capacità di manovra proprie dei multicotteri e descritte all'inizio. Le caratteristiche principali della piattaforma APM con il software ArduCopterMega (ACM) sono:

- Possibilità di gestire diverse configurazioni di modelli ad ala rotante, sia elicotteri che multicotteri,
- Capacità del software di determinare automaticamente la configurazione hardware e la tipologia dei componenti collegata; ad esempio è in grado di riconoscere automaticamente quale tipo di GPS è collegato e con quale tipologia di firmware è configurato,



Fig. 1

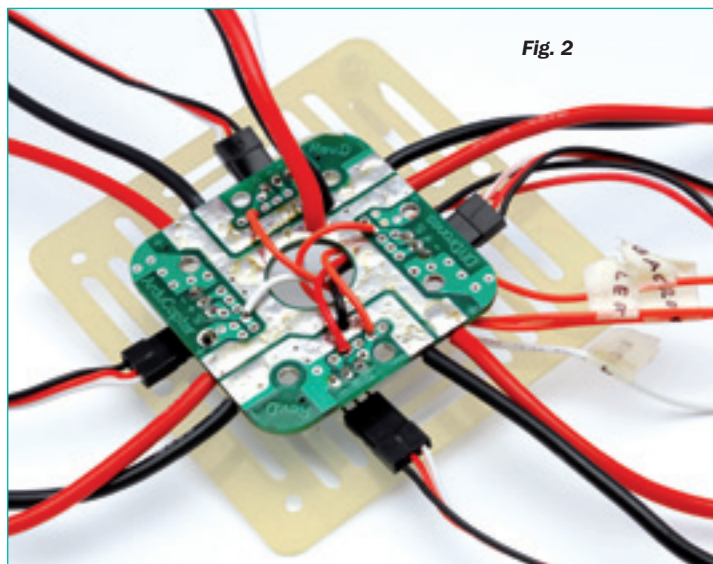


Fig. 2

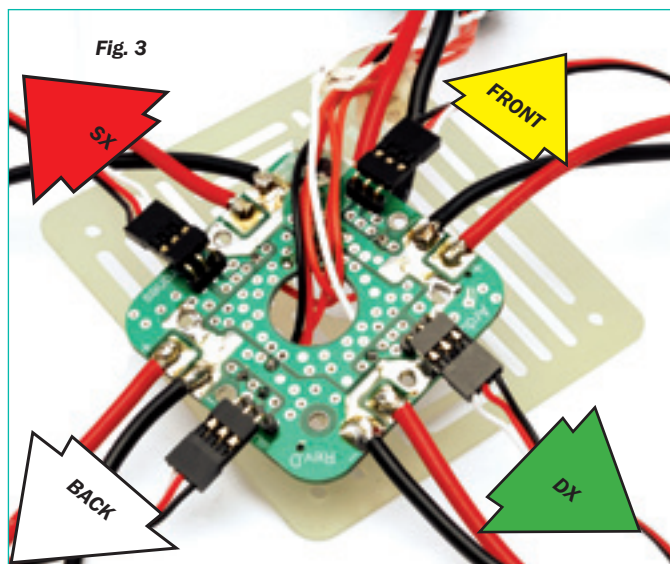


Fig. 3

- Disponibilità della applicazione software di configurazione ArdupilotMega Planner che permette di:
 - caricare il software su APM senza necessità di compilazione e quindi senza la obbligatorietà di utilizzare l'IDE di sviluppo di Arduino,
 - possibilità di modificare i parametri di configurazione e di calibrare i controllori PID on line, anche in volo,
 - possibilità di impostare le caratteristiche di una missione in modo grafico e di caricarla nel sistema,
- Possibilità di integrare e configurare nuovi sensori come il magnetometro, ed il sonar per il controllo del modello a poca distanza dal suolo,
- Possibilità di sfruttare appieno la telemetria bidirezionale, basata sul protocollo Mavlink, e quindi di impartire comandi e modificare le modalità di funzionamento del software da remoto,
- Nuove modalità di controllo di volo come Auto take off e Auto landing,
- Disponibilità di un registratore dei dati di volo per conservare i dati di una missione e poterli analizzare in differita.

Procediamo con ordine partendo con l'elenco dei componenti utilizzati nel nostro progetto.

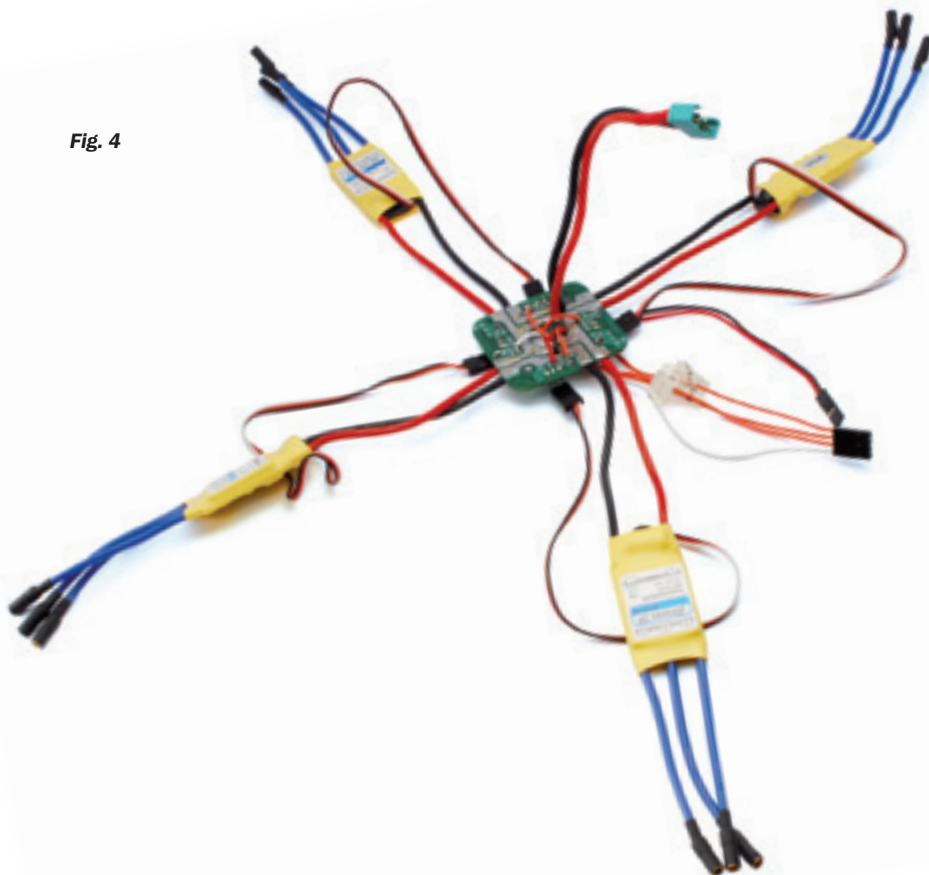
Per il modello abbiamo utilizzato:

- Il modello di quadricottero ArduCopter Quad v1.0, in scatola di montaggio, che permette di costruire un quadricottero di 60 cm di interasse tra i motori (ingombro in volo 90cm) e del peso

di circa 1200 grammi,

- Per la motorizzazione possiamo optare per due scelte:
 - La motorizzazione di base prevede quattro motori brushless AC 2830-358 850 Kv, con quattro regolatori ESC da 20 A e quattro eliche 10 x 45, due destre e due sinistre,
 - in alternativa, se preferite

Fig. 4





UAVForge, anche il DARPA interessato agli UAV amatoriali

Si chiama UAVForge la nuova iniziativa del DARPA (Defense Advanced Research Projects Agency), l'agenzia governativa americana incaricata dello sviluppo di nuove tecnologie per uso prevalentemente militare. Con un investimento di 2 milioni di dollari, UAVForge (<http://www.uavforge.net/>) si propone di supportare e raccogliere attorno al sito appositamente creato una comunità di appassionati di questo genere di velivoli, ovvero di autocostruttori, progettisti e geek che sono interessati alla costruzione di piccoli UAV. Insomma, una sorta di DIYDRONES sponsorizzato dai militari.

L'obiettivo finale è quello di selezionare il miglior progetto possibile in grado di rispondere a determinate specifiche e di utilizzare tale progetto in ambito militare realizzando un certo numero di esemplari: un modo per ridurre notevolmente le spese di progettazione e sviluppo e per fare emergere nuovi talenti in questo settore. Ovviamente nulla si fa per nulla, e il progettista o il team di progettisti che vincerà si aggiudicherà un premio di 100.000 dollari.

L'iniziativa è rivolta prevalentemente ai gruppi di appassionati che potranno trovare nell'apposito sito risorse di vario genere e che potranno confrontarsi con i team concorrenti.

Tutti dovranno proporre un video della propria realizzazione: a poche ore dall'avvio dell'iniziativa sono già numerosi i video postati, compresi alcuni provenienti dal nostro paese.

L'iniziativa prevede anche una sorta di gara finale durante la quale i 10 team che avranno raggiunto il punteggio più alto si sfideranno dal vivo tra loro; il vincitore si aggiudicherà l'appalto per la produzione di un lotto iniziale di 15 UAV.

realizzare una versione più performante possono essere montati quattro motori brushless AC 2836-358 880 Kv con quattro regolatori da 30 A e quattro eliche 12 x 45, sempre due destre e due sinistre. Noi abbiamo preferito questa opzione,

- Una batteria LIPO 3 celle da 3200 mA,
- Una trasmittente RC funzionante su una frequenza omologata per modellismo con emissione PPM per il controllo dei servocomandi,
- Una ricevente compatibile con la trasmittente, sempre con

uscita di segnale PPM verso i servocomandi con almeno 8 canali.

Per la parte elettronica abbiamo utilizzato:

- Un modulo ArduPilot Mega - Arduino Compatible UAV Controller w/ ATmega2560,
- Un modulo ArduPilotMega IMU Shield/OilPan Rev-H,
- Un modulo GPS mediatek (MediaTek MT3329 GPS 10Hz + Adapter Basic) con relativo cavo di prolunga da 25 cm (EM-406/uBlox/MTK Adapter Cable 25 cm),
- Un magnetometro HMC5843 - Triple Axis Magnetometer Rev 1.2 ,
- Un modulo Sonar XL-Maxsonar EZ0,
- Un modulo FTDI USB (FTDI Basic Breakout - 5V Cod. Sparkfun DEV-09716) per la programmazione del modulo GPS con il PC,
- Modulo USB per la telemetria lato PC XBee Explorer USB cod. Sparkfun: WRL-08687)
- Modulo FTDI per la telemetria lato aeromodello (XtreamBee Board),
- Due moduli XBee PRO lato PC e lato modello per la trasmissione e ricezione della telemetria (XBee Pro - Wire Antenna cod. Sparkfun WRL-08742); sono possibili altre combinazioni di moduli XBee su frequenze diverse e con antenna cablata o esterna; unica accortezza i due moduli devono essere compatibili tra di loro,
- Un circuito stampato Power Distribution Board, per la distribuzione della tensione e per il cablaggio razionale dei cavi segnale dei regolatori dei motori.

Contrariamente a quanto avviene per gli aerei, dove le esigenze costruttive ed aerodinamiche im-

pongono di fatto la distribuzione dei componenti meccanici ed elettronici in spazi ed in posizioni obbligate e non sempre ottimali, un quadricottero, un pò come i veicoli spaziali, non ha particolari esigenze aerodinamiche da rispettare, se non alcuni aspetti dimensionali e di centraggio.

La struttura meccanica di un quadricottero è dettata unicamente da esigenze di robustezza e di supporto della meccanica ed elettronica e di conseguenza la costruzione avviene attorno ai componenti, un pò come una impalcatura.

Nel montaggio del nostro modello seguiremo lo stesso criterio, preparando prima tutti i componenti che verranno poi montati man mano unitamente alla costruzione del telaio.

MONTAGGIO DEL DISTRIBUTORE DI TENSIONE E SEGNALE PER I MOTORI

Questo circuito stampato è stato predisposto per razionalizzare i collegamenti per la alimentazione dei regolatori e per semplificare i collegamenti dei cavi segnale dei regolatori verso il modulo APM.

Come prima operazione è opportuno ispessire le piste che portano la tensione ai regolatori con l'aggiunta di stagno allo scopo di renderli

ideali a reggere il flusso di corrente richiesto dai motori in funzione.

Saldare i connettori a tre pin maschi per i cavi dei segnali negli alloggiamenti sui quattro lati della basetta, inserendoli tenendo la basetta rivolta verso l'alto.

Tenete presente che la basetta è orientata, guardandola dall'alto i lati Fronte, Retro, Destra e Sinistra DEVONO corrispondere alla posizione dei bracci e dei motori del modello, così come i relativi collegamenti ai motori, al modulo APM ed alla ricevente.

Saldare i cavi di collegamento della batteria facendo attenzione a rispettare le polarità. Saldare i cavi del connettore con il modulo APM rispettando il corretto orientamento dei bracci, il cavo bianco è il segnale per il regolatore destro, seguono nell'ordine i cavi arancione per i regolatori sinistro, fronte e dietro.

Convieni marcare i cavi con un segnacavo. Saldare i cavi di alimentazione di ciascun regolatore sulle piazzole presenti ai quattro lati della basetta facendo molta attenzione a rispettare le polarità.

Inserite le spinette dei cavi segnale dei regolatori sui rispettivi connettori rispettando le polarità; le scritte in serigrafia sulla basetta aiutano nell'operazione.

E' opportuno marcare la direzione

Front sulla basetta per evitare errori di montaggio nelle fasi successive.

Verificate con un multimetro commutato come ohmetro la correttezza dei collegamenti, con particolare attenzione alla polarità dei cavi di alimentazione.

Riverificate ancora.

PREPARAZIONE DEL MODULO APM E DEI SUOI ACCESSORI

Il processo di assemblaggio dei moduli APM, GPS e telemetria è stata descritta nel numero di Giugno 2011 di Elettronica In al quale si rimanda per le istruzioni da seguire. L'unica differenza è che il software ArduCopterMega, per la gestione del quadricottero richiede obbligatoriamente la installazione sul GPS del firmware in versione 1.6.

Le istruzioni per modificare il firmware del GPS le trovate nel numero 155 di Aprile 2011 di Elettronica In. Per aggiornare il firmware del GPS scaricate applicazione e firmware da

<http://code.google.com/p/ardupilot/downloads/list>
file:

MTK_DIYdrones.zip

Nel file scaricato si trova la versione 1.4 del firmware, la applicazione necessaria a caricarlo sul GPS attraverso la porta seriale e le istruzioni in inglese.



