

160 PAGINE!

Mensile di progettazione elettronica, attualità scientifica, novità tecnologiche.

€ 6,00

Anno XVII - n. 160
Ottobre 2011

Elettronica In

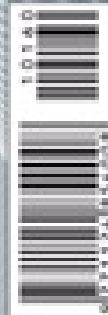
oltre l'elettronica

www.elettronicain.it

Tecnologia
Al via
GALILEO



Wi-Fi shield:
aggiungi link Wi-Fi
al tuo Arduino



Oscilloscopio
palmare monotraccia

Robot Area

- Quadricottero: software e test di volo
- Verso Nessie 2012
- News ed Eventi



- Arduino Motor Shield
- Analizzatore di rete: note finali
- Lampeggiante programmabile basato su Arduino
- FPGA SmartFusion
- Corso NI Multisim
- Le nuove board Arduino
- Corso iPhone

• NOVITÀ •
Corso **Processing**



ANALIZZATORE DI RETE: NOTE FINALI

Concludiamo il discorso sulla programmazione presentando gli strumenti hardware e software del caso. Ultima puntata.

..... dell'ing. SALVATORE COSENTINO

Se avete seguito i due precedenti articoli dedicati all'analizzatore di rete, avrete notato che abbiamo volutamente tralasciato le note di programmazione del microcontrollore, rimandandole ad un futuro approfondimento; ebbene, trovate tale approfondimento in queste pagine, dove vi spiegheremo come si carica il firmware nel micro, ma non solo: coglieremo l'occasione per proporvi tre schede di programmazione

utilizzabili allo scopo. Come abbiamo già detto nelle scorse puntate, a meno di montare un microcontrollore già programmato esternamente, una volta montato il circuito sarà indispensabile procedere alla programmazione in-circuit del micro tramite la porta BDM presente a questo scopo. A tal fine è indispensabile procurarsi un'opportuna interfaccia BDM, supportata dall'ambiente di sviluppo (IDE) ufficiale CodeWarrior, che è

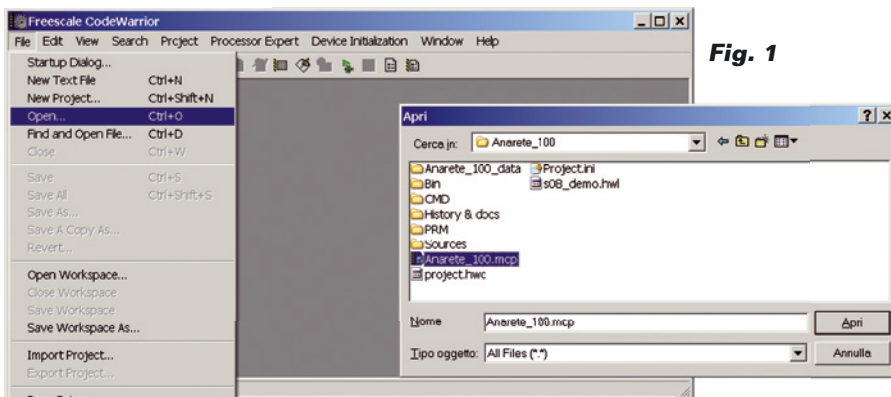


Fig. 1

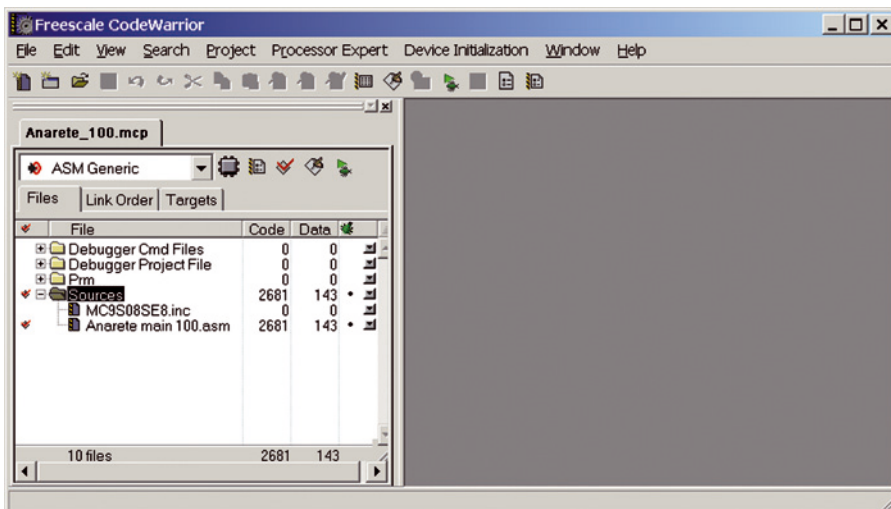


Fig. 2

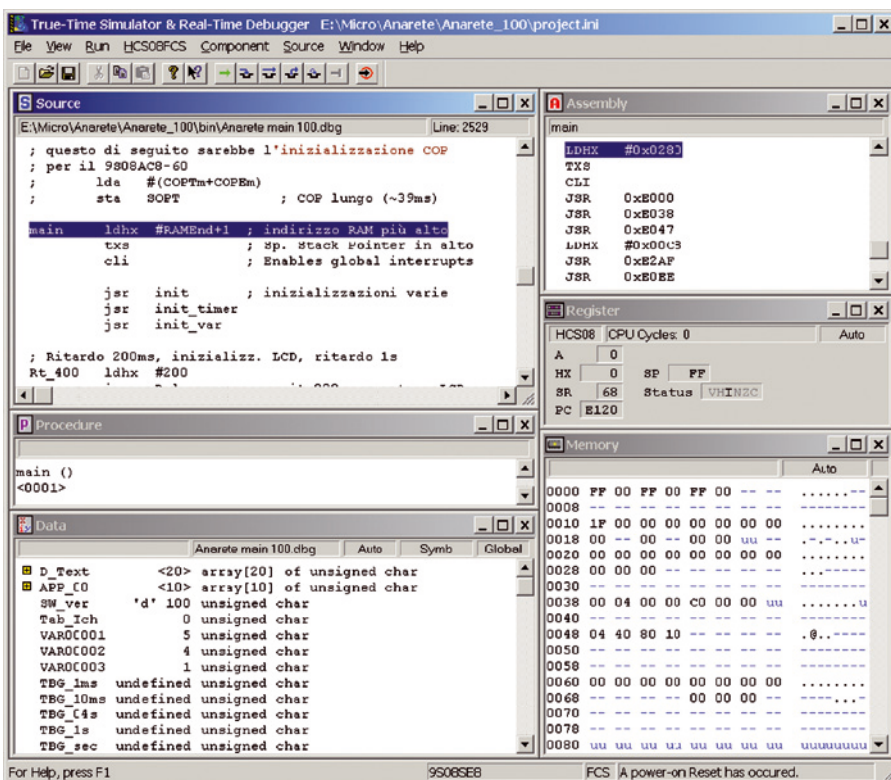


Fig. 3

quello tramite cui effettueremo il download del firmware sul microcontrollore. Prima di iniziare è indispensabile installare il CodeWarrior su un PC con sistema operativo Windows a 32 bit (da Windows XP a 7). Prove effettuate con Windows 7 a 64 bit hanno richiesto l'emulazione XP. La versione con cui sono state effettuate tutte le prove è la V6.3 classic, "special edition". Questo software è in genere fornito su disco con alcuni adattatori di interfaccia BDM, ma può essere comunque scaricato dal sito della Freescale. Conviene procurarselo per tempo, dato che il suo file di installazione "pesa" circa 500 MB. Esiste una versione più recente, la v.10.1 "Eclipse", valida anche per sistemi a 64 bit o per Linux, ma non l'abbiamo ancora provata. Supponendo di aver installato correttamente l'IDE e, successivamente, il driver del dispositivo di interfaccia, è possibile procedere con la programmazione della nostra scheda. A tal fine, sul nostro sito www.elettronica.in.it, nella sezione dedicata a questo articolo, trovate due versioni del progetto originale completo del software in assembler commentato. La prima versione è la "Anarete_100", quella che consigliamo di installare almeno inizialmente. La seconda, "Anarete_110", è del tutto analoga ma vi è stata inserita una routine di media delle letture, che ha rallentato notevolmente l'immediatezza di visualizzazione a favore di maggiore leggibilità. I due progetti sono contenuti in altrettanti file compressi che occorre scompattare in un'opportuna directory dati (ad esempio *D:/micro*) su cui fare le prove. Il nome e la collocazione di questa directory non sono vincolati. I progetti si trovano

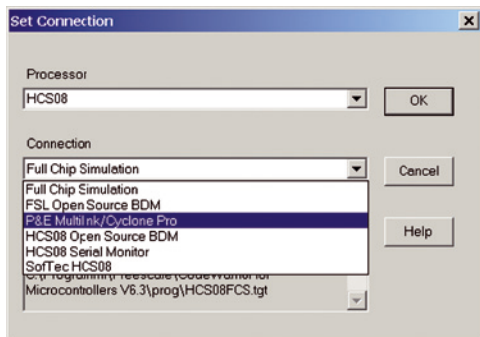


Fig. 4

all'interno di altrettante sottodirectory che comprendono file di inizializzazione, sorgenti, file oggetto e quant'altro necessario. Gli esempi qui descritti sono stati fatti con il progetto "Anarete 100" e con l'interfaccia BDM "Multilink USB-ML-12" della P&E. Per poter programmare la scheda del nostro circuito è anche indispensabile alimentarla con una sorgente di alimentazione ausiliaria isolata dalla rete; in nessun caso si potrà collegare la scheda alimentata direttamente dalla rete. Nel caso di interfacce in grado di alimentare direttamente la scheda da programmare, è necessario disabilitare questa possibilità per evitare di sovraccaricare i regolatori shunt montati sull'Anarete. Il circuito da noi preferito per la sua semplicità, fa uso di un alimentatore da laboratorio esterno collegato come illustrato nella Fig. 1 nella seconda puntata dell'articolo, ossia ai punti A e B visibili nella Fig. 3, della solita seconda puntata. Questa alimentazione è valida anche se sono stati montati i due condensatori C12 e C13 che vanno verso l'alimentazione

di rete, ma ha il difetto di non consentire l'illuminazione del display, il quale resta poco leggibile durante le prove (comunque si può leggere pur con qualche difficoltà, se il contrasto è regolato correttamente). Poiché non sono previsti appositi terminali, è indispensabile collegare una coppia di fili isolati di collegamento rosso (positivo) e nero (negativo) di piccolo diametro direttamente alle piazzole dei

componenti, alle cui estremità occorre connettere l'alimentatore esterno. Inizieremo la programmazione collegando la scheda all'alimentazione provvisoria come descritto e la porta di programmazione BDM alla interfaccia, ad alimentazione spenta. L'interfaccia deve quindi essere collegata ad una porta USB del PC dal quale effettueremo la programmazione; se i driver sono stati montati corret-

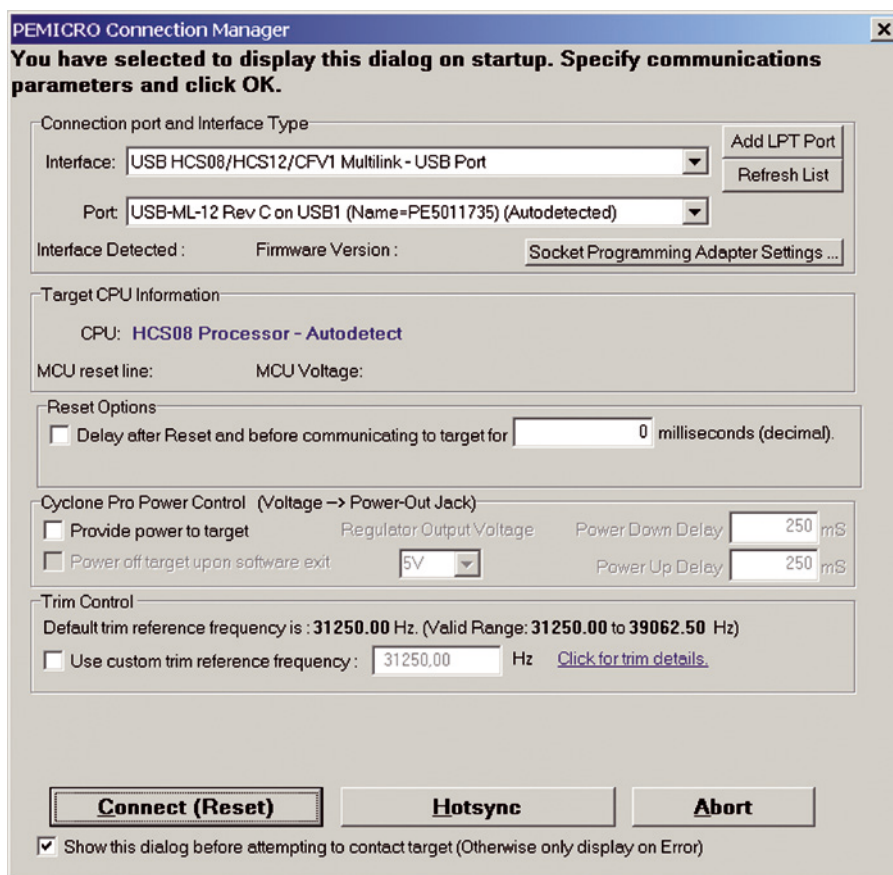


Fig. 5

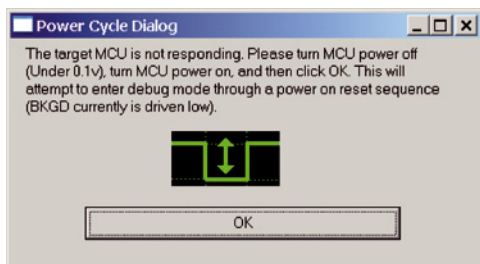


Fig. 6

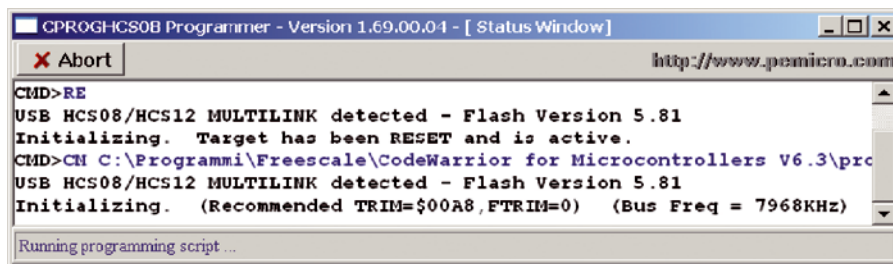


Fig. 7

tamente, l'interfaccia accenderà un LED, blu sul "Multilink", che ne segnalerà il riconoscimento. A questo punto occorrerà lanciare l'ambiente di sviluppo CodeWarrior ed accendere l'alimentatore esterno per la scheda. Dall'ambiente di sviluppo dobbiamo quindi aprire il file di progetto all'interno della cartella dove lo avremo copiato. Nel nostro esempio è il file *Anarete_100.mcp* (Fig. 1). Verrà così caricato l'intero albero del progetto nel desktop dell'IDE. Su questo dovremo spuntare la cartella "source" cliccandovi lateralmente (♥): così facendo verrà anche spuntato il file di programma vero e proprio, "Anarete main 100.asm". Fatto ciò dovremo compilare il programma e caricarlo sul micro, cliccando sul pulsante corrispondente, ossia (Fig. 2).

Dopo una veloce compilazione si aprirà un nuovo quadro relativo al programma separato di simulazione e debug (Fig. 3). Da questo quadro, relativo alla simulazione, si dovrebbe avviare in automatico l'interfaccia di comunicazione con il programmatore BDM. Se ciò non avviene, occorre aprire il menu a tendina "Component" e da questo il sottomenu "Set connection....". Nella finestra di dialogo che si apre (Fig. 4) bisogna scegliere il tipo di interfaccia (nel nostro caso la "P&E Multilink/Cyclone Pro") e cliccare su OK. Questo lancerà comunque l'interfaccia di comunicazione, la quale automaticamente rileverà il dispositivo di interfaccia, nel nostro caso "USB-ML-12 Rev C on USB1...." (Fig. 5).

In caso questa non fosse rilevata, staccate e riattaccate il connettore USB dell'interfaccia su PC, per forzarne il rilevamento e controllare l'accensione dei due

LED sull'interfaccia: quello blu di collegamento e quello giallo di rilevamento dell'alimentazione sulla scheda "target" (il nostro circuito). A questo punto bisogna cliccare sul pulsante *Connect (Reset)* per iniziare il download: si dovrebbe avviare la routine di Flash in memoria del firmware.

A volte questo non avviene immediatamente ed appare la finestra di Fig. 6, che ci invita a togliere e riapplicare alimentazione alla scheda da programmare per causarne il reset e subito dopo a fare clic sul pulsante OK. Così dovrebbe avviarsi il programmatore (Fig. 7), che si chiuderà poco dopo al termine del caricamento corretto, riaprendo il quadro di debug (Fig. 8). Il firmware sarà stato così caricato nella Flash del micro e diverrà possibile avviarlo con il programmatore inserito, facendo clic sul pulsante di avvio, quello con la freccia: ➡.

Osservando il display, è immediato dedurre se la nostra procedura ha avuto il successo sperato: dovrà infatti partire il programma e verrà mostrato il quadro di inizializzazione. A questo punto, supponendo che non occorra fare altri esperimenti, è possibile staccare l'alimentazione dal nostro circuito, disconnettere l'interfaccia di programmazione e i collegamenti provvisori, rimontare tutto e riaccendere il circuito tramite il suo collegamento alla rete elettrica, prendendo le opportune precauzioni per l'isolamento, onde evitare pericolose folgorazioni o altri guai.

DISPOSITIVI DI INTERFACCIA DI PROGRAMMAZIONE

Sono diverse le interfacce di programmazione della famiglia di microcontrollori Freescale a

8 bit che è possibile utilizzare per lo sviluppo e per le prove. Alcune sono di tipo industriale, altre contenute in schede dimostrative, altre ancora nate da progetti amatoriali (il cosiddetto progetto "Open BDM"). Di seguito illustreremo brevemente ciascuno dei tipi citati, il cui costo è ragionevolmente alla portata anche dell'hobbista che voglia portare avanti qualche esperimento.

La prima interfaccia BDM è quella proposta dallo sviluppatore ufficiale che affianca Freescale a questo scopo: la P&E. Il modello più economico è l'*USB-ML-12 rev.C Multilink Adapter*, che può essere acquistato in diversi siti in Italia ed all'estero ad un prezzo compreso tra i 75 e i 90 € + IVA. Per questo importo o poco più, le spese di spedizione sono spesso gratuite (anche dagli U.S.A.) ma è bene controllare e comparare le offer-



te. Il dispositivo in questione è altamente professionale e valido per tutta la famiglia Freescale 9S08 e l'economica RS08 a 8 bit, nonché la HC(S)12 a 16 bit e la recente ColdFire V1 a 32 bit, hardware compatibile con alcuni microcontrollori ad 8 bit. Per una descrizione dettagliata

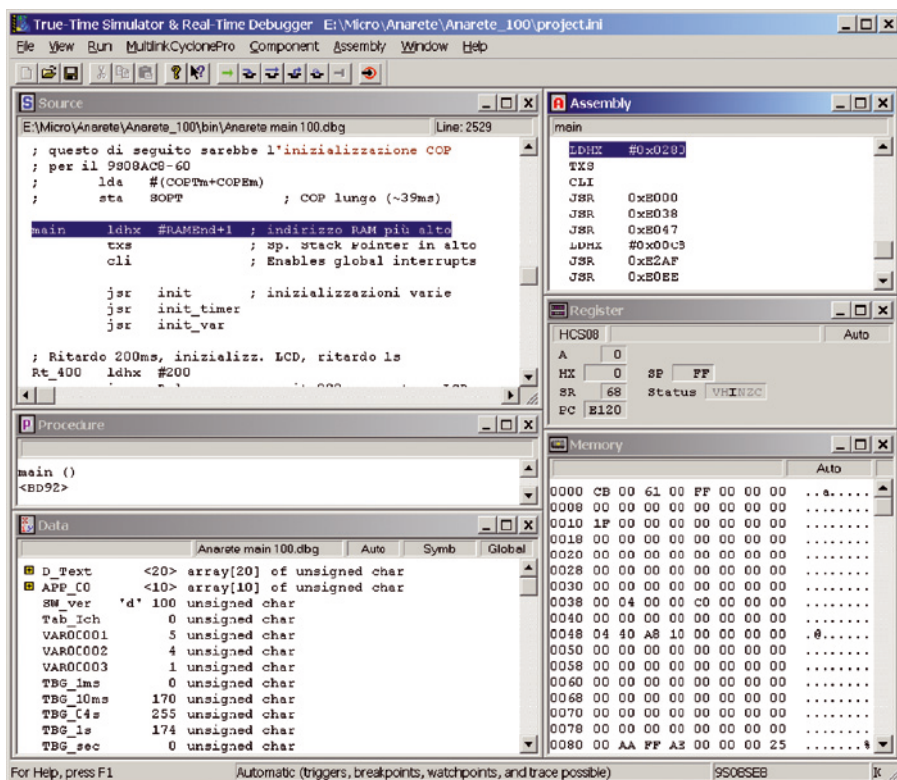
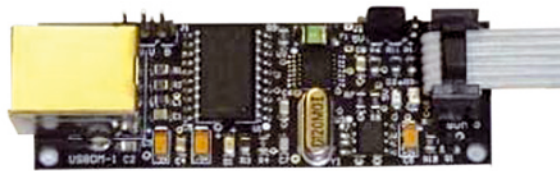


Fig. 8

pur essendo montata industrialmente dalla Wyttek. Si chiama *USBDM-1-JS16* ed è sostanzialmente equivalente all'interfaccia della P&E, pur costando circa 1/5. Per un pezzo singolo la spedizione dagli USA costa un po' più della scheda, ma il totale è comunque estremamente conveniente. Per tutti questi motivi "potrebbe" quindi essere la soluzione migliore per l'hobbista che non ha bisogno di un dispositivo "carrozato" come il *USB-ML-12 rev.C Multilink Adapter*.

Purtroppo la prova sul campo non è stata altrettanto soddisfacente: l'uso di driver non del tutto standard e una certa macchinosità di installazione non hanno consentito di conseguire un funzionamento corretto, per cui non ci sentiamo di consigliarla come primo dispositivo a neofiti che non abbiano mezzi per verificarne il buon funzionamento e non godano di una certa abilità nello "smanettare" con driver e configurazioni. Informazioni su questa scheda e su numerosi altri prodotti Wyttek a basso prezzo basati sulla famiglia 9S08, possono essere reperite sul sito www.evbplus.com, ove potete anche acquistare la scheda di programmazione.



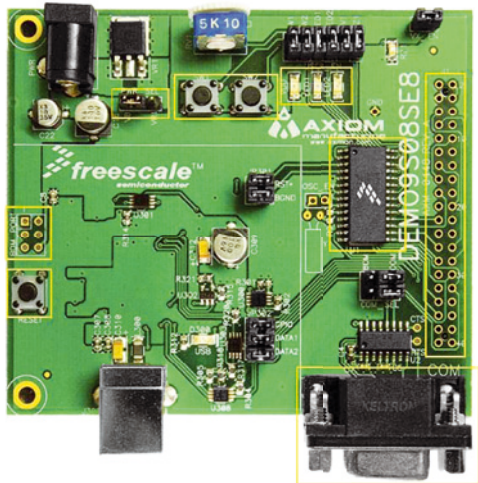
Tutte le schede descritte necessitano dell'IDE CodeWarrior V6.3 o superiore, "special edition". Per le prime due schede descritte, questo software è in genere fornito su disco, mentre per la terza va scaricato dal sito della Freescale.

si veda il sito Freescale (www.freescale.com) o quello P&E (www.pemicro.com).

La seconda interfaccia è quella contenuta nella scheda dimostrativa e di valutazione del microcontrollore 9S08SE8, quello utilizzato nel nostro circuito. Infatti questa scheda, costruita da AXIOM per Freescale, oltre a permettere di sperimentare il μC in questione riguardo a molte delle sue funzioni hardware più generiche (linee di I/O, pilotaggio di LED, ingresso da switch

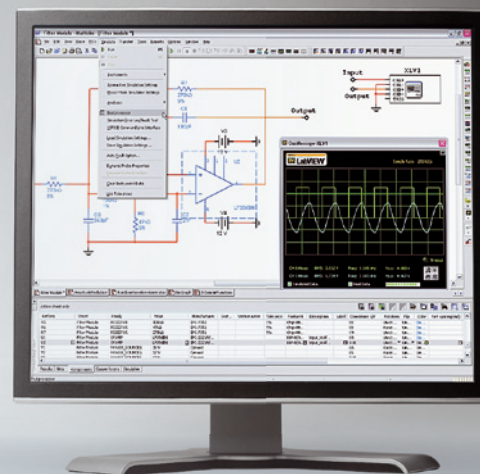
e da ADC, porta RS232) con semplici programmi, può essere facilmente usata per programmare e "debuggare" lo stesso microcontrollore montato su schede esterne. Una buona scelta, quindi, per chi volesse fare esperimenti hardware e software prima di montare un circuito applicativo.

Anche il prezzo è competitivo; infatti, nonostante l'uso non sia limitato alla semplice interfaccia, questa scheda costa meno della proposta P&E: la si può trovare intorno ai 60 € + IVA negli stessi posti e con le stesse modalità della *USB-ML-12 rev.C Multilink Adapter*. L'unico neo è che, sebbene non ci siano reali limitazioni hardware, l'uso per la programmazione esterna potrebbe essere limitato via software alla sola famiglia 9S08SE. La scheda si chiama *DEMO9S08SE8* e se ne può facilmente scaricare la documentazione, compresi schemi e guide, sul sito Internet della Freescale. L'ultima scheda è figlia del progetto open-source USBDM,



Conoscere e usare NI Multisim

Progettazione più semplice,
migliore simulazione.



Operiamo simulazioni in ambito digitale utilizzando la componentistica classica dell'elettronica digitale di tipo cablato e la strumentazione di misura disponibile. Ottava puntata.

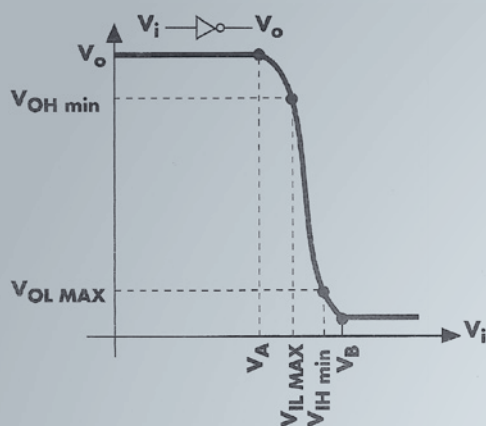
di Enrico Ambrosini

Riprendiamo lo studio di Multisim vedendo come sia possibile operare simulazioni in ambito digitale, avvalendoci di componentistica classica dell'elettronica digitale di tipo cablato e la strumentazione di misura del software; prossimamente, cioè nelle prossime puntate, ci occuperemo anche di microprocessori e PLD.

SIMULAZIONE DIGITALE E ANALOGICA

In Multisim la simulazione analogica e quella digitale possono coesistere ed essere utilizzate in contemporanea sullo stesso circuito; questa situazione, estremamente positiva, può però generare errori nella valutazione dei risultati della simulazione stessa, in quanto i componenti tipici dell'elettronica digitale disponibili in Multisim sono pensati per una simulazione digitale, ovvero per un'analisi in termini di livelli logici, secondo i valori previsti per la famiglia logica scelta, ma senza considerare le relative correnti e le variazioni di tensione, rispetto ai livelli logici teorici, per effetto delle correnti erogate o assorbite dai gate. Prima di procedere ricordiamo che *un dispositivo digitale* (di tipo binario) *opera solo con due valori di*

tensione: il livello alto (H) e il livello basso (L) che, normalmente, sono interpretati in binario, rispettivamente come 1 e 0. Un dispositivo analogico può invece lavorare con tutti gli infiniti valori di tensione e corrente previsti nel range di funzionamento. Per una piena comprensione di quanto sin qui detto, possiamo osservare il grafico della caratteristica di trasferimento di un NOT riportata in Fig. 1 (tratta dal volume *L'Elettronica Digitale* di E. Ambrosini, citato nella prima puntata in bibliografia), che esprime come varia la tensione in uscita V_o in funzione di quella in ingresso V_i . Sebbene si tratti di un componente digitale, il suo effettivo comportamento è di tipo analogico, visto che per ogni valore di tensione in ingresso V_i presenta un diverso valore di V_o ; infatti se V_i è inferiore a V_A , l'uscita ha il livello logico alto. Inoltre il livello logico alto in uscita è considerato tale finché la V_o non scende sotto il valore V_{OHmin} e in questi termini il comportamento è idealizzato a digitale. Se la tensione di ingresso V_i è superiore al valore V_B , l'uscita è a livello logico basso; inoltre il livello zero in uscita è considerato tale finché la V_o non sale sopra il valore V_{OLMAX} e in

Fig. 1 - La caratteristica di trasferimento di una NOT.

questi termini il comportamento è idealizzato a digitale. I valori limite in ingresso per il funzionamento digitale sono $V_{IL\ MAX}$ e $V_{IH\ min}$; per valori di V_i intermedi tra queste due tensioni il comportamento in termini digitali non è corretto e si può dire che variando con continuità la tensione d'uscita al variare dell'ingresso, il dispositivo opera in analogico.

In estrema sintesi, si può quindi affermare che il comportamento digitale dei componenti logici è solo una semplificazione della realtà, che invece è di tipo analogico.

Proviamo ora a tracciare la caratteristica di trasferimento di una porta logica NOT tramite Multisim. Per farlo, useremo una NOT CMOS del tipo 4069. Il circuito proposto è riportato in **Fig. 2** (file **NOT_1**).

Notate in particolare che:

- l'uso di una NOT della famiglia CMOS alimentato a 5 V (l'alimentazione è inserita in automatico da Multisim; per trovare il dispositivo, seguite il percorso **Place/Component/CMOS/CMOS_5V**);
- l'uso della massa digitale **GND** (**Place/Component/Sources/POWER_SOURCES/DGND**); in effetti potevamo anche usare

la massa (**GROUND**) sin qui usata ma abbiamo scelto quella digitale per evidenziare la possibilità, in Multisim, di separare la massa analogica da quella digitale;

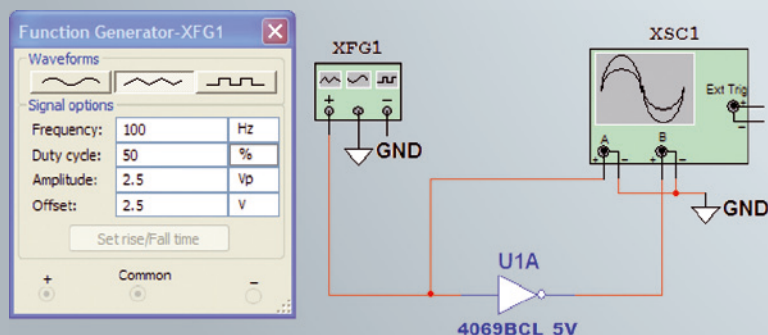
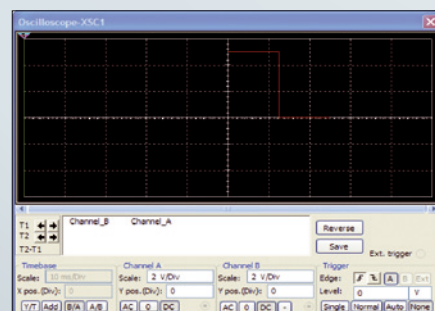
- il generatore di funzioni è predisposto per fornire una tensione triangolare che varia tra 0 e 5 V (ottenuta fissando un'ampiezza massima di picco di 2,5 V e aggiungendo un offset di 2,5 V in modo da traslare verso l'alto di 2,5 V la tensione alternata originaria di $\pm 2,5$ V);
- variando la V_i tra 0 e 5 V e usando l'oscilloscopio in x/y si traccia la curva, che esprime come varia la V_o in funzione della V_i .

Nella **Fig. 3** riportiamo il grafico della caratteristica di trasferimento ottenuta tramite questa simulazione digitale: come potete vedere, il grafico è idealizzato, infatti per V_i inferiore a 2,5 V (cioè 1/2 della tensione di alimentazione) la V_o è a livello alto (5 V) e per V_i maggiore di 2,5 V la V_o è a livello basso (0 V). Se non fosse chiaro che si tratta di una simulazione digitale, si potrebbero trarre conclusioni errate sul comportamento elettrico della NOT utilizzata.

Vediamo ora come si può fare una simulazione analogica che ci permetta di raggiungere risultati più corretti. Per farlo utilizziamo il circuito di **Fig. 4** (file **NOT_2**).

Questo circuito per ora non vi spieghiamo come realizzarlo (lo vedrete in un'altra puntata del corso) e quindi usate il nostro. Con un doppio clic del mouse con il puntatore sul NOT analogico (analogic_inverter) e cliccando su **Edit HB/SC** potrete vedere il circuito interno della nostra NOT (**Fig. 5**).

Ora il NOT è simulato con il suo effettivo circuito elettrico e quindi essendo i componenti

**Fig. 2** - Simulazione digitale del tracciamento della caratteristica di trasferimento di una NOT.**Fig. 3** - Caratteristica di trasferimento della NOT con simulazione digitale.

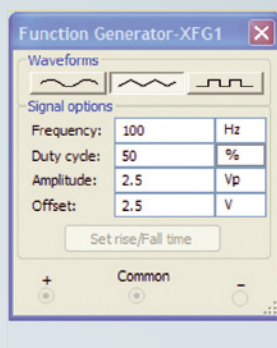


Fig. 4 - Circuito per la simulazione analogica della NOT CMOS.

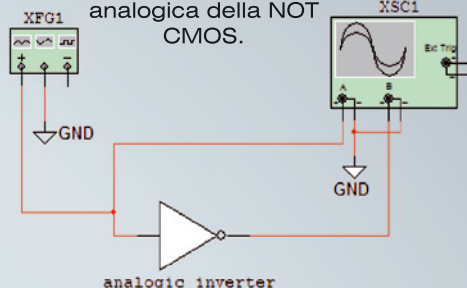
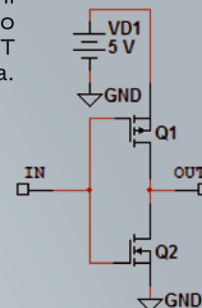


Fig. 5 - Il circuito interno della NOT analogica.



di tipo analogico tale sarà anche la simulazione. Una regola: *se non si usano componenti espressamente previsti per la simulazione digitale (vedremo poi quali sono) la simulazione è analogica.*

Se ora analizziamo la caratteristica di trasferimento di **Fig. 6**, vediamo che il suo comportamento evidenzia una situazione molto meno ideale che corrisponde a quella effettiva di questi dispositivi.

Ma allora a cosa serve la simulazione digitale, se fornisce risultati solo approssimati?

La spiegazione è che nella maggioranza dei casi ciò che conta in un circuito digitale è il livello logico e in tutti questi casi la simulazione digitale va benissimo e risulta molto efficiente. Inoltre un eventuale circuito analogico collegato a uno digitale richiede da quest'ultimo un livello logico o una transizione HL o LH, ed anche in questi ultimi casi non c'è alcun problema. La simulazione digitale può creare difficoltà solo in casi particolari: un esempio lo abbiamo appena visto e un altro è quello legato alla simulazione di generatori di clock realizzati con le porte logiche. Si tratta di circuiti che sfruttano i transistori di carica e scarica dei condensatori e quindi, trattandosi di una situazione tipicamente analogica, non potremo realizzare la simulazione utilizzando in Multisim dispositivi digitali. Dovremo, in alternativa, usare dispositivi analogici come il nostro inverter o più semplicemente sostituire, nel sistema da simulare, il clock ottenuto dalle porte con uno dei tanti clock disponibili in Multisim. Notate che se le porte da voi usate sono di tipo a trigger di Schmitt, non ci sono problemi: essendo concepite per avere due soglie di intervento, operano in termini sostanzialmente analogici.

I COMPONENTI PER LA SIMULAZIONE DIGITALE

Vediamo ora quali sono le famiglie di

dispositivi digitali disponibili in Multisim:

- famiglia TTL (**Place/Component/TTL**) con le diverse sotto-famiglie (STD, S, LS, F, AS e ALS); l'alimentazione già inserita è 5 V; nel caso di IC con all'interno singole porte logiche (ad esempio 7400 con quattro NOT) è possibile accedere alle singole porte senza necessariamente riportare l'integrato completo (qualora ciò interessi, si può fare scegliendo la variante **_IC**);
- famiglia CMOS (**Place/Component/CMOS**) della quale sono disponibili le serie 4xxx con le alimentazioni inserite a scelta di 5, 10 e 15 V (anche in questo caso esiste la variante **_IC**), la serie HC (a 2, 4 e 6 V con la variante **_IC**) e la serie TinyLogic (a 2, 3, 4, 5 e 6 V) qualora servano singole porte logiche senza considerare il dispositivo commerciale di riferimento;
- famiglia TIL (**Place/Component/Misc Digital**) che è una logica con livelli ideali a 0 e 5 V utile per realizzare schemi logici prescindendo dai dispositivi commerciali;
- memorie (**Place/Component/Misc Digital**);
- linee di trasmissione (**Place/Component/Misc Digital**) ad esempio IC per passare dai livelli 0-5 V a quelli RS232;
- dispositivi a cavallo tra digitale e analogica

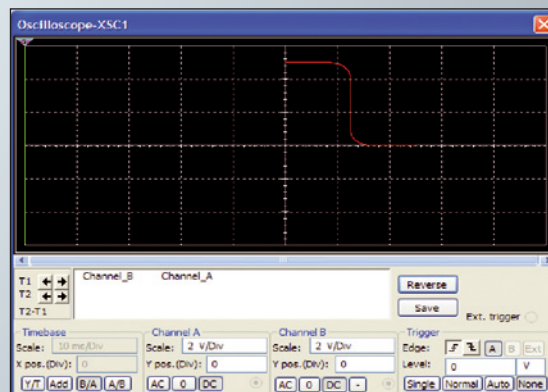


Fig. 6 - Simulazione analogica della caratteristica di trasferimento di una NOT CMOS.

Fig. 7 - I pulsanti del logic converter.

 → $\overline{101}$	pulsante 1: noto il circuito logico e collegati i suoi ingressi e la sua uscita allo strumento, permette di ottenere la tabella della verità.
$\overline{101}$ → A B	pulsante 2: inserita nello strumento la tabella della verità, permette di ricavarne l'espressione nella prima forma canonica.
$\overline{101}$ $\xrightarrow{\text{SIMP}}$ A B	pulsante 3: inserita nello strumento la tabella della verità, permette di ricavarne l'espressione logica minima ottenuta dalla prima forma canonica.
A B → $\overline{101}$	pulsante 4: data un'espressione logica, permette di ricavare la corrispondente tabella della verità.
A B → 	pulsante 5: data un'espressione logica permette di ricavare il circuito corrispondente in termini di somma di prodotti.
A B → NAND	pulsante 6: data un'espressione logica permette di ricavare il circuito corrispondente in termini di sole porte NAND.

(Place/Component/Mixed) quali switch, timer (ad esempio 555), ADC, DAC e IC per multivibratori digitali;

- alcuni microprocessori;
- PLD.

Un'analisi completa richiederebbe troppo spazio, quindi ci limiteremo solo ad alcuni esempi, nell'ambito della logica cablata, mentre per quanto riguarda quella programmata (microprocessori e PLD) ne parleremo nelle successive puntate. Prima, però, descriviamo il funzionamento di alcuni strumenti di Multisim specifici per l'ambito digitale.

GLI STRUMENTI PER L'ANALISI DEI CIRCUITI DIGITALI

Sebbene l'analisi dei circuiti digitali possa avvenire con molti degli strumenti già considerati e, in particolare, oscilloscopio, generatore di funzioni, multimetro e sonde, esistono in Multisim appositi strumenti, per i sistemi digitali, che meritano attenzione:

- il convertitore logico (*Logic Converter*);
- il generatore di parole (*Word Generator*);

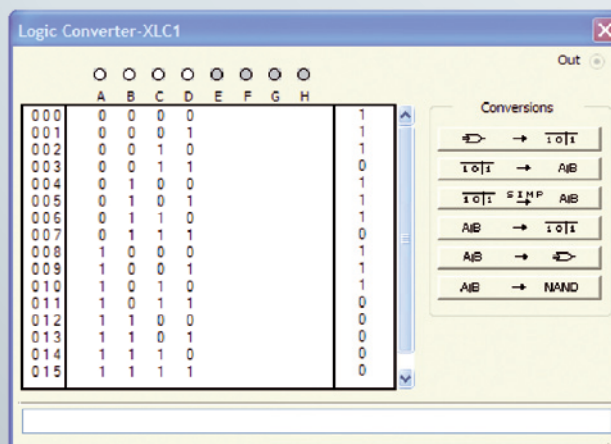
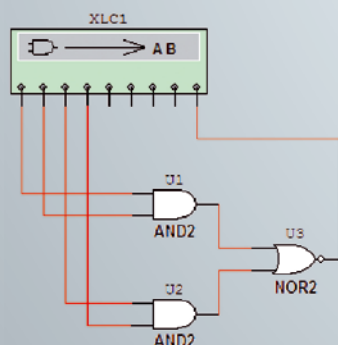
- l'analizzatore di stati logici (*Logic Analyzer*).