Fig. 5

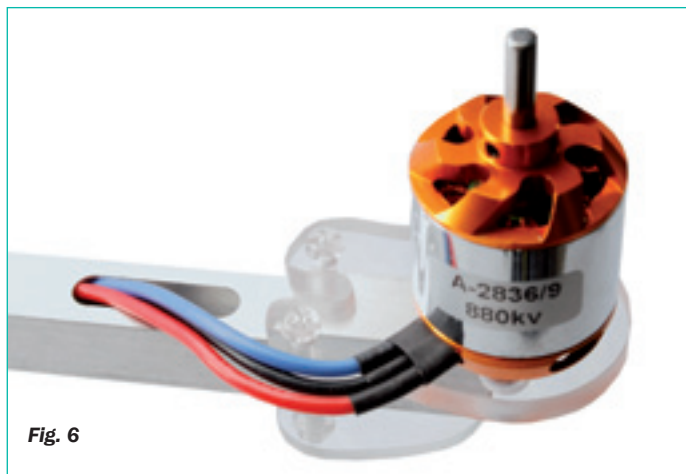


Fig. 6

Fig. 7

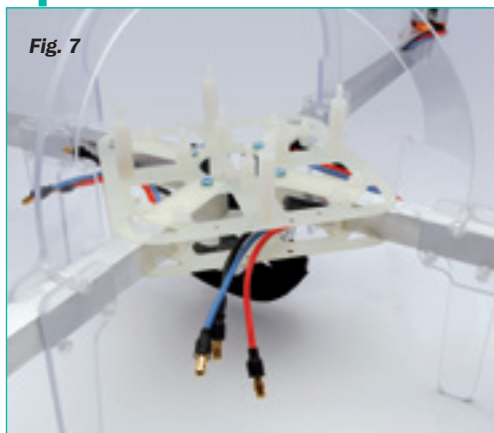


Fig. 8

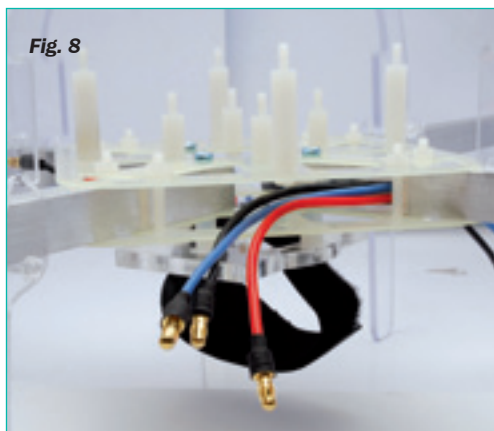


Fig. 9

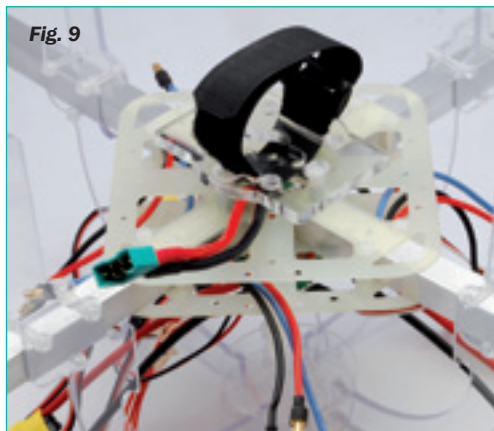
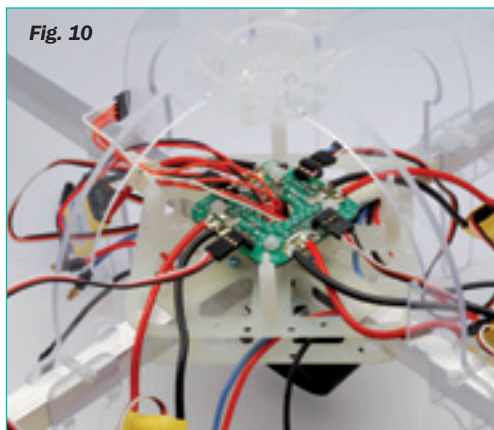


Fig. 10



La versione di firmware 1.6 è scaricabile da:

http://api.ning.com/files/taDZ5KJK**s3X*OdmnVy2mgVbaAZX0TpLaZYkQVvQi2GMhS-gID*N6S6rC5svYewSZfdABxz3GnRehZr4OQmA_/AXN1.30_2389_3329_384.1151100.1_v16.bin

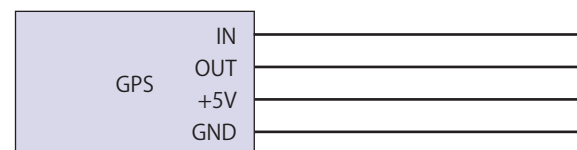
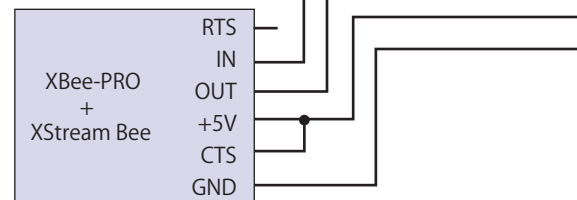
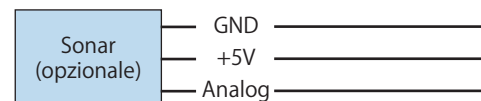
Quando arrivate al punto di eseguire il comando Download dal sottomenù Connection scegliete il file AXN1.30_2389_3329_384.1151100.1_v16.

MONTAGGIO DEL QUADRICOTTERO

La prima operazione consiste nell'inserire i cavi di alimentazione dei motori in ciascuno dei bracci a sezione quadrata del quadricottero. Da un lato i cavi sono collegati al motore mentre sulla estremità opposta sono montati tre connettori che si collegheranno ai regolatori di tensione. Inserite i connettori nella fessura ad asola del braccio dal lato dove la fessura ed il foro per la vite sono sullo stesso lato del braccio. Inserite i cavi fino a farli fuoriuscire dalla fessura ad asola sulla estremità opposta del braccio; da questa parte la fessura ed il foro per la vite sono su lati perpendicolari del braccio.

Successivamente montate i supporti dei motori sulla estremità di ciascun braccio. Inserite la rondella

Schema a blocchi dell'elettronica



in plastica su una vite lunga in metallo, inserite la vite nel foro centrale del contro supporto del motore, dal lato con la svasatura, fatela passare nei fori del braccio e poi nel foro centrale del supporto motore. Appoggiate il motore in modo che resti posizionato come nella foto e puntate la vite nel foro del motore. Fissate supporto e contro supporto al braccio con le due viti corte e relative rondelle; non stringete troppo per evitare di rompere le viti o di in-

Fig. 11

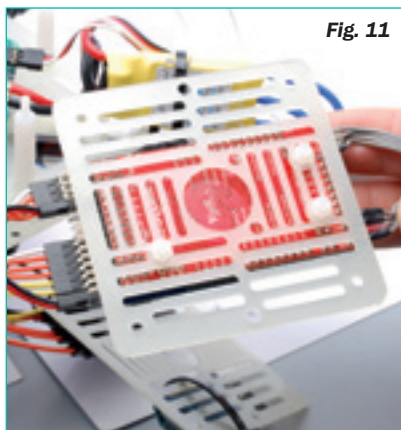
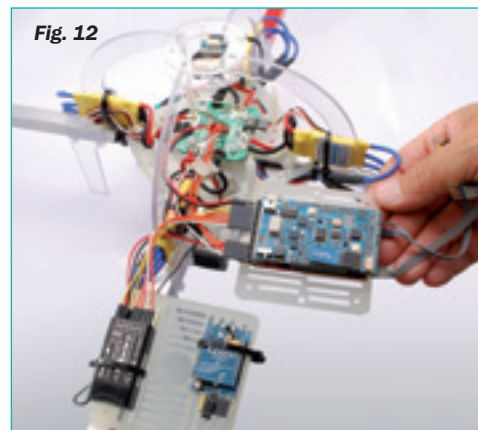
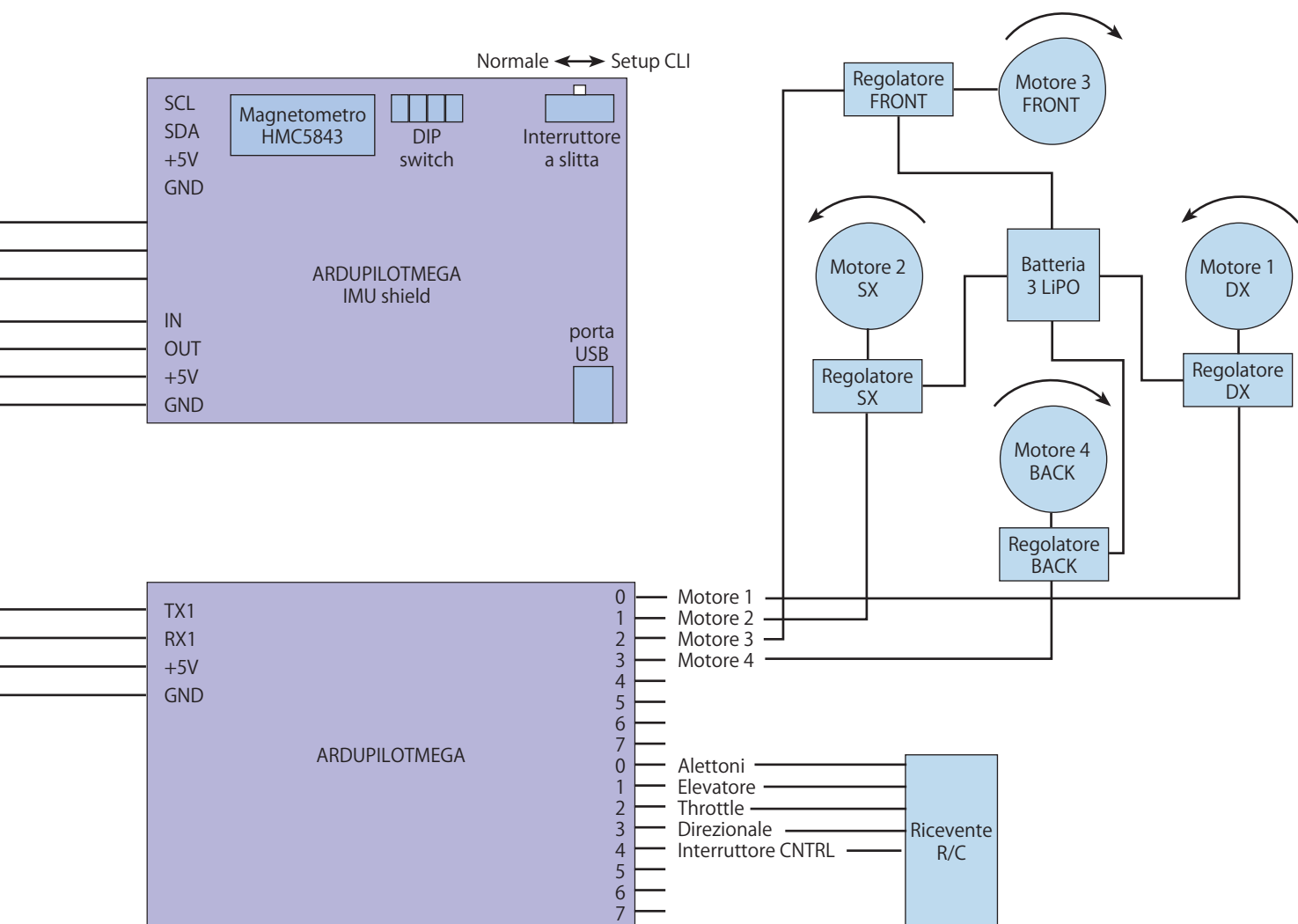


Fig. 12





crinare i supporti.

Posizionate e puntate le altre tre viti di fissaggio del motore e stringetele senza esagerare.

Ripetete le stesse operazioni per gli altri tre bracci.

Ora è il momento di assemblare i bracci dei motori al corpo centrale del modello. Il supporto centrale dei bracci è costituito dalle due placche quadrate più grandi, opportunamente sagomate.

Posizionate su una delle placche i distanziali per il circuito stampato dei regolari motori e per il supporto dell'APM.

Nella stessa placca inserite le quattro viti lunghe nei fori di testa dei

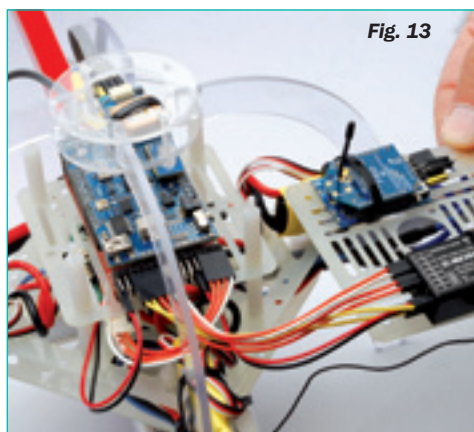


Fig. 13

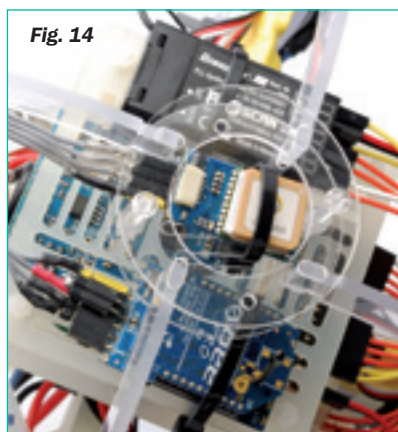


Fig. 14

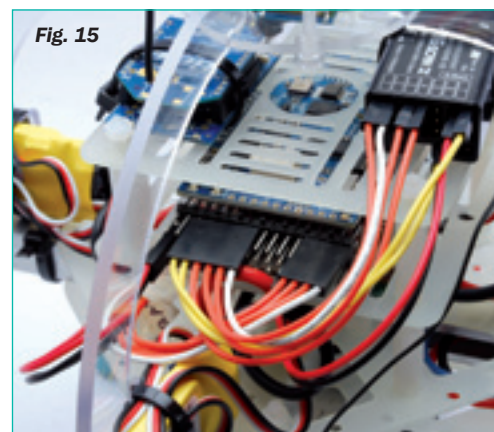


Fig. 15

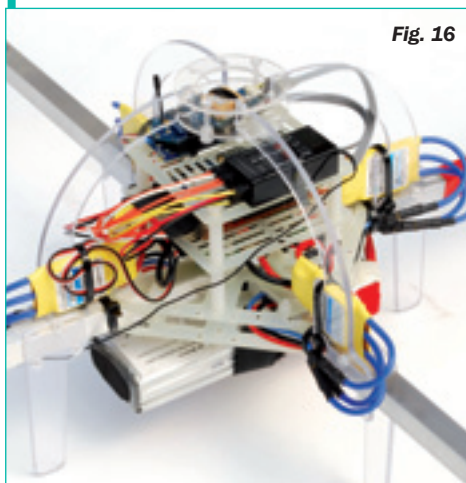


Fig. 16

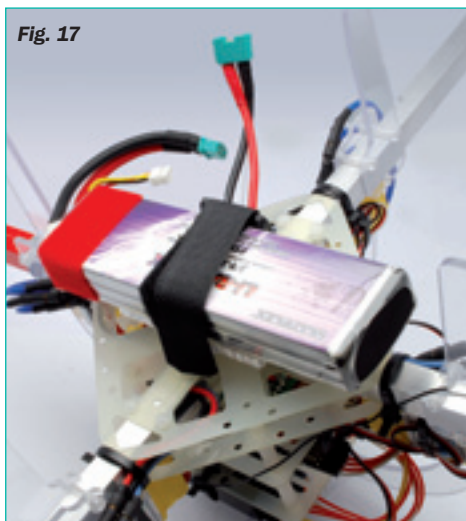


Fig. 17

supporti dei bracci; fate entrare le viti nei fori presenti sulle estremità libere dei bracci; la testa delle viti deve stare dalla parte dei motori. Appoggiate la seconda placca sopra le viti; appoggiate il supporto della batteria sulle stesse viti, posiziona-

te quattro dadi nelle sedi esagonali e stringete senza esagerare.

Completate il fissaggio dei bracci con le otto viti in plastica e relativi dadi, con l'accortezza di inserire le viti in modo che la testa si trovi dalla parte inferiore del telaio.

Ora scegliete uno dei bracci come braccio Front e marcatelo in modo riconoscibile, ad esempio con un giro di nastro colorato.

Posizionate la basetta del distributore di tensione sui suoi distanziali facendo coincidere il lato Front della basetta con il braccio Front del quadricottero.

Fate passare al disotto della struttura il cavo di alimentazione della batteria.

Completate il telaio con la costruzione della "cupola". Costruitela separatamente unendo le quattro nervature ad arco ai supporti centrali rotondi fissando il tutto con le viti in plastica.

Fissate la cupola ai bracci del telaio utilizzando le quattro gambe, che devono essere fissate con le viti agli elementi della cupola. Fate attenzione a tenere la parte scanalata delle gambe rivolta verso ciascun braccio. Non stringete troppo le viti.

Montate il modulo APM sulla piastra quadrata in plastica come illustrato nella foto. Posizionate la piastra sui distanziali del telaio

in modo che la parte anteriore di APM, quella senza i pin per i servo, sia rivolta verso il braccio Front.

Fissate la piastra al telaio con una nuova serie di distanziali e posizionate la seconda piastra. Questa è destinata ad accogliere la ricevente del radiocomando ed il modulo XBee della telemetria.

Posizionate il modulo GPS sul supporto circolare in cima alla cupola. Completate il cablaggio collegando al pettine Output di APM il cavo piatto proveniente dal distributore di tensione ed al pettine Input il cavo piatto per collegarsi alla ricevente RC.

Infine collegate i cavi di ciascun motore ai rispettivi regolatori.

Ricordiamo qui che i motori frontali e posteriori girano in senso orario e montano eliche destrorse mentre i motori destro e sinistro girano in senso antiorario e montano eliche sinistrorse.

Se durante le prove uno o più motori girano in direzione opposta al desiderato è sufficiente invertire i collegamenti di due dei cavi che collegano i motori ai regolatori per invertire il senso di rotazione.

Collegate i cavi alla ricevente rispettando lo schema elettrico.

Per verificare quali canali corrispondono a ciascun movimento degli stick potete aiutarvi con la funzione di "Verifica corsa servi", che dovrebbe essere disponibile nel menu di configurazione di ogni trasmettente RC.

Sul fascicolo di ottobre ci occuperemo della configurazione software del nostro quadricottero, della impostazione dei PID, della pianificazione missioni con l'APM Mission Planner e del funzionamento della telemetria. Descriveremo anche i primi test di volo e le possibili applicazioni pratiche del nostro quadricottero, specie per quanto riguarda l'utilizzo come velivolo autonomo. □



Fig. 18



Utilizziamo Arduino come sistema per programmare gli ATmega senza bootloader. Questa tecnica consente di utilizzare tutta la Flash per il codice e di realizzare schede usando degli ATmega vergini, più economici di quelli con caricato il bootloader. Prima puntata.

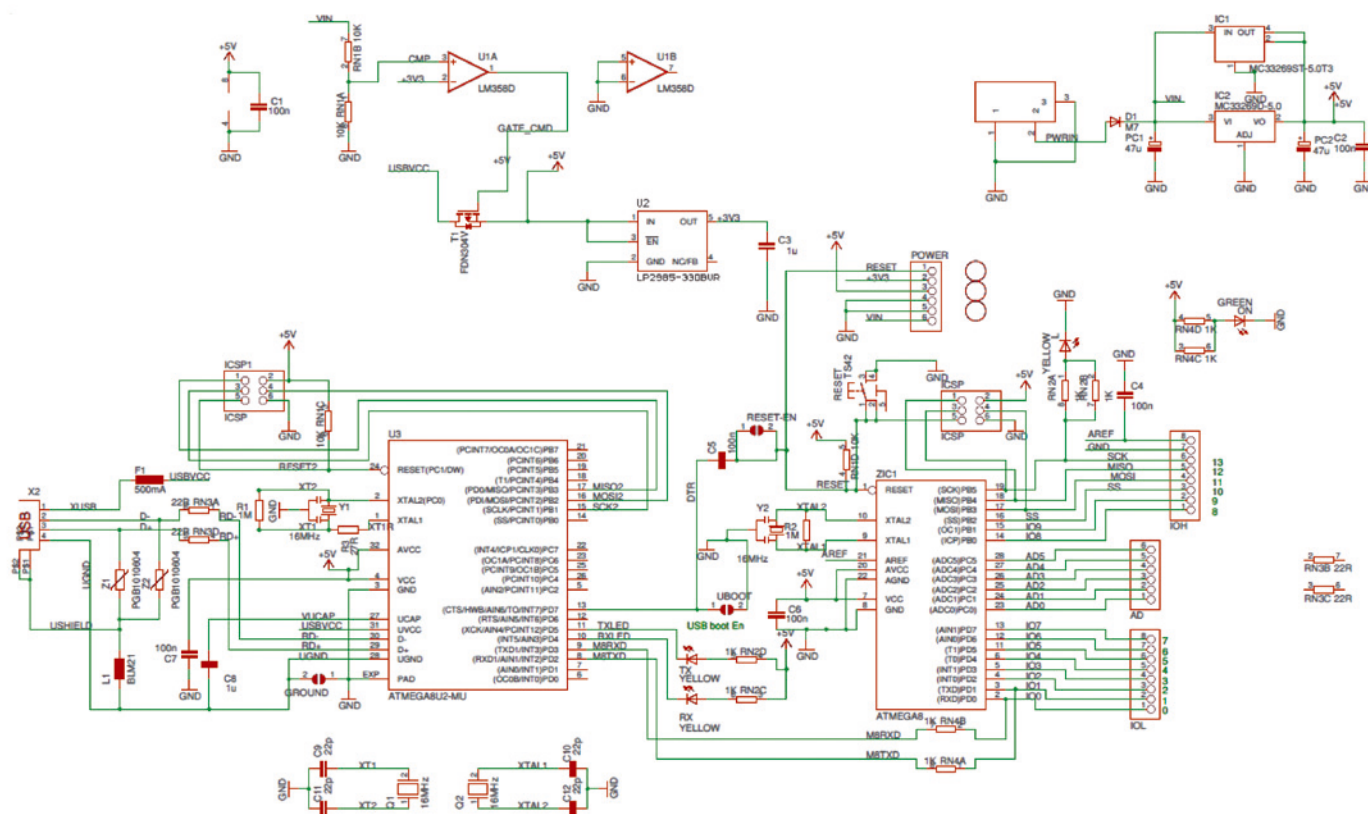
ARDUINO E CIRCUITI STAND-ALONE

..... del prof. MICHELE MENNITI

Le qualità che hanno decretato il successo di Arduino sono indubbiamente la progettazione open-source, un software IDE (giunto alla versione 0022), numerose librerie aperte a tutti, un hardware che mette a disposizione dell'utente tutti i pin utilizzabili del microprocessore, un Reference praticamente infinito che spiega ogni possibile risvolto dell'uso della piattaforma. Non ci dilungheremo oltre, perché sulle pagine di Elettronica In l'argomento è stato trattato in maniera esauriente molte volte; l'aspetto che vogliamo approfondire è quello

dell'uso di Arduino come piattaforma di sviluppo hardware. La scheda prevede quattro connettori, disposti in linea sui due lati lunghi, creati con l'intento di facilitare la realizzazione di schede aggiuntive, dette "shield", sovrapponibili ad Arduino, per aggiungere le funzionalità necessarie, di volta in volta, in base alle esigenze dell'utente. In tal modo Arduino resta il cuore di ogni progetto; è sufficiente installare lo shield e caricare nella memoria Flash del microprocessore il programma di gestione (sketch) e si ottiene un sistema funzionante.

Fig. 1 - Schema elettrico di Arduino Uno.



Questa configurazione va bene quando l'applicazione da sviluppare si basa su shield commerciali o se pensiamo di destinare una scheda Arduino a più applicazioni; ma se intendiamo destinare la piattaforma ad un impiego specifico, possiamo integrarla (lo schema elettrico è di pubblico dominio...) in un circuito ad hoc e programmare il micro in modo che esegua un solo firmware. Possiamo così rinunciare al bootloader e lasciare al firmware l'intera memoria di programma. Se diamo un'occhiata allo schema elettrico di Arduino UNO (Fig. 1) o Duemilanove, possiamo facilmente vedere come praticamente tutti i pin dei connettori siano collegati all'ATmega, eccezion fatta per la sezione di alimentazione (+5 V, +3,3 V, GND); questo ci fa intuire che il solo chip Atmega328,

programmato e tolto dall'Arduino e montato nella cosiddetta configurazione stand-alone (cioè con la componentistica minima necessaria al funzionamento) sia in grado di pilotare correttamente un circuito di qualsiasi tipo. La configurazione del micro ATmega328P in modalità stand-alone prevede semplicemente, oltre alle alimentazioni (+5 Vcc ai pin 7 e 20 del chip, GND ai pin 8 e 22), un quarzo da 16 MHz tra i pin 9 e 10 del chip, due condensatori ceramici da 22 pF tra questi due pin e GND, una resistenza da 10 kΩ tra il pin 1 e +5 Vcc per il pull-up della linea di reset (Fig. 2). Dobbiamo inoltre considerare che, laddove non fosse necessario far lavorare il microcontrollore a 16 MHz, potremmo far ricorso alla cosiddetta configurazione "minimal", ossia eliminare sia il quarzo che

i due condensatori, e perfino la resistenza di pull-up; quest'ultima, in realtà non c'entra con la configurazione minima, però, visto che i progettisti nel Reference l'hanno eliminata, non saremo certo noi ad insistere. In questo caso potremo far lavorare il microcontrollore con l'oscillatore interno, che può essere settato da 8 MHz (valore massimo) a 128 kHz (valore minimo), agendo sui cosiddetti "low, high ed extended_fuses", tre valori espressi in esadecimale che permettono, appunto, di configurare praticamente tutte le funzionalità interne dell'ATmega328, in base alla loro combinazione. La cosa ancora più interessante è che il microprocessore, una volta settato per lavorare con l'oscillatore interno, può essere alimentato a tensioni inferiori rispetto ai

canonici 5 Vcc, per arrivare a 3,3 V o addirittura ad 1,8 V; tutto ciò a vantaggio dei consumi, specialmente se l'applicazione che si va a realizzare dovrà essere alimentata a batterie.

L'altro aspetto che va chiarito, prima di entrare nel dettaglio della nostra sperimentazione, è quello della memoria Flash: l'ATmega328 dispone di 32 kbyte di Flash, che quando il chip è montato su Arduino non sono tutti disponibili, in quanto una porzione è riservata al bootloader, ossia il programma precaricato nel chip, scopo del quale è far dialogare Arduino con l'IDE per caricare i programmi (sketch) da eseguire. Lo stesso bootloader, ad ogni accensione o reset di Arduino, verifica la presenza nella memoria Flash di uno sketch e lo esegue. Il bootloader occupa uno spazio di 512 byte, nel caso di Arduino UNO, o di 2 kbyte (dovuti all'organizzazione della memoria, perché in realtà servirebbero solo 1,5 kbyte) nel caso di Arduino Duemilanove. Ebbene, nella versione stand-alone il bootloader non serve più; da ciò derivano due grandi vantaggi per il nostro lavoro, il primo dei quali è che possiamo utilizzare chip vergini, che costano di meno rispetto a quelli pre-programmati col bootloader. Il secondo vantaggio è che possiamo sfruttare, con opportune tecniche di programmazione del chip, anche lo spazio normalmente occupato dal bootloader, utilissimo nel caso di sketch molto lunghi.

IL NOSTRO PROGETTO

Per meglio comprendere la tecnica che andremo a studiare, sarà necessario effettuare delle prove "sul campo"; allo scopo, bisogna realizzare un progetto per un circuito in stand-alone, che pre-

veda appunto un core costituito dal chip ATmega328P. Il circuito è un lampeggiante, tipo quello montato sul tetto delle auto delle Forze di Polizia di tutto il mondo. Per la parte circuitale useremo un chip ATmega328P; la sigla finale P indica la versione PDIP, acronimo di Plastic Dual Inline Package, cioè un chip che si monta sul tipico zoccolo DIP, a due file di pin. Per completezza di informazione, diciamo che esiste anche l'ATmega328 (senza lettera finale), che invece identifica la versione QFN/MLF ("Quad Flat No-Lead/Micro Lead Frame Package"), progettata per i montaggi SMD; quindi fate molta attenzione, quando acquistate i chip, a controllarne la versione: quelli in SMD, oltre ad avere un costo maggiore, sono più difficili da saldare, quindi poco adatti ad impieghi hobbistici.

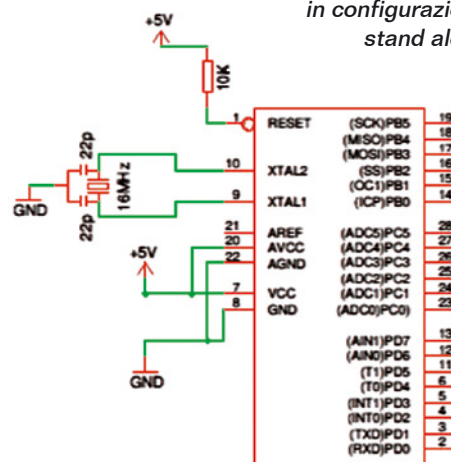
Il resto della componentistica è costituito da una piccola barra di LED e pochi passivi, che saranno montati, assieme al micro, su un circuito stampato di dimensioni minime.