Il Logic Converter

Questo strumento aiuta nello studio dei circuiti combinatori realizzati a porte logiche elementari; per meglio comprenderlo, analizziamo le funzionalità dei suoi pulsanti descritte nella Fig. 7.

- 1) Pulsante 1: noto il circuito logico e collegati i suoi ingressi e la sua uscita allo strumento permette di ottenere la tabella di verità.
- 2) Pulsante 2: inserita nello strumento la tabella della verità, permette di ricavarne l'espressione nella prima forma canonica.
- 3) Pulsante 3: inserita nello strumento la tabella della verità, permette di ricavarne l'espressione logica minima ricavata dalla prima forma canonica.
- 4) Pulsante 4: data un'espressione logica, permette di ricavare la corrispondente tabella della verità.
- 5) Pulsante 5: data un'espressione logica, permette di ricavare il circuito corrispondente in termini di somma di prodotti.
- 6) Pulsante 6: data un'espressione logica per-

Fig. 8 - Il Logic Converter ricava la tabella della verità.



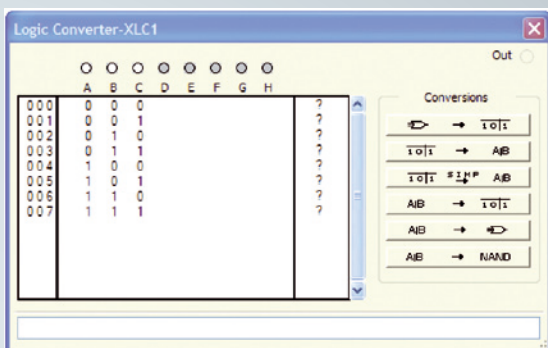
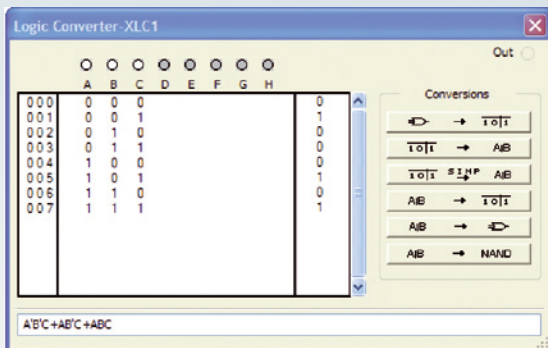


Fig. 10 - Definizione dei valori in uscita e della conseguente prima forma canonica.



mette di ricavare il circuito corrispondente in termini di sole porte NAND.

Pulsante 1: nel circuito d'esempio della Fig. 8 (file *Logic_converter_1*) premendo il pulsante si otterrà la relativa tabella della verità (A, B, C e D sono gli ingressi; l'uscita è la colonna a destra).

Per la realizzazione del circuito, trattandosi di una simulazione puramente logica, abbiamo usato i dispositivi **TIL**, raggiungibili con il tasto dei componenti **Misc Digital**.

Pulsante 2: con riferimento alla Fig. 9, se la tabella non è già inserita possiamo crearla cliccando con il tasto sinistro del mouse quando il cursore è posizionato sui pallini delle singole variabili di ingresso; noi abbiamo cliccato su A, B e C, ottenendo le combinazioni di ingresso. Nel caso della Fig. 10, per fissare i valori in uscita, posto il cursore del mouse

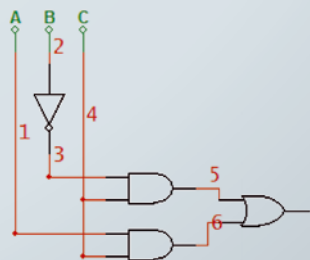


Fig. 11
Circuito minimizzato derivato dalla prima forma canonica.

Fig. 9 - La definizione delle variabili in ingresso.

sul singolo ? in uscita possiamo fissare con dei clic i valori 0, 1 e x (condizione di indifferenza). Ora, cliccando sul *pulsante 2* otterremo l'espressione della prima forma canonica (tenere presente che le complementazioni sono indicate con dei ').

Pulsanti 3, 4, 5 e 6: l'espressione logica corrispondente a questi pulsanti, se non già disponibile da precedenti passaggi, può essere direttamente scritta nello spazio previsto. Nella Fig. 11 vedete il circuito dell'espressione minima della tabella di Fig. 10, ottenuto con il Pulsante 5.

Uso del Word Generator e del Logic Analyzer

Il Word Generator genera parole binarie a 32 bit che possono modificarsi ciclicamente secondo diverse modalità. Il Logic Analyzer permette la visualizzazione dei singoli bit in termini di diagrammi temporali: in pratica è un oscilloscopio a 32 canali che però può funzionare solo con segnali binari.

Nella Fig. 12 proponiamo un esempio d'uso di entrambi gli strumenti (file *contatore_1*): si tratta di un collegamento diretto del generatore di parole (XWG1) all'analizzatore di stati logici (XLA1). In particolare, il collegamento avviene tra le uscite dei quattro bit meno significativi del Word Generator con quattro ingressi del Logic Analyzer.

Con il solito doppio clic possiamo visualizzare il pannello del Word Generator (Fig. 13): la visualizzazione delle parole binarie può avvenire scegliendo nel settore **Display** tra esadecimale (**EX**), decimale (**Dec**), binario (**Binary**) e **ASCII**. L'assegnazione delle singole parole può avvenire posizionando il cursore

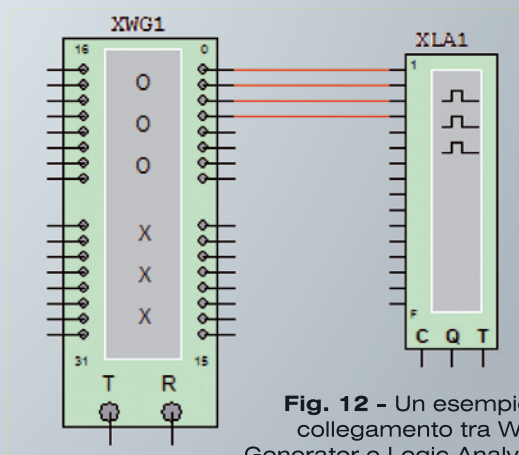


Fig. 12 - Un esempio di collegamento tra Word Generator e Logic Analyzer.

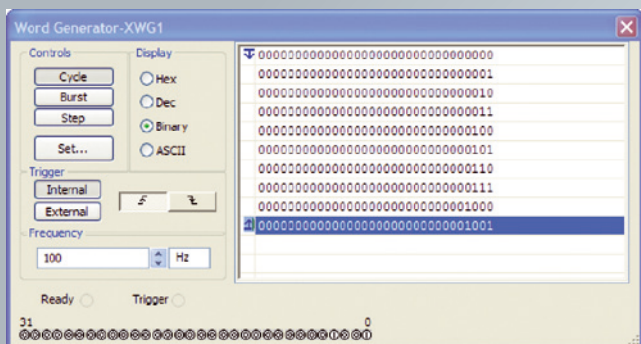


Fig. 13 - Il pannello del Word Generator.

del mouse su una di esse; facendo clic si può scrivere la parola con la tastiera del PC. Noi abbiamo scritto in sequenza le combinazioni binarie tra 0 e 9.

Per scegliere la modalità di successione delle parole in uscita al generatore di parole, si deve cliccare su **Set**; si ottiene la finestra **Setting** (Fig. 14).

Per impostazione predefinita, nel settore **Preset Patterns** sarà attiva l'opzione **No Change**; se i valori precedentemente impostati vanno bene, lasciamola e stabiliamo il numero (in binario o in esadecimale) di parole da considerare (che corrisponderà a quelle che avevamo precedentemente scritto) scrivendolo in **Buffer Size**. Con il tasto **Accept** potremo tornare al pannello del generatore e definire la modalità di funzionamento, che può essere:

- **Cycle** per ottenere in uscita le parole scritte una dopo l'altra ciclicamente;
- **Burst**: per avere la sequenza una sola volta;
- **Step**: per avere un'esecuzione passo-passo.

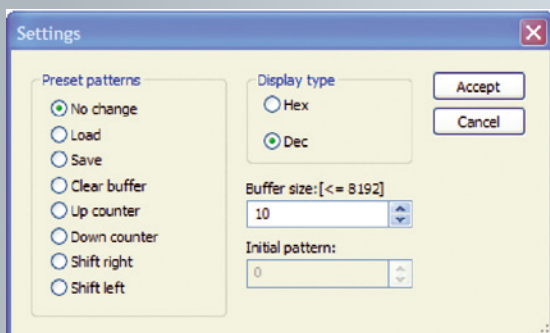


Fig. 14 - La scelta della modalità di successione delle parole.

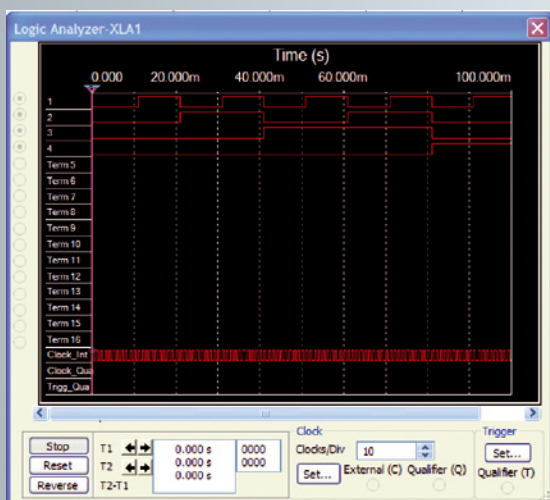
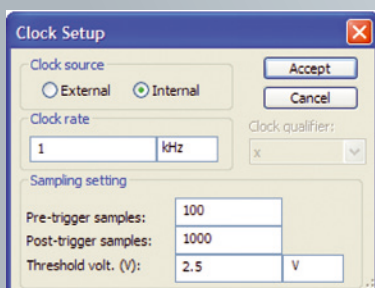


Fig. 15 - I diagrammi temporali visualizzati dal Logic Analyzer.

Fig. 16 - Il settaggio del clock del Logic Analyzer.



Nel settore **Frequency** dovremo poi fissare la frequenza di successione delle parole (è la frequenza della simulazione che potete valutare, ad esempio, con il Logic Analyzer, e non la frequenza che visivamente appare, la quale è invece sempre molto più bassa e dipende dalla velocità di elaborazione del nostro PC). Ritornando alla finestra **Setting** possiamo scegliere anche tra altre opzioni; qui di seguito ne vediamo alcune.

- **Up Counter**: stabilito il numero iniziale (**Initial Pattern**) si avrà una sequenza di conteggio in avanti definita dal numero che fisseremo in **Buffer Size** (nell'esempio in Fig. 14 il conteggio avverrà da 0 a 9);

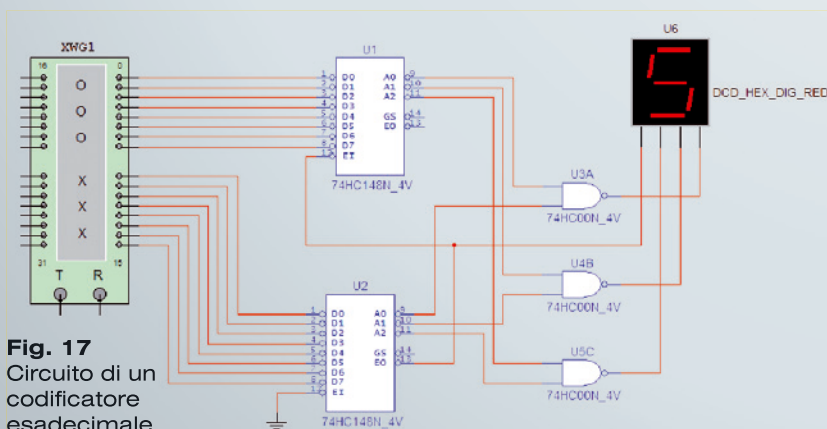


Fig. 17
Circuito di un codificatore esadecimale.

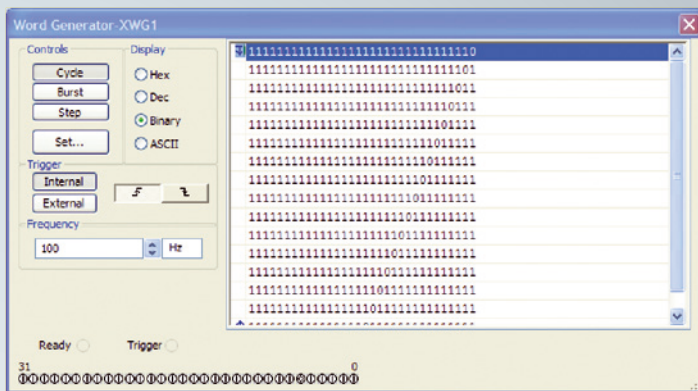
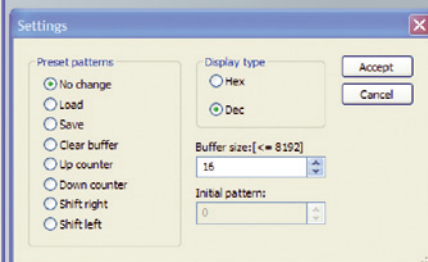


Fig. 18 - Settaggio del Word Generator.



- **Down Counter:** come sopra, ma con il conteggio all'indietro;
- **Shift Right:** fissato il valore iniziale con **Initial Pattern** e il numero di parole con **Buffer Size**, otterremo, passando da una parola alla successiva, uno shift binario verso destra;
- **Shift Left:** come il precedente, ma lo shift è verso sinistra.

L'analizzatore di stati logici visualizza le sequenze di bit in uscita al generatore di parole (Fig. 15): nel nostro esempio, il generatore conta ciclicamente da 0 e 9 e questo è ciò che viene visualizzato. In basso vedete il clock interno dello strumento, che va settato a una frequenza molto maggiore di quella dei segnali da visualizzare; per farlo, cliccate su

Set e fissate il valore del **Clock Rate** (Fig. 16), mentre sul pannello dell'analizzatore, alla voce **Clock**, potete fissare il numero di questi clock corrispondenti a una divisione.

ALCUNI ESEMPI DI SIMULAZIONI DIGITALI

Il primo esempio che proponiamo (Fig. 17, file *codificatore*) è tratto dal volume *L'Elettronica Digitale* di E. Ambrosini (vedi bibliografia nella prima puntata del corso). Si tratta di un **codificatore esadecimale** realizzato con due 74HC148 alimentati a 4 V (**Place/Component/CMOS**) e tre NAND, sempre di tipo HC a 4 V. In ogni possibile combinazione, i 16 bit in ingresso assumono tutti il valore 1 eccetto uno che vale 0; la posizione del bit a 0 corrisponde al numero esadecimale in uscita il cui valore è visualizzato tramite un display a LED, con

Dispositivo																Numero			
I ₁₅	I ₁₄	I ₁₃	I ₁₂	I ₁₁	I ₁₀	I ₉	I ₈	I ₇	I ₆	I ₅	I ₄	I ₃	I ₂	I ₁	I ₀	O ₃	O ₂	O ₁	O ₀
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	0
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	0	0
1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	1	0	1
1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	0
1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1
1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabella 1 - Tabella della verità del circuito di Fig. 17.

DATO BINARIO				VISUALIZZAZIONE ESADECIMALE
O ₃	O ₂	O ₁	O ₀	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

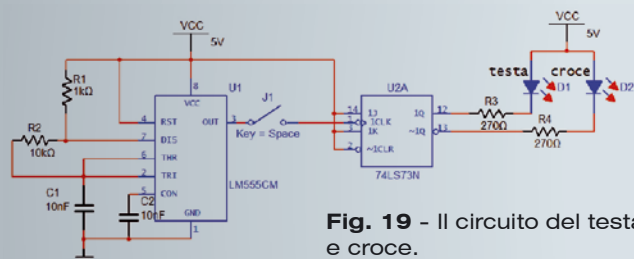


Fig. 19 - Il circuito del testa e croce.

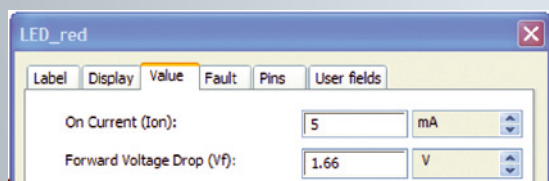


Fig. 20 - La definizione dei parametri del LED.

circuito di decodifica incorporato (**Place/Component/Indicators/HEX_DISPLAY**). Il tutto è sintetizzato nella **Tabella 1**, dove le uscite sono espresse dalle 16 possibili combinazioni binarie.

La visualizzazione esadecimale avviene secondo quanto espresso dalla **Tabella 2**. La **Fig. 18** riporta la definizione dei dati in ingresso con il Word Generator, coerentemente con la **Tabella 1**.

A questo punto, avviando la simulazione si vedrà sul display la sequenza di simboli da 0 a F (alcune lettere in minuscolo per limiti di visualizzazione dei display a 7 segmenti). Ai

Tabella 2 - Corrispondenza tra il dato binario a 4 bit e la corrispondente visualizzazione esadecimale.

fini pratici, se il Word Generator è sostituito da una tastiera con 16 pulsanti, che non premuti inviano al singolo ingresso un 1 e se premuti uno 0, abbiamo realizzato una tastiera che fornisce in uscita un codice binario a 4 bit. Un'osservazione per quanto riguarda il display: quello scelto è solo uno dei tanti, infatti possiamo scegliere tra diversi colori, tra display con o senza decodifica, ad anodo o a catodo comune, ecc.

Il secondo esempio che proponiamo è meno impegnativo: si tratta del **gioco del testa e croce** (tratto sempre dallo stesso libro). Il circuito (file *testa_e_croce*) lo trovate in **Fig. 19**: il timer 555 (**Place/Component/Mixed/TIMER**) genera un'onda (quasi) quadra alla frequenza di 6,8 kHz; questo segnale, quando il tasto J1 è chiuso, fa da clock per il flip-flop 74LC73 (**Place/Component/TTL**) che a ogni fronte di discesa commuta. In conclusione, i LED in uscita (**Place/Component/Diodes**) per come sono collegati (uno all'uscita Q e l'altro all'uscita $\bar{Q}$) si accenderanno alternativamente alla frequenza dell'onda quadra e quindi al nostro occhio appariranno entrambi sempre accesi. Aprendo il tasto J1, uno a caso dei due LED si spegnerà. Attenzione che la simulazione è visivamente più lenta e quindi in effetti in questo caso i due LED si vedranno accendersi e spegnersi ma nella realtà andrà tutto bene.

Quello appena fatto è un tipico esempio di simulazione analogica (il timer 555) e digitale (il flip-flop) che coesistono senza problemi. Nella finestra di **Fig. 20** avete la possibilità, con doppio clic, di agire su tensione e corrente di conduzione diretta del LED. Infine, in **Fig. 21** proponiamo (sempre dal solito libro) il circuito di un contatore/contapezzi UP-DW (file *contatore_2*). Tramite J1 è possibile azzerare il contatore, mentre con J2 possiamo assegnare il conteggio in avanti (UP) o all'indietro (DW). Il tasto J4 permette di scegliere tra un clock interno e uno esterno (fissato da un generatore da 400 Hz ai fini della simulazione, ma che in realtà potrebbe essere una sequenza di impulsi corrispondenti ai singoli pezzi da contare). Il tasto J3 permette di attivare/disattivare il clock interno, ottenuto da porte trigger e quindi in modalità analogica. Le reti di diodi formano degli OR: il transistor

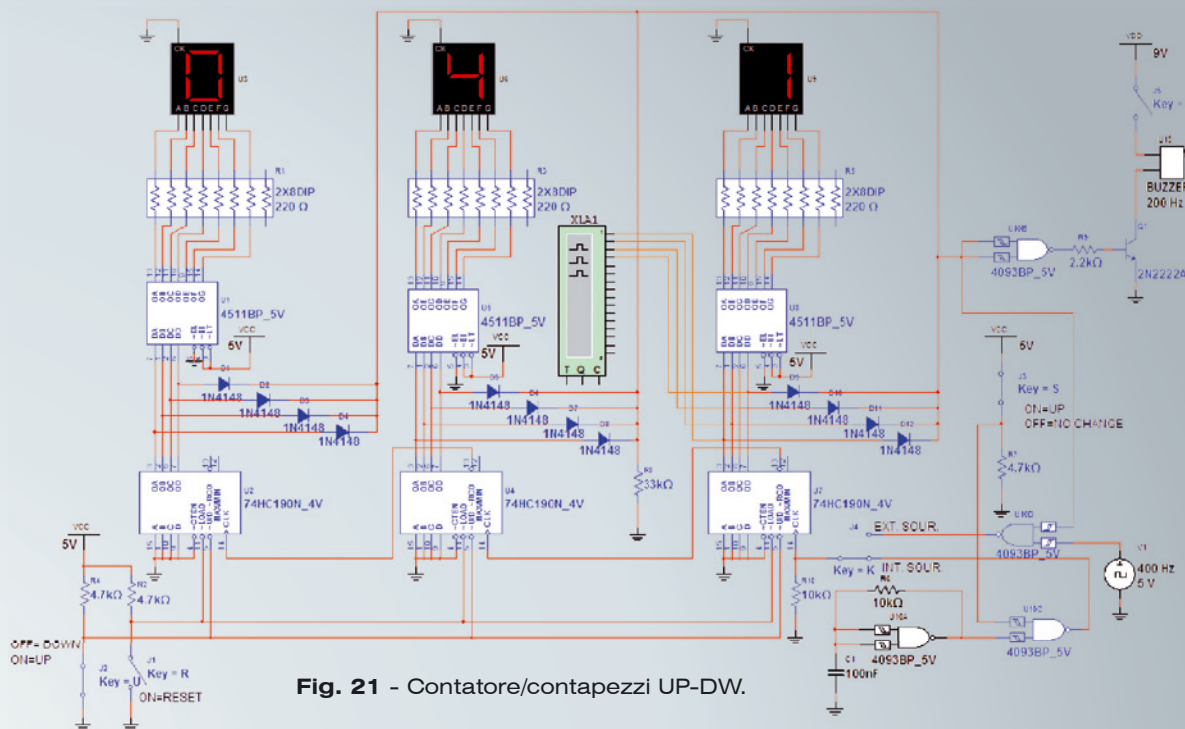


Fig. 21 - Contatore/contapezzi UP-DW.

Q1 ha in ingresso un livello alto e pilota l'accensione del buzzer (cicalino sonoro, disattivabile con J5) solo quando tutti i display sono a zero.

Vediamo la corretta sequenza d'uso:

1. con J3 aperto si chiude J2 per predisporre il conteggio in avanti;
2. sempre con J3 aperto, si predispone J4 per il clock interno;
3. tramite J1 si azzerava il contatore (chiuderlo e poi riaprirlo);
4. si chiude J3 e si inizia così il conteggio in

avanti tramite il clock interno;

5. raggiunto il conteggio desiderato si riapre J3, interrompendo così il conteggio e fissando, di conseguenza, il numero di pezzi da contare;
6. con J2 si imposta il conteggio all'indietro;
7. si porta J4 in modo che inizi il conteggio a decremento tramite il clock esterno;
8. raggiunto l'azzeramento del conteggio, il cicalino indicherà il raggiungimento dei pezzi stabilito.

Il valore di C1 è adatto per la simulazione; nell'uso reale sarà opportuno un valore di 22 μ F ($4 \div 5$ Hz). Per trovare il cicalino in Multisim, seguite il percorso: **Place/Component/Indicator/BUZZER**; per le reti resistive, andate su **Place/Component/Basic/RPAK**.

In Fig. 22 trovate i diagrammi temporali del contatore modulo 9, relativi alla cifra delle unità, tracciati con il Logic Analyzer.

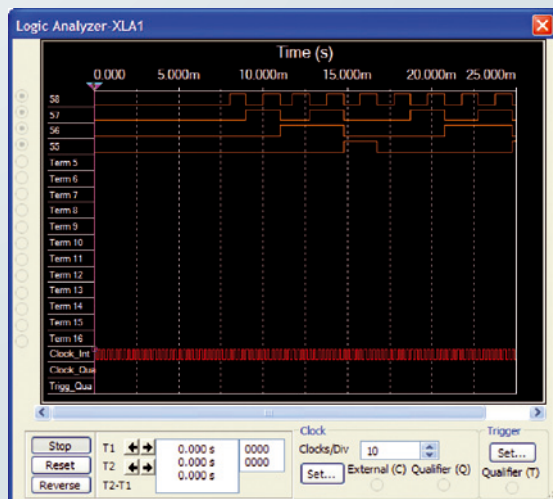


Fig. 22 - Diagramma temporale del contatore delle unità.

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.

AGGIUNGI UN LINK WI-FI AL TUO ARDUINO

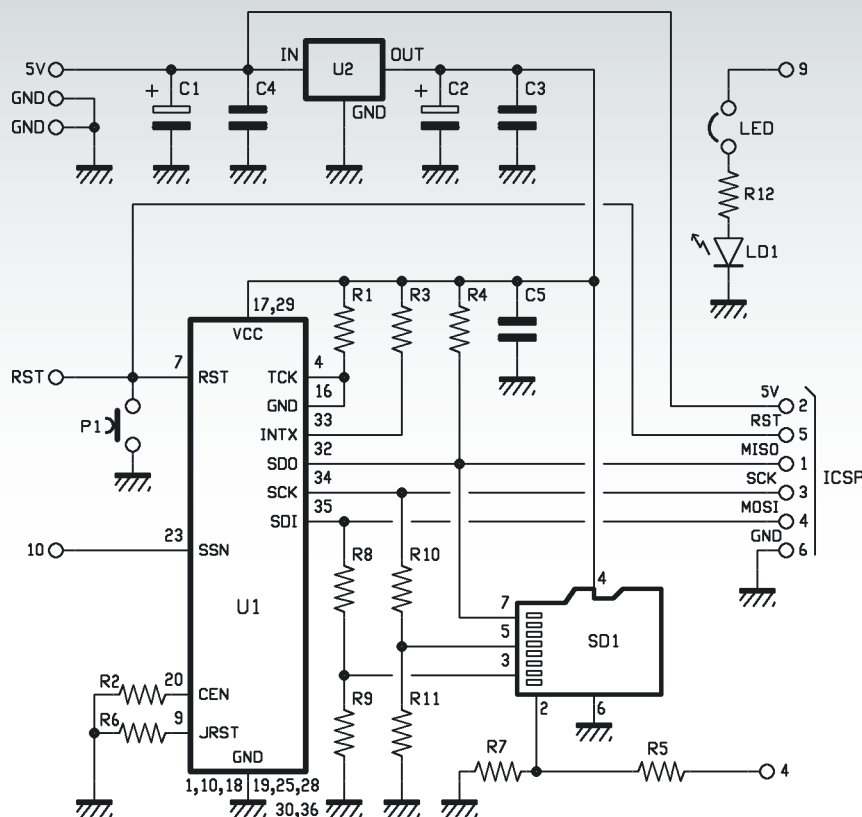
Realizziamo uno shield basato sul modulo MRF24WB0MA della Microchip, facilmente utilizzabile con librerie di schede commerciali.

..... degli Ingg. TOMMASO E ALESSANDRO GIUSTO

Come avrete notato, è da parecchio tempo che ci stiamo interessando all'hardware Arduino e che presentiamo diversi prodotti basati su questa architettura. Oltre all'ambiente di sviluppo open-source, ciò che rende interessante il progetto Arduino è la possibilità di espandere le funzionalità della scheda base attraverso l'aggiunta di diversi livelli hardware, denominati "shield", capaci di svolgere funzioni limitate praticamente solo dalla fantasia del progettista. È possibile affermare che tra tutti quelli sviluppati e disponibili, gli shield più utilizzati nella pratica sono quelli che espandono le possibilità di comunicazione della scheda Arduino e in particolare quelli che permettono di aggiungere

il collegamento a una rete di tipo TCP/IP. I primi shield di questo tipo a uscire sul mercato sono stati quelli basati sulla tecnologia ethernet, che consentivano ai sistemi Arduino di collegarsi a una LAN ethernet basata sul protocollo TCP/IP e quindi ad Internet. Col tempo, però, soprattutto dopo l'avvento della tecnologia Wi-Fi, questi shield hanno iniziato a mostrare i loro limiti ed in un certo senso sono diventati scomodi da usare, in quanto richiedono la stesura di cavi di collegamento e il successivo riposizionamento nel caso si decida di spostare i dispositivi dalla posizione originaria. Per ovviare a queste ed altre limitazioni, fortunatamente da poco tempo si sono resi disponibili gli shield





prodotti da tale casa. In aggiunta, abbiamo previsto uno slot per inserire memorie di tipo microSD. La compatibilità pin-to-pin con l'Ethernet Shield della AsyncLabs permette di sostituire, in sistemi basati su Arduino, tale shield con il nostro; ciò significa che se avete realizzato un progetto che fa uso dello shield ethernet AsyncLabs (ad esempio il controllo RGB descritto nel fascicolo n° 159) e volete passare dal controllo da LAN cablata a quello Wi-Fi, vi basta togliere lo shield ethernet e montare il nostro. Per gestirlo, potete utilizzare tranquillamente librerie esistenti, che nello specifico sono le stesse fornite dalla AsyncLabs per il suo Wi-Fi Shield.

ANALISI HARDWARE

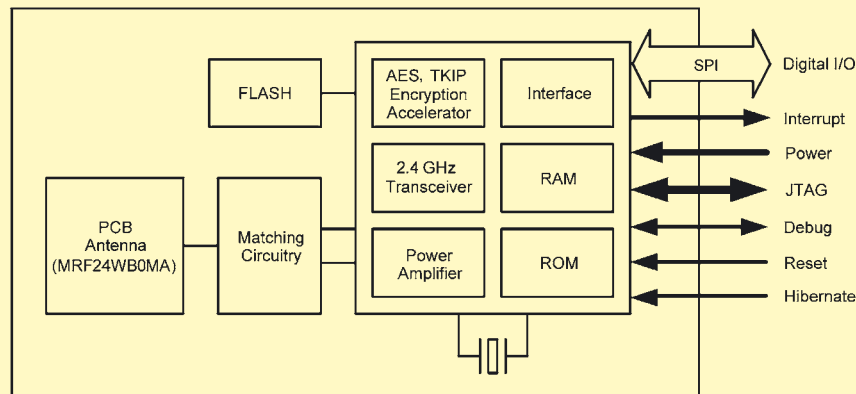
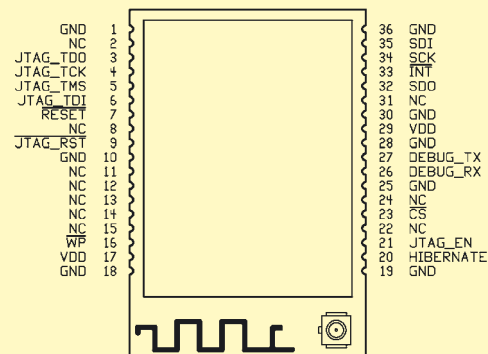
Il componente base dello shield Wi-Fi che abbiamo realizzato e

Wi-Fi, che permettono sempre di accedere a una rete TCP/IP e ad Internet, ma non più attraverso livelli hardware di tipo ethernet, bensì tramite connessioni wireless. La comodità della tecnologia Wi-Fi è ormai nota a tutti: niente più cavi da stendere (che aumentano il costo e il tempo di realizzazione di qualsiasi impianto) e piena libertà nel posizionamento dei diversi nodi della rete wireless.

Una delle prime aziende a concentrarsi sulla tecnologia Wi-Fi è stata la AsyncLabs, che ha proposto al mercato i primi shield del genere, corredandoli con le librerie del caso.

In questo articolo vi proponiamo una nuova soluzione Wi-Fi: si tratta di uno shield che a livello hardware prende spunto da quello della AsyncLabs, pertanto è stata mantenuta la piena compatibilità hardware sia con l'ether-net shield, sia con gli shield Wi-Fi

A destra, la piedinatura del modulo Wi-Fi MRF24WB0MA visto dall'alto, ossia da dove si vede il connettore per l'antenna esterna. Sotto, lo schema a blocchi interno del dispositivo, dal quale appare la composizione: vediamo il transceiver ed il generatore di codifica che attinge ad una propria Flash, oltre all'interfaccia SPI per la comunicazione dati ed all'amplificatore RF.



[piano di MONTAGGIO]

che qui descriviamo è un modulo Wi-Fi siglato MRF24WB0MA e prodotto dalla Microchip, una casa che certo non ha bisogno di presentazioni.

Il dispositivo è un Wi-Fi RF transceiver certificato IEEE 802.11, ha un data-rate compreso tra 1 e 2 Mbps, dispone di antenna interna con elettronica di adattamento integrata (è comunque predisposto per l'utilizzo di un'antenna esterna dedicata), ha un indirizzo MAC univoco, lavora nella banda di frequenze 2.400÷2.484 GHz ed è l'ideale per applicazioni di domotica, controlli remoti (sia consumer che industriali) e per impieghi wireless in generale. Nella pagina accanto, trovate lo schema elettrico della nostra scheda shield. Le dimensioni e la forma sono compatibili con le schede Arduino Diecimila e Duemilanove.

Nel nostro progetto viene utilizzata la connessione SPI (velocità massima di 25 MHz) per la comunicazione con il controllore host principale; sono supportate sia le reti wireless del tipo infrastrutturato (BSS) che ad hoc (IBSS) ed è permesso il collegamento anche a reti sicure (sono supportate le crittografie WEP a 64 e a 128 bit, nonché WPA/WPA2 TKIP, AES e PSK).

Analizzando lo schema elettrico, facciamo notare il modulo Microchip MRF24WB0MA (indicato con U1) collegato al bus SPI tramite le linee SDO, SCK e SDI (pin 32, 34 e 35) e che, in caso di necessità, può essere resettato tramite un pulsante (P1 nello schema).

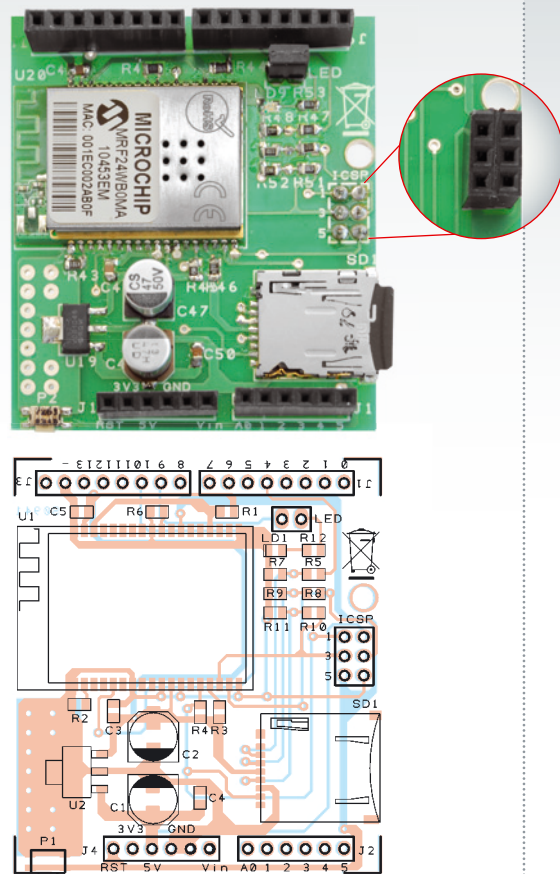
Peculiarità del nostro shield è che, a differenza di quello prodotto dalla AsyncLabs, nel quale è prevista una memoria fissa, dispone di uno slot per montare le sempre più utilizzate e diffuse memorie SD (siglato SD1 nello

Elenco Componenti:

- R1 ÷ R4: 4,7 kohm (0805)
- R5: 1 kohm (0805)
- R6: 4,7 kohm (0805)
- R7: 2,2 kohm (0805)
- R8: 1 kohm (0805)
- R9: 2,2 kohm (0805)
- R10: 1 kohm (0805)
- R11: 2,2 kohm (0805)
- R12: 330 ohm (0805)
- C1: 220 µF 6,3 VL elettrolitico (D)
- C2: 220 µF 6,3 VL elettrolitico (D)
- C3: 100 nF multistrato (0805)
- C4: 100 nF multistrato (0805)
- C5: 100 nF multistrato (0805)
- LD1: LED verde (0805)
- U1: MRF24WB0MA/RM
- U2: TC1262-3.3 (SOT-223)
- SD1: Connettore µSD-Card (MICROSDSOCK2)
- P1: Pulsante miniatura SMD

Varie:

- Strip maschio 2 poli
- Strip M/F 6 poli (2 pz.)
- Strip M/F 8 poli (2 pz.)
- Strip M/F 3 poli (2 pz.)
- Jumper
- Circuito stampato



schema elettrico), gestito da Arduino sempre attraverso la porta SPI; ciò permette di estrarre i dati e trasportarli in altri sistemi. È anche presente un LED (siglato LD1) utilizzato per indicare se la connessione Wi-Fi è attiva o meno; la relativa gestione è realizzata tramite una porta hardware, che pertanto normalmente non potrà essere utilizzata da software. Nel caso in cui fosse necessaria proprio questa porta, è comunque presente un ponticello (indicato con LED) che, una volta aperto, scollega il LED rendendo nuovamente disponibile il pin di I/O.

Infine, a completare l'hardware dello shield c'è una sezione di alimentazione dedicata alla logica, composta da un regolatore di tensione a 3,3 V (indicato con U2); quanto ai 12 volt, vengono

prelevati direttamente dalla scheda Arduino.