Il progetto pratico sarà presentato nella prossima puntata; per il momento dedichiamoci alla programmazione e, a scopo di test, utilizzeremo il classico sketch "blink".

LA PROGRAMMAZIONE DI UN ATMEGA IN STAND-ALONE

Chiunque possieda una scheda Arduino sa bene che per farla funzionare è necessario caricarvi uno sketch, mediante il software chiamato IDE; l'operazione è piuttosto semplice: si collega Arduino alla porta USB del PC, nell'IDE si seleziona il modello di Arduino e la porta di connessione (COMx, nel caso dei sistemi Windows), si scrive o si richiama lo sketch e si preme il pulsante UPLOAD. A questo

Fig. 2 - ATmega328P in configurazione stand alone



punto i LED TX/RX di Arduino iniziano a lampeggiare allegramente, segno che la comunicazione col PC sta avvenendo regolarmente e dopo qualche istante il messaggio "done uploading" ci indicherà che tutto è andato a buon fine e che lo sketch è già stato avviato su Arduino (la prova di ciò arriva dal "Blink", che, una volta inviato, fa iniziare subito il lampeggio del LED collegato al pin 13 di Arduino). Nella Fig. 3 possiamo vedere un'immagine della schermata dell'IDE con lo sketch ArduinoISP caricato e, nella Fig. 4, Arduino UNO durante la fase di ricevimento dello sketch (notate i LED gialli accesi).

Chi ha già realizzato dei circuiti in stand-alone con ATmega328 sa anche che il chip, dopo aver ricevuto lo sketch, può essere tolto da Arduino e montato su un circuito (assieme ad un quarzo e due condensatori ceramici da 22 pF, oltre alla restante circuiteria prevista dal proprio progetto) dove svolgerà il proprio compito. Questa tipologia di programmazione però comporta ogni volta il dover togliere il chip dallo zoccolo di Arduino, per sostituirlo con un altro; inoltre molti, non riuscendo a far funzionare la tecnica seriale spiegata nel Reference, sono costretti a fare questa manovra ogni qualvolta devono

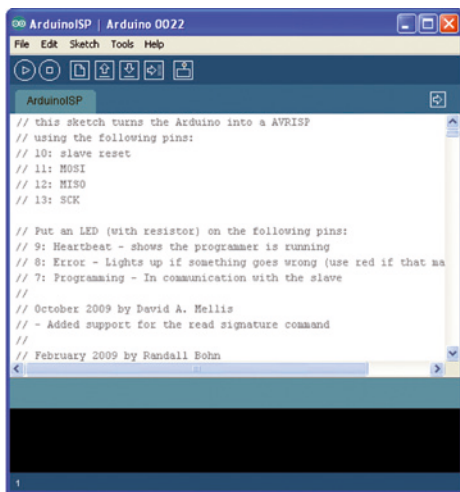


Fig. 3 - L'IDE con lo sketch ArduinoISP.

modificare lo sketch.

Tutto quanto appena descritto ha un grande svantaggio: oltre alla notevole perdita di tempo, c'è il problema dell'inevitabile danneggiamento dello zoccolo dell'ATmega, saldato sulla board Arduino, i cui contatti alla lunga si allentano.

Per non parlare, infine, di tutte le volte che un pin del chip si piega durante l'inserimento nello zoccolo, con la conseguenza che l'IDE inizierà a dare messaggi di errore. Inoltre non dimentichiamo che con questo metodo il

nostro chip avrà una parte della Flash inutilmente occupata dal bootloader.

Le problematiche appena descritte portano tutte alla stessa necessità: trovare una tecnica che permetta di salvaguardare Arduino ed il suo microprocessore riuscendo comunque a programmare il chip che dovrà essere montato in stand-alone.

La tecnica che esploreremo è una variante di quella già nota a molti, che consente l'uso di una board Arduino come Programmatore ISP.

Iniziamo con l'elenco del materiale occorrente (Fig. 5):

- Arduino UNO/Duemilanove;
- una Breadboard e relativi jumper di varia misura;
- un chip ATmega328P (vergine o già programmato);
- un quarzo da 16 MHz, due condensatori ceramici da 22 pF, una resistenza da 10 K Ω 1/4 W, una resistenza da 560 Ω 1/4 W, un LED da 3 o 5 mm;
- sette cavetti jumper maschio-maschio da 10÷15 cm.

Inoltre, vi consigliamo di tenere a portata di mano una resistenza da 120 Ω 1/4 di watt e un condensatore elettrolitico o al tantalio da 10 μ F 10÷16 volt.

Ora prepariamo il nostro circuito target su Breadboard e prima di tutto inseriamo il chip in un punto qualsiasi della Breadboard; questi sono i collegamenti da fare (Fig. 6):

- tramite i jumper da Breadboard colleghiamo i pin 7 e 20 del chip alla linea di alimentazione positiva (+5 volt);
- allo stesso modo colleghiamo i pin 8 e 22 del chip alla linea di massa dell'alimentazione (GND);
- colleghiamo il pin 1 del chip alla linea +5 V mediante la resistenza da 10 k Ω ;
- inseriamo il quarzo in modo che i suoi piedini siano collegati ai pin 9 e 10 del chip;
- inseriamo i due condensatori ceramici da 22 pF; entrambi dovranno avere un piedino collegato a GND, mentre l'altro servirà per collegare un condensatore al pin 9 e l'altro ancora al pin 10 del chip;
- inseriamo un capo della resistenza da 560 Ω in corrispondenza del pin 19 e l'altro capo in un punto vuoto della breadboard; a questo capo colleghiamo l'anodo (il pin più lungo) del LED, il cui altro capo (catodo) va a GND;

La resistenza da 120 Ω e il condensatore da 10 μ F per ora non servono.

A questo punto possiamo collegare Arduino alla Breadboard usando i cavetti jumper da 15 cm (vedere la Fig. 7) rispettando le seguenti corrispondenze:

- il pin 10 di Arduino va al pin 1 del chip;
- il pin 11 di Arduino va al pin 17 del chip;

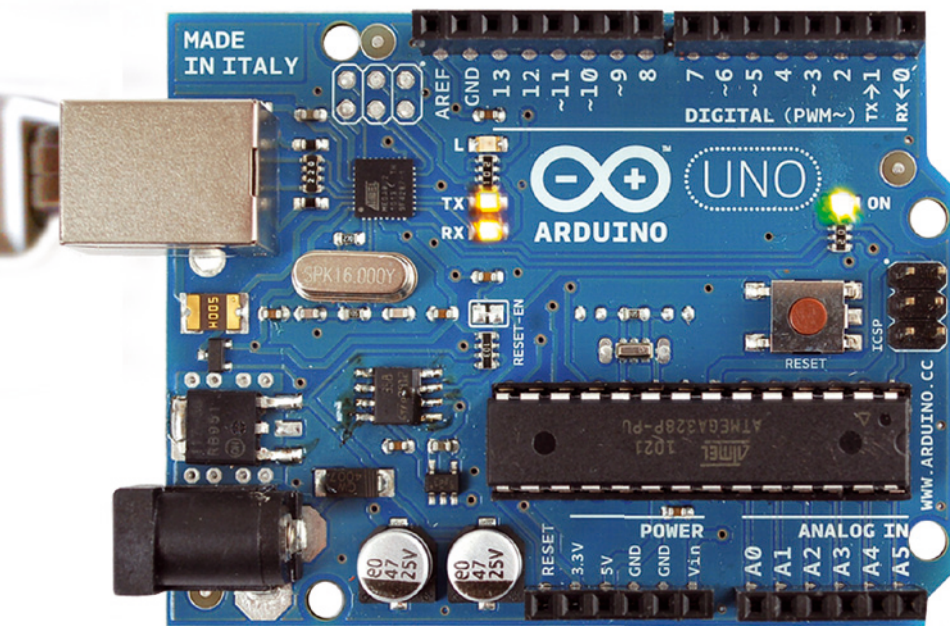


Fig. 4 - Arduino UNO durante l'Uploading dello sketch.

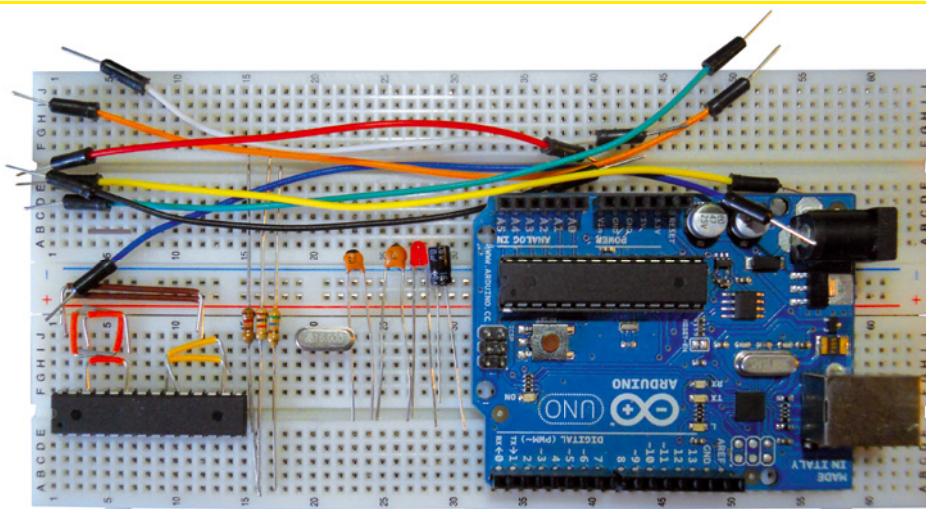


Fig. 5 - I componenti necessari per la programmazione dell'ATmega.

- il pin 12 di Arduino va collegato al pin 18 del chip;
- il pin 13 di Arduino va sul pin 19 del chip;
- il pin +5V di Arduino va alla linea di alimentazione positiva della breadboard;
- uno qualsiasi dei tre pin GND di Arduino va alla linea di massa della breadboard.

L'ultimo dei sette cavetti jumper, al momento non serve.

Adesso passiamo al software, iniziando dallo svelarvi il "trucco" che ci permetterà di inviare uno sketch, tramite l'IDE, al chip della Breadboard, bypassando Arduino, che pur sarà collegato al PC e che svolgerà appunto il ruolo di Programmatore ISP. Quello che dobbiamo fare è creare una board virtuale, partendo

da quella originale (corrispondente al modello di Arduino che stiamo usando) ed apportando alcune semplici ma essenziali modifiche. Dobbiamo innanzitutto localizzare il file *boards.txt* che è quello contenente tutte le informazioni relative alle varie board che l'IDE ci mostra quando eseguiamo il comando *Tools→Board*. Tipicamente questo file si trova, all'interno della cartella del software IDE, nel percorso *X:\MyPath\arduino-0022\hardware\arduino*, in cui X è la lettera che indica l'unità logica e MyPath la cartella o il percorso contenente il programma (0022 indica la versione del programma).

Nella **Fig. 8** possiamo osservare la cartella contenente il file *boards.txt* (ed il relativo percorso) e l'elenco delle board mostrate dall'IDE versione 0022.

Ora apriamo il file, con Blocco Note di Windows (o un programma equivalente se usiamo altri sistemi operativi) e vedremo una lunghissima serie di righe organizzate in gruppi separati da una linea costituita da una ripetizione del simbolo "#"; ogni gruppo rappresenta una board diversa. Le righe sono identificate dal codice iniziale, uguale per tutte, ma diverso da una board all'altra; il nome che apparirà nel sottomenu *Tools→Board* è inserito nella prima riga del gruppo. Per meglio comprendere l'organizzazione di queste informazioni, osserviamo attentamente la **Fig. 9**, che mostra le righe di configurazione della board Arduino UNO.

Il codice è rappresentato dalla parola "uno" che, come vedete, dà inizio ad ogni riga ed è da essa separata tramite un ".".

La riga contenente la parola "name" (in genere è la prima) è seguita da "=" e quindi dal nome che la board avrà nell'IDE.

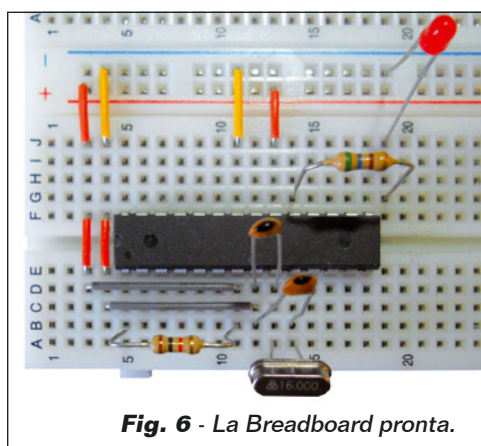


Fig. 6 - La Breadboard pronta.

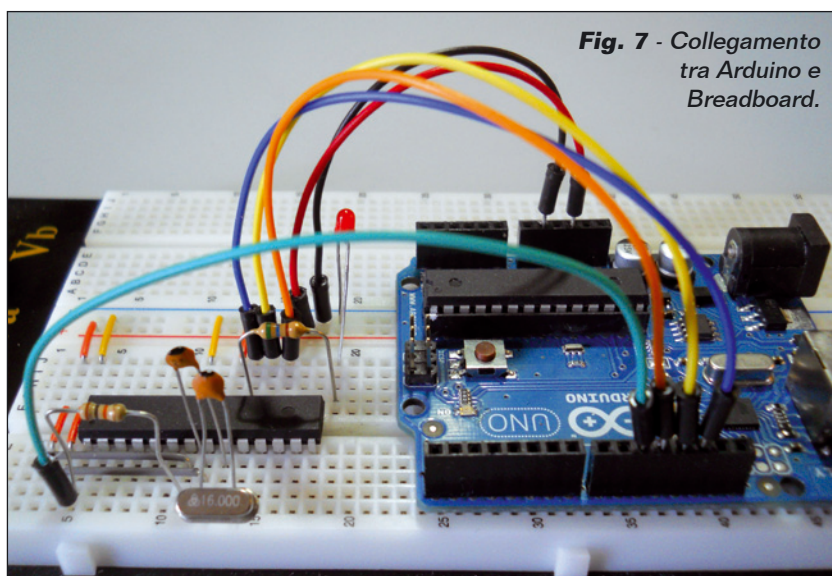


Fig. 7 - Collegamento tra Arduino e Breadboard.