ANALISI SOFTWARE

Come già anticipato, il nostro shield è compatibile con le librerie software WiShield fornite dalla AsyncLabs (al momento della stesura di questo articolo l'ultima versione disponibile è la 1.3.0); a nostro parere questa compatibilità è un grosso vantaggio, in quanto permette di passare da una versione di shield hardware all'altra in modo indolore, semplicemente rimuovendo il vecchio e inserendo il nuovo, senza dover eseguire alcuna modifica al software già scritto. La libreria WiShield supporta varie modalità di funzionamento, che sono APP_WEB-SERVER, APP_WEBCLIENT, APP_SOCKAPP, APP_UDPAPP

Listato 1

```
/******  
SoftwareDemoWebServerWi-Fi  
Esempio codice Web Server tramite Wi-Fi  
Autori: Ingg. Tommaso Giusto e Ing. Alessandro Giusto  
Email: tommro@libero.it  
*****/  
  
// Inclusione Libreria per Server Web Wi-Fi  
#include <WiServer.h>  
  
// Definizione pin INPUT/OUTPUT  
int Pin_Led_Rosso = 7; // Led rosso uscita digitale 7  
  
// Definizione Variabili Globali Stato Led  
byte Stato_Led_Rosso = 0; // Stato led rosso  
  
// Definizione Parametri Rete Wireless  
#define WIRELESS_MODE_INFRA 1 // Infrastrutturata (basata su Access Point)  
#define WIRELESS_MODE_ADHOC 2 // Ad-hoc (senza Access Point)  
  
unsigned char local_ip[] = {192, 168, 1, 250}; // Indirizzo IP  
unsigned char gateway_ip[] = {192, 168, 1, 91}; // Indirizzo gateway IP  
unsigned char subnet_mask[] = {255, 255, 255, 0}; // Subnet Mask  
const prog_char ssid[] PROGMEM = {"Sitecom"}; // SSID access point  
  
// Selezione tipo di cifratura rete Wireless  
unsigned char security_type = 0; // 0 -> nessuna cifratura  
// 1 -> cifratura WEP  
// 2 -> cifratura WPA  
// 3 -> cifratura WPA2  
  
// Password cifratura per WPA/WPA2 (max. 64 caratteri)  
const prog_char security_passphrase[] PROGMEM = {"12345678"};  
  
// Password cifratura per WEP 128-bit keys  
prog_uchar wep_keys[] PROGMEM = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00};  
  
// Selezione tipo di rete Wireless infrastrutturata  
unsigned char wireless_mode = WIRELESS_MODE_INFRA;  
  
// Variabili per lunghezza SSID e password di cifratura  
unsigned char ssid_len;  
unsigned char security_passphrase_len;  
  
// Inizializzazione Scheda  
void setup() {  
    // Inizializzo pin usati come INPUT/OUTPUT  
    pinMode (Pin_Led_Rosso, OUTPUT);  
  
    // Inizializzo WiServer (Gestione_Richieste_Web per creare/trasmettere pagine HTML)  
    WiServer.init (Gestione_Richieste_Web);  
    // Inizializzo porta seriale  
    Serial.begin (9600);  
    WiServer.enableVerboseMode (false);  
  
    // Spengo led rosso  
    Led_Rosso_OFF();  
}  
  
// Programma Principale  
void loop() {  
    // Avvio WiServer  
    WiServer.server_task();  
}  
  
// Gestione diverse richieste provenienti dal WEB  
// INPUT: URL pagina web richiesta  
// OUTPUT: Flag URL riconosciuto/non riconosciuto  
boolean Gestione_Richieste_Web (char* URL) {  
    // Se URL richiesto corrisponde a "/" (pagina index)  
    if (strcmp (URL, "/") == 0) {  
        // Secondo gli I/O creo e invio le pagine Web  
        Invia_Pagina_Web();  
  
        // Ritorno URL è stato riconosciuto  
        return (true);  
    } // Chiusura if URL richiesto corrisponde a "/" (pagina index)
```



```

// Se URL richiesto corrisponde a "?OPERATION=ACCENDI_ROSSO"
if (strcmp (URL, "?OPERATION=ACCENDI_ROSSO") == 0) {
    // Accendo led rosso
    Led_Rosso_ON();
    Stato_Led_Rosso = 1;

    // Secondo gli I/O creo e invio le pagine Web
    Invia_Pagina_Web();

    // Ritorno URL è stato riconosciuto
    return (true);
} // Chiusura if URL richiesto corrisponde a "?OPERATION=ACCENDI_ROSSO"

// Se URL richiesto corrisponde a "?OPERATION=SPEGNI_ROSSO"
if (strcmp (URL, "?OPERATION=SPEGNI_ROSSO") == 0) {
    // Spengo led rosso
    Led_Rosso_OFF();
    Stato_Led_Rosso = 0;

    // Secondo gli I/O creo e invio le pagine Web
    Invia_Pagina_Web();

    // Ritorno URL è stato riconosciuto
    return (true);
} // Chiusura if URL richiesto corrisponde a "?OPERATION=SPEGNI_ROSSO"

// Ritorno URL non riconosciuto
return (false);
}

// Funzione che, secondo gli I/O, crea e invia le pagine Web
void Invia_Pagina_Web() {
    // Usando le funzioni WiServer.print trasmette al pagina Web da visualizzare
    WiServer.print ("<html>");
    WiServer.print ("<head>");
    WiServer.print ("<meta http-equiv='refresh' content='10;url=http://'>");
    WiServer.print (local_ip[0], DEC);
    WiServer.print (".");
    WiServer.print (local_ip[1], DEC);
    WiServer.print (".");
    WiServer.print (local_ip[2], DEC);
    WiServer.print (".");
    WiServer.print (local_ip[3], DEC);
    WiServer.print ("/" "/>");
    WiServer.print ("</head>");
    WiServer.print ("<p align='center'>");
    WiServer.print ("Hello World!<br>");
    WiServer.print ("Esempio Web Server tramite librerie WiShield<br>");
    WiServer.print ("By Ingg. Tommaso Giusto e Ing. Alessandro Giusto<br>");
    WiServer.print ("tommro@libero.it<br>");

    // Se led rosso spento
    if (Stato_Led_Rosso == 0) {
        WiServer.print ("<form>Led rosso spento<br>");
        WiServer.print ("<method=GET>");
        WiServer.print ("<input type=submit name=OPERATION value=ACCENDI_ROSSO></form><br>");
    } // Chiusura if led rosso spento
    // Se led rosso acceso
    else {
        WiServer.print ("<form>Led rosso acceso<br>");
        WiServer.print ("<method=GET>");
        WiServer.print ("<input type=submit name=OPERATION value=SPEGNI_ROSSO></form><br>");
    } // Chiusura if led rosso acceso

    // Termino pagina HTML
    WiServer.print ("</html>");
}

// Accende il led rosso
void Led_Rosso_ON() {
    digitalWrite (Pin_Led_Rosso, LOW);
}

// Spegne il led rosso
void Led_Rosso_OFF() {
    digitalWrite (Pin_Led_Rosso, HIGH);
}

```

e APP_WISERVER, corrispondenti rispettivamente a Web Server, Web Client, Socket, UDP e WiServer. L'impostazione predefinita è APP_WEBSERVER, che funziona sulla maggior parte dei sistemi Arduino ma presenta diverse limitazioni, la principale delle quali è l'incapacità di funzionare simultaneamente sia come client che come server. Una modalità alternativa sicuramente più interessante, maggiormente diffusa, utilizzata e consigliata, è la APP_WISERVER (libreria denominata "WiServer.h") che permette ad un sistema Arduino equipaggiato con shield Wi-Fi di essere configurato e funzionare sia come Web Server sia come Web Client. Nel primo caso sarà possibile servire richieste di connessione provenienti da web client esterni ed inviare delle pagine HTML in risposta; invece nel secondo ci si potrà collegare a un web server e, tramite la tecnologia request GET o POST, inviare richieste di esecuzione comandi e ricevere pagine HTML di risposta. Per configurare la libreria WiShield scaricata come APP_WISERVER è necessario aprire il file *<directory di arduino>\libraries\WiShield\apps-conf.h*, andare circa alla riga numero 39, commentare il "#define" di APP_WEBSERVER e rimuovere il commento al "#define" di APP_WISERVER.

La sezione di codice in esame deve risultare qualcosa come:

```

// #define APP_WEBSERVER
// #define APP_WEBCLIENT
// #define APP_SOCKETAPP
// #define APP_UDPPAPP
#define APP_WISERVER

```

FUNZIONAMENTO DA WEB SERVER

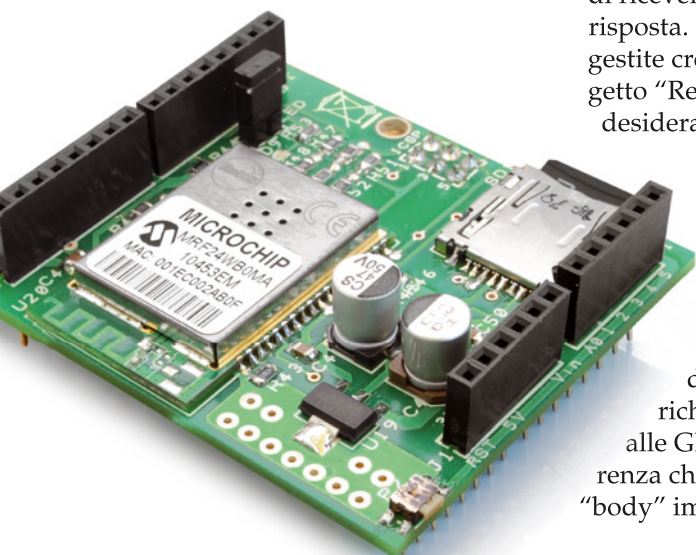
La libreria WiServer permette facilmente di configurare un

Listato 2

sistema Arduino+shield Wi-Fi come Web Server, in modo che sia in grado di generare dinamicamente (cioè a runtime) ed inviare pagine web il cui contenuto può essere modificato a seconda dello stato interno della scheda. Affinché il sistema si comporti come un server web, il codice sorgente deve implementare una funzione di "creazione pagina" (page serving function) richiamabile dalla libreria quando venga ricevuta una richiesta HTTP. All'interno di questa funzione, il software deve verificare se l'URL della richiesta ricevuta è corretto e riconosciuto come comando, quindi utilizzare le funzioni "print" e "println" per generare "runtime" le stringhe di testo che costituiscono la pagina HTML di risposta. Infine, la funzione dovrebbe tornare il valore boolean "true" se l'URL è stato riconosciuto come valido; viceversa deve ritornare "false". In conclusione di questo discorso, facciamo notare che la libreria WiServer invierà in automatico l'appropriata intestazione HTTP.

FUNZIONAMENTO COME WEB CLIENT

La libreria WiServer supporta anche comunicazioni in cui il si-



```
/******  
SoftwareDemo2WebClientWi-Fi  
Esempio codice Web Client tramite Wi-Fi  
Autori: Ingg. Tommaso Giusto e Ing. Alessandro Giusto  
Email: tommro@libero.it  
*****/  
  
// Inclusione Libreria per Server Web Wi-Fi  
#include <WiServer.h>  
  
// Definizione Parametri Rete Wireless  
#define WIRELESS_MODE_INFRA 1 // Infrastrutturata (basata su Access Point)  
#define WIRELESS_MODE_ADHOC 2 // Ad-hoc (senza Access Point)  
  
unsigned char local_ip[] = {192, 168, 1, 250}; // Indirizzo IP  
unsigned char gateway_ip[] = {192, 168, 1, 91}; // Indirizzo gateway IP  
unsigned char subnet_mask[] = {255, 255, 255, 0}; // Subnet Mask  
const prog_char ssid[] PROGMEM = {"Sitecom"}; // SSID access point  
  
// Selezione tipo di cifratura rete Wireless  
unsigned char security_type = 0; // 0 -> nessuna cifratura  
// 1 -> cifratura WEP  
// 2 -> cifratura WPA  
// 3 -> cifratura WPA2  
  
// Password cifratura per WPA/WPA2 (max. 64 caratteri)  
const prog_char security_passphrase[] PROGMEM = {"12345678"};  
  
// Password cifratura per WEP 128-bit keys  
prog_uchar wep_keys[] PROGMEM = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,  
0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19,  
0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29,  
0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39,  
0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f, 0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49,  
0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, 0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59,  
0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69,  
0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79,  
0x7a, 0x7b, 0x7c, 0x7d, 0x7e, 0x7f, 0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89,  
0x8a, 0x8b, 0x8c, 0x8d, 0x8e, 0x8f, 0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97, 0x98, 0x99,  
0x9a, 0x9b, 0x9c, 0x9d, 0x9e, 0x9f, 0xa0, 0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9,  
0xaa, 0xab, 0xac, 0xad, 0xae, 0xaf, 0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9,  
0xba, 0xbb, 0xbc, 0xbd, 0xbe, 0xbf, 0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9,  
0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf, 0xd0, 0xd1, 0xd2, 0xd3, 0xd4, 0xd5, 0xd6, 0xd7, 0xd8, 0xd9,  
0xda, 0xdb, 0xdc, 0xdd, 0xde, 0xdf, 0xe0, 0xe1, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6, 0xe7, 0xe8, 0xe9,  
0xea, 0xeb, 0xec, 0xed, 0xee, 0xef, 0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf9,  
0xfa, 0xfb, 0xfc, 0xfd, 0xfe, 0xff};  
  
// Selezione tipo di rete Wireless infrastrutturata  
unsigned char wireless_mode = WIRELESS_MODE_INFRA;  
  
// Variabili per lunghezza SSID e password di cifratura  
unsigned char ssid_len;  
unsigned char security_passphrase_len;  
  
// Definizione Parametri Google Search  
// Indirizzo IP per server www.google.it  
uint8 google_ip[] = {209, 85, 148, 106};  
  
// Richiesta GET verso GOOGLE
```

stema Arduino si comporta come Web Client; è quindi in grado di inviare sia richieste HTTP GET che POST ad un Web Server e di ricevere le pagine HTML di risposta. Le richieste Client sono gestite creando un opportuno oggetto "Request" con i parametri desiderati e quindi inviandolo sulla rete. Le richieste GET vengono gestite creando un oggetto "GETrequest" e indicando l'indirizzo IP del server, il nome dell'host e l'URL della richiesta. Invece le richieste POST sono simili alle GET, con l'unica differenza che in più includono un "body" immediatamente dopo

l'intestazione e che in questo caso bisognerà creare un oggetto "POSTrequest".

ESEMPIO SOFTWARE 1: IL WEB SERVER

Ora facciamo un primo esempio applicativo (il codice è nel Listato 1): analizziamo come configurare e programmare un sistema Arduino+shield Wi-Fi in modo che funzioni da Web Server capace di accettare comandi provenienti dall'esterno; scopo del nostro esempio è permettere, tramite pulsanti presenti nella pagina HTML, di accendere/spegnere un LED collegato a un'uscita del sistema e segnalare attraverso la stessa pagina il relativo stato di accensione.

```

GETrequest getGoogleSearch (google_ip, 80, "www.google.it", "/search?q= ElettronicaIn");

// Inizializzazione Scheda
void setup() {
  // Inizializzo WiServer (NULL indica non dobbiamo servire pagine Web)
  WiServer.init (NULL);
  // Inizializzo porta seriale
  Serial.begin (9600);
  WiServer.enableVerboseMode (false);

  // Inizializzazione richiesta GET (parametro indica funzione a cui verrebbe passata la risposta)
  getGoogleSearch.setReturnFunc (Gestione_Risposte_Web);
}

// Variabile memorizzazione riavvio (millisecondi) di esecuzione ricerca
long updateSearch = 0;

// Programma Principale
void loop() {
  // Se passato periodo di attesa esecuzione ricerca
  if (millis() >= updateSearch) {
    // Eseguo ricerca
    getGoogleSearch.submit();
    // Aggiorno tempo di riavvio (attesa di 1 ora)
    updateSearch = updateSearch + (1000 * 60 * 60);
  }

  // Avvio WiServer
  WiServer.server_task();

  // Attesa
  delay(10);
}

// Gestione diverse risposte provenienti dal WEB
void Gestione_Risposte_Web (char* data, int len) {
  // Stampo la risposta proveniente dal WEB su porta seriale
  int i;

  // Stampo i singoli caratteri della risposta proveniente dal WEB su porta seriale
  for (i = 0; i < len; i++) {
    Serial.print(*(data));
    data++;
  }
}

```

Analizzando il sorgente, vediamo che, come prima operazione, viene definito il contatto cui è collegato fisicamente il diodo luminoso; in seguito vengono definiti alcuni parametri di configurazione della rete wireless (indirizzo IP, gateway, subnet mask, selezione del tipo di crittografia della rete, eccetera).

A questo punto, in **setup()**, tramite la funzione **WiServer.init()** viene inizializzato il Web Server (il parametro di questa funzione, **Gestione_Richieste_Web** nel nostro esempio, indica la funzione che verrà richiamata quando in ingresso arriverà una richiesta proveniente da un client). In seguito vengono inizializzati la porta seriale per il debug e lo

stato (spento) del LED.

In **loop()** viene ciclicamente richiamata la funzione **WiServer.server_task()** che rappresenta la task di gestione delle richieste.

A questo punto possiamo ad analizzare la funzione di gestione delle richieste (**Gestione_Richieste_Web**): come visto in precedenza, questa riceve come parametro l'URL della pagina web richiesta e, analizzandolo, determina quale comando eseguire. Dal codice si vede come, oltre alla

pagina principale (identificata dalla stringa "/"), siano presenti due comandi (accensione/spegnimento del LED rosso). Infine è presente la funzione di generazione ed invio delle pagine HTML di risposta, per la quale si utilizza la funzione **WiServer.print()** che invia le singole stringhe di testo costituenti le pagine HTML complete.

A conclusione del codice sorgente sono presenti le ultime due funzioni che gestiscono l'accensione e spegnimento hardware del LED.

Nella **Fig. 1** viene mostrata la pagina HTML che si ottiene collegandosi alla scheda tramite un browser (nella **Fig. 1** usiamo Internet Explorer). Per farlo è sufficiente digitare l'indirizzo IP della scheda (192.168.1.250) nella barra dell'indirizzo di Explorer (chiaramente il PC dovrà essere collegato alla stessa rete wireless). Premendo il pulsante viene invertito lo stato di accensione del LED e, in risposta, si riceve la nuova pagina HTML aggiornata.

ESEMPIO SOFTWARE 2: IL WEB CLIENT

In questo secondo esempio applicativo (codice in **Listato 2**) vediamo come possiamo configurare e programmare il sistema



Fig. 1

Installare le librerie WiShield

Per funzionare correttamente, gli shield di tipo WiShield necessitano delle librerie software WiShield sviluppate dalla AsyncLabs, le quali chiaramente devono essere installate nell'IDE Arduino utilizzato per compilare i sorgenti. Per fare ciò, come prima operazione è necessario scaricare le librerie (in questo articolo ci riferiamo alla versione 1.3.0), estrarle nel PC locale, se necessario rinominare la directory ottenuta dalla scompattazione del file scaricato dandole il nome "WiShiled" e infine copiare le librerie all'interno della cartella "<directory di arduino>\libraries" in modo che l'IDE Arduino sia in grado di vederle ed utilizzarle.

La libreria WiShield supporta varie modalità di funzionamento: APP_WEBSERVER, APP_WEBCLIENT, APP_SOCKET, APP_UDPAPP e APP_WISERVER, che permettono il funzionamento rispettivamente da Web Server, Web Client, Socket, UDP e WiServer. APP_WEBSERVER è la predefinita e funziona sulla maggior parte degli Arduino, ma presenta diverse limitazioni; ad esempio non è in grado di funzionare sia come client che come server simultaneamente.

Una modalità alternativa sicuramente più interessante, maggiormente diffusa, utilizzata e consigliata, è la

APP_WISERVER (si ottiene con la libreria "WiServer.h") che permette a un sistema Arduino equipaggiato con shield Wi-Fi di essere configurato e funzionare sia come Web Server sia come Web Client.

Nel primo caso sarà possibile servire richieste di connessione provenienti da web client esterni ed inviare delle pagine HTML in risposta; nel secondo sarà invece possibile collegarsi a un web server e, tramite la tecnologia request GET o POST, inviare richieste di esecuzione comandi e ricevere pagine HTML di risposta. Per configurare la libreria WiShield scaricata come APP_WISERVER è necessario aprire il file "<directory di arduino>\libraries\WiShield\apps-conf.h", andare circa alla riga numero 39, commentare il "#define" di APP_WEBSERVER e rimuovere il commento al "#define" di APP_WISERVER. La sezione di codice in esame deve risultare come di seguito:

```
//#define APP_WEBSERVER
//define APP_WEBCLIENT
//define APP_SOCKET
//define APP_UDPAPP
#define APP_WISERVER
```

Arduino+shield Wi-Fi in modo che funzioni come Web Client in grado di collegarsi al sito internet di Google (www.google.it), inviare una richiesta di ricerca (nello specifico ricerchiamo la parola ElettronicaIn), ricevere la pagina HTML di risposta ed inviare questa pagina sulla porta seriale. In questo esempio, come prima operazione vengono definiti al-

cuni parametri di configurazione della rete wireless (indirizzo IP, gateway, subnet mask, selezione del tipo di crittografia della rete, eccetera). Oltre a ciò, viene definito l'indirizzo IP del server di Google (rappresentato da un array di quattro byte) determinato tramite ping dello stesso e creato l'oggetto di tipo GETrequest utilizzato per gestire la richiesta GET. Per quest'ultimo è necessario specificare l'indirizzo IP del server, la porta TCP da utilizzare (80 nel nostro caso), il sito Internet in formato stringa e il testo che rappresenta la richiesta da inviare (in questo caso costituita da "/search?q=ElettronicaIn"). In **setup()**, tramite la funzione **WiServer.init()** viene inizializzato il Web Client (il parametro NULL indica che, essendo il codice un esempio di Client, non dobbiamo servire nessuna pagina Web) ed inizializza la porta seriale per il debug e l'invio della pagina di risposta ricevuta. Successivamente viene inizializzato l'oggetto di gestione della richiesta GET (il parametro **Gestione_Risposte_Web** indica la funzione cui verrà passata la

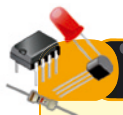
risposta ricevuta).

In **loop()**, ciclicamente, a distanza di un'ora una dall'altra, mediante la funzione **getGoogleSearch.submit()** vengono inviate a Google le richieste GET.

Alla fine del codice è presente la funzione **Gestione_Risposte_Web()**: quest'ultima riceve come primo parametro un puntatore a una stringa di testo contenente la risposta completa ricevuta dal Server e come secondo parametro la lunghezza della stringa stessa. Come già anticipato in precedenza, nella funzione le risposte HTML ricevute vengono inviate sulla porta seriale.

Nella **Fig. 2** vedete ciò che viene inviato sulla porta seriale facendo girare il software su un sistema reale (abbiamo utilizzato il Serial Monitor dell'IDE di Arduino).

È facile intuire che otteniamo il codice sorgente della pagina HTML ricevuta in risposta dal server di Google quando eseguiamo la ricerca. Nella **Fig. 3** appare invece quello che si ottiene digitando nella barra degli indirizzi di Internet Explorer la stringa "<http://www.google.it/search?q=>



per il MATERIALE

Le librerie e gli sketch sono scaricabili gratuitamente dal sito della rivista (www.elettronicaonline.it). Lo Shield WiFi è disponibile già montato e può essere acquistato presso Futura Elettronica al costo di 65,00 Euro (cod. WIFI_SHIELD). Il modulo transceiver WiFi (cod. MRF24WB0MA/RM) può anche essere acquistato separatamente al costo di 44,00 Euro. Tutti 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>

ElettronicaIn” corrispondente alla ricerca da effettuare e visualizzando il codice di origine dal menu *Visualizza Origine*. Confrontando le due immagini è abbastanza evidente che la risposta ricevuta dalla scheda Arduino è la stessa ricevuta dal browser. Chiaramente Explorer visualizza le informazioni ricevute in un formato più leggibile da noi umani, ma le informazioni sono equivalenti a quelle scambiate nella comunicazione su seriale.

CONCLUSIONI

L’implementazione del protocollo SPI e della relativa interfaccia, oltre alla presenza di un lettore per le card microSD, rendono il nostro shield Wi-Fi un prodotto valido e versatile. La compatibilità pin-to-pin con l’Ethernet Shield della AsyncLabs ci consente di montare il nostro shield in sistemi Arduino che montavano il prodotto AsyncLabs, mentre la compatibilità con le librerie Wi-Fi Shield di detta casa ci permette di controllare il nostro shield senza dover scrivere librerie specifiche, ma consente anche di sostituire il prodotto Asynclabs con il nostro senza fare alcuna modifica allo sketch. ■

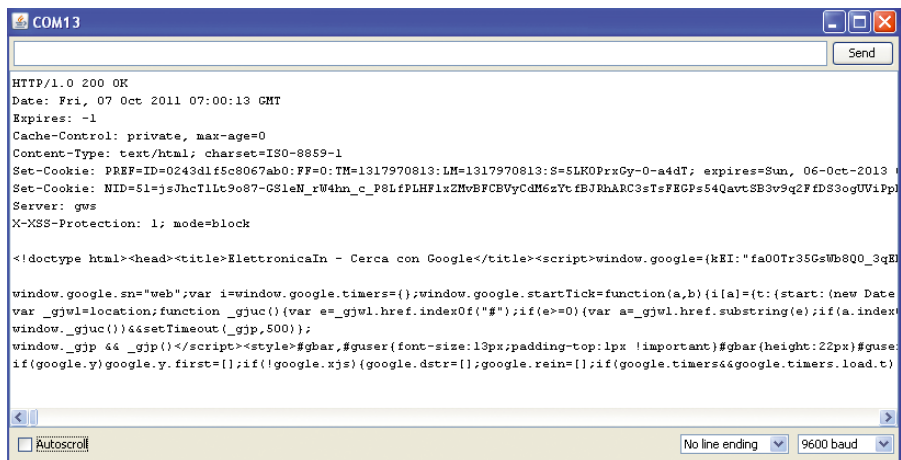


Fig. 2

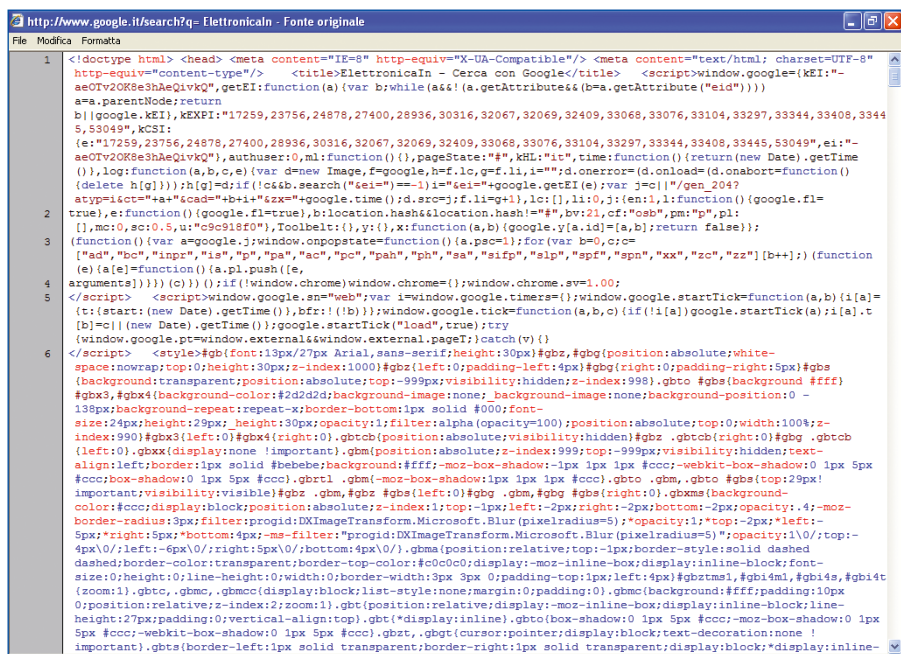


Fig. 3

RM ELETTRONICA

Forniture per hobbisti, scuole e industria

Distributore autorizzato:

Il tuo punto di riferimento per l'elettronica a ROMA

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

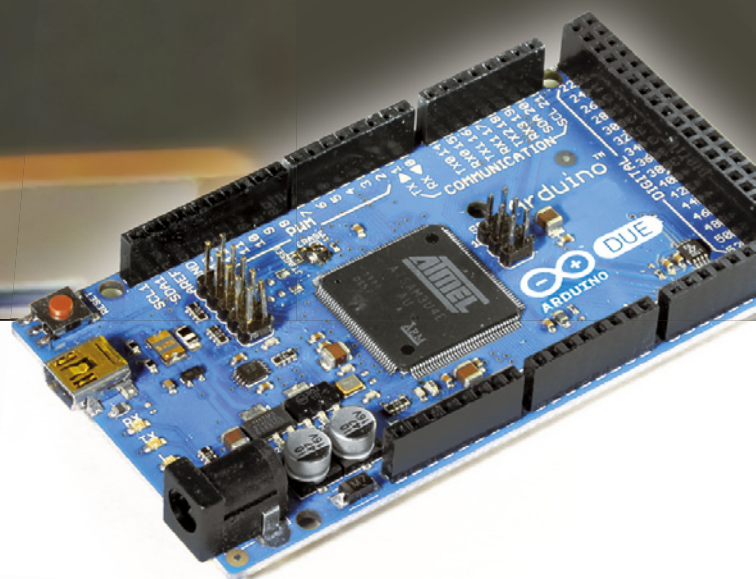
ARDUINO DUE, LEONARDO, Wi-Fi: TUTTE LE NOVITÀ DEL MONDO ARDUINO



di ARSENIO SPADONI



Finalmente ecco la board Arduino a 32 bit basata su core ARM e il nuovo Shield Wi-Fi. Ne ha parlato Massimo Banzi al Maker Faire di New York pochi giorni fa presentando queste e tutte le altre novità Arduino.



Negli ultimi anni le proposte del team di Arduino si sono limitate a semplici, piccoli e costanti miglioramenti della piattaforma nata sei anni fa ad Ivrea; se si esclude l'utilizzo di un chip più potente che equipaggia la board Arduino Mega e le modifiche all'interfac-

cia USB, la piattaforma è rimasta praticamente la stessa con al centro un microcontrollore AVR ad 8 bit prodotto dall'Atmel. Dopo sei anni dunque, è la volta del grande salto verso sistemi decisamente più complessi ma che, grazie all'approccio ed alla semplificazione di cui

il team di Arduino è capace, dovrebbero poter essere alla portata di chiunque, o perlomeno di quanti si sono fatti "le ossa" con le attuali schede di

prototipazione. I nuovi prodotti e le linee di sviluppo per il futuro sono stati presentati da Massimo Banzi durante l'edizione del Maker Faire che



	ADK	Ethernet	Leonardo	Arduino DUE
Microcontroller	ATmega2560	ATmega328	ATmega32U4	Atmel SAM3U4E ARM Cortex M3
Clock	16 MHz	16 MHz	16 MHz	96 MHz
Flash Memory	256 KB	32 KB	32 KB	256 KB
SRAM	8 KB	2 KB	3.3 KB	50 KB
Digital I/O Pins	54	14 (10)	14	54
Analog Pins	16	6	6	16 (12bit)

Android ADK Compatible USB Host Develop your own android accessory!	Wiznet W5100 Ethernet interface Optional PoE Module Bring your project online!	Onboard USB controller Build your own USB devices!	Onboard dual-channel DAC Bringing 32 bit power to Arduino!
Avr32 co-processor with fully open-source firmware H&D wifi module Easy to upgrade firmware Fully Hackable!	Breadboard-free electronic prototyping 30+ different modules Easy to use instructions & tutorials!	Arduino based dual platform robot Multiprocessor TinkerKit-compatible Program your own behaviours!	Look for us at Maker Faire 2011/New York www.arduino.cc

si è tenuta a New York il 17 e 18 settembre scorsi. Ecco dunque le novità in arrivo.

Arduino 1.0: L'intenzione è quella di dare una forma stabile e definitiva alle librerie, all'IDE ed al circuito elettrico entro pochissimo tempo, uno o due mesi al massimo. Insomma, mettere un punto fermo su tutti questi aspetti con pochissime modifiche. Questo lavoro sarà portato avanti con la collaborazione di tutta la community che è chiamata a testare la RC (Release Candidate) nonché ad apportare le necessarie modi-

fiche alle librerie attualmente in circolazione. Dopodiché l'intenzione è quella di congelare la versione 1.0 per un tempo ragionevolmente lungo. Come dice Banzi "...chi oggi scrive un libro su Arduino, non sarà costretto tra pochi mesi a dover fare degli aggiornamenti". Dunque un core 1.0 con un'interfaccia aggiornata per l'IDE ed alcune modifiche al layout delle schede. Queste ultime riguardano l'aggiunta di due piedini che saranno disponibili nelle release 3 sia di Arduino UNO che di Arduino Mega, due piedini che renderanno disponibile il bus

I2C o TWI che dir si voglia (durante la sua presentazione Banzi non ha mancato di tirare una "frecciatina" a Philips per la sua politica). Le nuove schede dovrebbero anche essere in grado di comunicare agli shield la tensione di funzionamento (5V o 3,3V). Infine Banzi ha annunciato la presenza di un altro pin "segreto", come lo ha definito. In altri tempi si sarebbe detto "per future applicazioni". In pratica è un pin che al momento non ha alcuna funzione ma che in futuro potrebbe evitare di cambiare un'altra volta il layout nel caso di aggiunta di una ulteriore funzionalità alla scheda.

Leonardo: è una scheda a basso costo basata sul chip Atmega32u4 che dispone al proprio interno dell'interfaccia USB per cui sulla basetta è presente un solo chip pur essendo disponibile la connessione USB. Ha la stessa forma e gli stessi connettori di Arduino Uno ma il circuito è più semplice e verrà fornita con i connettori da saldare. Il firmware permette di simulare dispositivi USB quali mouse, tastiera e porta seriale. Il tutto, come al solito, in versione open-source (Core, Bootloader, hardware). La scheda dovrebbe essere disponibile entro 1-2 mesi quando il core avrà raggiunto il necessario livello di stabilità.

Arduino Wifi Shield: Consente di aggiungere la funzionalità Wi-Fi a qualsiasi board Arduino. Invece di utilizzare un classico modulo commerciale, la scelta è caduta su un micro sistema WiFi prodotto da H&D Wireless accoppiato con un potente processore AVR a 32 bit che consente un utilizzo



dello stack TCP-IP in maniera semplificata ma completamente personalizzabile. È proprio su questo punto che Banzi ha insistito, sottolineando la necessità di poter entrare nel codice per poterlo modificare, aggiungendo protocolli personalizzati e tanto altro. Le risorse disponibili dovrebbero rendere molto semplice il passaggio da uno Shield Ethernet al nuovo sistema wireless.

Questa scheda consentirà, se quanto annunciato troverà riscontro nella realtà, di "aprire" facilmente Arduino al mondo della comunicazione e di Internet, consentendo agli appassionati di scatenare la propria fantasia anche in questo campo. La comodità della tecnologia Wi-Fi è ormai nota a tutti: niente più cavi da stendere (che aumentano il costo e il tempo di realizzazione di qualsiasi impianto) e piena libertà nel posizionamento dei diversi nodi della rete wireless.

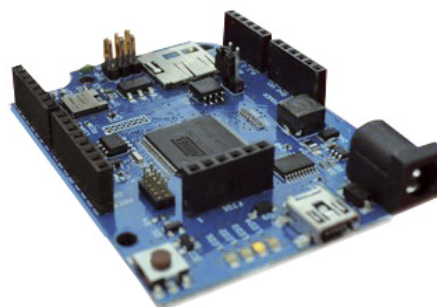
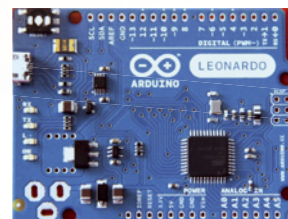
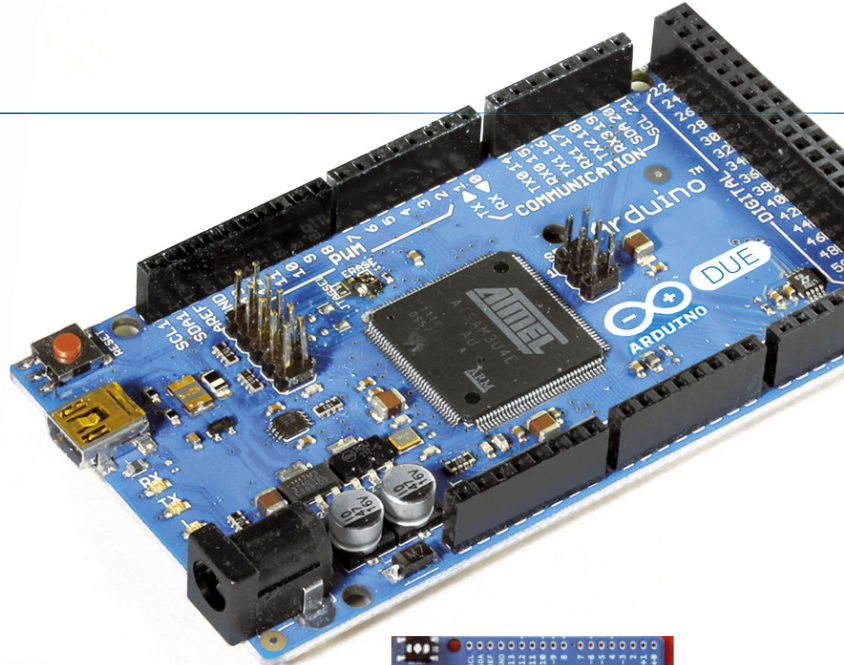
Certo, sul mercato esistono già degli Shield Wi-Fi, ma tutti sono piuttosto complessi da utilizzare: vedremo se con l'hardware (e con il supporto software) di Arduino, potremo accedere facilmente a questo tipo di applicazioni.

Da parte nostra, proprio in questo numero – ed in attesa della versione ufficiale di Arduino – presentiamo un nostro progetto di Shield Wi-Fi che ha il pregio di essere economico e, a nostro avviso, facile da utilizzare.

Il nostro shield Wi-Fi prende spunto da quello della AsyncLabs e ne mantiene la piena compatibilità hardware sia con l'Ethernet shield, sia con gli shield Wi-Fi prodotti da tale casa; nel nostro caso abbiamo previsto anche uno slot per inserire memorie di tipo microSD.

La compatibilità pin-to-pin con l'Ethernet Shield della AsyncLabs permette di sostituire, in sistemi basati su Arduino, tale shield con il nostro; ciò significa che se avete realizzato un progetto che fa uso dello shield ethernet AsyncLabs (ad esempio il controllo RGB descritto nel fascicolo n° 159) e volete passare dal controllo da LAN cablata a quello Wi-Fi, vi basta togliere lo shield ethernet e montare il nostro. Per gestirlo, potete utilizzare tranquillamente librerie esistenti, che nello specifico sono le stesse fornite dalla AsyncLabs per il suo Wi-Fi Shield.

Nel nostro caso utilizziamo il modulo Wi-Fi MRF24WB0MA della Microchip; il dispositivo è un Wi-Fi RF transceiver certificato IEEE 802.11, ha un data-rate compreso tra 1 e 2 Mbps, dispone di antenna interna con elettronica di adattamento integrata (è comunque predisposto per l'utilizzo di un'antenna esterna dedicata), ha un indirizzo MAC univoco, lavora nella banda di frequenze 2.400÷2.484 GHz ed è l'ideale per applicazioni di domotica, controlli remoti (sia consumer che industriali) e per impieghi wireless in generale. Il progetto è descritto a pagina 41.



Arduino Mega ADK: non è proprio una novità in quanto la board era già nota e disponibile per la vendita. Ricordiamo che questa scheda nasce per sviluppare dispositivi hardware da utilizzare in abbinamento con gli smartphone che utilizzano il sistema operativo Android. L'Arduino Mega ADK utilizza un microcontrollore ATmega 2560 e dispone di un'interfaccia USB Host; la board ha 54 I/O digitali (14 dei quali PWM), 16 ingressi analogici, un clock a 16 MHz, una porta USB, una presa di alimentazione, una connessione ICSP, e un tasto di reset. È sicuramente questo uno dei progetti più impegnativi del

Guarda il video di Massimo Banzi al Maker Faire di New York



Vuoi vedere il discorso di Massimo Banzi al Maker Faire? Nulla di più semplice: collegati al nostro blog (<http://blog.elettronica.in.it/2011/09/17/arduino-due-32bit-arm-leonardo-wifi-shield-tutti-i-nuovi-prodotti-del-mondo-arduino/>) e guarda la conferenza del co-fondatore di Arduino.

team di Arduino, un sasso lanciato nello stagno della community, che richiederà tantissimo lavoro per far sì che anche questo genere di applicazioni possano essere alla portata anche dei meno esperti. Un esempio di questa difficoltà è la board di DIYDRONES "PhoneDrone board for Android", basata anch'essa su Arduino e che, per quanto riguarda lo sviluppo software sembra essere rimasta al palo. L'argomento sembra essere piuttosto ostico anche per chi, come gli sviluppatori di DIYDRONES, hanno fatto un lavoro eccezionale negli ultimi due anni. Vedremo se questo rinnovato impegno del team di Arduino e di tutta la community riuscirà a dare nuova linfa a questo progetto.

Arduino Due: Finalmente la scheda a 32 bit targata Arduino. Per il chip la scelta è caduta su un processore ARM Cortex-M3 prodotto dall'ATMEL, il

modello SAM3U che funziona a 96MHz, dispone di 256Kb di Flash, 50Kb di Sram, 5 bus SPI, 2 interfacce I2C, cinque UART, 16 ingressi analogici con risoluzione 12bit e altro ancora. Nel comunicato di presentazione, il team di Arduino sottolinea il fatto che si tratta di prodotti ancora in fase di sviluppo e che da questo punto di vista l'apporto della community sarà fondamentale. Un limitato lotto di schede – contraddistinte dalla scritta "edizione per sviluppatori" – sarà disponibile per quanti vorranno contribuire allo sviluppo dopo il Maker Faire. L'intenzione è quella di arrivare alla versione definitiva entro i primi mesi dell'anno prossimo.