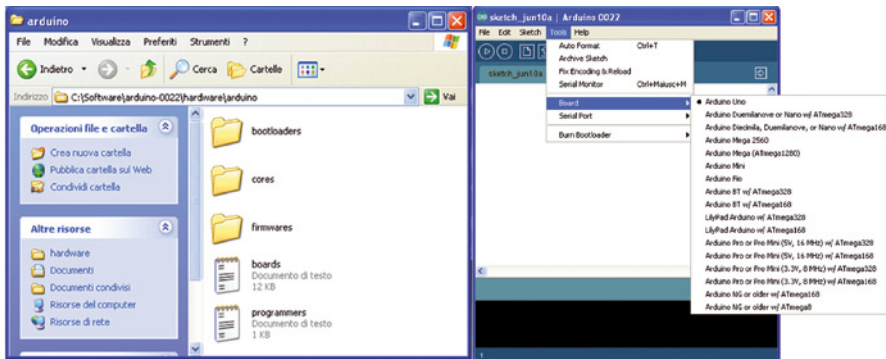


Fig. 8 - Il file boards.txt e il sottomenu Tools→Board.

Le altre informazioni che ci interessano sono:

- `uno.upload.maximum_size=32256`: imposta la capacità massima di memoria Flash che possiamo usare; in pratica dai 32 kbyte di Flash totali di cui è dotato l'ATmega328P dobbiamo sottrarre lo spazio occupato dal bootloader; nel caso di Arduino UNO si tratta di 512 byte, infatti, trasformando 32 kB byte in bit otteniamo 32.768, ai quali sottraendo 512 otteniamo 32.256;
- `uno.bootloader.low_fuses=0xff`; `uno.bootloader.high_fuses=0xde`; `uno.bootloader.extended_fuses=0x05`; queste tre righe sono i "fuse" (fusibili), servono per settare il comportamento del chip e si esprimono con valori esadecimali;
- `uno.build.f_cpu=16000000L`: questa riga deve corrispondere alla frequenza di clock per la quale il chip è stato impostato, mediante i fuse,

espressa in Hz; 16.000.000 di hertz corrispondono a 16 MHz, appunto la frequenza del quarzo o, più precisamente, dell'oscillatore esterno presente su Arduino UNO; tale valore serve come riferimento per i comandi di temporizzazione del software, come `delay()` e `millis()`.

E ora creiamo la nostra board virtuale, scrivendo, in un punto qualsiasi del file; nell'esempio abbiamo scritto subito sotto quelle della UNO, le righe di codice contenute nel **Listato 1**. Seguendo l'impostazione del file separeremo questo gruppo di righe da quelli delle altre board, inserendo una riga di "#". Il risultato finale dovrebbe essere quello della **Fig. 10**. Possiamo notare che, rispetto all'originale board di Arduino UNO, sono variati: il codice (`atmsa16` invece di `uno`), il `maximum_size` (portato alla massima capacità della Flash, visto che non dobbiamo riservare spazio per il bootloader); c'è poi una nuova riga (`atmsa16.upload.using=arduino:arduinoisp`) che ci permette di far capire all'IDE che dovrà programmare il chip in stand-alone e non quello su Arduino. Un'altra nuova riga è preceduta da alcuni "####" che la disattivano; il motivo è presto spiegato: l'`extended_fuses` è impostato a 0x05 e in alcuni particolari casi, nella fase di invio dello

sketch potrebbe essere segnalato un errore legato all'impostazione di tale valore. Come vedremo più avanti, sarà sufficiente modificare le seguenti due righe di codice:

```
#### atmsa16.bootloader.extended_fuses=0x05
atmsa16.bootloader.extended_fuses=0x07
```

attivando così il valore 0x07 invece di 0x05; tutto si risolverà. Naturalmente questa variazione non va effettuata a priori, ma solo se si dovesse verificare l'errore. È indispensabile un altro chiarimento: avrete notato che per creare la nostra board virtuale abbiamo usato come base di partenza le righe riferite ad Arduino UNO, quindi qualcuno ora potrebbe pensare che, disponendo della Duemilanove, non potrebbe usarle. In realtà non è così, perché con questa tecnica di programmazione le righe relative allo speed (la seriale della Duemilanove lavora a 57.600 bps e non a 115.200 come quella della UNO) ed al tipo di bootloader saranno ignorate dall'IDE, il che rende perfettamente intercambiabili le due board,



Fig. 9 - La configurazione della board Arduino UNO.

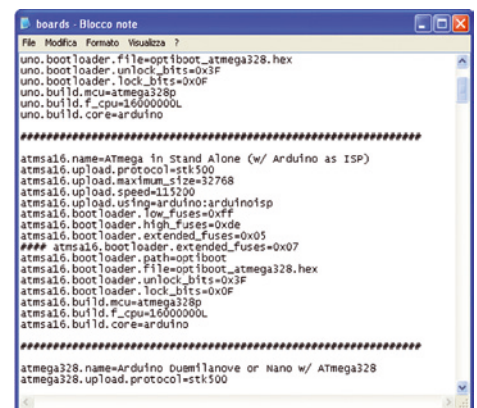


Fig. 10 - La configurazione della board virtuale.

Listato 1

```
atmsal6.name=ATmega in Stand Alone (w/ Arduino as ISP)
atmsal6.upload.protocol=stk500
atmsal6.upload.maximum_size=32768
atmsal6.upload.speed=115200
atmsal6.upload.using=arduino:arduinoisp
atmsal6.bootloader.low_fuses=0xff
atmsal6.bootloader.high_fuses=0xdf
atmsal6.bootloader.extended_fuses=0x05
##### atmsal6.bootloader.extended_fuses=0x07
atmsal6.bootloader.path=optiboot
atmsal6.bootloader.file=optiboot_atmega328.hex
atmsal6.bootloader.unlock_bits=0x3F
atmsal6.bootloader.lock_bits=0x0F
atmsal6.build.mcu=atmega328p
atmsal6.build.f_cpu=16000000L
atmsal6.build.core=arduino
```

che avranno la sola funzione di Programmatore ISP. Le abbiamo lasciate, diciamo così, per usi futuri; laddove dovessero servire ci preoccuperemo di realizzare una nuova board specifica per la compatibilità con la Duemilanove.

PROGRAMMARE IL MICRO

Giunti a questo punto siamo pronti per il passaggio finale: invieremo il nostro sketch al chip montato in stand-alone e poi testeremo il funzionamento separandolo da Arduino.

Per leggere la nuova board presente nel file, l'IDE deve essere riavviato, quindi se tale programma era aperto, durante la modifica del file *boards.txt* bisogna chiuderlo e farlo ripartire. Per verificare che la nostra modifica sia andata a buon fine, è sufficiente ora eseguire il comando Tools→Board e controllare se nell'elenco ora è presente la nostra "stand-alone" (v. Fig. 11); in caso contrario dobbiamo richiudere l'IDE e ricontrollare il file *boards.txt*, perché certamente abbiamo commesso un errore. La tecnica di invio dello sketch al chip in stand-alone è davvero semplice; per descriverla ci rifaremo a quella che si usa per caricare il bootloader: per prima cosa selezioniamo la board Arduino che stiamo usando

(per esempio Arduino UNO o Duemilanove), esattamente come facciamo per il normale uso di Arduino. Poi selezioniamo la porta seriale cui l'Arduino è collegata (la COM, per chi usa Windows) e richiamiamo dall'IDE lo sketch *ArduinoISP*; eseguiamo il comando *Upload* cliccando sul relativo pulsante dell'IDE. Dopo diversi secondi di lampeggio dei tre LED di Arduino e naturalmente di quello della breadboard (in questo momento è fisicamente collegato al pin 13 di Arduino che, come sappiamo, controlla uno dei tre LED della board) sull'IDE uscirà il messaggio "done uploading". Arduino è pronto per svolgere il ruolo di Programmatore ISP; selezioniamo, ora, nell'IDE la nostra board "ATmega in Stand Alone (w/ Arduino as ISP)", senza modificare la COM.

Richiamiamo dall'IDE lo sketch "blink" ed eseguiamo nuovamente Upload cliccando sul relativo pulsante dell'IDE: i LED di Arduino e breadboard lampeggeranno nuovamente, questa volta per un periodo decisamente più breve, alla fine del quale nuovamente l'IDE mostrerà il messaggio "done uploading". In alcuni casi l'intervento dell'autoreset di Arduino non permette il buon fine di questa operazione; vedremo dopo come risolvere

tale evenienza. A questo punto vedremo lampeggiare il LED della breadboard e quello del pin 13 di Arduino. Ma quale dei due chip sta inviando il segnale? Per scoprirlo e capire quindi se siamo riusciti nel nostro intento, c'è da fare un ultimo passaggio: stacciamo tutti i cavetti jumper che collegano Arduino alla breadboard ad eccezione di quelli dell'alimentazione +5 V e GND. Con questa semplice operazione separiamo il chip stand-alone da Arduino (che ora stiamo usando solo come alimentatore) e vedremo che il LED di Arduino si spegnerà mentre quello della breadboard continuerà tranquillamente a lampeggiare (vedere Fig. 12).

Quindi il nostro chip ATmega328P è stato programmato senza doverlo montare fisicamente su Arduino ed ora vive di vita propria. È quindi pronto per essere montato nel circuito cui è destinato.

Naturalmente il chip può essere riprogrammato a piacimento con qualsiasi tipo di sketch ed è ciò che faremo quando presenteremo il nostro lampeggiante.

RISOLUZIONE DEI PROBLEMI

Giunti a questo punto dobbiamo

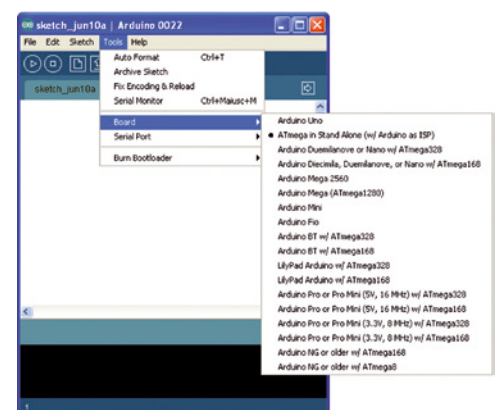


Fig. 11 - La board virtuale "stand-alone".

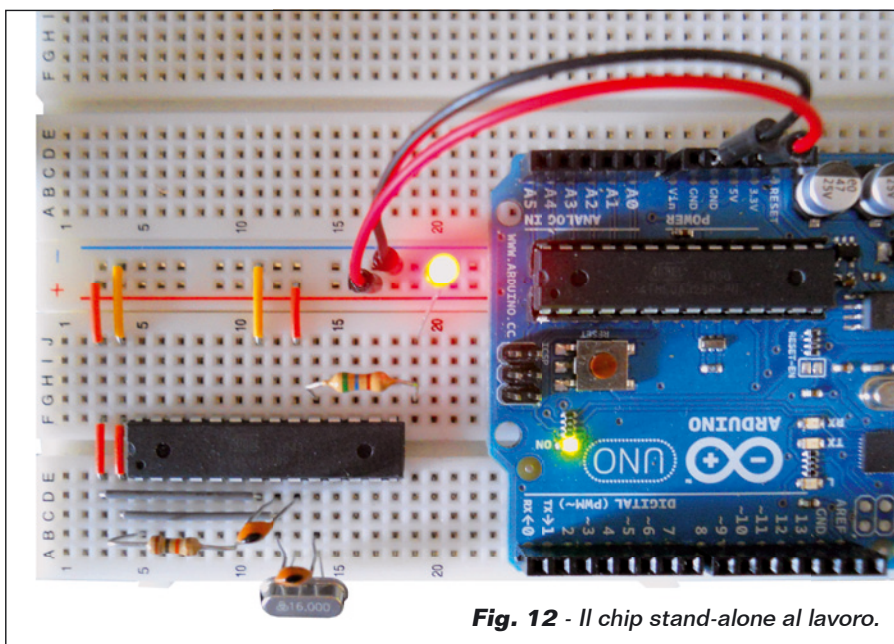


Fig. 12 - Il chip stand-alone al lavoro.

trattare e risolvere tre tipi di problema che potrebbero verificarsi quando inviamo lo sketch al chip stand-alone; chiariamo sin da subito che le problematiche dell'*extended_fuses* e dell'autore-set si possono verificare indifferentemente su Arduino UNO o Duemilanove, senza che sia possibile stabilire una regola certa, come dimostrano le innumerevoli prove fatte in laboratorio con alcune board dell'uno e dell'altro tipo. Dobbiamo inoltre sottolineare che i rimedi che illustreremo risolvono al 100 % i problemi, ma vanno applicati solo se il problema dovesse verificarsi. Naturalmente alla base di ogni troubleshooting che si rispetti deve esserci una certezza: i collegamenti devono essere perfetti, quindi ricordate che anche un solo ponticello o cavetto o componente mancante causerà il sicuro insuccesso della tecnica. Perciò se qualcosa non dovesse funzionare come indicato, prima di tutto verifichiamo attentamente, senza dare nulla per scontato, di aver collegato tutto come spiegato. Iniziamo dalla situazione che

può verificarsi se programmiamo un chip vergine; la Atmel imposta in fabbrica i fuse per far lavorare il chip ad 1 MHz con l'oscillatore interno. Se inviamo direttamente uno sketch succede che il chip in stand-alone ignorerà il quarzo esterno e i tempi risulteranno sfasati: ad esempio il lampeggio del LED con lo sketch blink durerà circa 16 secondi invece di 1 secondo. Ovviamente ciò rappresenta per noi un problema se il chip deve lavorare col quarzo a 16 MHz, mentre se invece optiamo per la configurazione "minimal" descritta all'inizio dell'articolo, dobbiamo semplicemente adeguare i valori di comandi come *delay()* e *millis()*. Ma a noi ora interessa lavorare a 16 MHz quindi, semplicemente, dobbiamo settare i fuse; l'operazione si effettua facilmente caricando sul chip il bootloader prima di inviare lo sketch. Prima di spiegare questa semplice manovra chiariamo subito due punti: il bootloader va inviato una sola volta al chip vergine e servirà solo per settare i fuse, poi diverrà inutile e lo sketch lo sovrascriverà; se dovessimo pri-

ma caricare lo sketch, accorgerci dell'errore, e poi caricare il bootloader, nessun problema: il chip viene settato e dobbiamo semplicemente inviare nuovamente lo sketch. I semplici passaggi che andiamo a descrivere torneranno utilissimi anche per i casi in cui volessimo preparare un chip vergine per lavorare direttamente su Arduino; ad esempio è sempre buona norma avere in casa un chip di riserva col bootloader della nostra board; se malauguratamente si dovesse danneggiare quello originale, una semplice sostituzione ci risolverebbe subito il problema.