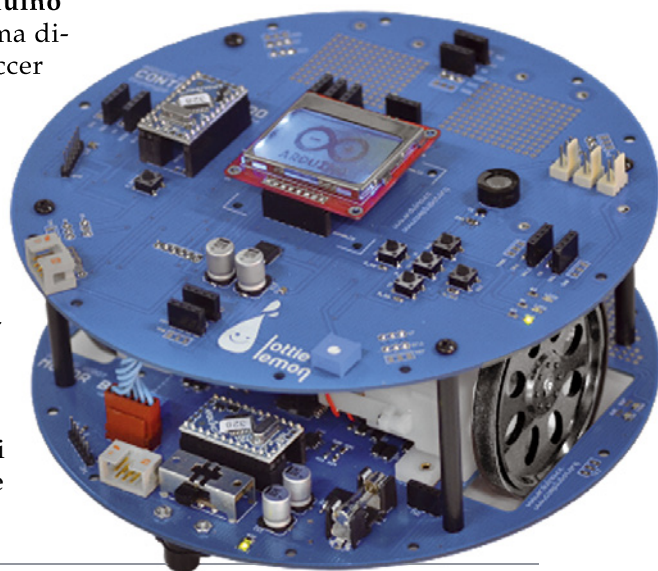
Nel presentare questo nuovo prodotto, Banzi ha più volte ringraziato i progettisti dell'Atmel, sia per l'hardware messo a disposizione, che per la fattiva collaborazione nella realizzazione della nuova scheda: un "mostro", come lo ha definito, con una interfaccia USB full speed e 16 ingressi per conversione AD con 12 bit di risoluzione. Infine Banzi, ricordando l'interesse di Arduino per la robotica e più in generale per la didattica, ha presentato l'**Arduino Robot System**, un sistema didattico per le gare di soccer della RoboCup, sviluppato da due ragazzini spagnoli che hanno utilizzato Arduino Mini ma che nella versione finale utilizzerà un Leonardo. Banzi ha voluto ricordare come il progetto sia stato completamente ed autonomamente sviluppato da due ragazzini di 17 anni, a dimostrazione della facilità di utilizzo

della piattaforma Arduino. Concludendo la sua presentazione, Banzi ha sottolineato per l'ennesima volta come *"Arduino siete voi: noi non facciamo altro che lanciare delle idee che vengono sviluppate dalla community: ogni singolo elemento di Arduino è il frutto di un lavoro comune, di tante persone che hanno contribuito alla sua realizzazione"*.

Che dire: complimenti ancora una volta a Massimo Banzi e a tutto il suo team.

Il lavoro di semplificazione e di "spacchettamento" dei problemi tecnici hardware e software realizzato con le schede ad 8 bit, se riuscirà – come siamo convinti – anche con i sistemi a 32 bit e con tecnologie complesse come le connessioni TCP/IP, decreterà il definitivo successo di Arduino che ormai possiamo considerare come uno "standard".

Un sistema di prototipazione e sviluppo che - anche per applicazioni industriali e su larga scala – può prescindere dalla board tradizionale come stiamo dimostrando su queste stesse pagine con le applicazioni del Prof. Michele Menniti.



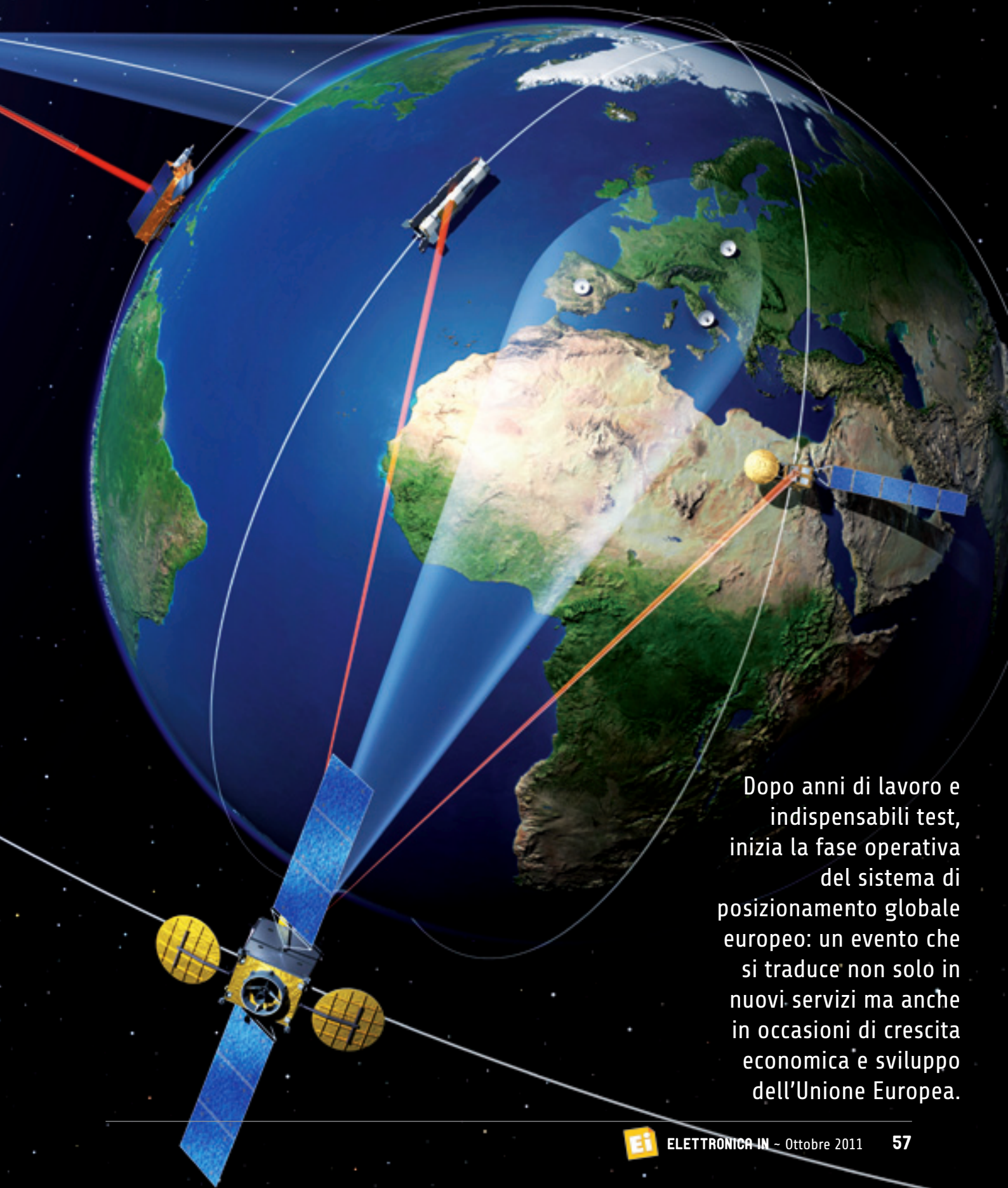


FINALMENTE IN ORBITA GALILEO

di DAVIDE SCULLINO

Se ne parla da tanto tempo, al punto che ormai quasi sembrava come il famigerato "ponte sullo stretto di Messina": è sulla bocca di tutti, ma nessuno l'ha visto; eppure, fortunatamente Galileo è alla griglia di partenza e dopo il lancio dei satelliti di completamento, quelli che già formano l'EGNOS (il sistema di correzione dell'errore del GPS) e di quelli di test e validazione, ora sono lì per prendere il volo i primi due satelliti dell'effettiva

costellazione, che forniranno una limitata ma pur efficace copertura a livello mondiale. Di Galileo abbiamo già parlato più volte, accennando alle sue peculiarità, tuttavia lo abbiamo fatto come si fa con qualcosa che ha da venire; ma ora che il sistema di posizionamento europeo si appresta a diventare



Dopo anni di lavoro e indispensabili test, inizia la fase operativa del sistema di posizionamento globale europeo: un evento che si traduce non solo in nuovi servizi ma anche in occasioni di crescita economica e sviluppo dell'Unione Europea.

precisione curve/segmentati). Il segmento terrestre del sistema EGNOS è composto da 34 stazioni RIMS (Ranging Integrity Monitoring Stations) disseminate sul globo terrestre, principalmente in Europa (due sono collocate presso i siti ENAV di Ciampino e Catania) incaricate di raccogliere le misurazioni dai satelliti di navigazione GPS ed inviarle, attraverso una rete dedicata chiamata TWAN (Transport Wide Area Network) alle CPF (Central Processing Facility) ospitate negli MCC; ci sono poi 4 MCC (Mission Control Center) situati in Italia (a Ciampino, gestito dall'ENAV) Spagna, Germania e Regno Unito incaricati del monitoraggio e controllo della rete TWAN e di tutto il segmento terrestre. Cinque CPF elaborano i dati provenienti dalle RIMS e calcolano le correzioni di clock, ionosferiche e delle effemeridi; infine, 6 stazioni NLES (Navigation Land Earth Stations) operanti in coppia effettuano l'invio (uplink) del segnale di correzione elaborato dalle MCC sui rispettivi satelliti Geostazionari che irradiano verso i satelliti il segnale di correzione. Oltre al segmento terrestre, vi è il segmento spaziale composto dai satelliti GPS e da 3 satelliti Geostazionari (Inmarsat AOR-E e IOR-W, ed Artemis) che effettuano la trasmissione del segnale di correzione SIS (Signal in Space), nella stessa frequenza del segnale GPS. Oltre che per l'aeronautica, il segnale di EGNOS è stato reso disponibile via Internet per applicazioni sperimentali, nell'ambito del progetto SISNet, lanciato dall'ESA a scopo didattico ed offerto gratuitamente previa richiesta via e-mail (www.esa.eu). Una delle sperimentazioni del SISNet ha riguardato un sistema di guida per non vedenti appoggiato ad un palmare dotato di ricevitore GPS, capace di guidare la persona secondo un percorso predefinito. Con questa tecnologia è possibile ottenere le correzioni Wide Area del sistema EGNOS anche in zone dove non è disponibile il segnale inviato dai satelliti geostazionari ma si può accedere ad Internet, tipicamente con una connessione di tipo wireless (GSM/GPRS/UMTS), il che risulta particolarmente frequente nella navigazione terrestre nei centri urbani. Lo sviluppo tecnico del sistema EGNOS è stato completato nel 2006. Il 2 marzo 2011, la Commissione Europea (proprietaria del sistema EGNOS) ha dichiarato ufficialmente disponibile il sistema per le applicazioni aeronautiche e Safety of Life.

una realtà, vogliamo tornare sull'argomento per parlare di quello che sarà e farà. Per prima cosa bisogna spiegare, a chi ancora non lo conosce, che cos'è e cosa fa il sistema Galileo.

GALILEO IN PRATICA

Avviato ufficialmente nel 2003 e sviluppato congiuntamente dall'Unione Europea e dall'Agenzia Spaziale Europea (ESA), Galileo entrerà in funzione nel 2014, quando la costellazione dei satelliti che forniranno il segnale di localizzazione conterà il minimo indispensabile per garantire un servizio di geolocalizzazione. Il completamento della costellazione e la piena operatività invece slittano al 2019, anno in cui in cielo ci saranno tutti e trenta i satelliti (27 operativi e tre di riserva) previsti, orbitanti su tre piani inclinati rispetto all'equatore a 23.222 km di quota, in orbita non geostazionaria. Il programma di lancio, che si servirà di razzi russi Soyuz e francesi Ariane, dovrebbe partire a breve e una volta posizionati i primi quattro satelliti operativi sarà possibile cominciare a fornire i servizi di navigazione e testare la piena funzionalità dei segmenti spaziale e di terra.

Galileo si compone, come tutti i sistemi di posizionamento globale, di un segmento di terra ed uno spaziale; il primo conta su due centri di controllo a terra (GCC) situati uno in Germania (ad Oberpfaffenhofen) e l'altro in Italia (al Fucino, presso Telespazio) che inviano il segnale orario di riferimento alla costellazione di satelliti (componenti il segmento spaziale) ed altri dati di correzione, ma che servono anche a monitorare costantemente la salute dei singoli elementi in orbita. Più esattamente, ogni GCC è il punto da cui i satelliti

ricevono i dati di aggiornamento e inviano a terra tutte le informazioni di diagnostica e quelle utili ad informare il sistema sul cambiamento di rotta e posizione che subiscono.

Il programma Galileo si articola in tre fasi: definizione, sviluppo e validazione in orbita, operatività completa. La prima, completata nel 2003, ha portato alle specifiche di base del sistema, mentre la validazione in orbita (IOV) sarà effettuata dopo la collocazione dei primi quattro satelliti della costellazione effettiva (quattro è il numero minimo di satelliti che consentono la localizzazione certa di un punto nello spazio tridimensionale), insieme alle stazioni di terra e al centro di controllo.

La fase di validazione, chiamata GIOVE (Galileo In-Orbit Validation Element) è iniziata col lancio del satellite GIOVE-A il 28 dicembre 2005, seguito dal GIOVE-B, il 27 aprile del 2008; questi due elementi sono serviti per verificare l'accesso alle frequenze che l'Unione Internazionale per le Telecomunicazioni (ITU) ha assegnato a Galileo. Il progetto GIOVE è servito anche per studiare la propagazione e le radiazioni nelle orbite terrestri a quota media (MEO, ossia Medium Earth Orbits) dove saranno posizionati i satelliti di Galileo, mettendo alla prova gli orologi atomici di bordo, il generatore di segnale ed i ricevitori a terra. La validazione in orbita verrà completata con i primi quattro satelliti operativi, i primi due dei quali partiranno il 20 ottobre prossimo (salvo rinvii dell'ultim'ora) a bordo di un razzo Soyuz ST-B. Entro la fine del 2012 dovrebbero essere portati in orbita altri due satelliti della costellazione definitiva, così da avere in tutto quattro satelliti,

I numeri di Galileo

Sebbene sia l'ultimo nato, il sistema di localizzazione europeo ha tutti i numeri per affermarsi a livello mondiale. Ecco da cosa è composto.

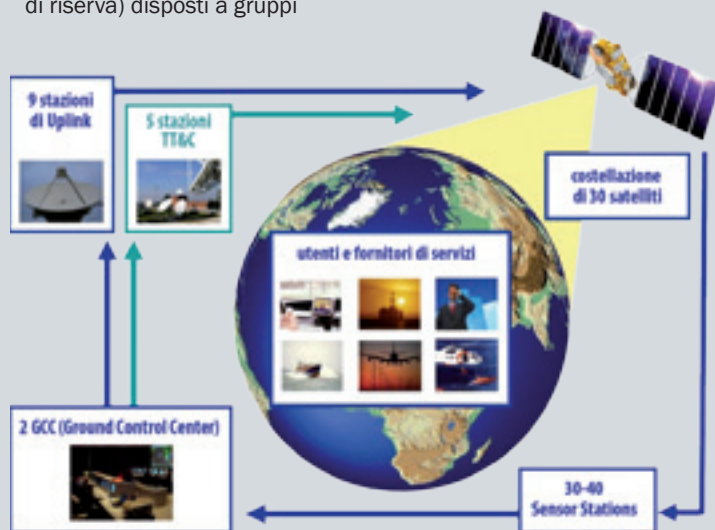
Segmento spaziale

È una costellazione di 30 satelliti (27 operativi più 3 di riserva) disposti a gruppi

di dieci su tre piani orbitali inclinati di 56° rispetto all'equatore, orbitanti ad un'altitudine di circa 23.222 km in modo da coprire costantemente il globo terrestre e ruotare lontano dagli apparati delle reti satellitari GLONASS e Navstar. I primi dieci verranno

no distribuiti sulle tre orbite previste e ruoteranno compiendo un giro di rivoluzione completo in 14 ore. Ogni satellite avrà a bordo due orologi atomici di nuovissima generazione e trasmetterà il proprio segnale in banda L, con un contenuto di dati ridondato. La posizione di ciascun satellite nell'orbita è stata studiata in modo da consentire ai pannelli solari che lo alimentano di ricevere costantemente i raggi del sole utili a ricavare elettricità.

dono oltre 100 stazioni in tutte le parti del mondo, anche se per arrivare a ciò bisognerà, vista la ridotta estensione dell'Europa, ottenere da altre nazioni il permesso di collocare le stazioni (si pensa ad ex colonie francesi ed inglesi) nel proprio territorio. Anche detto Ground Control System, il sistema di controllo di terra è composto da un Navigation System Control Center (NSCC), una rete globale di Stazioni di Orbitografia e Sincronizzazione (OSS) ed una serie di stazioni di tracking remoto, telemetria e comando (TT&C). Ogni OSS effettua delle misurazioni che vengono inviate al NSCC insieme ai Navigation Messages provenienti dai satelliti, ad informazioni meteorologiche ed ulteriori dati. Altri componenti del NSCC sono le Satellite Control



Segmento di terra

Galileo è controllato da un segmento a terra che dialoga con le stazioni di coordinamento dei segnali tra i vari satelliti; a riguardo, oltre alla realizzazione di una rete mondiale di sistemi di rilevamento dati (si spera che venga condivisa con quella del GPS) si preve-

che permettono un minimo di geolocalizzazione. Completata la fase di IOV con questi quattro satelliti, verranno lanciati uno alla volta gli altri elementi della costellazione. Il lancio del 20 ottobre sarà il primo di quelli previsti nel centro di lancio ESA di Kourou, nella Guiana Francese. I satelliti che partiranno ad ottobre sono stati inviati dalla Thales Alenia Space alla base di lancio; come gli altri che comporranno la

costellazione Galileo, peseranno 700 kg, misureranno più o meno $2,7 \times 1,2 \times 1,1$ metri e consumeranno poco meno di 1,6 kW (ricavati in buona parte da pannelli fotovoltaici aventi un'apertura di 13 m). Ogni satellite durerà in media 12 anni ed incorporerà un transponder per ricevere da terra i segnali di missione in banda C e trasmetterà verso terra il segnale UTC in banda L. Per il servizio di TT&C (Tracking, Telemetry & Command) verrà usata la banda S. A bordo, l'orario verrà scandito da due orologi atomici. Nello sviluppo di Galileo, il nostro Paese è in prima linea: Telespazio (gruppo Finmeccanica/Thales) svolge un ruolo di primo piano grazie al Centro Spaziale del Fucino, il più grande al mondo per le telecomunicazioni civili (da lì passano gran parte dei dati della TV da satellite). Il centro si

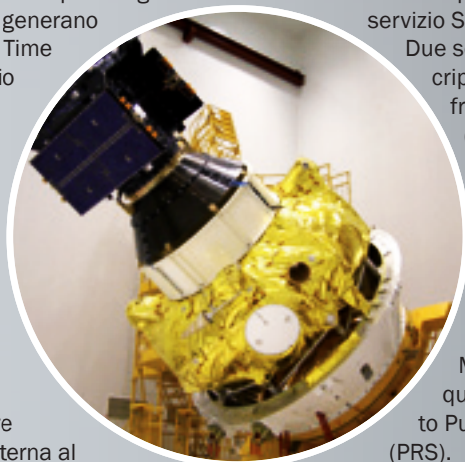
estende per 5.000 m² e a regime ospiterà oltre 100 tecnici e operatori specializzati che gestiranno i 30 satelliti e le attività operative relative al funzionamento del sistema; dalla sala di controllo principale si potrà valutare e modificare l'orbita di tutti i satelliti della costellazione e coordinare una rete di circa 40 stazioni di terra sparse per il mondo. Il centro del Fucino comprenderà il Galileo Control Segment, dedicato al controllo dei satelliti della costellazione e il Galileo Mission Segment, riservato alla fornitura, alla gestione e al monitoraggio dei servizi operativi. Avrà anche un centro di valutazione della qualità e affidabilità del segnale di Galileo.

TUTTO NASCE DA EGNOS

A parte i due del programma GIOVE, in orbita vi sono da tempo altri tre satelliti, che hanno uno stretto legame con Galileo, pur non appartenendo al pro-

Facilities (SCF) che effettuano la manutenzione dei satelliti, controllano le orbite ed operano telemetria e controllo a distanza tramite le stazioni TT&C; ci sono poi le Orbitography and Synchronisation Processing Facility (OSPF), che calcolano le effemeridi per ogni satellite, l'offset degli orologi a bordo, ed effettuano una predizione di questi parametri per generare il SISA (Signal In-Space Accuracy).

Ci sono anche le Precision Timing Station (PTS) che incorporano gli orologi atomici e generano il Galileo System Time (GST), cioè l'orario UTC. L'ultimo elemento del segmento di terra è la Navigation Control Facility (NCF), che si occupa di controllare, monitorare e gestire gli OSPF, OSS, PTS ed NCF. Esiste inoltre un'interfaccia esterna al



sistema, la Service Centres Interface, che si occupa di gestire la comunicazione con entità esterne. I Centri di Controllo (GCC) sono due: uno primario (in Germania) ed uno di riserva (Italia).

Caratteristiche del Segnale

I satelliti di Galileo trasmetteranno 10 tipi di segnale: sei segnali saranno accessibili a tutti gli utenti sulle frequenze portanti E5a (1.176,45 MHz), E5b (1.207,14 MHz) ed L1 (1.575,42 MHz) e offriranno l'Open Service (OS) ed il servizio Safety of Life (SoL).

Due segnali saranno criptati e trasmessi sulla frequenza portante E6 (1.278,75 MHz) ed offriranno il Commercial Service (CS). Infine due segnali saranno criptati e trasmessi sulle frequenze E6 (1.278,75 MHz) ed E2-L1-E1 (1.575,42 MHz) ed offriranno quello che viene chiamato Public Regulated Service (PRS).

gramma: si tratta di quelli del sistema EGNOS, che l'ESA sfrutta per trasmettere, su frequenze compatibili con quelle dei ricevitori GPS, i segnali di correzione dell'errore insito nel Navstar americano. Ciò, allo scopo di consentire una localizzazione più precisa di quella ottenibile con il solo GPS civile. Il livello di precisione raggiunto è tanto elevato che il sistema è stato di recente adottato per l'assistenza al volo, offrendo prestazioni di livello comparabile con quello dei migliori ILS (sistemi di atterraggio strumentale). Proprio EGNOS è stato il primo tassello di Galileo, sebbene sia nato come complemento del GPS, per migliorare la precisione della localizzazione in Europa (in realtà copre anche tutto il Nord Africa, compresi Marocco settentrionale, Tunisia, Algeria e Libia). A sua volta EGNOS, insieme al WAAS americano, forma la rete SBAS, che si appoggia sui satelliti dell'Inmarsat, la più datata rete

di telefonia satellitare, operante ormai da circa dieci anni e basata su quattro satelliti posti in orbita geostazionaria (cioè che ruotano insieme al nostro pianeta) intorno alla Terra, a una quota di 41.300 km dal livello del mare. WAAS è il sistema satellitare di correzione dell'errore di posizionamento nato per primo (negli U.S.A.): è stato sviluppato in due versioni (WAAS e CWAAS) che coprono l'intera America Settentrionale; WAAS (Wide Area Augmentation System) fornisce la copertura nella parte soprastante gli U.S.A. e CWAAS riguarda il Canada e il resto del Nord America. È operativo dal 2004. Invece EGNOS (European Geographic Navigation Overlay System) è operativo dall'inizio del 2006 e si appoggia sui due restanti satelliti Inmarsat più l'Artemis, che, come per il WAAS, vengono aggiornati da terra e ritrasmettono i dati di correzione su frequenze e con

formato compatibili con i ricevitori GPS ad uso civile.

Il funzionamento dello SBAS si spiega così: le stazioni a terra dei due sistemi rilevano l'errore dei dati trasmessi dai satelliti GPS, imputabile principalmente alla ionizzazione degli strati più bassi dell'atmosfera, confrontando la propria posizione con i dati generati localmente; per estendere l'area monitorata e dare, con una sola stazione, dati inerenti a più zone geografiche, è stata creata una "rete" di punti di rilevamento capaci di valutare il margine di errore relativo a ciascuno di essi e informarne la stazione. Per quanto riguarda EGNOS, le stazioni di elaborazione dell'errore dei dati GPS sono sparse su tutto il territorio europeo e rilevano ciascuna un modello di errore valido per la zona di pertinenza, inviando poi i relativi dati ad una stazione centrale di elaborazione. Ciò realizza un reticolo di fattori di



Cosa ci offrirà?

Il sistema Galileo sarà in grado di offrire, oltre al segnale per la geolocalizzazione, servizi di vario tipo, che sono i seguenti.

- **Open Service:** è il servizio gratuito che fornisce un segnale preciso un po' più di quello del GPS e accessibile a chiunque, per navigazione, controllo di mezzi di trasporto, ricerca e salvataggio; Open Service non garantisce la continuità del servizio e neppure l'informazione di integrità del segnale.
- **Servizio di Sicurezza (SoL, ossia Safety-of-Life)** migliora il servizio OS avvertendo tempestivamente gli utenti quando l'OS non raggiunge certi margini di accuratezza o di continuità (integrità); è sempre garantito.
- **Servizio Commerciale (CS, ossia Commercial Service):** permette l'accesso a due segnali aggiuntivi per un'elaborazione dati più veloce e una precisione centimetrica; il suo segnale è criptato e ne viene garantita la continuità, ma è disponibile a pagamento.
- **Public Regulated Service (PRS):** si basa su un segnale criptato (in realtà i segnali sono due con codici di classificazione e dati criptati) di cui sono garantiti continuità ed integrità, utilizzato essenzialmente da organi di Pubblica Sicurezza, come ad esempio le forze di Polizia, la Protezione Civile e le Forze Armate. I servizi PRS sfrutteranno un canale di comunicazione controllato dai governi degli stati membri dell'UE e potrebbero essere utili, fra l'altro, per migliorare gli strumen-

ti e i mezzi utilizzati dall'Unione Europea nella lotta contro le esportazioni illegali e l'immigrazione clandestina, nonché nell'Homeland Security.

- **Supporto al servizio di Ricerca e Recupero,** che rappresenta il contributo dell'Europa al sistema internazionale COSPAS-SARSAT; Galileo giocherà un ruolo importante nel sistema di Ricerca e Recupero MEOSAR (MEO Search & Rescue system) in quanto i suoi satelliti potranno ricevere segnali di emergenza emessi da navi, aerei e singoli individui e trasmetterli ai centri di soccorso nazionali in modo da rendere loro nota la precisa localizzazione dell'incidente (in ogni punto della Terra, in qualsiasi momento, almeno un satellite della costellazione di Galileo sarà visibile, in modo da poter sempre distribuire un segnale di soccorso in tempo reale.

Molto importante sarà il Safety-of-Life service, destinato alla sicurezza dei trasporti e delle persone; oltre alle prestazioni dell'Open Service, garantirà un segnale di integrità che avviserà tempestivamente l'utente nel caso in cui le prestazioni del sistema dovessero risultare degradate. Il PRS rientra nell'ambito di un accordo europeo mirato a creare un servizio speciale di navigazione per gestire meglio servizi critici di trasporto e di emergenza, per migliori interventi delle forze dell'ordine (polizie), per migliorare la sicurezza interna (controlli alle frontiere) e per rendere più sicure

correzione molto fitto, i cui dati vengono aggiornati in tempo reale, in quanto le condizioni di propagazione del segnale GPS attraverso l'atmosfera mutano rapidamente in relazione alla situazione meteo. I fattori di correzione vengono poi inviati ai satelliti WAAS/EGNOS e da essi sono ritrasmessi a terra utilizzando la banda radio L1 e ricevuti dai ricevitori GPS e, in

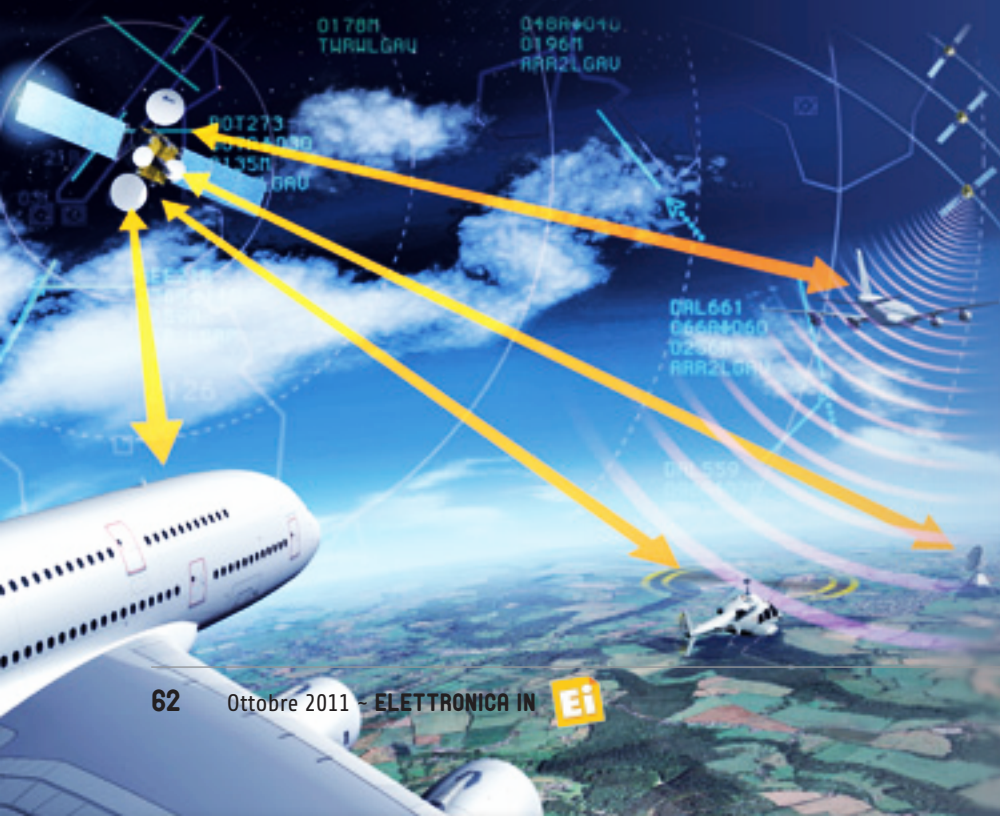
futuro, Galileo. Il ricevitore seleziona i dati validi per i punti del "reticolo" ad esso più vicini e li applica ai satelliti che sta ricevendo in quel momento.

PERCHÈ È IMPORTANTE?

L'avvento di Galileo avrà molte implicazioni; indubbiamente sul piano economico (dove è stato ed è un banco di prova per la tenuta degli accordi di coo-

perazione economica tra i Paesi dell'UE) ma anche su quello dell'arricchimento tecnologico. Senza parlare dei benefici in tema di sicurezza.

Ma che senso ha realizzare un sistema di localizzazione satellitare, quando nel mondo già opera il GPS (Navstar americano più WAAS ed EGNOS) e considerato che i russi hanno riportato in auge il loro GLO-NASS e che i due sistemi già forniscono un servizio fruibile da tutti, gratis? Ebbene, Galileo nasce con l'intento di realizzare qualcosa in più: fornire, insieme ad un ulteriore segnale localizzazione, numerosi altri servizi; se così non fosse, questo progetto (per il quale si fatica a reperire i necessari stanziamenti, che pure sono poca cosa rispetto alla spesa pubblica dei singoli Paesi aderenti al progetto: si parla di circa 3,4 miliardi di euro...) sarebbe davvero nato solo per rendersi indipendenti da Stati Uniti, Russia e Cina. Galileo dovrebbe offrire una localizzazione satellitare con un'accuratezza inferiore ai 10



le missioni di pace. Il PRS è un servizio protetto dalle minacce alle infrastrutture che dipendono dalle tecnologie di navigazione assistite da satellite, per far sì che gli utenti del PRS possano ricevere informazioni di vitale importanza anche quando altre infrastrutture non sono più disponibili.

Lo sviluppo del PRS con i suoi segnali cifrati rappresenta una risposta alle crescenti minacce alle infrastrutture che dipendono dalle tecnologie di navigazione assistite da satellite. Il PRS tutela queste infrastrutture contro tali minacce, tra cui lo "spoofing", che è una tecnica attraverso la quale si possono alterare i segnali che guidano un veicolo o una nave, ma anche fornire posizioni inesatte, ad esempio a un'automobile della polizia o a un'ambulanza. È una minaccia concreta, in quanto in Internet già si trovano in vendita dispositivi di oscuramento facilmente accessibili a tutti, capaci di provocare danni se finiscono nelle mani sbagliate. L'uso del PRS sarà controllato e limitato ad organismi governati negli Stati membri dell'UE e nelle istituzioni europee. Paesi terzi ed organizzazioni internazionali che concludono gli opportuni accordi con l'UE potranno accedere all'uso del servizio.



centimetri, tanti servizi complementari e nessuna interruzione o limitazione per l'uso civile; ciò perché nasce per usi civili, quindi, a differenza del Navstar, proprietà degli U.S.A. e sviluppato per scopi militari, funziona anche quando esso o il GLONASS dei Russi vengono "oscurati" (per il Navstar, ciò è accaduto, ad esempio, ai tempi della guerra in Bosnia). Senza contare che Galileo non prevede due livelli del segnale di localizzazione, ossia quello grezzo e il preciso, come invece fa il Navstar: fornisce solo il segnale preciso, seppure ne subordini l'utilizzo a determinate condizioni e fornisca il massimo della precisione solo agli operatori istituzionali o a chi si abbona a pagamento ai servizi correlati. Va comunque detto che il segnale standard di Galileo è molto più preciso di quello del Navstar. Oltre a ciò, tra gli scopi del programma Galileo c'è aumentare la copertura globale soprattutto nelle regioni con latitudine sopra i 75° e rendere maggiormente disponibile il

segnale nelle aree urbane. Sebbene possa vivere di vita propria, Galileo potrà collaborare con GPS e GLONASS nell'ambito della realizzazione di un sistema mondiale integrato di posizionamento globale: quello che conosciamo con il nome di GNSS (Global Navigation Satellite System); in quest'ottica, è assai probabile che presto vedremo in commercio ricevitori combinati in grado di ricevere e sfruttare i segnali di GPS, GLONASS e Galileo. Sulla base di queste considerazioni si può dire che Galileo nasce come servizio multimodale e non come semplice sistema di localizzazione; sarà in grado di migliorare la sicurezza e l'efficienza dei voli e del trasporto marittimo e terrestre, dove farà quel che attualmente fanno i sistemi GPS: fornirà i dati per trovare le strade più corte o (in collaborazione con sistemi di monitoraggio del traffico) meno trafficate. Grande vantaggio ne potranno trarre anche i mezzi di soccorso, ai quali potrebbe essere indicata la strada migliore per

arrivare sul luogo di un incidente, evitando gli ingorghi.

In Galileo il segnale preciso è liberamente accessibile ai servizi di pubblica utilità e a quelli di soccorso, mentre viene reso a pagamento a tutti gli utenti privati o commerciali. Questa è un'altra significativa distinzione esistente tra Galileo e il GPS tradizionale.

Altra caratteristica di Galileo (che GLONASS e Navstar non hanno) è il "messaggio di integrità", che permette di capire quando il segnale inviato dai satelliti garantisce la localizzazione con i parametri di certezza e precisione previsti; se manca tale messaggio, il ricevitore sa che il segnale è diventato inaffidabile. Questa forma di validazione del segnale consente l'impiego nelle applicazioni "critiche" (cosiddette Safety Of Life) come ad esempio l'assistenza al volo nell'aviazione civile, dove il solo GPS non è ritenuto affidabile; proprio nell'aviazione, la recente adozione a complemento dell'ILS fa presagire che in un futuro pros-

SISNet: quando Galileo arriva da Internet

I dati di correzione Wide Area del sistema GPS ottenuti da EGNOS, sono da tempo disponibili anche dal web; in questo modo, anche chi ha un ricevitore GPS base e non predisposto per ricevere il segnale del sistema europeo, può ottenere una notevole precisione paragonabile a quella ottenibile dalla localizzazione con Galileo. I dati, accessibili da Internet mediante una connessione autenticata ottenibile gratuitamente dietro semplice richiesta all'ESA via e-mail (scrivere a SISNET@esa.int) possono essere ricevuti su uno smartphone o PC palmare dotati di ricevitore GPS e connessione Internet GPRS, UMTS, HSDPA o semplicemente di WiFi. Il servizio SISNet è stato realizzato per molte ragioni, alcune delle quali sperimentali ed altre di pubblica utilità; con i suoi dati sono state condotte varie sperimentazioni, tra le quali la guida di non vedenti nello scenario cittadino: caricando il tragitto in un palmare dotato di ricevitore GPS e connessione Internet, è stato possibile guidare con una precisione di meno di un metro gli spostamenti, garantendo la massima sicurezza soprattutto nell'attraversamento delle strade. Questo è un esempio di applicazione

irrealizzabile con il solo GPS, perché i 5÷10 metri di precisione garantiti dal Navstar combinato con il DGPS possono fare la differenza fra trovarsi al sicuro su un marciapiede o nel mezzo di una strada alla mercé delle automobili! I dati del SISNet possono comunque essere impiegati per mille altre applicazioni, quali ad esempio la guida di macchine agricole e tanto altro che Galileo potrà



offrire una volta operativo. È peraltro probabile che SISNet sia offerto gratuitamente in attesa della reale operatività di Galileo e dell'arrivo in commercio dei ricevitori basati su di esso, dato che il sistema di localizzazione europeo ancora non è fruibile; quando lo sarà, facilmente l'accesso a SISNet non verrà più concesso, almeno quello libero.

simo il GNSS potrà rimpiazzare molte apparecchiature, tra cui i radar di terra e gli ILS (Instrumental Landing Systems). E, non è escluso, potrà servire per i sistemi di guida automatica degli UAV (che sono gli aerei senza pilota cui abbiamo dedicato un articolo nel fascicolo n° 159) già impiegati in alcuni scenari di guerra, ma utili anche in tempo di pace, ad esempio per fare rilievi o prestare soccorso in occasione di catastrofi di ingenti proporzioni. Siccome oggi sono tantissime le applicazioni basate sul GPS e molti di noi non riescono a fare a meno della localizzazione satellitare, comprendete bene quanto importante sia, sul piano economico e strategico per noi europei, avere un servizio proprio e non "preso a prestito" come è per l'attuale GPS.

IL BUSINESS DIETRO L'ANGOLO

Alla luce di quanto detto, appare evidente l'importanza del progetto Galileo per noi europei; un'importanza che, purtroppo, ancora sfugge ai più e che non viene compresa fino in fondo: lo dimostra il poco impegno economico profuso dai Paesi dell'UE. A dimostrazione di ciò, se domandate in giro che cos'è Galileo, qualcuno vi chiederà se state parlando del celebre "Galilei"... Qualcun altro, se gli spiegate di cosa si tratta, a stento capirà e vi risponderà che in fondo il navigatore della sua automobile già funziona bene... Eppure Galileo è importante per il nostro sviluppo; lasciamo stare per un istante il servizio che può darci, che pure ha i suoi bei perché... Guardiamola dal punto di vista economico: realizzare e ancor più mantenere efficiente un

sistema come Galileo, significa muovere un giro d'affari notevole: serve costruire (è già stato fatto) e far funzionare i centri di controllo a terra, che sono un insieme di antenne, cavi, computer, reti locali, personale ecc. Occorre provvedere al lancio dei satelliti, quindi preparare o noleggiare i razzi vettori, assistere il lancio, allestire le basi di lancio. Tutto ciò richiede personale e materie prime, ma soprattutto crea posti di lavoro: infatti l'industria del settore spaziale del nostro continente, che già occupa 40.000 persone, nella realizzazione e manutenzione di Galileo potrebbe creare in tutto 100.000 posti di lavoro. Il tutto verrebbe ripagato mediante una strategia commerciale ben definita e che conta, oltre che sulle entrate dirette, anche sugli stanziamenti dei Paesi UE che investiranno in questa infrastruttura (va comunque detto che l'intenzione di rendere parte dei servizi a pagamento ha creato qualche attrito con la Comunità Scientifica Internazionale, la quale contrappone la gratuità del GPS e del GLONASS). Le entrate dirette deriveranno dal fatto che parte dei servizi di Galileo verrà concessa in abbonamento a soggetti privati e istituzionali: i satelliti trasporteranno, oltre al segnale orario (orario UTC) altri dati resi accessibili in cambio di denaro. Già questo porterà un consistente introito. Mettiamoci poi i servizi di accessibilità del segnale ad altissima precisione (si parla di centimetri...) affittato, per così dire, ad aziende e professionisti che devono fare rilievi geodetici, cartografici e catastali; un esempio lo vediamo in Lombardia, dove



la Regione ha da tempo attivato un servizio di geolocalizzazione precisa a pagamento basata sulla combinazione dei segnali Navstar ed EGNOS. Il servizio è rivolto a chi opera nel campo della cartografia e dell'edilizia, per tracciare e verificare i confini di terreni e fabbricati; lo stesso può essere sfruttato dall'edilizia stradale per dirigere opportunamente le macchine operatrici, ma anche nell'agricoltura di precisione, per guidare i trattori e le mietitrici o le macchine di raccolta automatica verso le piante da cui cogliere i frutti, oppure quelle per la semina teleguidata. C'è poi tutto il discorso che riguarda la pubblica sicurezza, l'ausilio nelle emergenze e in generale i servizi di pubblica utilità: tutte cose che non portano un utile indiretto ma fanno risparmiare un sacco di denaro pubblico. In questo discorso rientra anche il risparmio, in termini di impianti e loro conduzione, che deriverà dalla sostituzione delle apparecchiature di assistenza al volo come l'ILS e il radar di terra, resa possibile dalla qualità del segnale combinato di Galileo ed EGNOS. La realizzazione del progetto Galileo è finanziata sia da fondi pubblici, sia dall'intervento di sponsor privati che per quest'anno hanno già stanziato 100 milioni di euro. Anche la Cina, nel 2003 ha annunciato di partecipare al progetto Galileo, con un investimento di 230 milioni di euro offerto in cambio dell'accesso ai servizi. I finanziatori privati potranno avere un ritorno economico dalla partecipazione agli utili derivanti dalla vendita dei servizi; ma anche dai risparmi legati all'uso del sistema: ad esempio, si stima che per ogni punto percentuale di riduzione del tempo di viaggio stradale (e perciò di inquinamento, di incidenti, di traffico) ottenti-

Ogni satellite operativo avrà a bordo un transponder per ricevere da terra i segnali di missione in banda C e trasmettere verso terra il segnale UTC in banda L. L'orario verrà scandito da due tipi di orologio atomico: il principale, a maser passivo a idrogeno, ha un errore di 1 miliardesimo di secondo al giorno, mentre il secondario, a rubidio, ha una tolleranza di 10 miliardesimi di secondo al giorno. A bordo ci sono 2 orologi per tipo, ma ne funziona una sola coppia alla volta. L'orologio a maser produce una frequenza di riferimento dalla quale viene generato il segnale di navigazione; se dovesse guastarsi, subentrerebbe l'orologio a rubidio e verrebbero accesi i due orologi di riserva. L'unità di controllo e monitoraggio dell'orologio e l'interfaccia tra i



quattro orologi è l'unità di generazione del segnale di navigazione (NGSU) la quale, insieme ai circuiti che convertono la frequenza, genera il segnale di navigazione partendo dagli input che provengono dall'unità di controllo dell'orologio e dai dati inviati da terra (relativi alla navigazione ed all'integrità). Il segnale di navigazione viene quindi convertito in banda L per essere trasmesso a terra verso i ricevitori degli utenti. L'orologio a rubidio è basato su un risonatore atomico contenente vapori ad alta temperatura; gli atomi vengono eccitati e portati ad uno stato energetico più alto dalla luce di una lampada al rubidio, che si trova ad un'estremità del risonatore. All'altra estremità, un fotodiodo rivela la quantità di luce che attraversa la cella. Dopo l'eccitazione, gli atomi tornano allo stato energe-

tico inferiore, ma sono nuovamente eccitati a un livello intermedio a causa della proiezione nel risonatore di una radiazione a microonde a una ben determinata frequenza. Quando gli atomi sono nello stato intermedio, l'assorbimento di luce è massimo. Il fotodiodo realizza una retroazione che modifica la frequenza delle microonde emesse, calibrando la sorgente a microonde in modo da determinare il massimo assorbimento di luce. La risonanza è

mantenuta dall'energia della lampada al rubidio, poiché gli atomi nello stato intermedio vengono eccitati di nuovo nello stato più elevato e poi decadono nello stato di energia minore, a partire dal quale si ripete l'intero ciclo.

L'orologio maser passivo a idrogeno è invece basato su un risonatore atomico dove un serbatoio rifornisce di idrogeno

molecolare un bulbo a scarica di gas, nel quale le molecole sono dissociate in atomi di idrogeno. A questo punto gli atomi entrano in una cavità di risonanza attraverso un collimatore e un selezionatore a campo magnetico. Il selezionatore permette l'ingresso solo agli atomi che si trovano al livello energetico voluto. Nella cavità, gli atomi sono immagazzinati in un bulbo di quarzo, dove tendono a tornare al loro stato energetico fondamentale, con una transizione che dà luogo a emissioni di microonde a una ben determinata frequenza. L'emissione è rivelata mediante un circuito che misura la frequenza esterna d'interrogazione rispetto a quella della risonanza di orologio (detto circuito d'interrogazione). La transizione atomica avviene solo quando la frequenza immessa coincide con quella di risonanza atomica.

bile da Galileo, si possano risparmiare circa 200 miliardi di euro. Nell'aviazione, la stessa riduzione può essere di circa 0,5 miliardi di euro. Si calcola anche che i benefici economici legati all'uso di Galileo saranno, nel ventennio 2000÷2020, quantificabili in 74 mi-

liardi di euro; cifre estremamente elevate, che garantiscono guadagni con margini di rischio piuttosto ridotti. Le stime dicono inoltre che Galileo fornirà 90 miliardi di euro all'economia europea in circa 12 anni, in termini di entrate per l'industria e di utilità pubblica.

OSCILLOSCOPIO PALMARE MONOTRACCIA

dell'ing. MIRCO SEGATELLO

Costruiamoci uno strumento per tutte le tasche, dotato di display LCD e capace di misurare segnali con un'ampiezza di banda fino a 700 kHz, ideale per visualizzare segnali audio, tensioni di alimentatori switching, impulsi.



Con l'aumentare delle prestazioni dei microcontrollori e con la diminuzione del prezzo dei display grafici, è stato possibile realizzare circuiti elettronici più impegnativi dei classici timer o controllori per processi e motori. Oggi riusciamo anche a realizzare piccoli oscilloscopi palmari alimentati a bat-

teria, che tanto vanno di moda in questo periodo; si tratta di veri e propri oscilloscopi che, pur ammettendo alcuni limiti nelle prestazioni, hanno un costo decisamente

ragionevole, fattore che, in campo hobbistico, è decisivo. In commercio se ne trovano di svariati modelli, con prestazioni variabili e persino anche in kit di montaggio. Quello che andiamo a descrivere in queste pagine è proprio un piccolo oscilloscopio portatile adatto a misure in bassa frequenza, di facile ed immediato utilizzo, ma non per questo limitato nelle funzioni. Un punto fermo di questo progetto è stato proprio conservare, per quanto possibile, tutte le funzioni di un oscilloscopio professionale, compresa una sofisticata gestione via software di tutte le funzionalità.

Il nostro strumento è a singola traccia (e perciò dispone di un unico connettore BNC) ed ha le tipiche regolazioni di base-tempi, volt/divisione, trigger, oltre ad altri comandi. Ma vediamo bene di cosa si tratta, partendo dall'analisi dello schema elettrico del circuito che lo realizza.

Paragonato a quello di un oscilloscopio pro-

fessionale, lo schema è davvero semplice e i componenti utilizzati, veramente pochi, in quanto tutte le funzioni sono state implementate tramite un microcontrollore della Microchip di tipo 18F452. Ecco che quindi basta una manciata di componenti per completare il progetto. Visto il carattere hobbistico dello strumento proposto, per la realizzazione ci siamo affidati a soli componenti in formato DIP, decisamente più facili da gestire. Adottare componentistica SMD avrebbe forse permesso di limitare ulteriormente gli ingombri finali (che comunque sono già molto contenuti) ma per contro la costruzione sarebbe stata più difficile. La particolare attenzione alla realizzazione del PCB e l'utilizzo di un adeguato contenitore, hanno permesso di ottenere un risultato finale molto apprezzabile sen-

CARATTERISTICHE TECNICHE

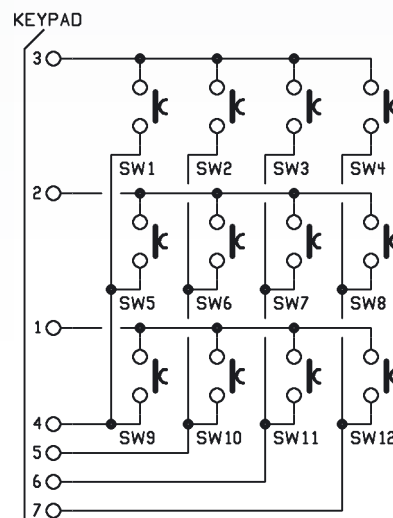
- Frequenza di campionamento massima: 1 Msps
- Larghezza di banda: 700 kHz
- Numero dei canali: 1
- Impedenza d'ingresso: 1 Mohm/20 pF
- Ingresso: BNC standard
- Sensibilità in verticale: da 10 mV/div a 5 V/div a passi 1,2,5
- Base dei tempi: da 20ms/div a 10 μ s/div a passi di 1,2,5
- Modo di funzionamento: run, stop, singola acquisizione
- Livello e pendenza del trigger regolabili
- Regolazione livello di offset
- Accoppiamento AC, DC, GND
- Misure automatiche di tensione Vpp, Vrms, Vmed
- Misure tramite cursori di tempo e tensione
- Doppia modalità di visualizzazione
- Protezione in ingresso fino a ± 50 V
- Alimentazione: a batteria Li-Ion 8,4 V
- Completo di circuito di ricarica
- Autospegnimento: dopo 5 o 10 minuti
- Totale gestione tramite pulsanti
- Dimensioni: 145 x 85 x 27mm

za alcuna lavorazione industriale.

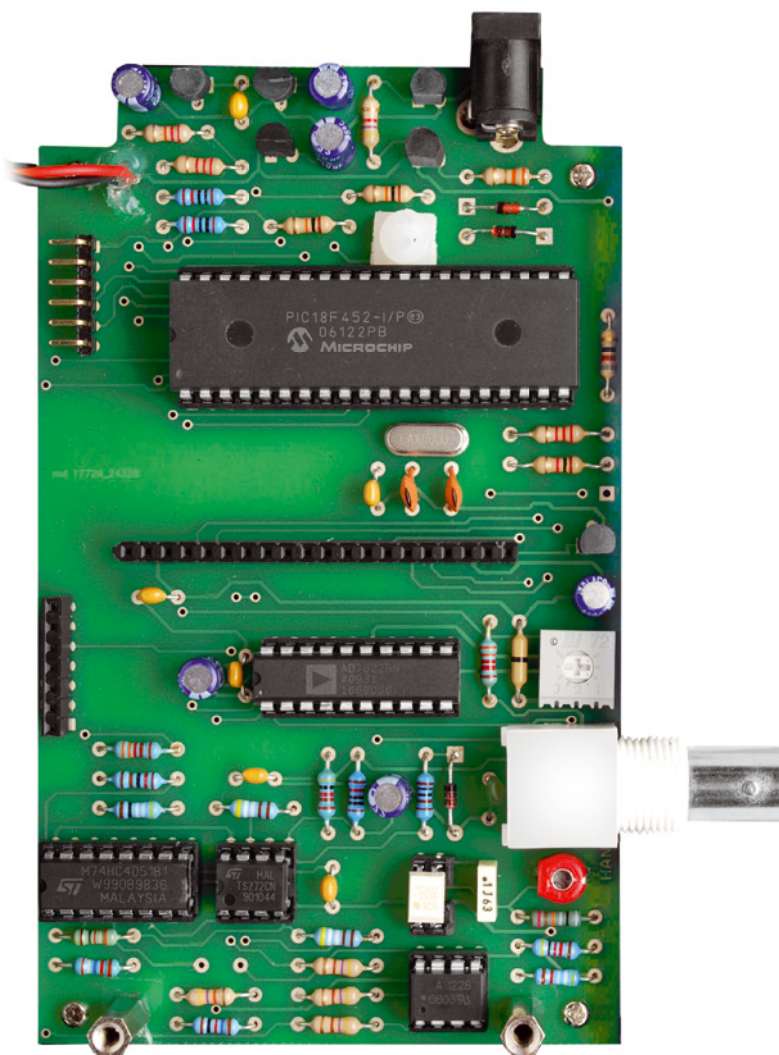
SCHEMA ELETTRICO

Lo schema elettrico è essenzialmente diviso in tre sezioni: la parte analogica adibita al condizionamento del segnale in ingresso, una parte digitale con la conversione analogico-digitale ed uno stadio di

alimentazione. L'insieme si trova su due stampati: uno (*base*) contiene l'elettronica, ossia i blocchi appena descritti, mentre l'altro realizza la *tastiera*: l'insieme è visibile nelle immagini. Il segnale di ingresso viene prelevato tramite un connettore BNC da circuito stampato e giunge allo stadio di



[schema elettrico TASTIERA]



Elenco Componenti:

R1: 470 ohm
 R2: 909 kohm 1%
 R3: 40,2 kohm 1%
 R4: 49,9 kohm 1%
 R5: 47 kohm
 R6, R7: 470 ohm
 R8: 162 1%
 R9: 392 1%
 R10: 19,6 kohm 1%
 R11: 499 ohm 1%
 R12: 1,2 kohm 1%
 R13: 3,3 kohm 1%
 R14: 4,7 kohm
 R15: 22 kohm
 R16, R17: 10 kohm 1%
 R18: 0 ohm
 R19: 10 kohm
 R20: 4,99 kohm 1%
 R21: 4,02 kohm 1%
 R22: 10 kohm
 R23: 4,99 kohm
 R24: 33 kohm
 R25: 1 kohm
 R26: 2,2 kohm
 R27: 3,3 ohm
 R28: 4,7 kohm
 R29: 47 kohm
 R30: 82 ohm
 R31: 1 kohm
 R32: Trimmer 20 kohm
 C1, C5, C8: 100 nF multistrato
 C11, C15, C17: 100 nF multistrato
 C2: Compensatore 4-20 pF
 C3, C4, C6: 10 μ F 16 VL elettrolitico

accoppiamento AC/DC gestito, in questo caso, da un relé allo stato solido controllato dalla linea digitale AC/DC del microcontrollore. Segue uno stadio di attenuazione gestito da un secondo relé allo stato solido, comandato dalle linee E e D del microcontrollore. I relé a stato solido sono costituiti essenzialmente da un MOSFET che va a sostituire il contatto meccanico dei normali relé; ciò presenta il vantaggio di ridurre l'ingombro. Il comando di chiusura del contatto viene dato tramite un fotodiodo emettitore, consentendo un facile interfacciamento con un microcontrollore. Lo stadio

di amplificazione del circuito è affidato ad un doppio operativo (TS272) in abbinamento ad un interruttore analogico in tecnologia CMOS (74HC4051) comandato dalle linee A, B e C, che definiscono le varie portate in tensione. Il secondo operativo dell'integrato TS272, è utilizzato per gestire il livello di offset del segnale, che deve giungere al convertitore ADC in modo unipolare e traslato rispetto il livello d'ingresso. Per la conversione abbiamo utilizzato un ADC Flash della Analog Devices di tipo AD7822, capace di conversioni sino a 2 Msps (milioni di campionamenti al secondo).

Le linee dati del convertitore sono utilizzate anche dal display grafico LCD, che ha una risoluzione di 128x64 pixel. Per ridurre "all'osso" l'uso di componenti, la tensione negativa necessaria allo stadio analogico di amplificazione è stata prelevata dal pin VEE del



C13, C14, C18: 10 μ F 16 VL elettrolitico

C7: non usato

C9, C10: 22 pF ceramico

C16: 150 pF ceramico

C18: 680 pF ceramico

(Saldare direttamente sulla piazzole di R4)

C19: 100 pF ceramico

(Saldare direttamente sulla piazzole di R20)

U1: PIC18F452-I/P (MF951)

U2: Display grafico GDM12864HLCM

U3: AD7822BNZ

U4: TS272

U5: MOSFET relay G3VM-351

U6: MOSFET relay ASSR-1228

U7: 74HC4051

U8: MCP1702-5

U9: LP2950-5

Q1: Quarzo 10 MHz

D1÷D3: 1N4148

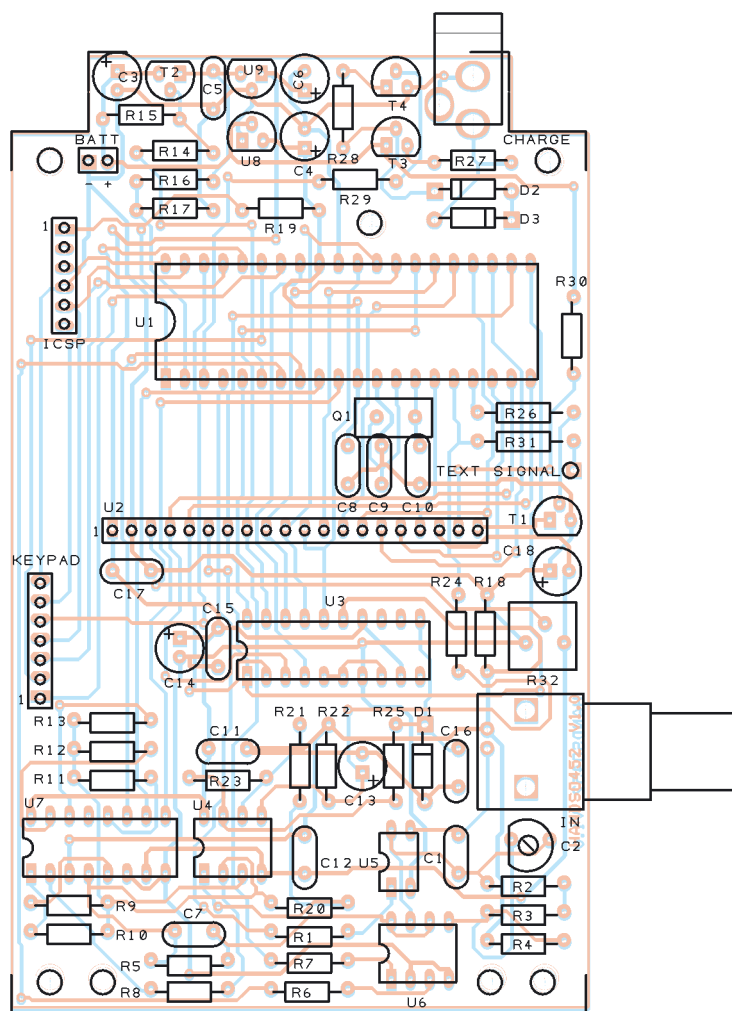
T1: BC337

T2, T3: BC327

T4: BC547

Varie:

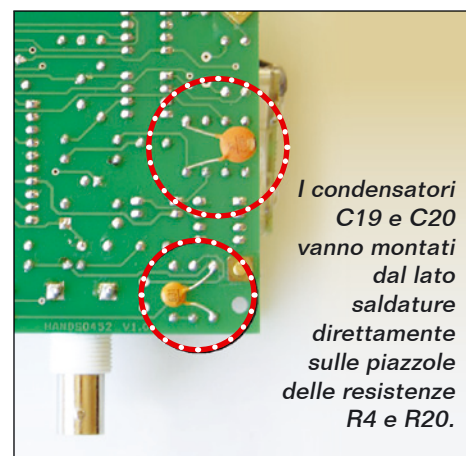
- Zoccolo 2+2
- Zoccolo 4+4 (2 pz.)
- Zoccolo 8+8
- Zoccolo 10+10
- Zoccolo 20+20
- BNC da cs
- Plug alimentazione
- Strip maschio 6 poli
- Strip femmina 7 poli
- Strip maschio 20 poli
- Strip femmina 20 poli
- Circuito stampato



display, che la genera internamente per gestire il contrasto delle immagini. Il transistor T1 viene comandato dal microcontrollore ed è utilizzato per la gestione della retroilluminazione del display. Per l'alimentazione abbiamo previsto l'utilizzo di una batteria agli ioni di litio da 8,4 volt e capacità di 350 mA/h, che, grazie alle sue ridotte dimensioni, trova posto all'interno del contenitore. I transistor T3 e T4 formano il circuito di ricarica a corrente costante, gestito interamente dal micro, il quale rileva la tensione ai capi della batteria tramite il partitore formato dalle resistenze R16 e R17. Invece di

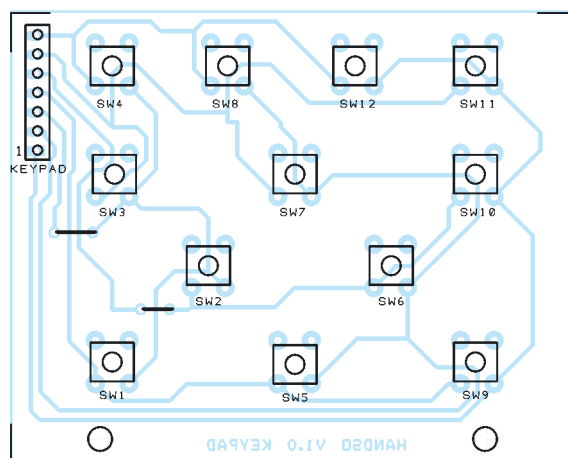
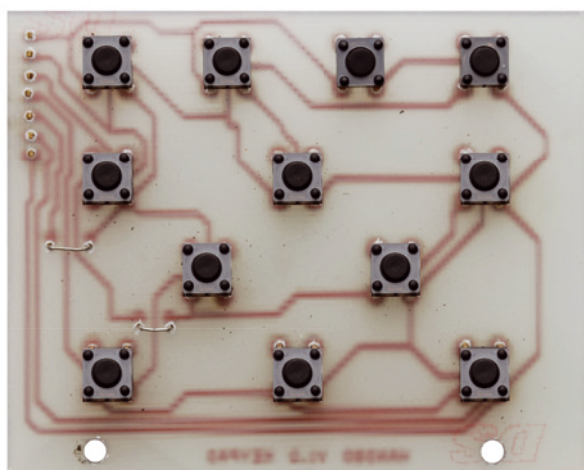
utilizzare un classico interruttore, per controllare l'alimentazione abbiamo preferito una soluzione più sofisticata con gestione via software, uniformandoci all'uso comune che vuole, in quasi tutte le apparecchiature alimentate a batteria, la funzione auto-power-OFF, ovvero la possibilità di spegnere automaticamente il sistema dopo un certo tempo di inattività. Questa funzione è ormai da tempo adottata in cellulari, lettori MP3 e multimetri digitali ed ha lo scopo di evitare la scarica completa della batteria in caso di inutilizzo; evitare la completa scarica, è vitale per batterie come quelle al litio, che al disotto di

un certo livello di tensione si danneggiano. L'idea è quella di lasciare sempre



I condensatori C19 e C20 vanno montati dal lato saldature sulle piazzole delle resistenze R4 e R20.

[piano di montaggio **TASTIERA**]



Elenco Componenti:

SW1, SW2: Microswitch
SW3, SW4: Microswitch
SW5, SW6: Microswitch

SW7, SW8: Microswitch
SW9, SW10: Microswitch
SW11, SW12: Microswitch

Varie:
- Strip maschio 7 poli
- Circuito stampato

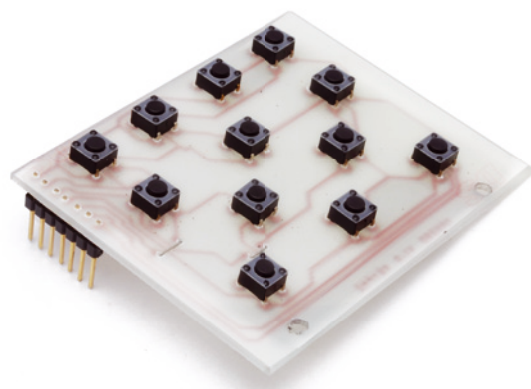
alimentato il PIC, mentre il resto del circuito viene messo sotto tensione solo quando serve. Per fare questo sono necessari due accorgimenti: il primo consiste nel far funzionare, a circuito spento, il PIC in modalità di sleep, mentre il secondo prevede di utilizzare uno stabilizzatore di tensione con bassa corrente di funzionamento (Low Quiescent Current). Il microcontrollore dovrà essere in grado di comandare l'accensione e lo spegnimento di tutti gli altri componenti dell'oscilloscopio. Nel nostro circuito l'integrato U6 (MCP1702) è appunto uno stabilizzatore di tensione che richiede, per funzio-

nare, una corrente di soli 2 μ A ed alimenta il PIC, che, in modalità *sleep*, assorbe solo pochi microampere, quindi questi due componenti sono sempre alimentati. Il rimanente circuito è alimentato tramite U9, il quale riceve tensione solo quando il transistor T2 viene polarizzato dal comando del PIC. Ovviamente la gestione di T2 e della modalità *sleep* avvengono a livello software, mediante una specifica funzione di *interrupt* in grado di rilevare la pressione del pulsante di accensione anche quando il circuito è nello stato di *Power-down*.

REALIZZAZIONE PRATICA

La costruzione dell'oscilloscopio non presenta grosse difficoltà, ma vista la complessità del circuito è necessaria la massima diligenza. I file relativi alle tracce (il c.s. è a doppia faccia) per costruire lo stampato sono disponibili per il download sul nostro sito www.elettronica.in.it. Un secondo stampato, a singola faccia, è necessario per realizzare la tastiera. Tutti i componenti

sono di facile reperibilità tranne i relé a stato solido, che sono un po' particolari, ma eventualmente sostituibili con interruttori classici; ad esempio al posto di U5 potete inserire un interruttore per la funzione AC/DC che, evidentemente, non sarà più gestibile a livello software. U6 potrà essere sostituito con un deviatore. I condensatori C19 e C20 (ceramici) vanno saldati direttamente sui pin delle relative resistenze sul lato opposto a quello dei componenti. Utilizzate le foto riportate in queste pagine e il disegno di disposizione dei componenti, per vedere come posizionare i singoli elementi. Completato il montaggio, non vi sono da fare particolari tarature, se non la regolazione del trimmer R32 per definire il contrasto dell'LCD ed eventualmente la regolazione del valore di C2 per ottimizzare la banda passante dello stadio analogico. Il circuito è predisposto per essere inserito in un contenitore della Teko (codice Teko 660F.9) che consente di ottenere un aspetto



finale molto soddisfacente; in alternativa, per chi utilizza delle batterie più voluminose, esiste il modello 660.5 (più spesso). Il contenitore va sagomato rispettando le dimensioni della presa di alimentazione e del connettore BNC; la parte superiore dovrà presentare una finestrella in corrispondenza del display ed alcuni fori per l'accesso ai pulsanti della tastiera. La batteria (ai polimeri di litio) va inserita tra lo stampato principale e la tastiera. Dopo aver sagomato adeguatamente il coperchio, potete usare il file *HANDSO452_maschera_color.doc* per stampare la mascherina frontale in scala 1:1. Il circuito di ricarica è sempre attivo e quindi non appena inserirete una fonte di alimentazione esterna la batteria verrà ricaricata; per il corretto funzionamento utilizzate un piccolo alimentatore non stabilizzato da 500 mA, impostato su una tensione di 9 volt. Quando la batteria sarà completamente carica, il circuito di ricarica sarà disabilitato (per evitare

Listato 1

```
RunFast:
    MOVLW 0x03          ' Load 3 to Working
    LFSR FSR0, 0x200    ' Load to FSR the address pointer of bank 2 (200-2FF)

LoopFast:
    BCF PORTB,0 ,0      ' Sampling procedure 1Ms/sec 256 samples
    BSF PORTB,0 ,0      ' Go sampling (1 cycle)
    nop                 ' (1 cycle)
    nop                 ' (1 cycle)
    nop                 ' (1 cycle)
    nop                 ' End conversion (1 cycle)
    MOVFF PORTD, POSTINC0 ' Save ADC value to POSTINC0 register (2 cycles)
    CPFSEQ FSR0H, 0      ' Compare FSRH with 3 (W) Jump if equal (1 cycle if not jump)
    bra LoopFast         ' (2 cycles if jump)
    nop
    goto EndFast        ' Bank 2 full (256 sampling) then exit
```

dannose sovraccariche). Notate che il firmware del microcontrollore è ottimizzato per gestire due celle LiPo con tensione nominale di 8,4 volt; la corrente di carica è di circa 180 mA. Sulla sinistra del display è previsto il connettore per la programmazione, compatibile con il programmatore PIC Kit2 della Microchip.

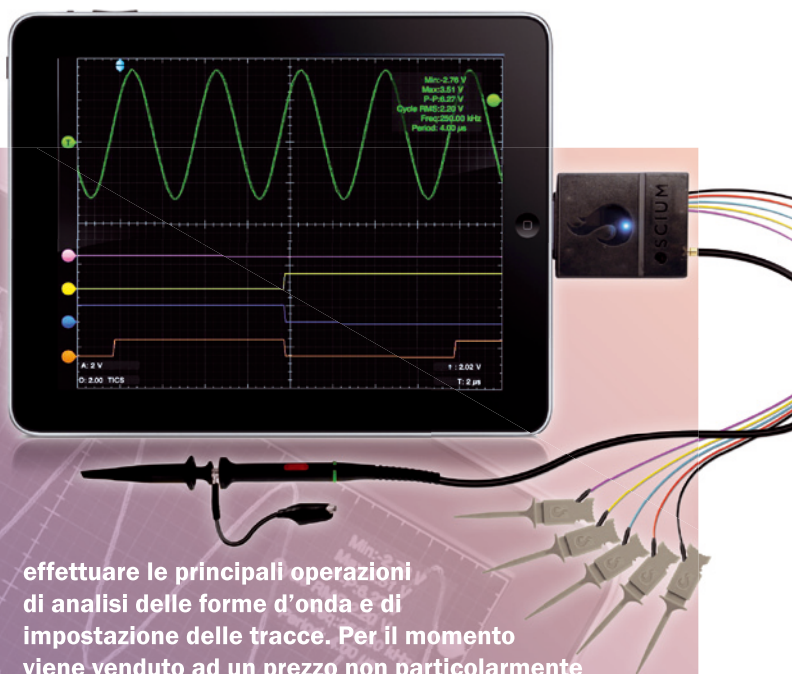
IL FIRMWARE

Descrivere tutto il firmware sarebbe lungo, ma possiamo farlo

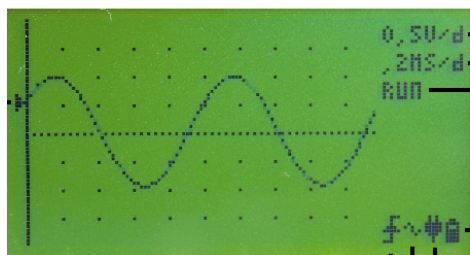
nelle linee essenziali: la maggiore difficoltà è ottenere un'elevata frequenza di campionamento senza ricorrere a circuiti dedicati. Il PIC è utilizzato alla sua massima frequenza di clock (48 MHz) però per eseguire un'istruzione sono necessari 4 cicli di clock, il che significa che una semplice istruzione in assembler richiede 83 ns, mentre un'istruzione di salto ne richiede 166 ns. La parte essenziale del firmware, scritta in assembler, deve permettere

L'iPad diventa oscilloscopio

Arriva da Oscium (www.oscium.com) e si chiama iMSO-104: è un eccezionale accessorio per iPad e iPhone che permette di trasformare questi dispositivi della "mela" in veri e propri oscilloscopi. Il cuore dello strumento è un PSoc 3 della Cypress Semiconductor il quale gestisce la comunicazione bidirezionale tra l'iMSO-104 e l'iPad (quest'ultimo è l'ideale per via dell'ampio schermo a colori e la funzionalità touch-screen) o iPhone tramite il connettore dock, oltre alla digitalizzazione dei segnali rilevati dalla sonda. Per utilizzarlo, dopo averlo connesso bisogna scaricare l'applicazione software (gratuitamente disponibile su App Store): si tratta di un'App che permette di

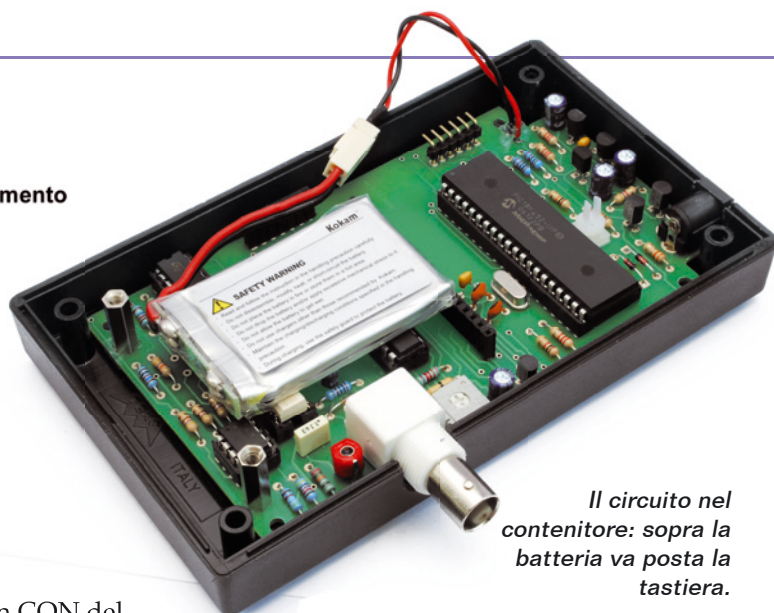


effettuare le principali operazioni di analisi delle forme d'onda e di impostazione delle tracce. Per il momento viene venduto ad un prezzo non particolarmente abbordabile (si parla di 300 dollari U.S.A.) ma non è escluso che col tempo divenga più alla portata dell'hobbista.

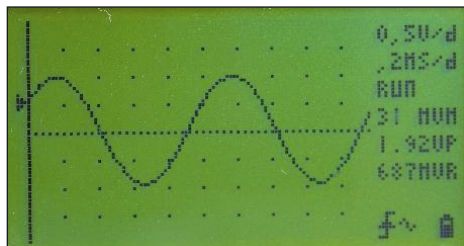


Il visualizzatore.

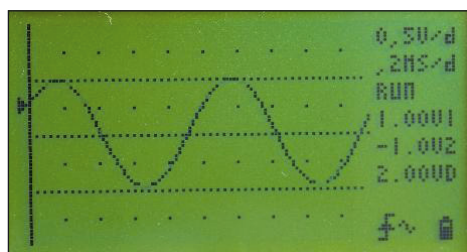
Volt division
Time division
Modo di funzionamento
Livello batteria
Alimentazione esterna
Accoppiamento
Pendenza trigger



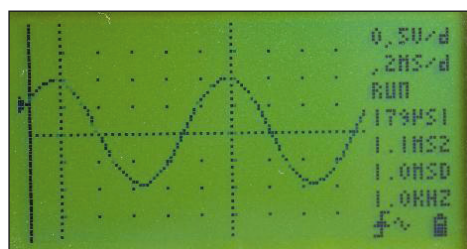
Il circuito nel contenitore: sopra la batteria va posta la tastiera.



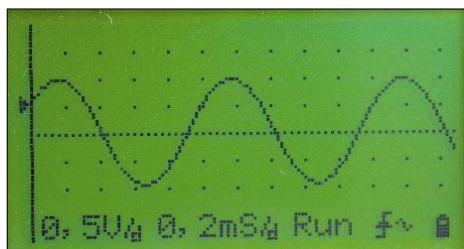
Misure automatiche.



Misure di tensione manuali.



Misure di tempi manuali.



Modalità di visualizzazione in orizzontale.

di gestire la linea CON del convertitore, acquisire il dato dal convertitore, salvarlo in memoria e verificare quando il buffer di acquisizione sia completo. La parte di programma che esegue queste operazioni è riportata nel **Listato 1**.

Da esso vedete che sono necessarie otto istruzioni da un ciclo macchina e due da due cicli, per un totale di 12 cicli macchina; lavorando a 12 milioni di cicli macchina al secondo, si ottiene una massima frequenza di campionamento di 1 Msps. Prima di iniziare il campionamento è però necessario rilevare il livello e la pendenza del trigger, operazione che viene fatta, sempre a livello software, analizzando il valore di due campioni consecutivi in un tempo massimo predefinito. Per ottenere tutte le altre frequenze di campionamento, essendo più basse, è sufficiente aggiungere dei ritardi calibrati. La struttura generale del firmware prevede i seguenti punti:

1. impostazione dei parametri di campionamento;
2. ricerca del trigger;
3. campionamento;
4. eventuali misure sul segnale (se richiesto);
5. scalatura dei campioni secondo risoluzione del display;
6. visualizzazione dei campioni;
7. visualizzazione della configurazione.

In qualsiasi istante venga premuto un pulsante della tastiera sarà generata un'interruzione che provvederà a rilevare quale pulsante è stato premuto e successivamente eseguire l'operazione richiesta. Nel caso venga premuto il pulsante di spegnimento, sarà attivata una procedura particolare che provvede a:

1. disattivare l'alimentazione degli integrati;
2. salvare lo stato attuale delle variabili;
3. porre a zero tutte le uscite;
4. attivare la funzione sleep del PIC.

Nello stato di sleep, premendo il pulsante di accensione viene generata nuovamente un'interruzione che provvede a:

1. ripristinare lo stato delle variabili;
2. ripristinare lo stato delle uscite;
3. abilitare l'alimentazione degli integrati.

Questo sistema permette, dopo una nuova accensione, di ripartire nelle esatte condizioni precedenti lo spegnimento, senza dover reimpostare tutti i parametri di cattura. C'è da notare che quando l'oscilloscopio è spento è necessario disattivare tutte le

uscite, per evitare inutili sprechi di corrente, quindi il circuito di ricarica sarà disattivato, impedendo di fatto la ricarica delle batterie, funzione necessaria in quanto il PIC in modalità sleep non sarebbe comunque in grado di verificare lo stato della carica della batteria. Se per qualche motivo la batteria fosse sotto la minima carica utile, l'oscilloscopio non si accenderebbe e non sarebbe possibile ricaricarla. In modalità di visualizzazione orizzontale vengono usati i classici caratteri 8x5 pixel, standard per i display LCD di testo, mentre nella modalità di visualizzazione verticale abbiamo ridefinito i caratteri utilizzando una risoluzione di 6x3 pixel che permette una maggiore compattezza nella visualizzazione ma sacrifica leggermente la visibilità. Ad esempio il carattere "m" dei *milli* è sostituito dal carattere "M" in quanto è l'unico rappresentabile con 6x3 pixel. La ridefinizione dei caratteri ci ha permesso di creare anche tutti i caratteri speciali relativi ai simboli specifici presenti negli oscilloscopi.

UTILIZZO PRATICO

All'accensione viene mostrato il nome e la versione del firmware, poi l'oscilloscopio riprende a funzionare con le esatte impostazioni precedenti lo spegnimento. Il funzionamento è quello classico che si riscontra in oscilloscopi di simili caratteristiche; l'uso di un display a bassa risoluzione ci obbliga ad un compromesso tra l'area in cui è visualizzato il segnale e l'area utilizzata per i parametri di funzionamento. Possono essere utilizzate due modalità di visualizzazione verticale o orizzontale, a seconda di quali dati si vogliono mettere in evidenza. Vediamo ora le funzioni dei tasti dell'oscilloscopio:

- *ON/OFF* permette l'accensione e lo spegnimento dello strumento;
- i tasti freccia consentono, a seconda della funzione abilitata, la modifica dei parametri oppure lo spostamento dei marker per le misure in manuale tramite cursori;
- *Volt/div* abilita la modifica del valore relativo alla portata in tensione tramite i due pulsanti freccia in su e in giù;
- *Time/div* abilita la modifica del valore della base dei tempi tramite i due pulsanti freccia su e giù.

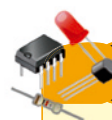
Alla prima pressione del tasto *Trigger* è possibile selezionare la pendenza, mentre con una seconda pressione si regola il livello; quest'ultimo è mostrato tramite una piccola freccia sulla sinistra del grafico, che diventa una linea quando si sta impostando il livello. Alla prima pressione del tasto *Measure* sono visualizzate le misure automatiche del valore medio (M), valore di picco (P) e valore efficace (R) del segnale. Una seconda pressione attiverà i marker per la lettura manuale dei valori di tensione. Una terza pressione attiverà i marker per la lettura manuale del valore dei tempi. Per entrambe le funzioni sarà visualizzato il valore assoluto dei due marker più il valore differenza. I due tasti freccia su e giù imposteranno il marker di destra, mentre i tasti freccia destra e sinistra imposteranno il marker di sinistra. Per visualizzare tutte le informazioni sul nostro piccolo display abbiamo dovuto accettare alcuni compromessi.

Vediamo ora il tasto *mode*: normalmente l'oscilloscopio funziona in modalità RUN, appena agganciato il trigger inizia l'ac-

quisizione; alla prima pressione del tasto si passa alla modalità STOP, bloccando sullo schermo l'ultima sequenza campionata. Premendo nuovamente il tasto *mode* si entra nella modalità di singola acquisizione (segnalata dalla scritta TRG?) e l'oscilloscopio si pone in attesa del segnale di trigger per eseguire il campionamento, dopodiché passa in modalità di STOP. Questa funzione è utile per catturare singoli eventi.

Il tasto *Setup* permette di entrare nel menu di modifica, dove è possibile impostare il tipo di visualizzazione orizzontale o verticale, l'intensità della retroilluminazione e il tempo per l'autospegnimento.

Il tasto *AC/DC/GND* attiva in sequenza le omonime funzioni per quanto riguarda l'accoppiamento del segnale in ingresso. Quando si è in modalità GND è anche possibile modificare il livello di offset del segnale per compensare eventuali errori interni allo strumento; due piccole frecce sono mostrate in corrispondenza della linea di livello.



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista così come il firmware utilizzato per programmare il microcontrollore PIC18F452-I/P; quest'ultimo è anche disponibile già programmato al prezzo di 20,00 Euro, IVA compresa.