Ecco i passaggi da seguire:

- prepariamo la Breadboard e colleghiamo Arduino come già visto in precedenza;
- Apriamo l'IDE e selezioniamo il modello di Arduino che stiamo usando e la porta cui esso è collegato;
- carichiamo lo sketch ArduinoISP su Arduino;
- eseguiamo ora il comando Tools→Burn Bootloader→w/Arduino as ISP;
- dopo circa un minuto avremo caricato nel chip stand-alone il bootloader della board Arduino che stiamo usando come Programmatore ISP (avrete certamente notato che questa volta nell'IDE abbiamo lasciato sempre settata la nostra board Arduino).

Come detto, il chip può essere subito montato su Arduino per ricevere gli sketch con la normale operazione di Upload dell'IDE, oppure lo lasciamo dov'è e ripetiamo/eseguiamo l'operazione di invio dello sketch in stand-alone; questa volta il blink funzionerà alla perfezione. Naturalmente potrebbero verificarsi altri errori; niente paura, continuiamo a leggere questa

sezione e certamente tutto si risolverà.

Passiamo quindi al problema dell'*extended_fuses*: come già accennato in precedenza, nelle righe della nostra board virtuale abbiamo previsto un doppio valore per questo fuse; infatti può capitare (anche se è piuttosto raro) che qualche board non riesca a programmare tale fuse col valore 0x05. Ne deriva che, al momento dell'Upload dello sketch sul chip stand-alone, l'IDE mostrerà un messaggio d'errore (scritto in colore rosso su sfondo nero) che indicherà la necessità di usare il valore 0x07; in pratica se appare tale avviso significa che dobbiamo chiudere l'IDE, aprire nuovamente il file *boards.txt* e attivare la relativa riga, disattivando contestualmente l'altra (quella con 0x05), come spiegato in precedenza. A questo punto possiamo rifare la prova e tutto andrà a buon fine. Chiariamo che se l'errore si verifica su una certa board si verificherà sempre, quindi la variazione del file dovrà essere fatta una sola volta e definitivamente. Ed ora vediamo il problema, un po' più frequente, dell'autoreset; allo scopo conviene che spieghiamo il funzionamento di Arduino in questa circostanza. Quando il chip seriale (FT232RL su Duemilanove o ATmega8U2 su UNO) riceve un segnale dalla porta USB, cioè l'IDE comunica che sta per inviare dei dati, immediatamente invia l'impulso di reset all'ATmega328, che quindi si predispone a ricevere tali dati. Questa operazione corrisponde a quella che effettuiamo ogni volta che premiamo il tastino "RESET" presente su Arduino.

Se i dati poi non arrivano o se il reset è stato dato manualmente, subito dopo viene eseguito l'eventuale sketch presente nella

Non tutti sanno che Arduino può svolgere la funzione di un Programmatore ISP ("In-System Programmer"), né più né meno dei tanti programmatori che si trovano in commercio; questa potenzialità ci dà l'indubbio vantaggio di non dover acquistare un ulteriore apparecchio per svolgere le operazioni di cui andremo a parlare.

Nello specifico la funzionalità di Programmatore ISP permette all'utente di caricare il bootloader su un chip vergine, montato su un circuito target; questo può essere costituito indifferente da un altro Arduino o da una comune Breadboard in configurazione stand-alone (come già spiegato, servono solo quattro componenti aggiuntivi: un quarzo da 16 MHz, due condensatori ceramici da 22 pF ed una resistenza da 10 kΩ).

Tutto ciò che serve è uno sketch (ArduinoISP), già fornito di serie con il software IDE e un cavetto a 6 poli, che può essere facilmente sostituito da 6 cavetti jumper maschio-maschio.

I 6 fili servono per collegare Arduino al circuito target, lo sketch va caricato sull'Arduino collegato fisicamente al PC, con un normale Upload, quindi si esegue il comando Tools→Burn Bootloader→w/Arduino as ISP e, nel tempo massimo di un minuto (dipendente dalla versione di bootloader che si carica), il chip sarà pronto per essere installato su Arduino e ricevere il suo sketch.

I collegamenti ed altri approfondimenti su questa tecnica sono descritti nell'articolo.

memoria flash del chip ATmega328. Quando Arduino viene usato come Programmatore ISP può succedere che, se l'autoreset viene inviato con troppo anticipo (ma si parla di millisecondi), l'operazione di Upload dello sketch sul chip stand-alone non vada a buon fine. In questo caso l'IDE restituisce il seguente errore: **"avrdude: stk500_getsync(): not in sync: resp=0x15"**.

Il problema si risolve bloccando l'autoreset; e qui entrano in gioco i due componenti che finora erano rimasti inutilizzati: la resistenza da 120 Ω ed il condensatore da 10 μF. La resistenza si deve collegare tra il pin RESET di Arduino e +5 V, mentre il condensatore va collegato col polo positivo al RESET di Arduino e quello negativo a GND; con tale configurazione, lo scopo della resistenza sarà far caricare rapidamente il condensatore mentre quest'ultimo, una volta carico, riuscirà a fissare il pin RESET ad un livello logico alto, bloccandolo di fatto, visto che, affinché il chip ATmega328 venga resettato, esso deve poter ricevere un impulso a zero logico.

Nemmeno in questo caso, dalle varie prove effettuate con diverse board, è emersa una regola, quindi anche questo problema, se si verifica, dipende dalla singola board. Inoltre abbiamo osservato che spesso è sufficiente il solo condensatore; in casi sporadici (soprattutto su Arduino Duemilanove) è bastata la sola resistenza, mentre in diversi altri casi sono stati necessari entrambi i componenti. Infine, da nuove prove effettuate è emerso che è possibile risolvere il problema anche mettendo il condensatore col polo positivo a +5 volt ed il polo negativo sul pin RESET; in questo caso il meccanismo d'azione è diverso, in quanto esso diminuisce drasticamente la durata dell'impulso di RESET, vanificandolo e quindi ottenendo lo stesso risultato.

Nella **Fig. 13** vediamo il collegamento globale con l'inserimento di resistenza e condensatore; ora comprendiamo l'utilità del settimo cavetto jumper, restato fino a questo momento inutilizzato: esso porta sulla Breadboard il segnale di RESET di Arduino, per facilitare l'aggiunta

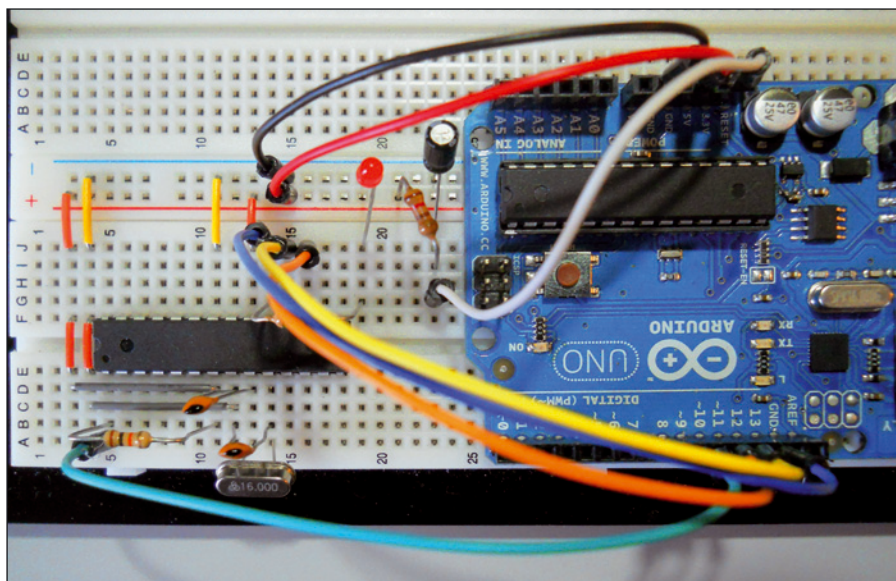


Fig. 13 - Autoreset: resistenza, condensatore e cavetto jumper.

dei due componenti. Com'è facile intuire, quando basta il solo condensatore si potrebbe inserirlo direttamente tra i pin "RESET" e "GND" (o, come detto, tra i pin "+5 V" e "RESET") di Arduino, ma noi lo sconsigliamo: in genere i piedini di un condensatore sono più grossi di quelli di un cavetto jumper e alla lunga tenderebbero ad allargare le molle del connettore di Arduino, danneggiandolo.

Gli accorgimenti descritti vanno usati solo se strettamente necessario; quale adottare dipenderà dalle prove che farete nel caso v'imbattiate nell'errore dell'autoreset. Anche in questo caso, se il vostro Arduino presenta il problema, esso sarà stabile, per cui i componenti necessari per risolvere il problema dell'autoreset andranno utilizzati sempre per questa operazione; essi dovranno essere collegati, anche senza togliere l'alimentazione, dopo aver caricato su Arduino lo sketch ArduinoISP.

Nota importante: la necessità di collegare tali componenti solo nel momento in cui servono, è dettata dal fatto che per caricare uno sketch su Arduino occorre l'autoreset, altrimenti il caricamento non riuscirà ed otterre-

mo l'errore **avrdude: stk500_getsync(): not in sync: resp=0x00 - avrdude: stk500_disable(): protocol error, expect=0x14, resp=0x51**; quindi se vedete questo errore, sappiate che dovete semplicemente "liberare" il pin "RESET" di Arduino dal collegamento con i componenti anti-autoreset.

Chiudiamo con una considerazione importante: incontrando una o anche entrambe queste problematiche, potreste essere erroneamente indotti a credere che il vostro Arduino sia difettoso. Niente di più errato! Si tratta di un più semplice "effetto temporale" (ricordiamo che parliamo solo di alcuni millisecondi), dovuto alle tolleranze dei componenti interni di Arduino che, comunque, influenzano solo questo particolare tipo di operazione. D'altra parte, nei tutorial ufficiali del Team Arduino la problematica dell'autoreset è stata affrontata ed in alcuni casi risolta con interventi drastici, come ad esempio il taglio del ponticello RESET-EN presente sulla board Arduino; ma ciò comporta la disabilitazione definitiva dell'autoreset e l'obbligo di ricorrere all'uso del tastino presente sempre sulla board, per

dare il reset manualmente ad ogni sketch da caricare. A titolo di curiosità, ma è una tecnica che riteniamo decisamente scomoda, il problema dell'autoreset si potrebbe anche risolvere usando il solo tastino di "RESET"; i passaggi da fare sono questi:

- quando siamo pronti per inviare lo sketch al chip in stand-alone premiamo e manteniamo premuto il tastino "RESET" di Arduino;
- eseguiamo il comando *Upload* e l'IDE inizia la compilazione dello sketch;
- appena terminata la compilazione e prima di inviare lo sketch, l'IDE mostra nella sua finestra inferiore la dimensione dello stesso sketch espressa in byte;
- in quel preciso istante bisogna rilasciare il tasto "RESET".

Purtroppo l'operazione non è semplice come sembra, perché quando parliamo di "preciso istante" intendiamo dire che una minima differenza di tempo non fa andare a buon fine l'operazione; dobbiamo anche considerare che il tempo di compilazione varia in base alla dimensione dello sketch.

Per esperienza possiamo dire che, se avrete la pazienza di fare 40÷50 tentativi, alla fine raggiungerete una discreta dimestichezza con questo metodo; ma noi continuiamo a ritenere tale metodica molto scomoda, visto che, con un paio di componenti montati all'occorrenza, ci liberiamo dall'incombenza di stare lì col patema del "carpe diem". Nella prossima puntata presenteremo il progetto del lampeggiante, basato su chip ATmega328, che programmeremo con la tecnica vista in questo articolo. ■

Visualizzatore a quattro digit a bandiera inglese capace di mostrare numeri e scritte formate da simboli alfanumerici.

DISPLAY ALFANUMERICO A CONTROLLO SERIALE

di ALESSANDRO SOTTOCORNOLA

Il principale problema nel controllo mediante microprocessori e microcontrollori dei visualizzatori a più cifre realizzati con display a 7 segmenti o a bandiera inglese, è il numero elevato di linee che servono a gestire i singoli segmenti; ciò si può mitigare in parte ricorrendo

alla tecnica di multiplex, che permette di ridurre le linee di controllo ad un bus che interessa in parallelo i segmenti e ad una linea di comando per l'accensione di ciascun digit. Questa tecnica presenta, però, il problema di complicare i circuiti di controllo, dato che richiede il comando sincronizzato dei display. La soluzione al problema, almeno quando si devono pilotare display relativamente grandi, consiste nel realizzare dei controller posti localmente, ossia dove si trovano i digit, dotati di interfaccia seriale; ciò consente sia di collegare i visualizzatori a distanza portando in giro un numero ridottissimo di fili (ne basterebbero due: segnale dati e massa di riferimento) sia di pilotare il visualizzatore su un circuito stampato consumando una sola linea di microcontrollore, quale



CARATTERISTICHE TECNICHE

- Display a 4 digit + separatore a doppio punto;
- Ridotte dimensioni per facilitare l'implementazione;
- Alimentazione 5 Vcc/100 mA;
- Comunicazione unidirezionale (solo in ricezione) a livello TTL;
- Tabella caratteri di base implementata per facilitare la gestione;
- Possibilità di accendere i segmenti a proprio piacimento.
- Velocità di comunicazione 9.600 bps.

ad esempio la TXD (se il micro ha l'UART integrato) o un I/O digitale configurato come output ed assegnato ad una routine di comunicazione seriale. La gestione seriale permette di lasciare le altre linee del microcontrollore per altre applicazioni.

Il dispositivo descritto in queste pagine nasce proprio in questa filosofia: si tratta di un display a quattro cifre con digit a bandiera inglese, controllato serialmente. Il nostro display potrebbe essere usato ad esempio in sveglie, distributori automatici, contapezzi, e tanti altri apparati dove basti rappresentare valori con quattro caratteri; l'aver adottato, per realizzarlo, display a bandiera inglese, permette di comporre sia messaggi numerici, sia testi, pur rimanendo nei limiti delle quattro lettere. Ciò consente anche di impiegare il dispositivo per pro-

porre menu e messaggi di vario genere in macchinari, distributori, serrature elettroniche ed apparati che richiedono programmazione.

COME FUNZIONA

Il visualizzatore qui proposto consta essenzialmente di un display monoblocco a quattro digit a bandiera inglese con due punti separatori posti tra i primi due digit ed i seguenti, ed un microcontrollore che lo gestisce, presentando verso l'esterno un'interfaccia di comunicazione seriale. Il micro svolge dunque il compito di convertitore dai comandi seriali a quelli relativi alle singole linee del display. Per rendere più facilmente utilizzabile il dispositivo anche da PC, via firmware è stata implementata la gestione della tabella ASCII, il che ci permette di comandare il display mediante codici ASCII allo scopo di visualizzare i caratteri corrispondenti, fermo restando che potranno essere mostrati solo caratteri componibili con la bandiera inglese, quindi almeno tutte le lettere e i numeri.