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

PUNTATA 5

Corso di programmazione iPhone

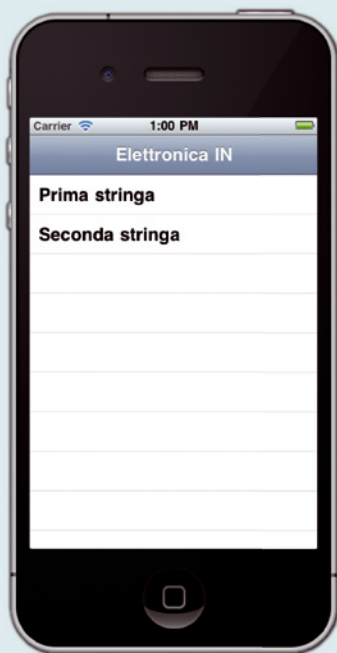


In queste pagine partiremo usando il progetto di tipo "Navigation Based Application" descritto nella scorsa puntata, per spiegarvi come gestire la navigazione all'interno dei contenuti. Poi faremo una veloce incursione nel mondo dei pickers per la selezione di date, orari o nostri contenuti e concluderemo con un nuovo progetto pratico dove andremo ad utilizzare una TabBar e le collegheremo un Navigation Controller per la navigazione di ogni

singolo contenuto selezionato dalla tab. Per prima cosa occupiamoci di spiegare cos'è e come si usa la Navigation Bar, la quale è uno strumento utilissimo nelle applicazioni sviluppate per iPhone e iPad; allo scopo, dobbiamo "riesumare" i concetti esposti nella quarta puntata e in particolare il progetto "Navigation Based Application", perché questo ingloba la Table View. In tale progetto non abbiamo creato direttamente la Table View in quanto,

Scopriamo come possiamo utilizzare la Navigation Bar per gestire la navigazione dei contenuti e impariamo ad usare i pickers, utili per selezionare date, orari o contenuti. Vediamo anche un progetto pratico basato sull'uso di TabBar e Navigation Controller.

Fig. 1 - Il titolo nella Navigation Bar.



per la navigazione dei contenuti, si è dimostrata veramente utile e sufficiente la Navigation Bar.

Navigation Bar

Questo componente serve, ad esempio, per introdurre delle viste nei contenuti da navigare; se non lo utilizziamo, quando vogliamo aggiungere una vista dobbiamo farlo sfruttando il metodo "addSubview" (come è stato, pur sommariamente, spiegato nelle precedenti puntate) e dovremmo anche fare in modo da rimuovere le viste stesse

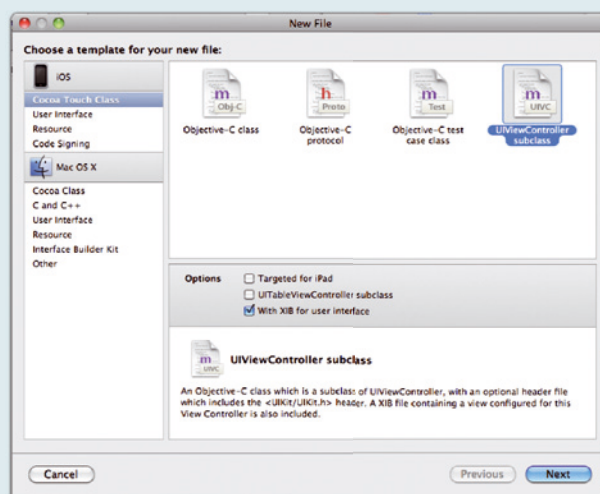
per ritornare nelle precedenti, utilizzando il metodo "removeFromSuperview". Lavorando con la Navigation Bar (NavBar) l'aggiunta e la rimozione vengono gestite al suo interno, il che ci garantisce una programmazione semplice e veloce.

Riapriamo ora il nostro vecchio progetto "TableViewExample" e iniziamo ad aggiungere le informazioni per riempire la nostra barra di navigazione; per prima cosa, nel metodo "viewDidLoad" andiamo ad aggiungere un titolo al nostro controller. Questa tecnica viene molto utilizzata per dare un nome ai ViewController; per applicarla, agiremo sulla proprietà "title", come mostrato nel **Listato 1**.

Se ora avviamo la nostra applicazione, vediamo che il titolo viene automaticamente inserito all'interno della barra di navigazione (**Fig. 1**).

A questo punto siamo pronti per andare a "spingere" i nostri controller personalizzati all'interno della Navigation Bar; allo scopo dobbiamo creare un nuovo controller che rappresenterà il dettaglio

Fig. 2 - Aggiungere un controller al nostro progetto.



della selezione di una riga della tabella. Per creare questo componente facciamo MELAN sulla cartella "Classes" del nostro progetto e selezioniamo "Add" e poi "New File...". Quando si apre la finestra di selezione, indichiamo che vogliamo creare un nuovo UIViewController (**Fig. 2**) facendo attenzione a selezionare la voce riguardante la creazione della vista associata. Chiamiamo il nostro controller "DetailViewController". Una volta generato il codice, Xcode mette i tre file del nostro nuovo controller all'interno della nostra folder "Classes"; per ottenere un progetto "pulito", spostiamo la vista, quindi il file ".xib", nella cartella "Resources" (**Fig. 3**).

All'interno di questo nuovo controller dobbiamo creare due elementi, il primo di tipo "NSString" (è il nostro modello di dati) ed un altro di tipo "UILabel", che serve a collegarlo, tramite l'Interface Builder, al codice e poi far agire il nostro controller dai dati alla visualizzazione.

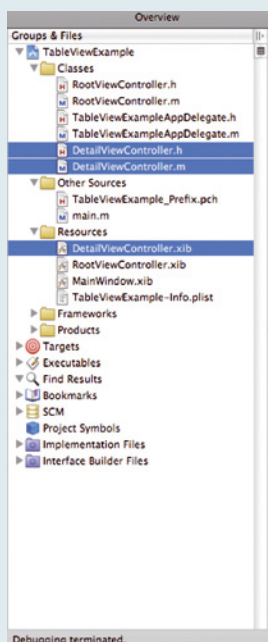
Listato 1 - Assegnare il titolo al controller.

```
- (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;

    self.title = @"Elettronica IN";
    //Resto dell'implementazione...
```

Fig. 3 - Situazione del nostro progetto.



Per collegare il modello serve solo un semplice collegamento nel metodo “viewDidLoad”, come mostra il **Listato 3**.

Adesso il controller è pronto; non ci resta che ritornare a lavorare sulla TableView, in modo da unire il tutto sotto un'unica bandiera; in pratica allocheremo il controller creato e diremo alla Navigation Bar di renderlo disponibile agli utenti. Sarà nostra premura andare ad informare il controller del dettaglio di tutte le informazioni che devono essere visualizzate; allo scopo passeremo il modello di dati al controller di dettaglio e questo si occuperà di passare alla sua vista i dati da visualizzare (**Listato 3**). Dunque dobbiamo aprire il nostro “RootViewController”, ovvero la TableView e qui

Il procedimento è identico a quello dell'HelloWorld che abbiamo già visto nelle precedenti puntate. Nella vista aggiungiamo la label ed agganciamola al codice tramite la direttiva IBOutlet; a questo punto eseguiamo il collegamento visuale. In aggiunta utilizziamo un oggetto di tipo “NSString”; ora il nostro file “.h” dovrebbe risultare come il **Listato 2**.

Ricordiamo che il compito di questo controller è accedere alla vista passando i dati da visualizzare; in questo caso abbiamo il modello dei dati (oggetto NSString) e l'etichetta che li rappresenta (UILabel).

Listato 2 - L'intestazione del controller di visualizzazione.

```
//DetailViewController.h
#import <UIKit/UIKit.h>
@interface DetailViewController : UIViewController {
    UILabel *etichetta;
    NSString *dato;
}

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

@end
```

cercare l'implementazione del metodo “didSelectRowAtIndexPath”; ciò perché, come dice il nome del metodo, si tratta di un'azione che viene eseguita quando un utente seleziona una riga della nostra tabella.

All'interno di questo metodo c'è una parte di codice che andremo a riutilizzare completamente, in quanto riguarda proprio il funzionamento incrociato di navigazione e visualizzazione. Nel **Listato 4** vediamo il codice necessario per il funzionamento completo della NavBar, dove notiamo che è stato semplicemente rimosso il commento multilinea. Alla riga 2 stiamo allocando il controller del dettaglio sulla selezione dell'utente, informando il compilatore che la sua vista deve essere elaborata tramite il “.xib” file chiamato “DetailViewController.xib”.

Alla riga 3, invece, stiamo lavorando con la

Listato 3 - Assegnare alla vista i dati del modello.

```
//DetailViewController.h
- (void)viewDidLoad {
    [super viewDidLoad];
    self.etichetta.text = dato;
}
```

Listato 4 - Utilizzare la navigazione.

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    DetailViewController *detailViewController = [[DetailViewController alloc] initWithNibName:@"DetailViewController" bundle:nil];
    // ...
    // Pass the selected object to the new view controller.
    [self.navigationController pushViewController:detailViewController animated:YES];
    [detailViewController release];
}
```

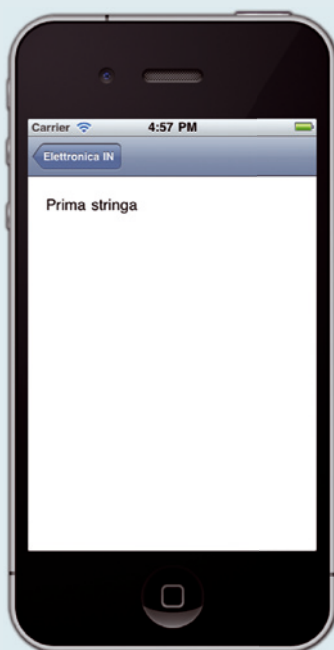
Listato 5 - Modificare il metodo per passare i dati al controller di dettaglio.

```

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    NSString *stringa = (NSString *)[self.dati objectAtIndex:indexPath.row];

    DetailViewController *detailViewController = [[DetailViewController alloc] initWithNibName:@"DetailView
Controller" bundle:nil];
    detailViewController.dato = stringa;
    // ...
    // Pass the selected object to the new view controller.
    [self.navigationController pushViewController:detailViewController animated:YES];
    [detailViewController release];
}

```

Fig. 4 - Vista del dettaglio inserita dal Navigation Controller.

barra di navigazione. In ogni controller è presente la proprietà "navigationController" che riguarda proprio la Navigation Bar; per "spingere" (push) un nuovo controller, dobbiamo agire su di essa. Successivamente vediamo la direttiva che rilascia il controller che abbiamo inserito nello stack di visualizzazione; ciò perché la Navigation Bar esegue una "retain" su tale controller, ovvero esegue la "release" quando, invece, rimuoveremo il controller dalla pila. La "release" garantisce il riciclo di memoria del componente allocato.

Se ora avviamo il nostro progetto e facciamo clic su una riga, vediamo che viene inserita la nostra vista di dettaglio con un'animazione. Modifichiamo ancora il metodo di selezione

della riga andando ad agire sulla proprietà "etichetta", facendo comparire il valore della riga selezionata; per fare questo dobbiamo prendere dall'array utilizzato per la table view il modello e assegnare la stringa all'etichetta del "DetailViewController" (**Listato 5**).

A questo punto siamo pronti per avviare il nostro progetto: quando il simulatore ha caricato il sistema, facciamo clic su una delle due righe che abbiamo settato e vedremo che il Navigation Controller si occuperà, tramite una transizione, di rendere disponibile la vista di dettaglio.

Questa utilizzerà il modello dei dati per visualizzare il testo selezionato (**Fig. 4**). Se analizziamo questa vista, notiamo che il Navigation Controller ha aggiunto un pulsante (sulla sua barra) che ci permette di ritornare alla vista precedente senza dover assolutamente aggiungere una sola riga di codice. Per rendere il tutto perfetto dobbiamo aprire il file del dettaglio ed aggiungere il titolo tramite la proprietà "title" del controller in questione, come fatto nel **Listato 6**.

A questo punto abbiamo terminato la stesura del nostro progetto iniziato nella scorsa puntata; ricordate che dovete sempre rilasciare la memoria anche nel dettaglio, perché il Naviga-

Listato 6 - Aggiungere il titolo al controller di dettaglio.

```

- (void)viewDidLoad {
    [super viewDidLoad];

    self.title = @"Dettaglio";

    self.etichetta.text = dato;
}

```

Listato 7 - Liberare la memoria nel controller di dettaglio.

```

- (void)dealloc {
    dato = nil;
    [etichetta release];
    [super dealloc];
}

```


tion Controller utilizzerà la “release” quando facciamo clic sul pulsante per ritornare al controller precedente. Il **Listato 7** mostra come liberare tutte le risorse allocate.

Progetto pratico con il TabBar controller

Il tab bar controller è un altro componente fondamentale delle applicazioni iOS: il suo principale obiettivo è permettere la navigazione indiretta tra i contenuti. Esso si dispone graficamente nella parte bassa della nostra applicazione e permette all'utente di selezionare quale vista visualizzare, passando da una all'altra senza perdere la navigazione eseguita. In **Fig. 5** possiamo vedere un tipico esempio di App strutturata con TabBar, che eseguiamo tra poco.

Come potete vedere, la barra di navigazione indiretta è posizionata in basso e permette la navigazione tra due diversi controller/viste. Per realizzare questo nuovo progetto avviamo il nostro Xcode e iniziamone uno di tipo “Tab Bar Application”; quando ci viene richiesto, impostiamo come nome “TabBarExample”. Una volta creato il progetto, possiamo compilare ed eseguire il codice per avere un risultato simile a quello mostrato nella **Fig. 5**.

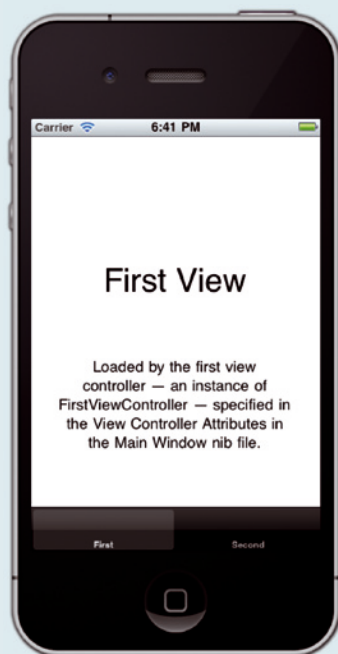


Fig. 5 - Esempio di TabBar application.

Adesso analizziamo il codice che è stato generato: apriamo il nostro delegato dell'applicazione, quindi il file “TabBarExampleAppDelegate”; vediamo che qui è stato utilizzato il controller di tipo “UITabBarController”.

La configurazione di questo componente non è immediata ed esula dagli scopi di questa puntata del corso; è più interessante costruire il nostro nuovo progetto per integrarvi la NavigationBar mediante

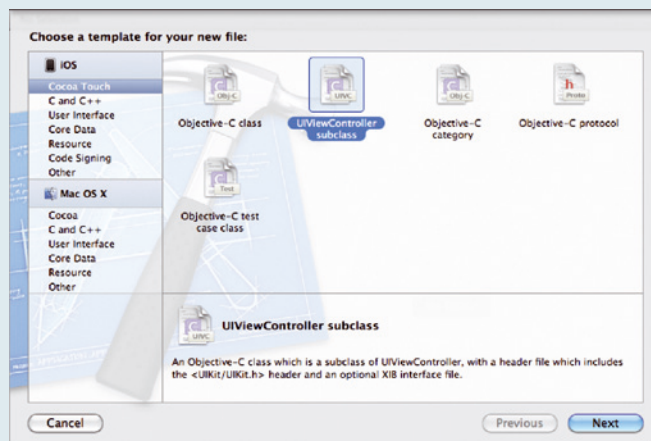


Fig. 6 - Generare un UITableViewController.

un controller di tipo UITableViewController ed un secondo di tipo UIViewController. Questa configurazione, che andremo a realizzare, è infatti il tipico standard delle App gestionali che trovate nell'App store. Per realizzare questo set dobbiamo smontare la parte visuale dell'applicazione che abbiamo creato (tramite il generatore di Xcode) e ricomporla con tutti i componenti di cui abbiamo bisogno.

Ora dobbiamo creare un nuovo controller di tipo UITableViewController; allo scopo facciamo MELA+N sulla folder “Classes” e generiamo una classe di tipo UIViewController (**Fig. 6**); successivamente indichiamo che quest'ultima deve essere una sottoclasse di UITableViewController e che vogliamo collegare una vista in XIB.

I pickers

I pickers sono dei particolari componenti grafici che permettono la selezione, da parte dell'utente, di un set pre-organizzato di valori. Esistono due tipologie di tali componenti: il “Picker View” ed il “Date Picker”; il primo si occupa di far selezionare dei dati completamente gestiti dal programma, mentre il secondo serve prevalentemente per le date. Per l'utilizzo del primo componente si deve implementare due interfacce: la prima per la gestione dei dati ed una seconda, invece, per il delegato dell'applicazione e quindi per l'interazione utente-dispositivo. Il secondo tipo, invece, richiede l'implementazione di una sola interfaccia.

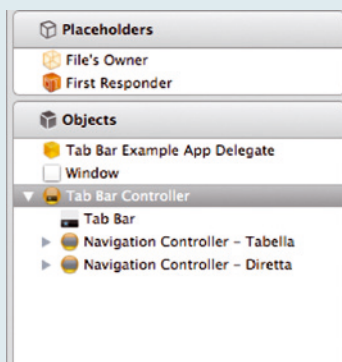


Fig. 7 - Il progetto del TabBar in InterfaceBuilder.

A questo punto, se aprite la vista associata a tale controller potete vedere che il risultato è proprio quello che volevamo. Ora tocca al tipo `UIViewController`, che dobbiamo generare come fatto per il `UITableView` Controller, ovvero rimuovendo la spunta di tipo *Table*, (vedere in Fig. 6); siccome quando abbiamo generato il nostro progetto, Xcode ci ha già regalato la vista classica (il relativo file ha il nome standard "FirstViewController") possiamo evitare di crearne una seconda e riutilizzare l'oggetto già generato da Xcode. Se volete potete comunque generare tutti i controller di cui avete bisogno ed inserirli nella TabBar. Facciamo clic su `MainWindow.xib` per aprire XIB e vedere la struttura di questa vista principale: nella TabBar sono presenti le due View. Il

nostro obbiettivo sarà scomporre questo progetto per arrivare alla situazione visibile in Fig. 7.

Nel nostro "inspector" dedicato ai componenti visuali, è stato inserito dal generatore di codice anche il controller di tipo TabBar; tutti i controller che andremo ad aggiungere verranno inseriti qui.

Per prima cosa rimuoviamo entrambe le viste; allo scopo, selezioniamole dal box e premiamo il tasto Canc della tastiera. Una volta rimosse, iniziamo ad aggiungere il primo

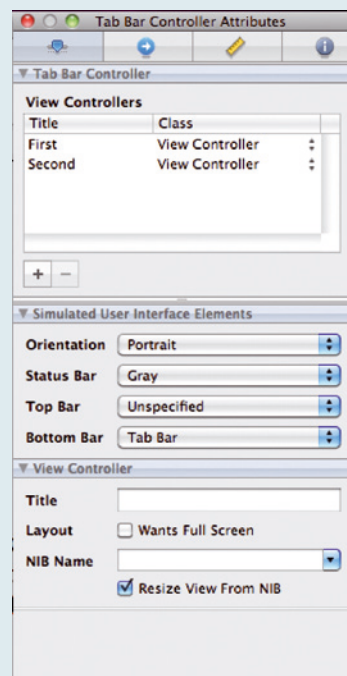


Fig. 8 - Aggiunta del controller.

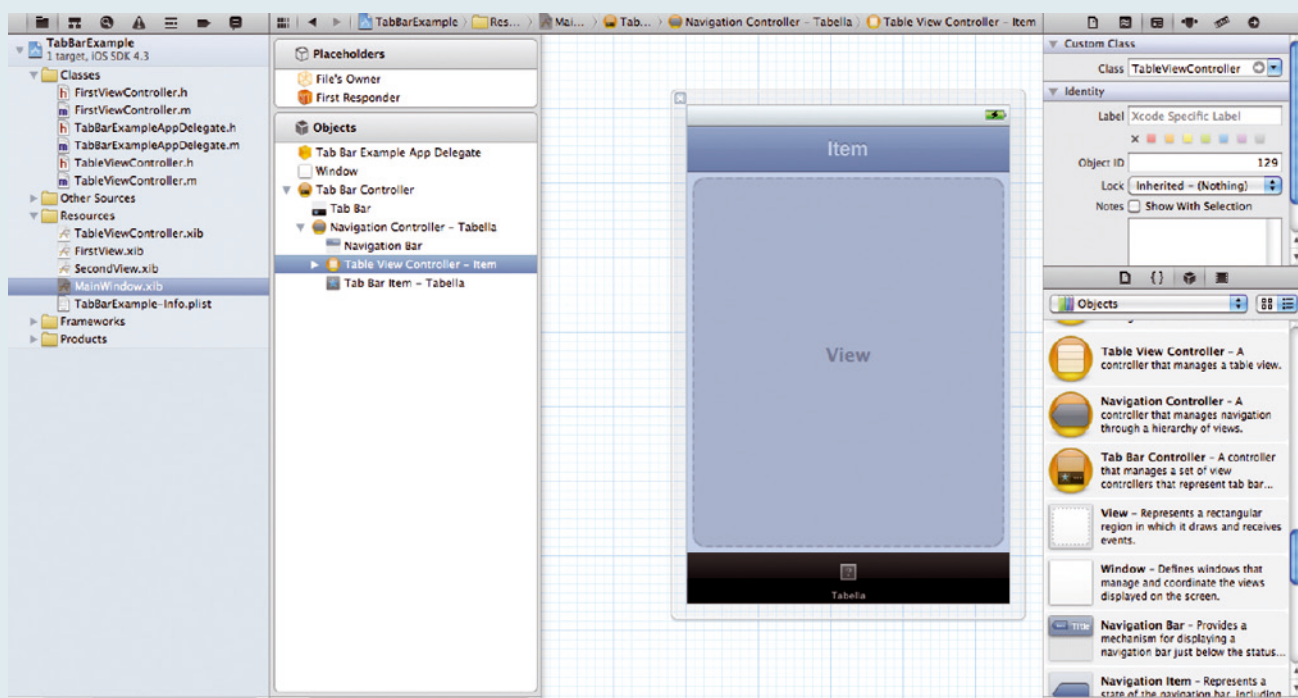


Fig. 9 - Assegnazione della Navigation Bar ad un Controller.

Listato 8 - Configurazione della nostra cella.

```
//Resto dell'implementazione della classe TableViewController
- (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...
    cell.textLabel.text = @"Elettronica IN";

    return cell;
}
//Resto dell'implementazione della classe TableViewController
```

Listato 9 - Utilizzare la Navigation Bar per visualizzare il secondo controller nella navigazione.

```
#pragma mark -
#pragma mark Table view delegate

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    // Navigation logic may go here. Create and push another view controller.

    FirstViewController *detailViewController = [[FirstViewController alloc] initWithNibName:@"FirstView"
bundle:nil];
    // ...
    // Pass the selected object to the new view controller.
    [self.navigationController pushViewController:detailViewController animated:YES];
    [detailViewController release];
}
```

controller: cerchiamo negli oggetti disponibili il tipo "Navigation Controller" e trasciniamolo nel box (**Fig. 8**); a questo punto possiamo "aprire" il componente facendo clic sulla freccina a sinistra e selezionando ViewController. Attraverso la sezione Identity Inspector (tipicamente a destra in Xcode) andiamo ad assegnare all'attributo class il nome "TableViewController" che abbiamo precedentemente creato. A questo punto la situazione dovrebbe essere come appare in **Fig. 9**. Ricordate di andare nell'implementazione del file "TableViewController.m" a sistemare il numero di sezioni e di celle da visualizzare; per fare presto, mettete una sola sezione ed una sola cella e, nella visualizzazione delle celle, assegnate a queste ultime una stringa fissa, come fatto nel **Listato 8**. Prima di avviare l'applicazione, aggiungiamo il titolo al controller (andando a decommentare il metodo

"viewDidLoad") e modifichiamo la proprietà "title" assegnandole il titolo "Tabella"; a questo punto possiamo avviare la nostra applicazione, allorché vedremo che è composta dalla barra in basso, ossia la TabBar, e dalla barra di navigazione (in alto). Il contenuto è la tabella che abbiamo appena generato (**Fig. 10**).

Prima di andare ad agganciare il secondo controller alla nostra vista, mettiamo al lavoro la nostra Navigation Bar, andando ad inserire il codice necessario per navigare nei contenuti. Invece di creare un nuovo controller, riutilizziamo il componente FirstViewController; allo scopo andremo a lavorare nel metodo di selezione della riga e inseriremo il controller. Aggiungiamo la direttiva di importazione del "FirstViewController.h" in testa al nostro TableViewController ed inseriamo il codice di utilizzo della Navigation Bar, come fatto nel **Listato 9**. Avviando ora l'applicazione e

Listato 10 - Utilizzo della navigazione da un UINavigationController.

```

- (void)premutoPulsante:(id)sender
{
    FirstViewController *detailViewController = [[FirstViewController alloc] initWithNibName:@"FirstView"
bundle:nil];
    // ...
    // Pass the selected object to the new view controller.
    [self.navigationController pushViewController:detailViewController animated:YES];
    [detailViewController release];
}

```

selezionando la riga nella tabella, il Navigation Controller “spingerà” il “FirstViewController” e lo renderà visibile.

A questo punto siamo pronti per aggiungere il secondo controller alla TabBar; agganciamo anche a questo componente la sua personale Navigation Bar, perché la Tab Bar separa completamente i contenuti e quindi ognuno di questi ha una sua personale linea della vita. Per aggiungere il secondo controller dobbiamo accedere ad Interface Builder tramite il file MainWindow.xib ed eseguire la stessa procedura, ma utilizzando il FirstViewController invece del TableViewController. Ora possiamo avviare l'applicazione, da cui risulteranno due componenti nella TabBar e due barre di

navigazione distinte per i due controller creati (Fig. 11).

Ora non ci resta che utilizzare anche il Navigation Controller nel FirstViewController. Per fare questo aggiungiamo un pulsante alla vista e con una Action andremo a gestire, dal codice, la navigazione. Per fare questo creiamo l'azione con nome “premutoPulsante”; in pratica basta aprire i file “.m” e “.h” del “FirstViewController” ed aggiungere il metodo necessario alla gestione della pressione del pulsante, mediante la IBAction. In un secondo

momento dobbiamo aprire la vista associata al controller “FirstView.xib” e collegare la IBAction con il pulsante all'evento “TouchUpInside”. Il codice per utilizzare la navigazione è lo stesso che abbiamo utilizzato nel metodo di selezione di una riga all'interno del TableViewController; il risultato per utilizzare la barra di navigazione sulla pressione del pulsante è espresso nel Listato 10.

Se avviamo ora l'applicazione, possiamo utilizzare le due tab e navigare nei contenuti, passando dall'una all'altra senza perdere la navigazione; infatti il concetto delle TabBar è proprio creare differenti linee di vita della nostra applicazione e permettere una navigazione indiretta che sarebbe altrimenti difficile.

Prima di chiudere il progetto, potete anche aggiungere un titolo al controller “FirstViewController” in modo da renderlo disponibile sulla Navigation Bar e rendere l'applicazione più accattivante e completa; non dimenticate di andare a fare il “release” di tutti componenti che avete utilizzato nell'applicazione, inserendo il codice necessario nel metodo “dealloc” dei controller.

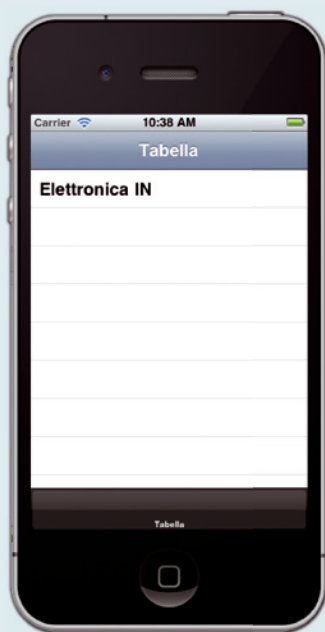


Fig. 10 - Esecuzione dell'applicazione completa delle due “Bar”.

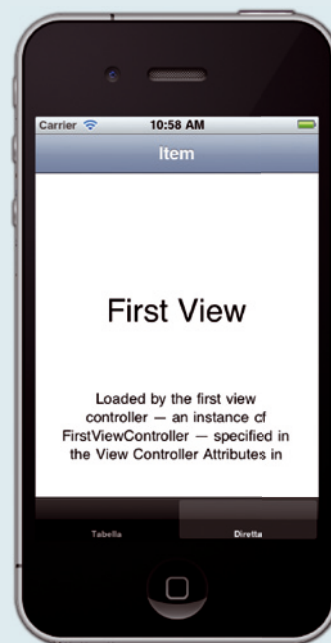
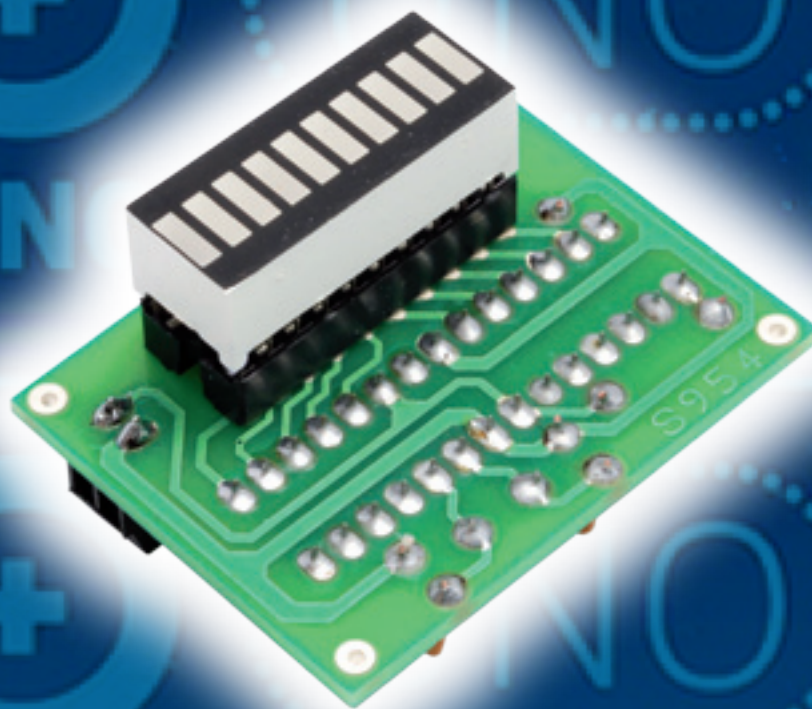


Fig. 11 - Due TabBar per la gestione di contenuti distinti.

Proseguiamo il discorso sulla realizzazione di circuiti in stand-alone impieganti microcontrollori ATmega programmati con Arduino. Seconda puntata.

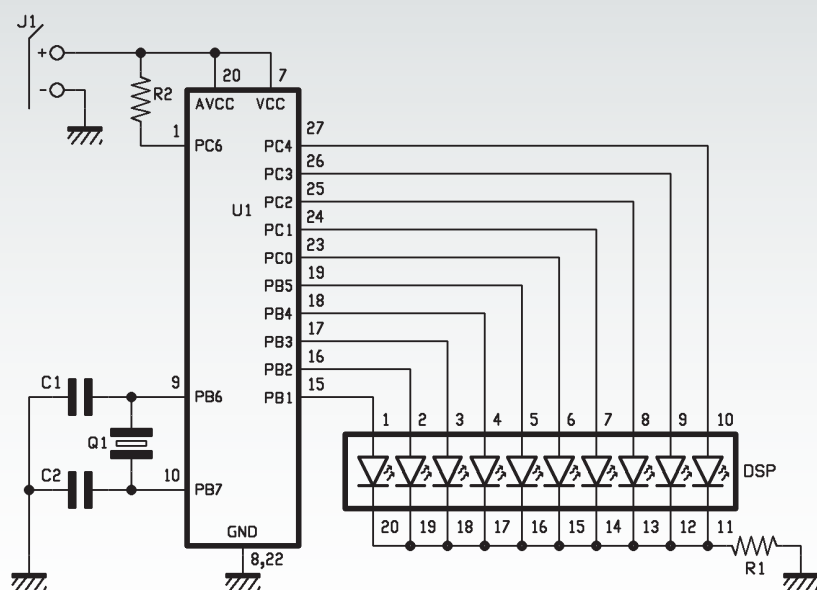


LAMPEGGIANTE PROGRAMMABILE BASATO SU ARDUINO

..... del prof. MICHELE MENNITI

Abbiamo visto, nella precedente puntata, come impiegare il nostro Arduino per programmare un microcontrollore ATmega da usare in modalità stand-alone, cioè separato dalla board. Questo artificio ci permette di creare svariate applicazioni di tipo definitivo, senza essere costretti ad utilizzare un Arduino per ognuna di esse; pensiamo per esempio a circuiti realizzati per svolgere una funzione di controllo ripetitiva: una volta che il micro è programmato per tale funzione, tutta la circuiteria di Arduino non serve più. Infatti il micro non dovrà essere più programmato (viene quindi meno l'esigenza della sezione di

interfaccia col PC, cioè il convertitore USB-seriale) né bisognerà fare test con sensori o altri componenti (allora non servono più i pin di interfacciamento col mondo esterno); in parole povere, tutto si riduce al solo chip con quei quattro componenti necessari a farlo funzionare, oltre ovviamente alla componentistica necessaria all'applicazione stand-alone. Per mettere in pratica quanto avete appreso (ma anche sperimentato, ricordate il blink stand-alone?) abbiamo pensato di realizzare un piccolo circuito, il cui funzionamento è più che sufficiente per i nostri scopi e permette un impiego immediato: si tratta di un lampeggiante



tipo quelli che vediamo in TV sulle auto della polizia di mezzo mondo. In un circuito del genere, il vantaggio di avere un micro consiste soprattutto nel fatto che con pochi comandi software è possibile risparmiare molta circuiteria esterna, sia digitale che analogica, sia attiva che passiva. Come vedremo tra poco, due soli componenti, oltre a quelli dello stand-alone, e un piccolo sketch scritto per Arduino, ci permetteranno di realizzare un semplice circuito che potrà essere alimentato a batteria e montato su una qualsiasi automobilina giocattolo.

SCHEMA ELETTRICO

Il cuore del progetto è, naturalmente, il microcontrollore ATmega328-P prodotto dalla ATMEL, da noi siglato U1. L'ATmega328-P è qui usato nella sua tipica configurazione stand-alone: il pin 1 (PC6) è anche l'ingresso per il reset, che però non useremo in quanto il circuito deve lavorare all'infinito; comunque possiamo notare la presenza della resistenza R2, da 10 kΩ, la cui funzione è quella tipica di pull-up. Ebbene, tale resistenza serve perché il reset del microcontrollore si ottiene ponendo il pin 1 a zero logico, cioè ad un

livello di tensione inferiore a 0,8÷0,9 volt; se lasciassimo libero il piedino, a causa di disturbi elettrici introdotti nel circuito dall'alimentazione o dall'etere, potrebbe verificarsi il reset accidentale del micro. Ponendo fisso a livello alto il pin 1 mediante un resistore, invece, siamo certi che ciò non accada.

Dal data-sheet (scaricabile dal sito ATMEL www.atmel.com/dyn/resources/prod_documents/doc8271.pdf) ricaviamo che i pin 7 e 20 vanno collegati a +5 volt, mentre i pin 8 e 22 vanno a GND dell'alimentazione; certamente avrete notato che sul pin 7 la sigla è AVcc mentre sul pin 20 la sigla è Vcc. In cosa consiste la differenza? Il pin Vcc serve per l'alimentazione generale del micro, mentre AVcc serve per l'alimentazione del solo convertitore Analogico/Digitale incorporato nel micro; in genere si collegano assieme, come già avviene su Arduino. I pin 9 e 10 sono riservati alla sezione dell'oscillatore interno; nel nostro caso, ad essi abbiamo collegato un quarzo da 16 MHz e due condensatori ceramici da 22 pF con un piedino a massa: questi tre componenti rappresentano appunto il circuito oscillatore esterno del micro-

controllore. Sul lato destro dello schema si nota chiaramente che sono stati utilizzati ben dieci pin del micro per pilotare altrettanti LED (in realtà si tratta di un'unica barra da dieci LED, ma ne parliamo più avanti); naturalmente ben sappiamo che con un semplice transistor, in veste di amplificatore di corrente, avremmo potuto pilotarli tutti con due soli pin (visto che li usiamo a gruppi di cinque), semplificando i collegamenti, ma questa scelta è stata fatta tenendo conto della necessità di creare un effetto simile a quello dei lampeggianti a striscia che si accorcia e si allunga; usando dieci pin, infatti, abbiamo avuto la possibilità di pilotare singolarmente i diodi, ottenendo un effetto visivo davvero accattivante (questo circuito ormai in laboratorio ha assunto funzione di gadget e resta acceso per tutto il tempo).

E naturalmente, ben sappiamo che normalmente ogni LED deve avere la sua bella resistenza di limitazione di corrente, perché usarne una sola per molti LED significa risentire di una variazione di luminosità in base al numero di LED accesi contemporaneamente; tuttavia abbiamo usato un'unica resistenza comune alla linea dei catodi dei 10 LED, perché vogliamo ottenere un effetto luminoso che possiamo avere solamente comandando l'accensione dei diodi a frequenza elevata, opportunamente divisi in due gruppi da cinque ciascuno. Come già accennato i LED (vedete lo schema elettrico) sono contenuti in una barra che ne contiene appunto dieci di colore blu (Fig. 1) incapsulati in un unico contenitore; questo sistema facilita le cose, perché la piedinatura ha le stesse misure di un comune circuito integrato in contenitore DIP da 10+10 pin ed ogni

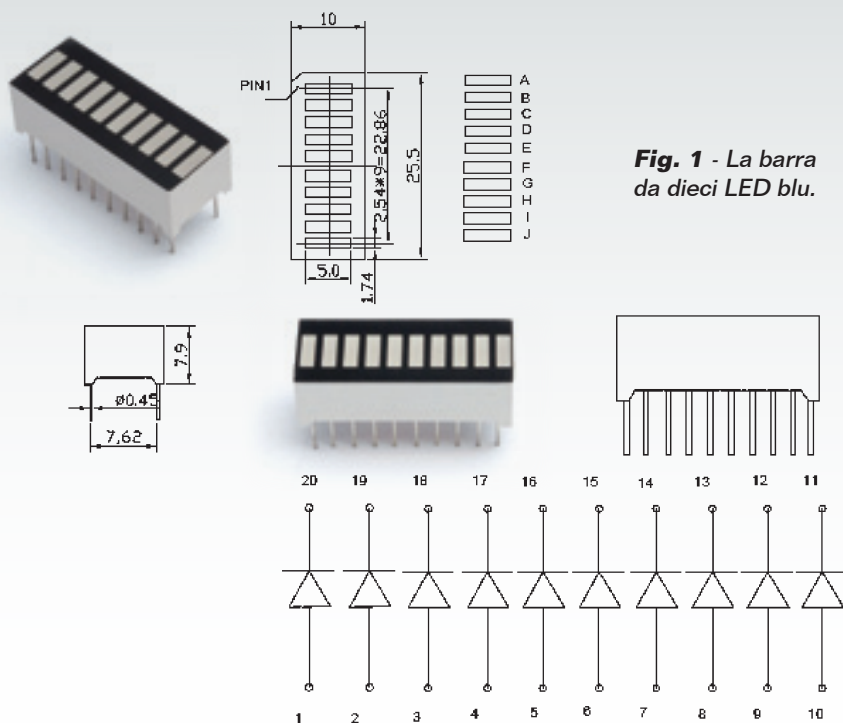


Fig. 1 - La barra da dieci LED blu.

LED è utilizzabile singolarmente, in quanto i propri due piedini sono disponibili separatamente rispetto a quelli degli altri nove (a questo link potrete vedere un video demo: www.youtube.com/watch?v=v8NcdahFjHM).

Nel nostro caso abbiamo creato una fila a catodo comune, unendo i dieci catodi tra loro e mettendoli a massa tramite la resistenza R1, da 390 Ω ; invece ogni anodo è stato collegato ad uno degli altrettanti pin del micro usati per questa applicazione.

Il data-sheet della barra di LED ci dice che la corrente tipica di funzionamento di ogni LED è circa 20 mA a 3,4 volt; se applichiamo la legge di Ohm (nel caso specifico $R=V/I$) per calcolare la resistenza di limitazione di ogni LED otteniamo: $5V-3,4V/0,02A=80\ \Omega$; in pratica, dalla tensione di alimentazione sottraiamo la tensione diretta del LED e dividiamo il risultato per la corrente di lavoro. La scelta di usare una resistenza del valore di 390 Ω è stata dettata dalla necessità di ridurre il consumo di corrente totale, prevedendo

quindi una possibile alimentazione del circuito con batterie; infatti tale resistenza limita la corrente ad un massimo di 4 mA (calcolati applicando sempre la legge di Ohm: $I=V/R$) e anche considerando le dovute tolleranze di tutti i valori indicati, difficilmente supereremo i 6 mA anche quando sarà acceso un intero gruppo di cinque LED. L'intero circuito durante il funzionamento assorbe circa 20 mA. Il circuito sarà alimentato applicando una tensione di 5 volt in corrente continua, facendo molta attenzione alla polarità, ai capi del connettore J1. Se dovessimo disporre di una tensione maggiore dovremo prevedere un opportuno regolatore di tensione anche se, con le correnti in gioco, un diodo zener da 5,1 volt ed una resistenza in serie opportunamente calcolata saranno più che sufficienti allo scopo. Disponendo, invece, di una tensione minore, il circuito funzionerà correttamente fino al limite minimo di 2,5÷2,7 volt; oltretutto il consumo del circuito diminuirà proporzionalmente alla tensione di alimentazione.

Bisogna comunque tenere da conto che, nel caso di una tensione di alimentazione inferiore ai 4 volt, bisognerà diminuire il valore della resistenza R1, altrimenti la barra dei LED perderà molta luminosità, fornendo un effetto molto scadente; in ogni caso non bisogna scendere al di sotto dei 150 Ω .

LO SKETCH

Per il funzionamento del nostro lampeggiante abbiamo predisposto il semplice sketch che potete vedere nel **Listato 1** e scaricare, assieme al master del c.s., dal nostro sito www.elettronica.in.it. Lo sketch è autoesplicativo, perché i numerosi commenti lo rendono estremamente comprensibile, tuttavia possiamo descriverlo a grandi linee.

Nella sezione SETUP vengono impostati come uscite dieci pin del micro.

Nella sezione LOOP vengono poste in HIGH, in rapidissima sequenza, cinque delle dieci uscite, facendo accendere i relativi LED della barra; dopo una pausa di 8 millisecondi, tali uscite vengono portate a LOW e i LED si spengono per 4 millisecondi (questo ciclo si ripete per cinque volte). A questo punto succede la stessa cosa con l'altro gruppo di cinque LED, e così via. L'effetto che si ottiene è quello di vedere due "luci" blu tremolanti e lampeggianti, proprio come quelle di alcuni lampeggianti della polizia. Naturalmente, vista la sua semplicità, lo sketch potrà essere modificato a piacimento per ottenere altri effetti; comunque sia, un gioco di luci con 10 LED di colore blu sarà sempre affascinante. Un particolare degno di nota riguarda i pin analogici: come ben sappiamo, i 14 pin digitali su Arduino sono denominati da 0 a 13 ed i 6 pin analogici sono

Listato 1

```

/*
Programma per il Lampeggiante a Barra di Led blu
***** Rivista Elettronica In *****
scritto da M. Menniti
*/

void setup() {
  for(int i=9; i<=18; i++)
    pinMode(i, OUTPUT);
  /*tutti i pin dal 9 al 18 sono impostati come uscite
   i pin 14-18 corrispondono ai pin analogici A0-A4
   usati in questo caso come normali uscite digitali */
}

void loop() {
  for(int t=1; t<=5; t++) // ciclo da 5 lampeggi totali del primo gruppo di
  5 LED
  {
    for(int k=9; k<=13; k++) // si inizia con le uscite dei pin da 9 a 13
    {
      digitalWrite(k, HIGH); // i LED si accendono
      delay (8); // restano accesi per 8 millisecondi
    }
    for(int x=9; x<=13; x++)
    {
      digitalWrite(x, LOW); // i LED si spengono
      delay (4); // restano spenti per 4 millisecondi
    }
  }

  delay(30); // pausa di 30 millisecondi

  for(int z=1; z<=5; z++) // ciclo da 5 lampeggi totali del secondo gruppo
  di 5 LED
  {
    for(int j=14; j<=18; j++) // questa volta si attivano le uscite dei pin
    da 14 a 18 (A0-A4)
    {
      digitalWrite(j, HIGH); // i LED si accendono
      delay (8); // restano accesi per 8 millisecondi
    }
    for(int w=14; w<=18; w++)
    {
      digitalWrite(w, LOW); // i LED si spengono
      delay (4); // restano spenti per 4 millisecondi
    }
  }

  delay(30); // pausa di 30 millisecondi
}

```

numerati da A0 ad A5; ma allora quali sono i pin che nel nostro sketch vengono denominati da 14 a 18? In realtà basterebbe dare un'occhiata allo schema elettrico per capirlo: infatti i sei pin analogici possono svolgere comunque il ruolo di uscite digitali. Ecco perché il software accetta la doppia denominazione; quindi il pin A0 corrisponde al 14 (in prosecuzione della numerazione dei pin digitali), il pin A1 al 15, fino ad arrivare al pin A6, che corrisponde al 19. L'aver usato questo secondo tipo di numerazione ci ha estremamente semplificato

la programmazione dello sketch, avendo potuto attivare con un semplice ciclo FOR cinque dei sei pin analogici.

LA PROGRAMMAZIONE DEL MICRO

Il nostro chip ATmega328-P, che sia vergine o già usato, ha bisogno ora di essere preparato per essere "accolto" nel suo nuovo circuito e poter quindi funzionare in modalità stand-alone, cioè senza l'ausilio della scheda Arduino. La programmazione di micro stand-alone, usando Arduino come programmatore

ISP, è stata già abbondantemente trattata nel precedente fascicolo di Elettronica In, perciò in questa sede ci limiteremo ad elencare le varie fasi per l'esecuzione della tecnica, rimandando alla lettura di tale articolo per i necessari approfondimenti.

I componenti preposti a compiere questa operazione sono in pratica parte di quelli che serviranno per realizzare il nostro lampeggiante, quindi è ovvio che prima di saldare i componenti sul c.s. li useremo con una breadboard per programmare il micro. In Fig. 2 possiamo vedere come preparare il micro su una breadboard. I componenti necessari sono:

- un quarzo da 16 MHz;
- due condensatori ceramici da 22 pF;
- una resistenza da 10 kΩ ¼ watt ed una da 560 Ω ¼ watt;
- un LED standard da 3 o 5 mm.

Da notare che la resistenza da 560 Ω ed il LED hanno qui lo scopo di "monitor": se tutto andrà a buon fine, il LED lampeggerà a frequenza piuttosto elevata. In realtà se ne potrebbe fare a meno, visto che comunque la stessa cosa accadrà al LED "L" montato sulla board Arduino; diciamo che il LED sulla breadboard è molto più evidente!

In Fig. 3 invece possiamo vedere i collegamenti tra Arduino e la breadboard, da fare per poter programmare il micro; eccoli descritti qui di seguito:

- il pin 10 di Arduino va al pin 1 del micro;
- il pin 11 di Arduino va al pin 17 del micro;
- il pin 12 di Arduino va al pin 18 del micro;
- il pin 13 di Arduino va al pin 19 del micro;
- il pin +5 V di Arduino va alla linea di alimentazione positiva della breadboard;

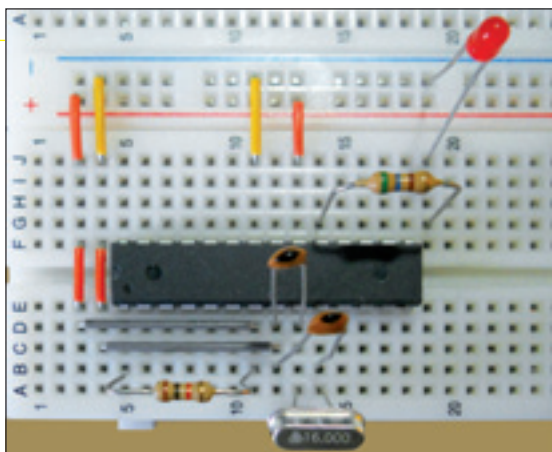


Fig. 2 - Preparazione del chip su Breadboard.

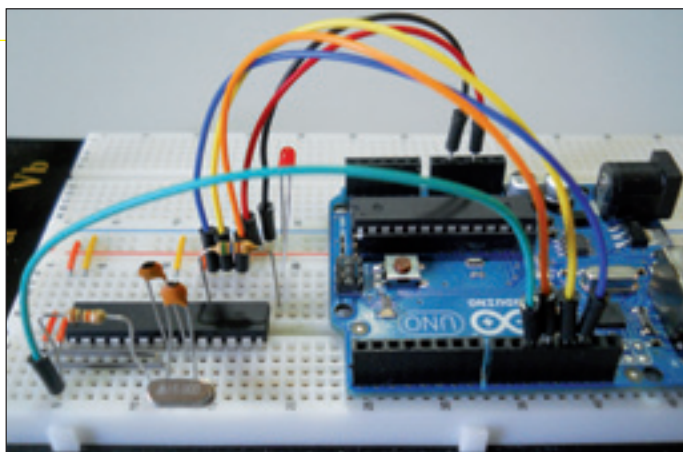


Fig. 3 - Collegamenti tra Arduino e Breadboard.

- uno qualsiasi dei tre pin GND di Arduino va alla linea di massa della breadboard.

La programmazione del micro si effettua eseguendo, nell'ordine descritto, i seguenti passaggi.

1. Modifica del file *boards.txt* con l'inserimento della board virtuale "ATmega in Stand Alone (w/ Arduino as ISP)", le cui righe sono scaricabili dalla sezione *download* del nostro sito internet e sono comunque visibili nel **Listato 2**. Il file *boards.txt*, lo ricordiamo, si trova 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 l'ultima versione del programma, al momento della stesura di questo articolo (potreste quindi trovare un codice diverso)).
2. Caricamento del bootloader per la corretta impostazione dei fuse (questa operazione è necessaria solo se il micro è vergine o in precedenza ha operato con frequenza di clock diversa da 16 MHz). Brevemente, tale operazione si esegue con questi passaggi:
 - a. caricamento dello sketch "ArduinoISP" nella vostra board Arduino;
 - b. selezione, tramite il comando Tools→Board, della board virtuale;

- c. esecuzione del comando Tools→Burn Bootloader→w/ Arduino as ISP;

3. caricamento ed upload dello sketch del lampeggiante, che andrà automaticamente nella

memoria flash del micro in stand-alone;

Le eventuali problematiche che potrebbero verificarsi durante

Arduino incontra LEGO

LEGO Mindstorms ha sbancato nel mondo della robotica sperimentale basandosi su hardware e software di programmazione ad hoc. Ma a questa piattaforma mancava ancora qualcosa: l'incontro con Arduino, il sistema che ha cambiato il modo di fare l'elettronica programmabile.

A colmare questo vuoto ha pensato la Mindsensors, che ha realizzato NXShield, un nuovo shield che permette di interfacciare le schede Arduino con i prodotti LEGO Mindstorms e di mettere a disposizione il ricco patrimonio di librerie e potenzialità dell'universo Arduino per sviluppare nuove applicazioni. NXShield, oltre ad essere dotato di quattro porte dedicate alla lettura di sensori NXT ed altrettante per il controllo dei motori originali LEGO, permette anche di controllare i normali servocomandi utilizzati nel modellismo dinamico.



Il micro ATmega328-P



- Architettura "Advanced RISC" a 20 MIPS
- 32 kbyte di memoria Flash
- 1 kbyte di memoria EEPROM
- 2 kbyte di memoria SRAM
- Cicli di lettura/scrittura: 10.000 per la Flash e 100.000 per la EEPROM
- Due Timer/Counter a 8-bit
- Un Timer/Counter a 16-bit
- Real Time Counter con oscillatore separato
- 23 linee programmabili I/O di cui sei canali PWM (utilizzabili anche come I/O digitali)
- 6 canali ADC (Convertitore analogico-digitale) a 10-bit, (utilizzabili anche come output digitali)
- 8 specifici canali digitali I/O
- Interfaccia Seriale compatibile col protocollo Philips I²C
- Oscillatore interno calibrato con ingresso per clock esterno fino a 20 MHz
- Gestione di Interrupt interni ed esterni
- Sei diverse modalità di Sleep: Idle, ADC Noise Reduction, Power-save, Power-down, Standby ed Extended Standby
- Range di Alimentazione: 1,8 ÷ 5,5V
- Range di Temperatura: -40 °C ÷ +85 °C
- Frequenze di Clock:
 - 0÷4 MHz a 1,8 ÷ 5,5 V
 - 0÷10 MHz a 2,7 ÷ 5,5 V
 - 0÷20 MHz a 4,5 ÷ 5,5 V
- Consumo minimo di corrente a 1 MHz, 1,8 V, 25 °C
 - Modalità attiva: 0,2 mA
 - Modalità Power-down: 0,1 µA
 - Modalità Power-save: 0,75 µA (Incluso l'RTC a 32 kHz).

l'operazione di programmazione del micro sono state abbondantemente descritte e risolte nell'articolo già citato.

REALIZZAZIONE PRATICA

Una volta che abbiamo compre-

so il funzionamento di questo semplice circuito e programmato il micro, possiamo passare alla sua realizzazione pratica, usando un piccolo circuito stampato a doppia faccia disegnato appositamente, che potrete realizzare

con le normali tecniche partendo dai file scaricabili dalla sezione *download* del nostro sito. Su di esso verranno montati tutti i componenti, come indicato dal piano di montaggio visibile in queste pagine.

Ciò che bisogna sottolineare subito è il fatto che tutta la componentistica prevista va montata su un lato e la sola barra dei LED (col relativo zoccolo) sul lato opposto. La ragione di ciò è semplice: prevedendo un'eventuale installazione sul tetto di un'automobilina, con questo tipo di montaggio sarà possibile far fuoriuscire la sola barra "nascondendo" nell'abitacolo gli altri componenti.

Una nota per coloro che ritenessero di incontrare difficoltà a realizzare lo stampato in doppia faccia: in realtà, a motivo della configurazione hardware adottata, è possibile montare anche la barra dei LED (col relativo zoccolo) sul lato componenti. È sufficiente fare in modo che i pin 1 e 20 (rispettivamente l'anodo e il catodo del primo LED) si vengano a trovare a ridosso della resistenza R1: in pratica la smussatura della barra dei LED dovrà trovarsi accanto al pin 26 del micro.

Qualcuno potrebbe osservare che la logica dello sketch verrebbe stravolta, ma possiamo assicurarvi che esso è perfettamente speculare per i due gruppi di LED e la sua velocità di esecuzione è tale da non mostrare alcuna differenza.

Per il montaggio conviene iniziare dalle due resistenze, proseguendo quindi col quarzo e i due condensatori; subito dopo si potrà procedere con la saldatura dello zoccolo a 14+14 pin destinato ad ospitare l'integrato, facendo attenzione alla tacca di riferimento, che tornerà utile quando

Listato 2

```
#####
```

```
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=1600000L
atmsal6.build.core=arduino
```

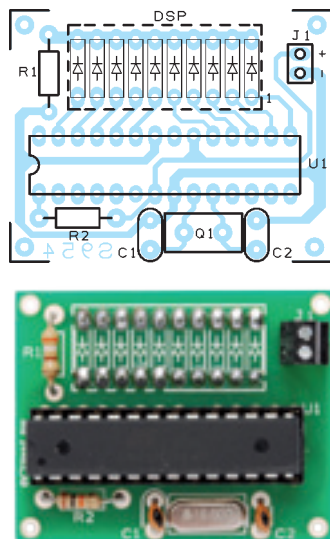
[piano di MONTAGGIO]

Elenco Componenti:

R1: 390 ohm
 R2: 10 kohm
 C1: 22 pF ceramico
 C2: 22 pF ceramico
 U1: ATmega328-P
 DSP: Barra 10 LED blu
 (YSLB-10251B5-10)
 Q1: quarzo 16 MHz

Varie:

- Zoccolo 10+10
- Zoccolo 14+14
- Morsetto 2 poli passo 2,54
- Circuito stampato



bisognerà inserire il componente. Infine bisogna saldare il morsetto per l'alimentazione.

A questo punto si dovrà girare lo stampato dal lato delle saldature ed inserire negli appositi fori lo zoccolo a 10+10 pin, saldandolo questa volta dal lato dove sono stati collocati tutti gli altri componenti, con la tacca di riferimento rivolta verso il connettore J1. Ora possiamo inserire l'ATmega328-P (naturalmente dopo averlo programmato, come spiegato in precedenza) ed il display

nei rispettivi zoccoli, facendo molta attenzione alla polarità di entrambi; per l'ATmega osservate il piano di montaggio controllando la tacca di riferimento presente sul chip.

La barra dei LED ha una piccola smussatura in corrispondenza dell'anodo del primo LED (pin 1); anche in questo caso riferitevi al piano di montaggio. Qualora tale smussatura non dovesse essere presente, poiché i LED sono alimentati con una corrente molto bassa potete comunque

inserire la barra dei LED a caso; se anche fosse a rovescio, non si potrebbe mai danneggiare. Naturalmente se vedete che i LED non si accendono appena alimentate il circuito, scollegate subito l'alimentazione ed invertite la barra dei LED.

Una volta localizzata la giusta polarità, sarebbe conveniente mettere un simbolo di riconoscimento sullo spigolo che dovrebbe riportare la smussatura; in tal modo, nel caso doveste essere costretti a togliere la barra dei LED dal suo alloggiamento, per rimetterla vi basterà controllare il segno, che corrisponderà al pin 1. Terminato il montaggio dei componenti non dovete fare altro che fornire al circuito un'alimentazione di 5 volt (attenzione alla polarità!) e subito vedrete la barra di LED al lavoro col suo bell'effetto scenografico.

Come detto lo sketch è molto semplice, quindi si potrà facilmente modificare per far eseguire al display giochi di luci; questo è uno dei vantaggi dell'aver usato un pin del micro per ogni LED. Nella Fig. 4 potete vedere il lampeggiante "al lavoro".

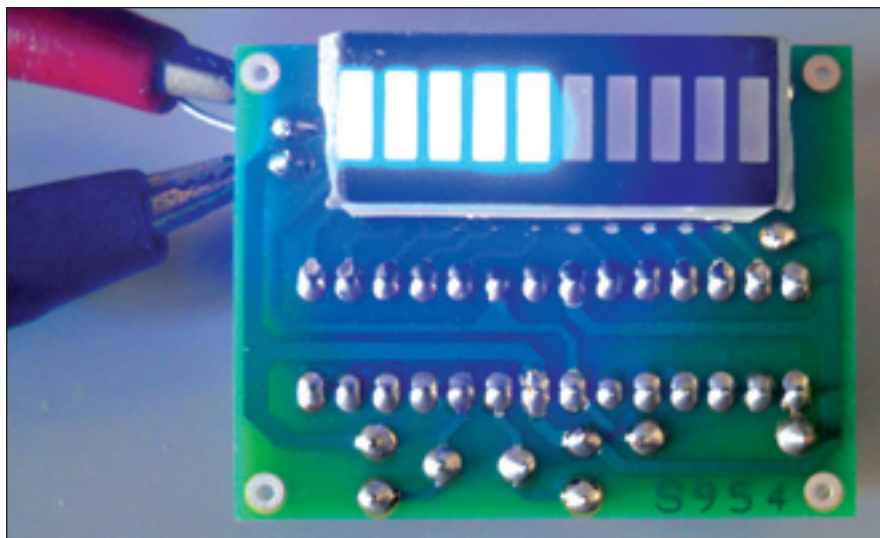
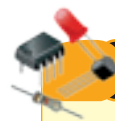


Fig. 4 - Il Lampeggiante in funzione.



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del CS e lo sketch sono scaricabili dal sito della rivista (www.elettronica.in.it). Il microcontrollore vergine ATmega328-P è reperibile presso Futura Elettronica al prezzo di 6 Euro (cod. ATMEGA328P-PU) così come la barra a LED blu al prezzo di 2,90 Euro (cod. BLULEDBAR).

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

Concludiamo la progettazione del firmware per il Data Logger su SD-Card realizzato con le FPGA basate su Cortex-M3. Ultima parte.



FPGA SMART FUSION

di MARIANO SEVERI

Dopo aver descritto l'architettura del progetto e la sintesi circuitale del Data Logger, riprendiamo e finiamo il discorso iniziato nella seconda parte di questo tutorial, passando alla programmazione della scheda di valutazione.

PROGRAMMAZIONE DELL'A2F-EVAL-KIT

Verifichiamo la corretta configurazione dei jumper, come riportato nella **Tabella 1**, e quindi connettiamo

la scheda ad un PC host mediante i due cavi USB in dotazione, uno dei quali connette il PC al programmatore integrato sulla scheda portandole anche l'alimentazione; il secondo cavo USB si connette, invece, al bridge UART-USB della stessa. La prima volta che eseguiamo l'operazione ci potrebbe essere richiesta l'autorizzazione ad installare i driver del cavo FlashPro; accettiamo. Dalla finestra principale di Libero, nell'area Design Flow, lanciamo quindi

Tabella 1

Connettore	Connettere pin
JP10	1 e 2
JP7	1 e 2
J6	1 e 2
JP6	2 e 3

Configurazione dei jumper della scheda A2F-EVAL-KIT per la corretta programmazione mediante programmatore integrato.

il comando *FlashPro*; si aprirà la finestra mostrata nella Fig. 1. Selezioniamo *Configure Device* e verifichiamo che siano spuntate entrambe le caselle *FPGA Array* ed *Embedded Flash Memory* e che risultino rispettivamente selezionati i file *../designer/impl1/DataLogger.fdb* e *../component/work/mss_core/MSS_ENVM_0/MSS_ENVM_0.efc*; i path indicati si intendono relativi alla cartella dove abbiamo creato il progetto Libero. Confermiamo la

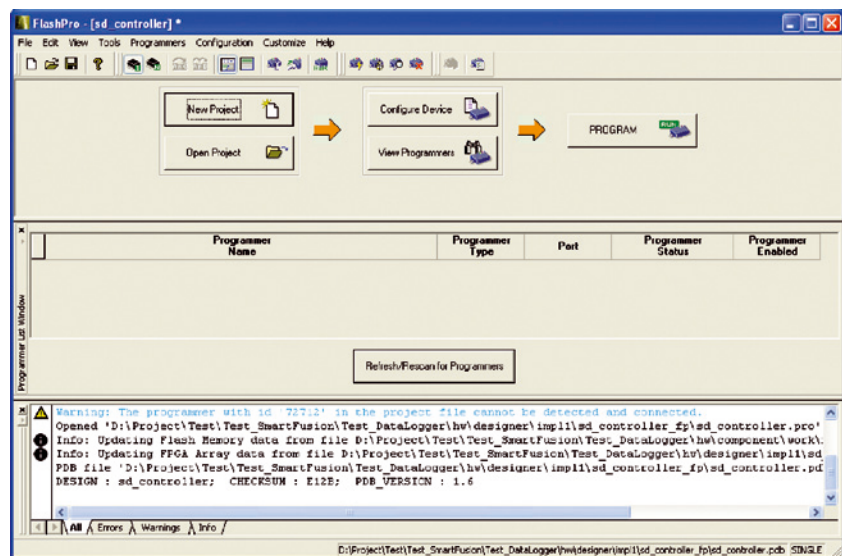


Fig. 1 - La finestra del tool FlashPro per la programmazione dei dispositivi SmartFusion.

selezione per ritornare alla finestra principale di FlashPro. Lanciamo infine il comando *Program*. L'operazione potrebbe richiedere qualche minuto, allo scadere del

quale l'FPGA della scheda sarà programmata per implementare il sistema di data logging che avevamo immaginato; ora non ci resta che scrivere il nostro applicativo.

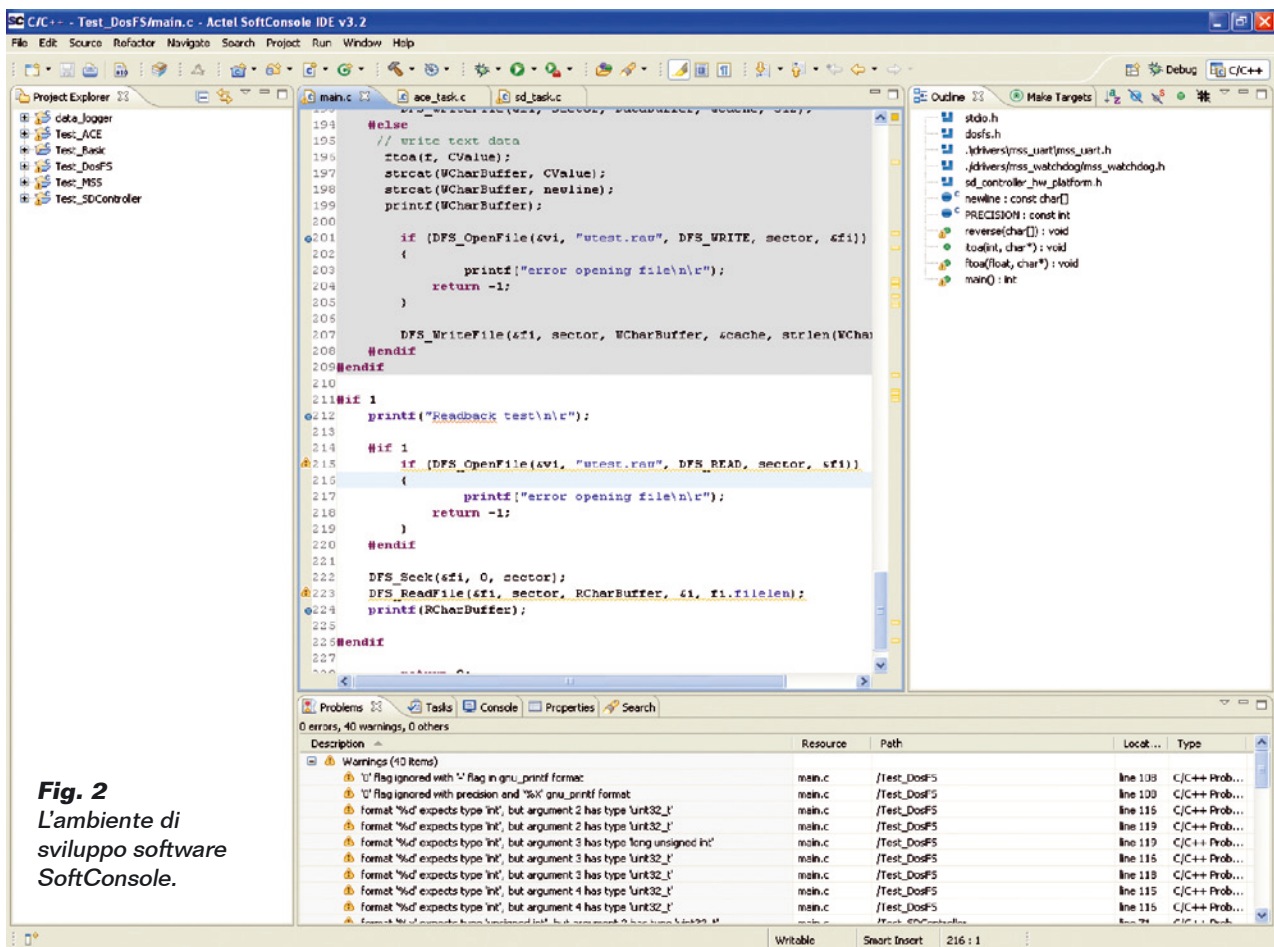


Fig. 2
L'ambiente di sviluppo software SoftConsole.

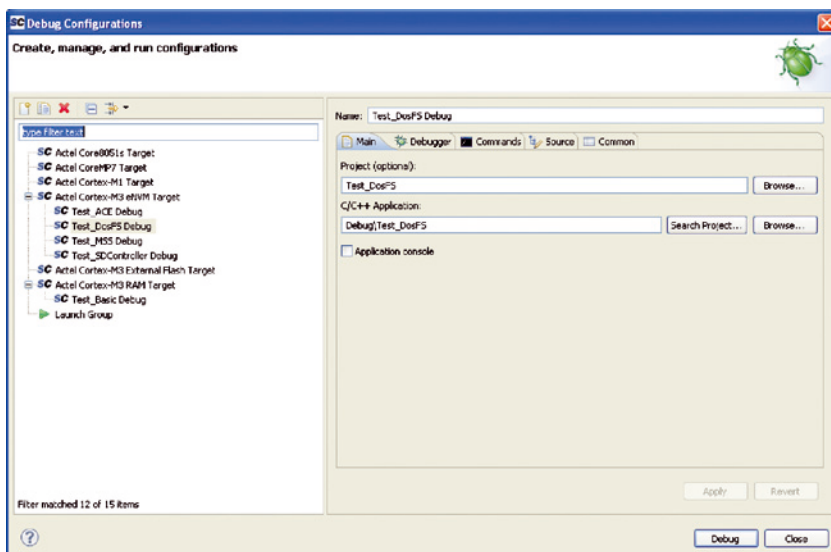


Fig. 3 - Definizione delle configurazioni di debug.

CREAZIONE DEL PROGETTO SOFTWARE

SoftConsole (Fig. 2) è l'ambiente di sviluppo software per SmartFusion; è basato sulla piattaforma open-source Eclipse. Lanciamo il programma, selezioniamo un nuovo workspace e creiamo un nuovo progetto, che nel nostro caso chiameremo *Test_MSS*; ci servirà per verificare il corretto funzionamento della sezione a microcontrollore. Da Project Explorer selezioniamo, con il pulsante destro del mouse, il progetto appena creato e scegliamo la voce *Import...*. Selezioniamo *File System* e navighiamo fino a trovare la directory firmware presente nella cartella del progetto Libero creato in precedenza. Quindi, scegliamo di importare le sotto-cartelle *CMSIS*, *drivers* e *drivers_config* e verifichiamo sia spuntata l'opzione *Create selected folders only*; confermiamo le impostazioni prima di tornare alla schermata principale di Soft Console. Ora selezioniamo una seconda volta il progetto in Project Explorer con il pulsante destro del mouse e scegliamo la voce *Properties*; selezioniamo quindi la voce *Settings* sotto C/C++ Build e poi il campo *Symbols* sotto GNU C compiler (all'interno del tab *Tool Settings*). Definiamo il nuovo simbolo

ACTEL_STUDIO_THRU_UART che permette di associare le istruzioni di I/O tipo *printf* alla porta seriale del microcontrollore Cortex-M3 presente nella FPGA. Quindi selezioniamo la voce *Miscellaneous* sotto GNU C Linker e, nel campo *Linker flags*, aggiungiamo l'opzione *-T ../CMSIS/startup_gcc/debug-in-actel-smartfusion-envm.ld*. L'applicativo sarà così linkato in modo da allocare le varie sezioni nella memoria NVRAM del dispositivo SmartFusion; la RAM embedded del componente è infatti in grado di contenere solo applicazioni piuttosto elementari e, d'altra parte, la scheda di sviluppo A2F-

EVAL-KIT non dispone di RAM esterna. Per applicazioni e configurazioni diverse, potrà essere definito un differente linker script; notate che la cartella *CMSIS/startup_gcc* contiene già un esempio di linker script per la compilazione di applicativi da eseguire da memoria esterna.

Chiudiamo la finestra di configurazione e ritorniamo alla schemata principale di Soft Console; creiamo quindi un nuovo file *main.c*. L'applicativo che intendiamo scrivere per verificare il corretto funzionamento della sezione a microcontrollore della FPGA prevede l'utilizzo del timer, che genera un'interruzione ogni secondo, in risposta alla quale il processore invia un messaggio sulla porta seriale. Come accennato in precedenza, la scheda A2F-EVAL-KIT utilizza un bridge UART-USB di Silicon Lab per la comunicazione seriale; i driver possono essere scaricati all'indirizzo indicato nei riferimenti al punto [9]. Sul PC host potrà essere eseguito Hyperterminal; la porta seriale da aprire, associata alla porta USB, potrà essere riconosciuta dalla voce Gestione Risorse del Pannello di Controllo. Il baud-rate della linea è 57.600, mentre la configurazione

Listato 2 - Applicativo di test della sezione a microcontrollore

```
int main()
{
    const uint32_t g_TimerFrequency = 1; /* Hz */
    const uint32_t g_TimerReloadValue = g_FrequencyPCLK0/g_TimerFrequency;

    /* Initialization all necessary hardware components */
    init_system();
    printf("Init system \r\n");

    /* Configure the Timer */
    MSS_TIM1_load_background(g_TimerReloadValue);
    MSS_TIM1_enable_irq();
    MSS_TIM1_start();

    /* Infinite Loop */
    while (1);

    return 0;
}
```


Listato 3 – Applicativo di test della sezione analogica : main()

```

/*****
/* Standard Includes */
/*****
#include <stdio.h>
#include <stdlib.h>

/*****
/* Firmware Includes */
/*****

#include "CMSIS/a2fxxm3.h"
#include "drivers/mss_watchdog/mss_watchdog.h"
#include "drivers/mss_uart/mss_uart.h"
#include "drivers/mss_ace/mss_ace.h"

/*****
/* Preprocessor Macros*/
/*****

#define SYS_TICK_CTRL_AND_STATUS_REG      0xE000E010
#define SYS_TICK_CONFIG_REG               0xE0042038
#define SYS_TICK_FCLK_DIV_32_NO_REF_CLK   0x31000000
#define ENABLE_SYS_TICK                   0x7

/*****
/*Extern Declarations*/
/*****
extern void ace_task();
extern void hyperterminal_task();

/*****
/*Functions */
/*****
void delay ( volatile unsigned int n)
{
    while(n!=0)
    {
        n--;
    }
}

/*****
/* Function to initialization all necessary hardware components for this*/
/* demo*/
/*****
void init_system()
{
    /* Disable the Watch Dog Timer */
    MSS_WD_disable( );
    /* Initialize and configure UART0. */
    MSS_UART_init
    (
        &g_mss_uart0,
        MSS_UART_57600_BAUD,
        MSS_UART_DATA_8_BITS | MSS_UART_NO_PARITY | MSS_UART_ONE_STOP_BIT
    );
    /* ACE Initialization*/
    ACE_init();
}

/*****
/* Entry to Main form user bootcode */
/*****
int main()
{
    /* Initialization all necessary hardware components */
    init_system();

    /* Periodically acquire the the analog input and display to hyperterminal */
    while (1)
    {
        ace_task();
        hyperterminal_task();
        delay(1000000);
    }
    return 0;
}

```

della porta è 8n1, senza controllo di flusso.

Il **Listato 2** riporta un estratto del codice sorgente dell'applicativo che abbiamo scritto per verificare il corretto funzionamento della sezione a microcontrollore della FPGA; il file completo è disponibile per il download dal sito web della rivista (*DataLogger/sw/Test_MSS/main.c*). Come si può vedere nel corpo del *main()*, dapprima si inizializza il sistema mediante chiamata della funzione *init_system()*, quindi viene configurato il valore di reload del timer. Viene abilitata la generazione dell'interruzione del timer ed il timer viene avviato. Il *main()* entra quindi in un ciclo infinito. La funzione *init_system()* disabilita il watchdog, configura la porta seriale ed imposta la modalità periodica per il timer; la funzione *Timer1_IRQHandler()* viene chiamata dal sistema in risposta all'interruzione del timer. Tale funzione (si veda il codice scaricato) dapprima esegue il clear dell'interruzione, quindi incrementa la variabile globale *TimerIrqNum* ed infine invia il messaggio sulla porta seriale.

Copiamo il contenuto del file scaricato nel *main.c* precedentemente creato, salviamo e compiliamo il progetto, selezionando il comando *Build Project* dal menu di pop-up che si apre facendo clic con il pulsante destro del mouse sulla voce *Test_MSS* nella finestra *Project Explorer di Soft Console*. Verifichiamo che non ci siano errori e quindi, dallo stesso menu, scegliamo l'opzione *Debug As -> Debug Configuration*. Creiamo quindi, come mostrato nella **Fig. 3**, una nuova configurazione sotto la voce *Actel Cortex-M3 eNVM Target*, verificando che nel campo *Project* appaia la dicitura *Test_MSS* e in quello *C/C++ Application* la selezione *Debug\Test_MSS*. Selezioniamo *Apply* e quindi *Debug*. *Soft Console*

Listato 4 – Applicativo di test della sezione analogica : ace_task()

```
/* ***** */
/* Standard Includes */
/* ***** */
#include <stdio.h>
#include <stdlib.h>

/* ***** */
/* Driver Includes */
/* ***** */
#include "drivers/mss_ace/mss_ace.h"

/* ***** */
/* Global Declarations */
/* ***** */

float          real_voltage_value;
float          real_current_value;
float          real_temperature_value;
float          real_temperature_value_tc;
float          real_temperature_value_tk;
float          real_temperature_value_tf;

/* ***** */
/* Functions */
/* ***** */
void update_ace_values( ace_channel_handle_t channel_handle,
                        uint16_t raw_value )
{
    channel_type_t channel_type;
    channel_type = ACE_get_channel_type( channel_handle );

    switch(channel_type)
    {
        case VOLTAGE:
            real_voltage_value = (float)ACE_convert_to_mV( channel_handle, raw_value )/(float)1000;
            break;

        case CURRENT:
            real_current_value = (float)ACE_convert_to_uA( channel_handle, raw_value );
            break;

        case TEMPERATURE:
            real_temperature_value_tk = (float)ACE_convert_to_Kelvin( channel_handle, raw_value );
            real_temperature_value_tf = (float)ACE_convert_to_Fahrenheit( channel_handle, raw_value );
            real_temperature_value_tc = (float)ACE_convert_to_Celsius( channel_handle, raw_value )/(float)10;
            break;

        default:
            break;
    }
}

void ace_task()
{
    uint16_t adc_result;
    ace_channel_handle_t first_channel;
    ace_channel_handle_t current_channel;
    /* Display all configured channels current sampled value. */

    current_channel = ACE_get_first_channel();
    first_channel = current_channel;
    do
    {
        adc_result = ACE_get_ppe_sample( current_channel );
        update_ace_values( current_channel, adc_result );
        current_channel = ACE_get_next_channel( current_channel );
    } while ( current_channel != first_channel );
}
```

scaricherà l'applicativo compilato nella memoria eNVM del dispositivo (assicurarsi che la scheda

target sia alimentata) e passerà automaticamente alla vista Debug. A questo punto, possiamo inserire

breakpoint nel codice o eseguirlo passo-passo. Lanciando il programma, se abbiamo eseguito tutti

Listato 5 - Applicativo di test della sezione analogica : `hyperterminal_task()` (vert. to uA)

```

/*****
/* Standard Includes */
/*****
#include <stdio.h>
#include <stdlib.h>

/*****
/* Driver Includes */
/*****
#include "../drivers/mss_uart/mss_uart.h"

/*****
/* Global Declarations */
/*****
float          real_voltage_value;
float          real_current_value;
float          real_temperature_value_tc;