Il fatto che il microcontrollore funzioni da decoder ASCII e comando di visualizzazione, permette di comandarlo componendo i codici ASCII, cosa che può fare sia un PC, sia un comune microcontrollore.

Per visualizzare simboli particolari o caratteri specifici, abbiamo previsto la possibilità di imposta-

re a proprio piacimento il contenuto del display. Il nostro visualizzatore prevede dunque due modalità operative: "Automatica" ("A") che permette la gestione mediante l'invio di codici ASCII, oppure "Personalizzata" ("P") se si desidera impostare a piacimento i segmenti.

Il protocollo è molto semplice e per un discorso di comodità è stato previsto l'invio forzato di 12 byte, anche se non sempre sono tutti utilizzati.

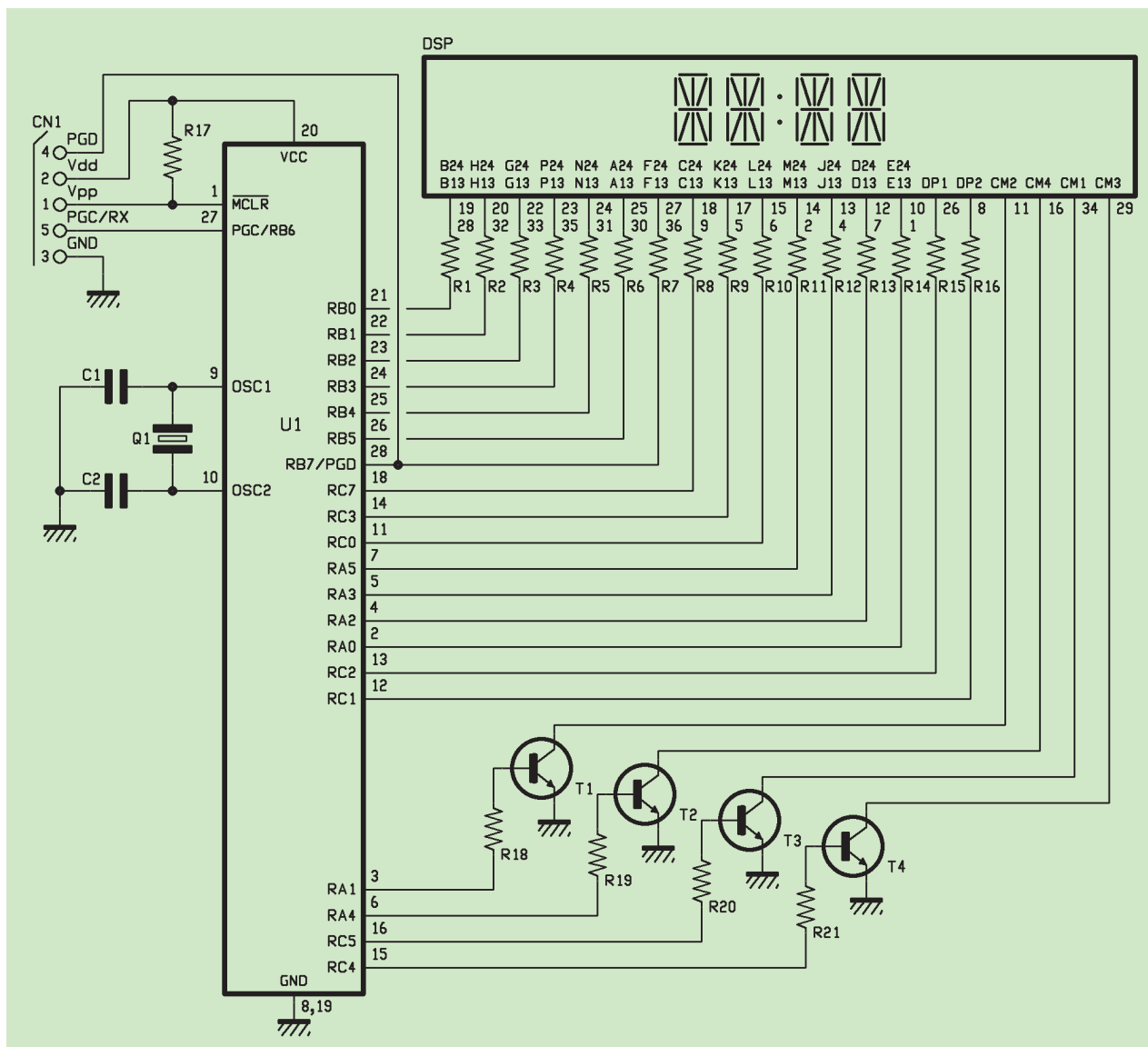
Prima di procedere con la spiegazione del protocollo di comunicazione, è meglio indicare quali sono i caratteri implementati per la gestione nella modalità "Automatica", ovvero tutti i caratteri ASCII richiamabili dalla memoria. La **Tabella 1** mostra i caratteri introdotti; in essa notate che molti caratteri non sono stati considerati, in quanto caratteri speciali e quindi non visualizzabili dal display a bandiera inglese (appaiono solo quelli effettivamente visualizzabili). L'invio al circuito di ogni altro codice non compreso nella tabella ASCII comporterà lo spegnimento del display, in quanto corrispondente a caratteri non contemplati dal firmware e perciò non visualizzabili.

SCHEMA ELETTRICO

Visto il funzionamento di massima, approfondiamo il discorso esaminando lo schema elettrico del circuito, dal quale appare

Tabella 1 - Caratteri visualizzabili e codici ASCII corrispondenti.

COD	CHR	COD	CHR	COD	CHR	COD	CHR	COD	CHR
48	0	49	1	50	2	51	3	52	4
53	5	54	6	55	7	56	8	57	9
60	<	62	>	65	A	66	B	67	C
68	D	69	E	70	F	71	G	72	H
73	I	74	J	75	K	76	L	77	M
78	N	79	O	80	P	81	Q	82	R
83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	193	-	195	+
220	-	254	FULL	255	BLANK				



l'estrema semplicità dell'insieme: tutto si riduce, infatti, ad un microcontrollore, al display monoblocco e a quattro transistor più un quarzo e pochi altri componenti passivi.

Il microcontrollore è un PIC18F2620 in versione SMD, che nella nostra applicazione funziona con il clock generato dal generatore interno ad esso con l'aiuto del quarzo Q1 e dei condensatori C1 e C2.

Le linee PGC, PGD e /MCLR sono utilizzate per la programmazione in-circuit (ICSP) insieme a Vpp e Vdd, cosa utile in quanto il micro è un SMD e verrà

programmato dopo essere stato montato sullo stampato; la linea PGC nel normale funzionamento è la RB6 (settimo bit del registro RB) ed è inizializzata come canale seriale di comunicazione utilizzata dal main program per ricevere i caratteri ASCII da mostrare. Invece la PGD nel normale utilizzo (ossia quando gira il programma principale) è la RB7 e concorre al comando degli anodi del display (i segmenti F della coppia di digit 1-2 e 3-4).

La routine di visualizzazione prevede il comando in multiplex del display: le linee che controllano l'accensione dei singoli segmenti

sono dirette (un I/O del micro gestisce il segmento della bandiera inglese di due digit) mentre i singoli digit vengono accesi collegandone il catodo comune a massa, in sequenza, mediante i transistor T1, T2, T3, T4, che vengono polarizzati in sequenza ed in perfetto sincronismo con il comando delle linee dei segmenti. In breve, quando si deve visualizzare un certo carattere su un determinato display, il microcontrollore porta a livello logico alto le linee dei segmenti corrispondenti e poi manda ad 1 logico la linea, tra RA1, RA4, RC4, RC5, corrispondente al catodo

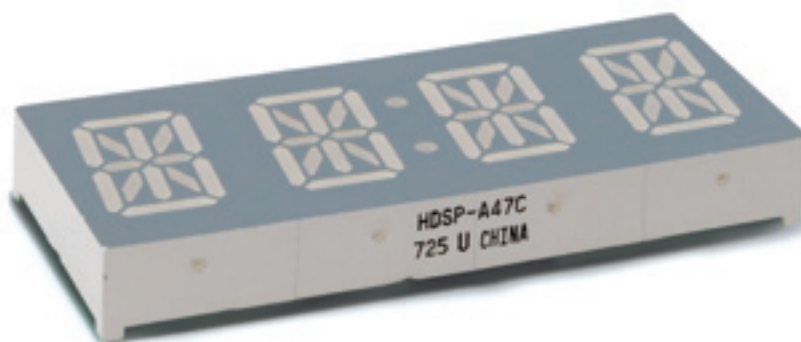
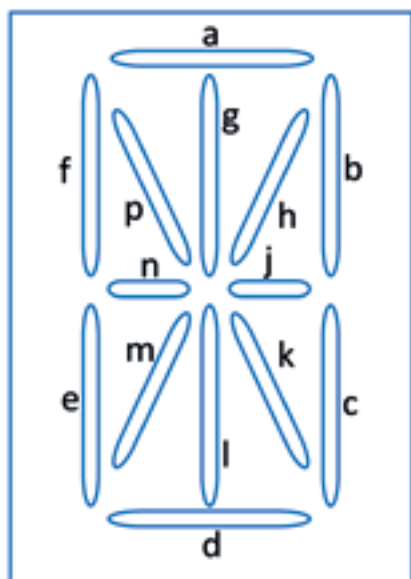


Fig. 1 - Il display alfanumerico utilizzato è catodo comune, composto da 4 digit.

del display corrispondente, in modo che solo quello si accenda e mostri il carattere.

Per fare in modo che il visualizzatore, quando deve mostrare più di un carattere, faccia apparire accesi simultaneamente tutti i digit corrispondenti, il pilotaggio in multiplex avviene ad una frequenza molto elevata, nel senso che tutti i display vengono pilotati sui segmenti ed accesi tramite i transistor, in un tempo minore di $1/24^{\circ}$ di secondo, che è il tempo di persistenza dell'occhio umano.

Quindi, se bisogna comporre una scritta con quattro caratteri, ogni digit deve essere acceso ed aggiornato in circa $1/100$ di secondo. A ciò provvede l'apposita routine di visualizzazione che gira nel microcontrollore e che governa il funzionamento in multiplex.

Per chiarirvi le idee su come viene gestito il display, possiamo vedere insieme, brevemente, come sono collegati i singoli digit: il visualizzatore scelto è monoblocco e contiene quattro caratteri a catodo comune, solo che per ridurre il numero dei pin di collegamento, gli anodi (ossia i terminali indipendenti dei singoli segmenti) non sono del tutto distinti, ma risultano collegati a due a due. Insomma, il digit 1 e il 2 hanno in comune gli anodi, mentre i catodi sono distinti e lo stesso dicasi per i digit 3 e 4; ne risulta un numero di pin pari a 14 (6 anodi dei segmenti numerici esterni e 8 di quelli della bandiera inglese) per ciascuna coppia di caratteri più due per ciascuno dei due punti, oltre a quattro elettrodi per i catodi comuni.

Notate che i catodi dei punti sono in comune con quelli dei di-

git, ovvero uno è in comune con la prima coppia di digit e l'altro con la seconda coppia.

Nel nostro circuito, visto che abbiamo scelto il multiplex totale (cioè multiplexiamo tutti e quattro i digit, mentre avremmo potuto farlo a coppie comandando i display a due a due) mettiamo in parallelo tra loro i pin che comandano le coppie di anodi dei digit, interponendo resistenze di limitazione calcolate per la corrente di un solo segmento, visto che di fatto accendiamo sempre un solo segmento alla volta (infatti il collegamento in parallelo è solo virtuale, in quanto i transistor chiudono il circuito solo per un digit alla volta e non ci sono mai due digit accesi simultaneamente).

Chiediamo la descrizione dello schema elettrico con un cenno all'alimentazione: come avrete notato, il circuito non ha stadio d'alimentazione né regolatori di tensione, ragion per cui va alimentato con una tensione continua stabilizzata di 5 volt (assorbe intorno ai 100 milliampere).

L'alimentazione viene prele-

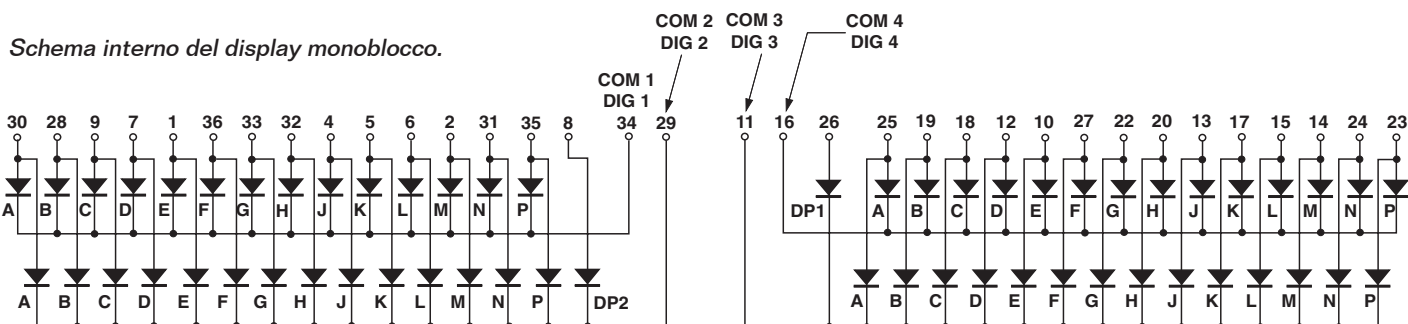


Fig. 2 - Formato della stringa dati.

12 Byte											
*	/	CMD	PUNTI	D1a	D2a	D3a	D4a	D1b	D2b	D3b	D4b

Tabella 2 - Composizione del byte che, nel modo personalizzato, indica quali sono i segmenti da accendere per quanto riguarda i caratteri numerici; vale per i byte dei digit D1a, D2a, D3a, D4a.

Dxa	1 Byte							
bit	7	6	5	4	3	2	1	0
Segmento	a	b	c	d	e	f	g	-

Tabella 3 - Composizione del byte che, nel modo personalizzato, indica quali sono i segmenti da accendere per quanto riguarda i caratteri numerici; vale per i byte dei digit D1b, D2b, D3b, D4b.

Dxb	1 Byte							
Bit	7	6	5	4	3	2	1	0
Segmento	h	k	j	l	m	n	p	-

vata dall'unico connettore di cui dispone, perché il progetto nasce per essere integrato in altri apparati, da cui quindi prende tensione: tipicamente circuiti a microprocessore o microcontrollore, già dotati di alimentatore stabilizzato per la logica.

PROTOCOLLO DI COMUNICAZIONE

Bene, descritto l'hardware spendiamo adesso due parole sul protocollo di comunicazione implementato dal firmware; questo protocollo è strutturato in un pacchetto dati fisso composto da 12 byte, nel senso che il circuito deve ricevere comandi seriali

sotto forma di stringhe di dati lunghe 12 byte. Nella modalità di funzionamento "Automatico" (ASCII) il pacchetto fondamentale è composto da 8 byte, mentre i rimanenti 4 byte non vengono considerati.

Nella modalità di funzionamento "Personalizzato" è obbligatorio inserire tutti e 12 i byte, dato che servono per comporre caratteri fuori standard. In pratica nel modo personalizzato sono i singoli byte ad esprimere quali segmenti del digit interessato debbono restare accesi.

La tipica stringa di dati del sistema è strutturata come mostra la Fig. 2.

In essa, possiamo andare a spiegare il significato dei singoli byte, descritto qui di seguito.

CMD: definisce la modalità di visualizzazione (1 Byte).

A (0x41): visualizza il carattere prelevandolo da tabella ASCII (considera solo D1a, D2a, D3a, D4a).

P (0x50): indica la scrittura personalizzata del display (considera tutti i byte D1a, D2a, D3a, D4a, D1b, D2b, D3b, D4b).

PUNTI; definisce se visualizzare e in che modo i due punti ":" separatori centrali (1 byte):

0 = punti spenti

1 = acceso solo punto basso

2 = acceso solo punto alto

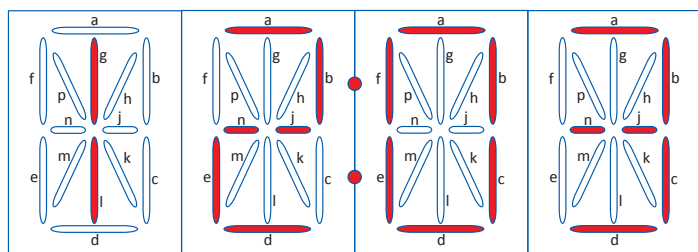


Fig. 3

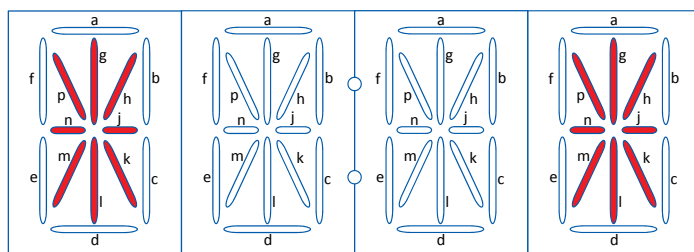


Fig. 4

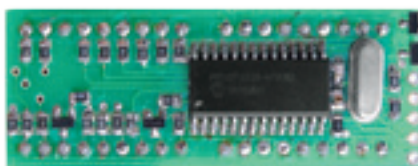
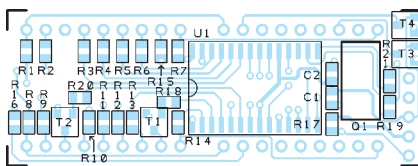
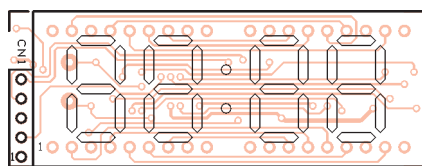
Tabella 4

Visualizzazione ad esempio di un orario, precisamente "12:03" in cui i punti sono lampeggianti											
*	/	0x41	0x04	0x31	0x32	0x30	0x33	0xFF	0xFF	0xFF	0xFF

Tabella 5

Visualizzazione solo sul primo e ultimo display del simbolo asterisco, quindi "*" *											
*	/	0x41	0x00	0x02	0xFF	0xFF	0x02	0xFE	0xFF	0xFF	0xFE

[piano di **MONTAGGIO**]



Elenco Componenti:

R1 ÷ R16: 150 ohm (0805)
 R17 ÷ R21: 4,7 kohm (0805)
 C1, C2: 15 pF ceramico (0805)
 Q1: Quarzo 20 MHz (HC49)
 U1: PIC18F2620-I/SP (MF929)
 T1 ÷ T4: BC817

DSP: Display alfanumerico C.C.
 (Farnell 1003351)

Varie:

- Strip maschio 5 poli
- Circuito stampato

3 = accesi entrambi i punti
 4 = accesi entrambi i punti ma
 in modo lampeggiante ogni
 500 ms.

D1a: in modalità "A" rappresenta il carattere ASCII del display 1; in modalità "P" esprime, in formato binario, lo stato dei primi sette segmenti del display 1 (vedi **Tabella 2**).

D2a: in modalità "A" rappresenta il carattere ASCII display 2. In modalità "P", esprime in formato binario lo stato dei primi sette segmenti del display 2 (vedi **Tabella 2**).

D3a: in modalità "A" rappresenta il carattere ASCII del

display 3. In modalità "P", esprime in formato binario lo stato dei primi sette segmenti del display 3 (vedi **Tabella 2**).

D4a: in modalità "A" rappresenta il carattere ASCII display 4. In modalità "P" esprime in formato binario lo stato dei primi sette segmenti del display 4 (vedi **Tabella 2**).

D1b: in modalità "A" non è gestito, mentre nella modalità "P" esprime lo stato dei secondi sette segmenti del display 1 (vedi **Tabella 3**).

D2b: in modalità "A" non viene gestito, mentre in modalità "P" esprime in formato binario lo

stato dei primi sette segmenti del display 2 (vedi **Tabella 3**).

D3b: in modalità "A" non è gestito; in modalità "P" esprime in formato binario lo stato dei primi sette segmenti del display 3 (vedi **Tabella 3**).

D4b: in modalità "A" > non viene gestito; in modalità "P" esprime in formato binario lo stato dei primi sette segmenti del display 4 (vedi **Tabella 3**).

Per rendervi più comprensibile il protocollo di gestione del display, vediamo alcuni esempi pratici di stringhe con le visualizzazioni corrispondenti.

Nel primo esempio, utilizziamo la modalità automatica ("A") e visualizziamo un orario, ossia 12:03, per cui dobbiamo accendere sul digit di sinistra la cifra 1, su quello subito a destra il 2, sul terzo lo zero e sul digit di destra il 3; dobbiamo anche accendere i due punti. Il comando è quello illustrato nella **Tabella 4**, mentre la situazione del display è mostrata nella **Fig. 3**.

In questo caso si adottano caratteri ASCII.

Il secondo esempio riguarda la modalità personalizzata ("P") e in esso vogliamo visualizzare due asterischi sui display più esterni (il primo ed il quarto) come illustrato nella **Fig. 4**; la stringa corrispondente è quella riassunta nella **Tabella 5**.

REALIZZAZIONE PRATICA

Bene, detto ciò riteniamo di aver spiegato quanto basta sul funzionamento del display ad interfaccia seriale; possiamo quindi spendere due parole sulla costruzione. Il circuito si realizza abbastanza semplicemente, anche se composto (eccetto il display) da soli componenti SMD.

Connettore CN1

- 1: Vpp
- 2: Vdd
- 3: GND
- 4: PGD
- 5: PGC/RX

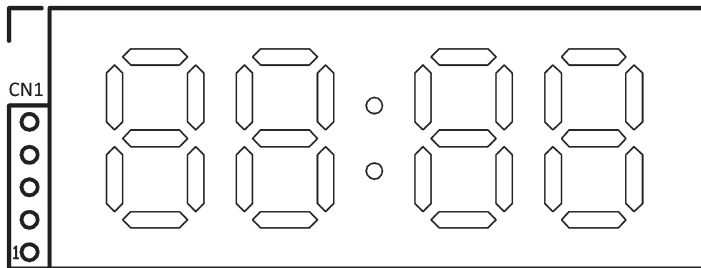


Fig. 5 - Piedinatura del connettore CN1.

L'unica criticità che presenta sta nel montaggio del microcontrollore, essendo questo in versione SMD; gli altri componenti, ossia i passivi ed i quattro transistor, si saldano abbastanza facilmente. Dunque un volta incisa e forata la basetta (che è del tipo a doppia faccia, quindi va realizzata per fotoincisione dopo aver scaricato le due tracce rame dal nostro sito www.elettronica.in.it) dovete realizzare le interconnessioni fra le due facce e poi montare per prime le resistenze; allo scopo, procuratevi un saldatore con punta sottile, da non più di 20 W di potenza, e del filo di lega saldante da 0,5 mm di diametro. Non guasta disporre di un po' di flussante, che aiuterà la saldatura soprattutto se usate lega saldante compatibile Rohs, dato che sovente questa aderisce meno e richiede, per sciogliersi bene, temperature più elevate della classica lega stagno/piombo. A questa dotazione, utile per tutte le fasi del montaggio, non guasta aggiungere una lente d'ingrandimento da tavolo provvista di lampada, grazie alla quale potete vedere meglio mentre sistemate i vari componenti e riuscite ad accertare la presenza di baffi di lega saldante o saldature che cortocircuitano contatti vicini. Allora, con tutta la strumentazione del caso, sistemate le resistenze ed i condensatori in SMD, quindi montate il quarzo e poi i quattro transistor, il cui posizionamento è obbligato in quanto se li tenete con la sigla a vista possono essere saldati solo nel verso giusto. Passate quindi al microcontrollore, che vi consigliamo di sistemare ben centrato sulle rispettive piazzole (sulle quali avrete avuto cura di spalmare della pasta flussante) e poi di saldare a partire dai piedini che stanno sugli angoli;

Samsung ha finalmente mostrato al pubblico del CES 2011 tenutosi a Las Vegas, i primi prototipi funzionanti di schermo i AMOLED flessibili e trasparenti. Questi dispositivi, a lungo teorizzati, sono una realtà che i visitatori della manifestazione americana hanno potuto toccare con mano. Volutamente, Samsung ha messo in mostra display nudi e crudi per farli osservare in tutte le loro caratteristiche, prima fra tutte la flessibilità, che permette loro di adeguarsi alla sagoma di supporti di vario genere, elementi architettonici ecc. Ad esempio, i display potrebbero essere impiegati in cellulari e smartphone arrotolabili o fungere da monitor per PC da portare in giro arrotolati, in un'apposita

custodia, come le tastiere in gomma arrotolabili che già esistono da tempo. Gli AMOLED flessibili presentati da Samsung sono da 4,5 pollici ed hanno una risoluzione di 800x480 pixel; il loro consumo ed il tempo di risposta sono i più bassi ottenibili dalla tecnologia AMOLED. Oltre a quelli flessibili, la casa coreana ha proposto i nuovi display trasparenti nelle misure da 7 e 19 pollici, che dimostrano uno studio di fattibilità a bordo di un tablet o PC touch-screen. Ora si attende la diffusione di questi nuovi dispositivi: per gli AMOLED flessibili, si prevede già l'anno prossimo l'uscita di uno smartphone con display multitouch.

Arrivano gli OLED flessibili

in pratica suggeriamo di stagnare nell'ordine i pin 1, 28, 14 e 15, allo scopo di bloccare bene il chip per poterlo poi saldare bene. Fatto ciò, proseguite con la saldatura dei restanti pin, badando di non cortocircuitare quelli adiacenti con la lega saldante. Terminato il montaggio dei componenti SMD, dal lato opposto

della basetta potete inserire il display, ricordando che deve essere montato come mostrato nel piano di montaggio visibile in queste pagine e rammentando che il lato lungo riportante la sigla deve essere rivolto verso l'alto quando il display è montato a destra delle piazzole riservate al connettore CN1.

8^a FIERA REGIONALE DELL'ELETTRONICA SCANDIANO (REGGIO EMILIA) 22/23 OTTOBRE CENTRO FIERISTICO 2011



**MERCATINO DELLE PULCI
RADIOAMATORIALI
TELEFONIA - COMPONENTISTICA
COMPUTER - HI-FI CAR - RADIANTISMO
CB e OM - VIDEOREGISTRAZIONE**

**ORARI: Sabato 23 Ottobre 9 - 18,30
Domenica 24 Ottobre 9 - 18**

**INGRESSO euro 7,50 RIDOTTO euro 5
GRATUITO ragazzi fino a 14 anni**

**INFO: 0522/764302-857436
www.fierascandiano.it**



con il patrocinio di
A.R.I. sez. Scandiano



A proposito di connettore, potete realizzarlo con un pin strip da 5 punte a passo 2,54 mm, da stagnare dal lato dei componenti, in modo che le punte escano da dove c'è il display.

A questo connettore collegherete il programmatore, una volta terminato e verificato il montaggio, per caricare il firmware nel microcontrollore via ICSP; una volta programmato il circuito, lo stesso connettore vi servirà per collegare il dispositivo all'apparato che dovrà comandarlo.

A tal proposito rammentate che vengono effettivamente utilizzati solo i pin 2, 3, 5, che sono rispettivamente l'alimentazione positiva (5 volt in continua) la massa e la linea dati; quest'ultima richiede segnali a livello TTL o compatibile (0/5V o 0/3V). La Fig. 5 illustra le connessioni del connettore CN1: seguitemela per realizzare il cavetto di collegamento con l'apparato da cui prendere i comandi. Se pensate di gestire il display dalla seriale di un PC, ricordate che vi occorre un convertitore RS232/TTL, ottenibile con una sezione dell'integrato MAX232 o, più semplicemente, con un transistor NPN collegato con la base (tramite una resistenza da 10 kohm) alla linea TXD della porta e l'emettitore a massa (GND della seriale); dal collettore, connesso al positivo dei 5V (Vdd) del circuito con una resistenza da 4,7 kohm, potete prelevare i dati da inviare al contatto 5 del CN1. Siccome la RS232 fornisce livelli logici sia positivi che negativi (tipicamente $\pm 12V$) inserite un diodo 1N4148 in parallelo alla giunzione base-collettore del transistor, collegato con l'anodo sull'emettitore ed il catodo sulla base; il diodo provvederà a tagliare le tensioni negative evitando che il transistor venga danneggiato. Che comandiate il circuito da un PC o da una scheda a microcontrollore, ricordate che per comandarlo dovete obbligatoriamente comporre delle stringhe che rispettino il protocollo spiegato nelle pagine precedenti. ■



per il MATERIALE

Tutti il materiale utilizzato per realizzare questo display è facilmente reperibile in commercio. Il master del CS ed il firmware col quale programmare il micro sono scaricabili gratuitamente dal sito della rivista (www.elettronica.in.it). Il microcontrollore vergine PIC18F2620-I/SP è reperibile presso Futura Elettronica al prezzo di 12,00 Euro, IVA compresa.

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