/*****
/* Task Will work with hyper Terminal */
/* Provides scrolling of demo option using Keyboard.*/
/*****

void hyperterminal_task()
{
    printf( "Voltage: %5.2f V \n\r", real_voltage_value );
    printf( "Current: %5.2f uA \n\r", real_current_value );
    printf( "Temperature %5.2f celsius\n\r", real_temperature_value_tc );
}

```

i passi correttamente, vedremo apparire all'interno del terminale seriale sul PC host, ogni secondo, un messaggio del tipo "Timer Irq Number :." seguito da un numero progressivo.

TEST DELLA SEZIONE ANALOGICA

Il Listato 3 mostra invece un estratto del codice sorgente scritto per verificare il corretto funzionamento della sezione analogica del nostro sistema. Per il codice completo si veda il file presente nell'allegato all'articolo scaricabile dal sito della rivista (*DataLogger/sw/Test_ACE/main.c*). I file *multi-meter_task.c* e *hyperterminal_task.c* contengono il codice per le corrispondenti funzioni richiamate dal *main()*; estratti di tali file sono riportati nei listati 4 e 5. L'esempio è tratto direttamente da alcuni progetti di riferimenti distribuiti da Actel.

Creiamo, anche in questo caso, un nuovo progetto in SoftConsole, seguendo le istruzioni descritte in precedenza. Copiamo il contenuto del file che abbiamo scaricato in

un nuovo file *main.c*, compiliamo il progetto, creiamo una nuova configurazione di debug e lanciamo l'applicativo.

Come si può vedere dal listato 3, la funzione *main()* dell'applicativo, dapprima inizializza il sistema, quindi entra in un ciclo infinito in cui periodicamente esegue i due task utente *ace_task()* e *hyperterminal_task()*. Il primo acquisisce il valore dei canali di misura, il secondo invia sulla porta seriale il valore convertito. Ogni ciclo prevede un tempo di attesa definito dalla funzione *delay()*. Eseguendo l'applicativo, quindi, vedremo apparire sulla finestra di Hyperterminal una serie di messaggi che riportano il valore letto per i tre canali di misura; modificando la posizione del potenziometro mentre il programma è in esecuzione, vedremo cambiare coerentemente la corrente e la tensione misurate. Si tenga presente che alimentando la scheda A2F-EVAL-KIT mediante porta USB, l'alimentazione non è certo pulita e quindi accuratezza e precisione delle misure potrebbero risentirne. Il task *ace_task*

(Listato 4) interroga l'ACE per leggere il valore degli ultimi campioni acquisiti; si ricordi che l'ACE è stato configurato in hardware in modo tale che il sequencer microcontrollato presente al suo interno esegua una scansione continua degli ingressi analogici. L'*ace_task* utilizza le funzioni di libreria del driver della periferica ACE. Dal punto di vista software, ad ogni canale è associato un handle di tipo *ace_channel_handle_t* che lo identifica univocamente e che viene quindi passato come parametro nelle chiamate alle funzioni del driver. Gli handle dei diversi canali sono concatenati in una lista. La funzione *ACE_get_first_channel()* ritorna l'handle del primo canale della lista.

Per ogni canale, quindi (come si vede dal Listato 4) la funzione *ace_task()* dapprima legge l'ultimo valore acquisito mediante chiamata alla funzione *ACE_get_ppe_sample*, quindi converte tale valore nella grandezza opportuna chiamando la funzione *update_ace_values()*; infine aggiorna l'handle per puntare al prossimo canale.

Il tipo di grandezza associato ad un canale può essere ricavato dall'handle relativo mediante chiamata alla funzione *ACE_get_chan-*

Tabella 2

Pin Number	Pin Function
1	spiDataIn
2	spiClkOut
3	spiCS_n
4	spiDataOut
5	DGND
6	VCC

Piedinatura dell'adapter per micro-SD.

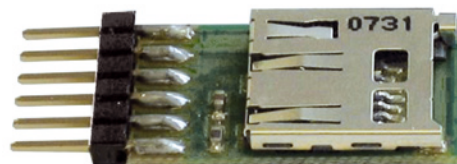


Fig. 4 - Un adapter per micro-SD.

nel_type. Come detto in precedenza, per i dettagli sull'utilizzo del driver dell'ACE si può far riferimento alla guida disponibile sul web [2]. La funzione *update_ace_values()* (si veda il codice scaricato) utilizza le funzioni *ACE_convert_to_mV*, *ACE_convert_to_Kelvin*, *ACE_convert_to_Fahrenheit* ed *ACE_convert_to_Celsius* usate per convertire il dato grezzo in valore ingegneristico sui canali di tensione e temperatura; tali funzioni sono di libreria.

La funzione *ACE_convert_to_uA* (si veda il file omonimo scaricato dal web) per la conversione del valore di corrente è stata invece adattata a partire da quella di libreria *ACE_convert_to_mA*; nel nostro esempio, infatti, il circuito di sensing di corrente della scheda A2F-EVAL-KIT è dimensionato in modo tale che la corrente misurata sia inferiore a 1 mA.

TEST DEL CONTROLLER DI MEMORIA SD

Passiamo ora a verificare il corretto funzionamento della periferica *sd_controller* per l'accesso alla memoria SD. Possiamo connettere una memoria micro-SD alla scheda A2F-EVAL-KIT utilizzando un adapter acquistabile on-line presso Mirifica [9]. L'adapter (Fig. 4) include un cartridge per micro-SD e riporta i segnali dell'interfaccia SPI unitamente alle linee di alimentazione e massa su di un header a 6 pin con passo standard, la cui pin-out è descritta, per semplicità, nella **Tabella 2**.

Realizziamo un adattatore femmina-femmina saldando tra loro due normali strip, così da poter connettere i pin da 1 a 4 dell'adapter ai pin da 5 a 2 (rispettando questa sequenza) dell'header J22 della scheda di sviluppo. I pin 5 e 6, rispettivamente massa ed alimentazione della memoria, potranno invece essere connessi a 4 ed 1 di J3. Si

Listato 5 - Applicativo di test della sezione analogica : *ACE_convert_to_uA()*

```

/*****
uint32_t ACE_convert_to_uA
(
    ace_channel_handle_t    channel_handle,
    uint16_t                sample_value
)
{
    uint32_t current = 0u;

    ASSERT(channel_handle < ACE_NB_OF_INPUT_CHANNELS);

    if((int32_t)channel_handle < ACE_NB_OF_INPUT_CHANNELS)
    {
        adc_channel_id_t channel_id;
        uint8_t current_monitor_idx;

        channel_id = g_ace_channel_desc_table[channel_handle].signal_id;
        ASSERT(channel_id < sizeof(channel_type_lut));
        if(CURRENT_CHANNEL == channel_type_lut[channel_id])
        {
            uint32_t resistor;
            uint32_t voltage;

            /* Compute index into g_ace_current_resistors[] from the current
             * channel number. This uses bit 2, 4 and 5 of the channel number
             * to derive the index as follows:
             * channel name : channel number : index
             * CM0          : 0x03          : 0
             * CM1          : 0x07          : 1
             * CM2          : 0x13          : 2
             * CM3          : 0x17          : 3
             * CM4          : 0x23          : 4
             * CM5          : 0x27          : 5
             */
            current_monitor_idx
                = (((uint8_t)channel_id & 0x04u) >> 2u) + (((uint8_t)channel_id &
                0x30u) >> 3u);

            if(current_monitor_idx < (uint8_t)ACE_NB_OF_CURRENT_MONITORS)
            {
                /* Retrieve the current sensing external resistor value from
                 * the ACE configuration data generated by the ACE configurator. */
                resistor = g_ace_current_resistors[current_monitor_idx];

                /* Compute mA current value taking into account the amplification
                 * factor of 50 used within the current monitor hardware. */
                voltage = ACE_convert_adc_input_to_mV(channel_handle, sample_value);
                voltage = voltage * 1000;
                current = (voltage / 50u) / resistor;
            }
        }
    }

    return current;
}

```

ricordi che la scheda A2F-EVAL-KIT è alimentata mediante porta USB, quindi prima di eseguire tali operazioni ricordatevi di sconnettere il cavo. Prima di riconnetterlo, è poi buona norma controllare di non aver creato cortocircuiti tra massa ed alimentazione. Il controller di memoria SD è una periferica custom integrata all'interno della FPGA Fabric; per potervi accedere abbiamo bisogno di scrivere un driver, seppur minimale. Come riferimento si può

prendere il progetto DataLogger/sw/Test_SDController contenuto nell'allegato al presente articolo. Il file *sd_controller_hw_platform.h* contenuto nell'allegato al presente articolo scaricabile dal nostro sito web, contiene le principali definizioni, tra cui, ad esempio, gli indirizzi relativi dei diversi registri e le istruzioni di accesso in lettura e scrittura a questi. La mappa dei registri del controller è derivata dal datasheet del core spiMaster scaricato da OpenCores. Per la

Listato 6 – Sd_controller.c : Init_SD()

```
uint32_t Init_SD(void)
{
    RegWrite(SPI_MASTER_APBWRAPPER_0 + SPI_TRANS_TYPE_REG, SPI_INIT_SD);
    RegWrite(SPI_MASTER_APBWRAPPER_0 + SPI_TRANS_CTRL_REG, SPI_TRANS_START);
    while (RegRead(SPI_MASTER_APBWRAPPER_0 + SPI_TRANS_STS_REG) == SPI_TRANS_BUSY);
    return RegRead(SPI_MASTER_APBWRAPPER_0 + SPI_TRANS_ERROR_REG);
}
```

La sequenza di istruzioni per la funzione DFS_ReadSector è invece la seguente :

- Set the SD block address registers by configuring registers SD_ADDR_7_0_REG, SD_ADDR_15_8_REG, SD_ADDR_23_16_REG and SD_ADDR_31_24_REG
 - Set SPI_TRANS_TYPE_REG = SPI_RW_READ_SD_BLOCK
 - Set SPI_TRANS_CTRL_REG = SPI_TRANS_START
 - Wait for SPI_TRANS_STS_REG != TRANS_BUSY
 - Check for SPI_TRANS_ERROR_REG[3:2] == READ_NO_ERROR
 - Read 512 bytes from SPI_RX_FIFO_DATA_REG and copy them to buffer. Increment buffer for each read access
 - Increment the SD block address by 512
 - Repeat previous steps for all the required sectors
- Il listato 8 mostra il codice corrispondente.

Listato 8. Sd_controller.c : DFS_ReadSector ()

```
uint32_t DFS_ReadSector(uint8_t unit, uint8_t *buffer, uint32_t sector, uint32_t count)
{
    int i, j;
    uint32_t status;
    uint32_t SD_ADDR;
    for (i=0; i<count; i++)
    {
        SD_ADDR=(sector + i) << 9;
        /* set the address */
        RegWrite(SPI_MASTER_APBWRAPPER_0 + SD_ADDR_7_0_REG, ( ( SD_ADDR >> 0 ) & 0xff ) );
        RegWrite(SPI_MASTER_APBWRAPPER_0 + SD_ADDR_15_8_REG, ( ( SD_ADDR >> 8 ) & 0xff ) );
        RegWrite(SPI_MASTER_APBWRAPPER_0 + SD_ADDR_23_16_REG, ( ( SD_ADDR >> 16 ) & 0xff ) );
        RegWrite(SPI_MASTER_APBWRAPPER_0 + SD_ADDR_31_24_REG, ( ( SD_ADDR >> 24 ) & 0xff ) );

        /* start the read */
        RegWrite(SPI_MASTER_APBWRAPPER_0 + SPI_TRANS_TYPE_REG, SPI_RW_READ_SD_BLOCK);
        RegWrite(SPI_MASTER_APBWRAPPER_0 + SPI_TRANS_CTRL_REG, SPI_TRANS_START);

        /* wait for write to complete */
        while (RegRead(SPI_MASTER_APBWRAPPER_0 + SPI_TRANS_STS_REG) == SPI_TRANS_BUSY);

        /* check status */
        status = RegRead(SPI_MASTER_APBWRAPPER_0 + SPI_TRANS_ERROR_REG);
        if ( status!=0 )
            return status;

        /* read data from rx fifo */
        for (j=0; j<512; j++)
        {
            *buffer=(unsigned char)(RegRead(SPI_MASTER_APBWRAPPER_0 + SPI_RX_FIFO_DATA_REG)&0xFF);
            buffer++;
        }
    }
    return 0;
}
```

nostra applicazione interessano le tre principali funzioni dichiarate all'interno del file .h:

- Init_SD, per l'inizializzazione della memoria così, ad esempio, da potervi accedere mediante porta SPI;
- DFS_ReadSector, per la lettura di n settori della memoria SD a partire da un indirizzo specificato (i dati letti sono copiati in un'area di memoria di sistema passata alla funzione mediante puntatore);
- DFS_WriteSector, per la scrittura di n settori della memoria SD a partire da un indirizzo

specificato (i dati da scrivere sono letti da un'area di memoria di sistema passata alla funzione mediante puntatore).

Il datasheet del controller spiMaster che stiamo utilizzando descrive la sequenza di istruzioni da fornire al core per implementare tali funzioni. Nel caso della funzione initSD, tale sequenza è :

- set SPI_TRANS_TYPE_REG = SPI_INIT_SD;
- set SPI_TRANS_CTRL_REG = SPI_TRANS_START;
- wait for SPI_TRANS_STS_REG != TRANS_BUSY;

- check for SPI_TRANS_ERROR_REG [1:0] == INIT_NO_ERROR.

Il codice corrispondente è riportato nel **Listato 6** (estratto dal file *DataLogger\sw\Test_SDController\sd_controller.c*).

La sequenza di istruzioni per la funzione DFS_ReadSector è invece la seguente:

- Set the SD block address registers by configuring registers SD_ADDR_7_0_REG, SD_ADDR_15_8_REG, SD_ADDR_23_16_REG and SD_ADDR_31_24_REG

Listato 7 – Sd_controller.c : DFS_ReadSector()

```
uint32_t DFS_ReadSector(uint8_t unit, uint8_t *buffer, uint32_t sector, uint32_t count)
{
    int i, j;
    uint32_t status;
    uint32_t SD_ADDR;
    for (i=0;i<count;i++)
    {
        SD_ADDR=(sector + i) << 9;
        /* set the address */
        RegWrite(SPIMASTER_APBWRAPPER_0 + SD_ADDR_7_0_REG, ( ( SD_ADDR ) & 0xff ) );
        RegWrite(SPIMASTER_APBWRAPPER_0 + SD_ADDR_15_8_REG, ( ( SD_ADDR >> 8 ) & 0xff ) );
        RegWrite(SPIMASTER_APBWRAPPER_0 + SD_ADDR_23_16_REG, ( ( SD_ADDR >> 16 ) & 0xff ) );
        RegWrite(SPIMASTER_APBWRAPPER_0 + SD_ADDR_31_24_REG, ( ( SD_ADDR >> 24 ) & 0xff ) );

        /* start the read */
        RegWrite(SPIMASTER_APBWRAPPER_0 + SPI_TRANS_TYPE_REG, SPI_RW_READ_SD_BLOCK);
        RegWrite(SPIMASTER_APBWRAPPER_0 + SPI_TRANS_CTRL_REG, SPI_TRANS_START);

        /* wait for write to complete */
        while (RegRead(SPIMASTER_APBWRAPPER_0 + SPI_TRANS_STS_REG) == SPI_TRANS_BUSY);

        /* check status */
        status = RegRead(SPIMASTER_APBWRAPPER_0 + SPI_TRANS_ERROR_REG);
        if ( status!=0 )
            return status;

        /* read data from rx fifo */
        for (j=0;j<512;j++)
        {
            *buffer=(unsigned char)(RegRead(SPIMASTER_APBWRAPPER_0+ SPI_RX_FIFO_DATA_REG)&0xFF);
            buffer++;
        }
    }
    return 0;
}
```

Listato 8 – Sd_controller.c : DFS_WriteSector()

```
uint32_t DFS_WriteSector(uint8_t unit, uint8_t *buffer, uint32_t sector, uint32_t count)
{
    int i, j;
    uint32_t status;
    uint32_t SD_ADDR;
    for (i=0;i<count;i++)
    {
        /* copy the data to tx fifo */
        for(j=0;j<512;j++)
        {
            RegWrite( SPIMASTER_APBWRAPPER_0 + SPI_TX_FIFO_DATA_REG,(uint32_t)(*buffer) );
            buffer++;
        }

        /* set the address */
        SD_ADDR=(sector + i) << 9;
        RegWrite(SPIMASTER_APBWRAPPER_0 + SD_ADDR_7_0_REG, ( ( SD_ADDR ) & 0xff ) );
        RegWrite(SPIMASTER_APBWRAPPER_0 + SD_ADDR_15_8_REG, ( ( SD_ADDR >> 8 ) & 0xff ) );
        RegWrite(SPIMASTER_APBWRAPPER_0 + SD_ADDR_23_16_REG, ( ( SD_ADDR >> 16 ) & 0xff ) );
        RegWrite(SPIMASTER_APBWRAPPER_0 + SD_ADDR_31_24_REG, ( ( SD_ADDR >> 24 ) & 0xff ) );

        /* start the write */
        RegWrite(SPIMASTER_APBWRAPPER_0 + SPI_TRANS_TYPE_REG, SPI_RW_WRITE_SD_BLOCK);
        RegWrite(SPIMASTER_APBWRAPPER_0 + SPI_TRANS_CTRL_REG, SPI_TRANS_START);

        /* wait for write to complete */
        while (RegRead(SPIMASTER_APBWRAPPER_0 + SPI_TRANS_STS_REG) == SPI_TRANS_BUSY);

        /* check the status */
        status = RegRead(SPIMASTER_APBWRAPPER_0 + SPI_TRANS_ERROR_REG);
        if ( status!=0 )
            return status;
    }
    return 0;
}
```

- Set SPI_TRANS_TYPE_REG = SPI_RW_READ_SD_BLOCK
- Set SPI_TRANS_CTRL_REG = SPI_TRANS_START
- Wait for SPI_TRANS_STS_REG
- Check for SPI_TRANS_ERROR_REG[3:2] == READ_NO_ERROR
- Read 512 bytes from SPI_RX_FIFO_DATA_REG and copy them to buffer. Increment buffer for each read access
- Increment the SD block address by 512

Listato 9 – Applicativo di test della libreria DosFS

```
int main()
{
    uint8_t sector[SECTOR_SIZE];
    uint32_t pstart, psize, i;
    uint8_t pactive, ptype;
    char CharBuffer[512];
    VOLINFO vi;
    uint32_t cache;
    FILEINFO fi;
    uint32_t LValue;

    /* Init system */
    MSS_WD_disable( );

    /* Init system */
    printf("Entering Main ( ) \n\r");
    LValue=RegRead(SPIMASTER_APBWRAPPER_0 + SPI_MASTER_VERSION_REG );
    printf("SD Controller Version Number : %x \n\r", (unsigned int) LValue);

    if ( Init_SD()!=0 )
    {
        printf("Cannot initialize device\n\r");
        return -1;
    }

    // Obtain pointer to first partition on first (only) unit
    pstart = DFS_GetPtnStart(0, sector, 0, &pactive, &ptype, &psize);
    if (pstart == 0xffffffff) {
        printf("Cannot find first partition\n\r");
        return -1;
    }

    printf("Partition 0 start sector 0x%-08.8lX active %-02.2hX type %-02.2hX size %-08.8lX\n\r", pstart, pactive, ptype, psize);

    if (DFS_GetVolInfo(0, sector, pstart, &vi)) {
        printf("Error getting volume information\n\r");
        return -1;
    }
    printf("Volume label '%-11.11s'\n", vi.label);
    printf("%d sector/s per cluster, %d reserved sector/s, volume total %d sectors.\n\r ", vi.secpersclus, vi.reservedsecs, vi.numsecs);
    printf("%d sectors per FAT, first FAT at sector %#d, root dir at %#d.\n\r ", vi.secpersfat, vi.fat1, vi.rootdir);
    printf("(For FAT32, the root dir is a CLUSTER number, FAT12/16 it is a SECTOR number)\n\r ");
    printf("%d root dir entries, data area commences at sector %#d.\n\r ", vi.rootentries, vi.dataarea);
    printf("%d clusters (%d bytes) in data area, filesystem IDd as ", vi.numclusters, vi.numclusters * vi.secpersclus * SECTOR_SIZE);
    if (vi.filesystem == FAT12)
        printf("FAT12.\n\r");
    else if (vi.filesystem == FAT16)
        printf("FAT16.\n\r");
    else if (vi.filesystem == FAT32)
        printf("FAT32.\n\r");
    else
        printf("[unknown]\n\r");

    /* File write test */
    if (DFS_OpenFile(&vi, "wtest.raw", DFS_WRITE, sector, &fi))
    {
        printf("error opening file\n\r");
        return -1;
    }

    CharBuffer[0]="\0";
    strcat(CharBuffer, "Hello world \n\r");
    DFS_WriteFile(&fi, sector, WCharBuffer, &cache, strlen(WCharBuffer) );

    /* read-back data */
    printf("Readback test\n\r");

    DFS_Seek(&fi, 0, sector);
    CharBuffer[0]="\0";
    DFS_ReadFile(&fi, sector, CharBuffer, &i, fi.filelen);
    printf(CharBuffer);

    return 0;
}
```

Listato 10 – Applicativo del DataLogger

- Repeat previous steps for all the required sectors.

Il **Listato 7** mostra il codice corrispondente.

Infine, l'accesso in scrittura alla memoria è gestito dalla seguente sequenza di istruzioni:

- Write 512 bytes from buffer to SPI_TX_FIFO_DATA_REG. Increment buffer for each access.
- Set the SD block address registers by configuring registers SD_ADDR_7_0_REG, SD_ADDR_15_8_REG, SD_ADDR_23_16_REG and SD_ADDR_31_24_REG
- Set SPI_TRANS_TYPE_REG = SPI_RW_READ_SD_BLOCK
- Set SPI_TRANS_CTRL_REG = SPI_TRANS_START
- Wait for SPI_TRANS_STS_REG != TRANS_BUSY
- Check for SPI_TRANS_ERROR_REG[5:4] == WRITE_NO_ERROR
- Increment the SD block address by 512
- Repeat previous steps for all the required sectors.

Il codice corrispondente è riportato nel **Listato 8**.

Per testare il corretto funzionamento del controller di memoria SD integrato nella FPGA possiamo quindi scrivere un semplice programma di esempio, utilizzando le funzioni fornite dal driver appena creato.

Nel creare un nuovo progetto sotto Soft Console, non dimentichiamo di applicare le configurazioni descritte in precedenza per quanto concerne simboli e script utilizzati durante la compilazione ed il linking dell'applicativo. Come riferimento per il test della memoria, si veda il file *DataLogger/sw/Test_SDController/main.c* all'interno dell'allegato all'articolo. Tale applicativo inizializza prima il sistema e quindi la memoria connessa, poi passa a scrivere due

```
/******  
/* Entry to Main form user bootcode */  
/******  
int main()  
{  
  
.....  
    /* Initialization all necessary hardware components */  
    init_system();  
  
    /* Configure the interrupt the processor to acquire data */  
    MSS_TIM1_load_background(g_TimerReloadValue);  
    MSS_TIM1_enable_irq();  
    MSS_TIM1_start();  
  
    /* Mount the file system for the SD memory card */  
    pstart = DFS_GetPtnStart(0, sector, 0, &pactive, &ptype, &psize);  
    if (pstart == 0xffffffff) {  
        printf("Cannot find first partition\n");  
        return -1;  
    }  
  
    /* Open the file in write mode to store the data */  
    if (DFS_OpenFile(&vi, "logfile.txt", DFS_WRITE, sector, &fi))  
    {  
        printf("error opening file\n");  
        return -1;  
    }  
  
    /* Periodically acquire the the analog inputs and store to file */  
    while (1);  
  
    return 0;  
}
```

settori contigui con un pattern di test predefinito, a rileggerli e a verificarne l'esatta programmazione.

UNA LIBRERIA PER LA GESTIONE DEL FILE SYSTEM

Per gestire la memorizzazione dei file sulla memoria SD utilizzeremo la libreria DosFS sviluppata da Lewin Edwards che implementa un file system FAT standard per sistemi embedded.

Il codice sorgente può essere scaricato dal web all'indirizzo indicato al punto [10] nei riferimenti. La libreria supporta file system FAT12, FAT16 e FAT32 su dispositivi con capacità di memoria fino a 2048 GB con o senza MBR; gestisce partizioni multiple sullo stesso supporto e supporta le sotto-directory. Non utilizza variabili globali e non necessita di funzioni di allocazione dinamica della memoria, né implementa funzioni di formattazione del supporto; un'utilità per tale scopo è *mkdosfs*, inclusa nel package *dosfstools* sotto Linux. Il porting

per Windows è reperibile sul web, ad esempio, all'indirizzo [11]. DosFS rappresenta un buon compromesso tra funzionalità e dimensioni del codice compilato, risultando ideale per applicazioni su piattaforme embedded con ridotta capacità di memoria programmi. Il porting della libreria su una generica piattaforma è piuttosto semplice e richiede la scrittura delle due funzioni *DFS_ReadSector* (*uint8_t unit, uint8_t *buffer, uint32_t sector, uint32_t count*) e *DFS_WriteSector* (*uint8_t unit, uint8_t *buffer, uint32_t sector, uint32_t count*) per l'accesso in lettura e scrittura a settori consecutivi della memoria. Nel nostro caso possiamo utilizzare le funzioni descritte nel precedente paragrafo. Per verificare il corretto funzionamento della libreria sul nostro sistema, possiamo usare un applicativo di esempio del tipo incluso nel progetto *DataLogger/sw/Test_DosFS/main.c*; un estratto di questo è riportato nel **Listato 9**.

Listato 11 - La funzione `sd_task()`

```
void sd_task()
{
    char CharBuffer[512];
    char c[512];

    ftoa(real_voltage_value, CharBuffer);
    strcat(CharBuffer, ",");
    ftoa(real_current_value, c);
    strcat(CharBuffer, c);
    strcat(CharBuffer, ",");
    ftoa(real_temperature_value_tc, c);
    strcat(CharBuffer, c);
    strcat(CharBuffer, newline);

    DFS_WriteFile(&fi, sector, CharBuffer, &cache, strlen(CharBuffer));
}
```

L'applicativo dapprima inizializza il sistema e la memoria SD, quindi monta il file system di quest'ultima; poi apre un file in scrittura e vi scrive un messaggio di testo, riposiziona il puntatore della posizione corrente nel file all'inizio e legge la stringa scritta, inviandola, infine, come messaggio sulla porta seriale.

APPLICATIVO PER DATALOGGER

Dopo aver provato singolarmente le diverse funzionalità del nostro progetto e portato sulla nostra piattaforma la libreria DosFS per l'accesso alla memoria SD mediante file system standard, non ci resta che mettere insieme il tutto e scrivere il codice applicativo del nostro DataLogger.

Un estratto del sorgente è riportato nel listato 10; il file completo si trova nel progetto *DataLogger/sw/data_logger/main.c* contenuto nell'allegato all'articolo.

Come si vede, dapprima il sistema viene inizializzato, quindi sono abilitate le interruzioni.

Ad intervalli regolari cadenzati dalle interruzioni generate dal timer, viene chiamata al funzione *Timer1_IRQHandler*, la quale richiama in sequenza i task:

- `ace_task()`, per l'acquisizione dei canali di ingresso;
- `sd_task()`, per la memorizzazione dei valori opportunamente convertiti in un file di testo.

I dati sono salvati in formato comma-separated, in modo tale che il file potrà essere successivamente importato in un foglio di calcolo elettronico per l'elaborazione ed, eventualmente, la rappresentazione degli andamenti. La funzione *ace_task()* è identica a quella riportata nel Listato 4. Un estratto del codice sorgente della funzione *sd_task()* è invece riportato nel Listato 11.

CONCLUSIONI

In questo tutorial dedicato all'FPGA Smart Fusion, abbiamo

proposto un esempio pratico di come si possa realizzare un data logger utilizzando le funzionalità dei dispositivi FPGA di Actel.

L'esempio ha valore meramente introduttivo e quindi è volutamente piuttosto semplice in alcuni suoi aspetti. Riteniamo, in ogni caso, che siano stati toccati da vicino tutti gli aspetti della progettazione con FPGA SmartFusion e che quindi si abbiano tutti gli strumenti per sviluppare applicazioni più complesse.

A seconda del caso specifico, potranno essere usate, ad esempio, le funzioni di gestione del sequencer integrato nell'ACE per configurare diverse sequenze di scansione dei canali di ingresso in funzione della modalità operativa, oppure potranno essere definite delle soglie in base alle quali abilitare la generazione automatica di flag di controllo.

Allo stesso modo, potranno essere introdotte nuove funzionalità specifiche mediante IP core implementati all'interno della FPGA Fabric, ad esempio per l'elaborazione on-line dei dati acquisiti. Il tutto senza dimenticare, poi, che i dispositivi SmartFusion dispongono anche di convertitori DAC e comparatori, che non abbiamo avuto modo di presentare negli esempi fatti in questo tutorial. Insomma, come spesso accade, il limite vero alle applicazioni, in una certa misura, sta nella fantasia del progettista!

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>
[8] <http://www.mirifica.it/store/51-trioflex-microsd-adapter.html> [9] http://www.actel.com/documents/CP2102_driver.zip
[10] <http://www.larwe.com/zws/products/dosfs/index.html> [11] <http://www1.mager.org/mkdosfs/>

Alla ricerca della "killer application"

Si apre tra poche settimane la terza edizione dell'unica manifestazione italiana dedicata alla robotica di servizio, Robotica 2011; l'appuntamento è a Milano dal 16 al 19 novembre. A questa manifestazione partecipano numerose aziende, enti di ricerca

e associazioni che presentano i loro progetti e prodotti robotici per le applicazioni più varie, dai robot utilizzati nella cosiddetta Life Assistance (medicina e riabilitazione) a quelli per Home Automation (pulizia, sicurezza, cucina), dai robot impiegati per

il monitoraggio ambientale a quelli per l'entertainment (personal robot, educazione, ecc.).

Secondo il recente rapporto "World Robotics 2009 Service Robots" dell'IFR (International Federation of Robotics), il mercato dei robot per impiego non industriale aveva un valore – a fine 2008 – di 11,2 miliardi di dollari, una cifra consistente ma ancora lontanissima dalle potenzialità di questo mercato. Anche perché molti dei robot classificati in questa categoria sono piuttosto distanti dal nostro quotidiano essendo destinati ad impieghi militari, per la sicurezza o utilizzati in agricoltura. Gli unici robot per impieghi "civili" che hanno avuto un discreto successo e che rappresentano il 10% circa di questo mercato sono quelli destinati alla pulizia dei pavimenti, i cosiddetti "cleaning robot": sicuramente un buon successo ma che non rappresenta ancora quella "killer application" di cui tutti sono alla ricerca e che potrebbe fare compiere un vero salto di qualità a questo mercato. Se ne discuterà sicuramente durante i numerosi convegni all'interno della manifestazione milanese e la presenza del Prof. Yoshiyuki Sankai, l'ideatore dell'esoscheletro Hal (Hybrid Assistive Limb) potrebbe essere un segno di quella che potrebbe essere, finalmente, l'applicazione vincente. Sono in molti, infatti, a ritenere che con l'invecchiamento della popolazione, uno dei problemi più seri da risolvere sarà quello della deambulazione. I prossimi anni ci diranno se avranno ragione.



IN QUESTO NUMERO

108 ■ NEWS ED EVENTI

Torna a Fieramilano la terza edizione di Robotica, l'unico evento italiano dedicato alla robotica umanoide e di servizio che quest'anno presenta numerose novità tra le quali l'area dedicata agli inventori e l'Arduino Day.

111 ■ DRONE ZONE QUADRICOTTERO, SOFTWARE E TEST DI VOLO

Proseguiamo la descrizione del nostro quadricottero occupandoci della configurazione software, della impostazione dei PID, della pianificazione missioni con l'APM Mission Planner e del funzionamento della telemetria. Descriviamo anche i primi test di volo e le possibili applicazioni pratiche.

123 ■ VERSO NESSIE 2012 CONSIDERAZIONI SU UN INSEGUITORE DI LINEA A 128 BIT

Iniziamo il percorso verso la RoboCup Junior 2012 aggiornando il progetto che ha partecipato alla precedente edizione. In questo articolo viene illustrato l'impiego di un nuovo sensore di linea in grado di offrire prestazioni decisamente superiori.



Al via ROBOTICA

2011

di Arsenio Spadoni



Torna a Fieramilano la terza edizione di Robotica, l'unico evento italiano dedicato alla robotica umanoide e di servizio che quest'anno presenta numerose novità tra le quali l'area dedicata agli inventori e l'Arduino Day.



La terza edizione di Robotica, in programma dal 16 al 19 novembre 2011 a Fieramilano Rho, si preannuncia particolarmente interessante. Tra le novità: I-Do Robot, il primo robot italiano per la telepresenza; Nao, uno dei più diffusi robot al mondo; Roborama, il primo robot show; l'area dedicata agli inventori, l'Arduino Day e la prestigiosa partecipazione alle conferenze del prof. Yoshiyuki Sankai, il "padre" dell'esoscheletro Hal. Robotica, l'unico evento fieristico italiano nel campo della robotica umanoide e di servizio organizzato da Artenergy nell'ambito di HTE-hi.tech.expo, la fiera delle tecnologie innovative, continua a crescere e ad arricchirsi di contenuti di grande interesse. Quest'anno, ad esempio, per la prima volta interverrà alle conferenze il prof. Yoshiyuki Sankai della University of Tsukuba, uno dei più prestigiosi esperti del settore, che giovedì 17 novembre presenterà l'esoscheletro Hal (Hybrid Assistive Limb), de-



Arduino Day

finito dallo stesso Sankai "La prima tuta robot al mondo che, utilizzando i segnali umani, permette di far funzionare i robot". Un'altra grande novità, che apre nuovi scenari per il futuro, è I-Do Robot di Nuzoo Robotics, il primo robot italiano per la telepresenza, termine che definisce la possibilità di compiere azioni a distanza quali interagire con persone e mettersi in relazione con oggetti. I-Do Robot, grazie a sensori e alle sue caratteristiche di autonomia e mobilità, permette anche la sorveglianza di ambienti in remoto.

A Robotica 2011 verrà inoltre presentato Robot Companions for Citizens, il più grande progetto europeo al servizio dei cittadini, che propone un nuovo approccio multidisciplinare verso i robot per utilizzarli nella società in modo da migliorare la qualità della vita delle persone. Per raggiungere questo obiettivo il consorzio di coordinamento del progetto, CA-RoboCom, adotta una strategia che mira a coinvolgere il mondo della scienza e della tecnologia, ma anche quello della finanza, della politica e dell'industria. A capo di questo progetto è il prof. Paolo Dario della Scuola Superiore Sant'Anna di Pisa, una fra le eccellenze della ricerca italiana. Un'altra interessante iniziativa della Scuola Superiore Sant'Anna è il Progetto HydroNet, sostenuto dalla Commissione Europea, che utilizza robot marini sensorizzati per migliorare il monitoraggio dell'ambiente.



Venerdì 18 novembre, dalle 11:45 alle 13:30 presso la Sala Asimov dell'Area Robotica (Pad. 3) verranno presentati alcuni progetti di robot e di dispositivi vari realizzati da studenti di diversi istituti utilizzando la piattaforma di prototipazione elettronica open-source Arduino. In quest'ambito, Massimo Banzi parlerà del futuro del progetto Arduino mentre altri oratori presenteranno alcune esperienze didattiche realizzate con questa piattaforma. Ci saremo anche noi per raccontare la nostra recente esperienza del settore degli UAV che ha portato alla realizzazione di due velivoli autonomi grazie alla piattaforma ArduPilot ed al relativo software open-source sviluppato dalle comunità di appassionati.

Nel nostro stand sarà possibile vedere in funzione, durante l'intero periodo della manifestazione, questo ed altri prototipi realizzati con le schede Arduino.



Grandi attrattive della manifestazione saranno i robot Nao, tra i più diffusi al mondo, che si esibiranno in fiera ballando sulle suggestive note del Bolero di Ravel grazie alla tecnologia sofisticata di Aldebaran Robotics, e Roborama, spettacolo teatrale di marionette robotiche, originale rappresentazione robotizzata li-



Area Inventors

Una grande novità di Robotica 2011 è l'Area Inventors, che offre agli inventori l'opportunità di presentare agli operatori del settore le realizzazioni frutto del loro ingegno. Tra le attrazioni che sarà possibile vedere in fiera, ad esempio, il prototipo di una mongolfiera Wi-Fi teleguidata da iPad o cellulare tramite controllo touch. La mongolfiera è dotata di tre motori con LED, che possono produrre 16 milioni di colori e creare effetti psichedelici e stroboscopici a tempo di musica. Un'altra attrazione dell'area riservata agli inventori sarà il robot tastierista di Teotronica, che con 19 dita può suonare qualsiasi genere musicale.

Se siete interessati ad esporre gratuitamente un vostro robot o più in generale un vostro progetto in questo ambito, ecco la mail di contatto: maurizio.lozza@zeroemission.eu



beramente ispirata alle vicende del Mago di Oz. Tra le sorprese di Roborama anche un robot che suona la batteria e il divertente balletto di cinque cagnolini robotici. Il RoboticsLab dell'Università di Palermo, che si propone di creare robot dotati di potenti capacità percettive e cognitive, dei "robot umanoidi emozionali" in grado di apprendere tramite l'imitazione, interverrà con il progetto Geminoid, già utilizzato nello spettacolo teatrale "Sayonara", recentemente presentato a Palermo.

Una grande novità di Robotica 2011 è l'**Area Inventors**, che offre agli inventori giovani e meno giovani l'opportunità di presentare agli operatori del settore le realizzazioni frutto del loro ingegno. Tra le attrazioni che sarà possibile vedere in fiera, ad esempio, il prototipo di una mongolfiera Wi-Fi teleguidata da iPad o cellulare tramite controllo touch. La mongolfiera è dotata di tre motori con led, che possono produrre 16 milioni di colori e creare effetti psichedelici e stroboscopici a tempo di musica. Un'altra attrazione dell'area riservata agli inventori sarà il robot tastierista di Teotronica, che con 19 dita

può suonare qualsiasi genere musicale. Giovedì 17 novembre, nell'ambito di Robotica, Siri (Associazione Italiana di Robotica e Automazione) consegnerà il Premio Antonio d'Auria, istituito per stimolare la creatività nello studio di progetti e prototipi di dispositivi meccatronici innovativi di ausilio a disabili, mentre venerdì 18 novembre si svolgerà, con la presenza di Massimo Banzi, l'**Arduino Day**, durante il quale verranno presentati numerosi prototipi di robot realizzati da studenti di diversi istituti utilizzando le schede di prototipazione open-source Arduino. Tutti i giorni di fiera sarà possibile ammirare la mostra fotografica di Massimo Brega, una rassegna di fotografie, unica nel suo genere, che rappresenta robot ispirati ad animali. Per la prima volta Robotica ospiterà anche un'area dimostrativa della **Rete Robocup Junior Italia**, le gare nazionali di rescue (salvataggio), dance e calcio con robot tra studenti di licei e istituti superiori che hanno inserito la robotica come metodo di studio e formazione.

A Robotica 2011 partecipa anche la Scuola media "Quintino di Vona" con esperimenti di minidomotica, che avvengono tramite il controllo di alcune funzioni domestiche basate su un calcolatore dotato di scheda MZ di Nuzoo. In questo modo, ad esempio, si può aprire automaticamente il portellone di un garage e accendere le luci. Tutto questo sarà in mostra a Robotica 2011, dove sarà possibile vedere anche i robot di salvataggio, le soluzioni robotiche per le persone diversamente abili e seguire le interessanti conferenze, che affronteranno temi quali, ad esempio, la roboetica, l'uomo bionico, i veicoli intelligenti e gli aerei robotizzati.

Robotica 2011 si svolgerà in contemporanea con HTE-hi.tech.expo, la fiera delle tecnologie innovative che rappresenta settori quali vuoto, ultravacuo e coating, fotonica, laser e optoelettronica, magneti e superconduttori, nanotecnologie, batterie e sistemi di ricarica, tecnologie della visione.

Alla manifestazione **saremo presenti anche noi** con i **prototipi dei più recenti progetti** robotici realizzati. □

QUADRICOTTERO, SOFTWARE E TEST DI VOLO

Proseguiamo la descrizione del nostro quadricottero occupandoci della configurazione software, della impostazione dei PID, della pianificazione missioni con l'APM Mission Planner e del funzionamento della telemetria. Descriviamo anche i primi test di volo e le possibili applicazioni pratiche.

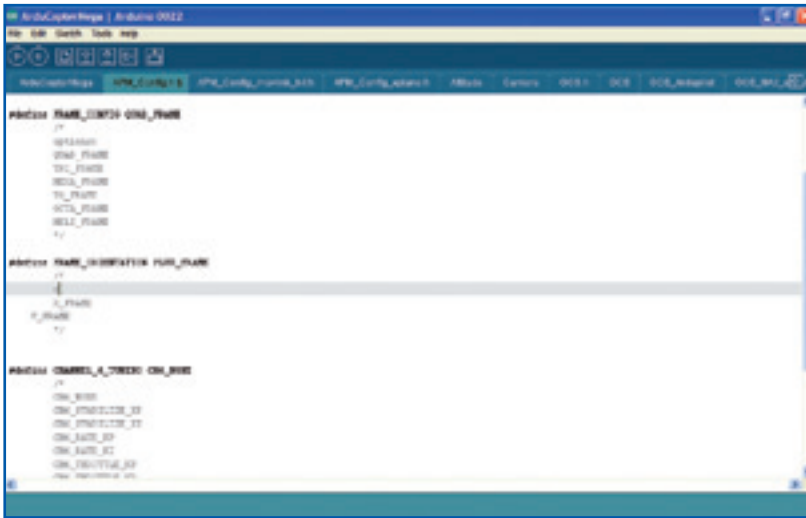


di MARCO MAGAGNIN

Dopo esserci occupati dell'hardware e della realizzazione del nostro quadricottero, vediamo ora come inizializzare e configurare il software ArduCopterMega. A questo proposito possiamo seguire due differenti strade: la via semplificata consiste nello scaricare unicamente il software APM Mission Planner ed utilizzarlo anche per caricare il firmware sul modulo APM; la seconda è quella classica che utilizza l'IDE di Arduino per compilare ed eseguire l'upload del software. Vi diamo le indicazioni per entrambe le possibilità, nel nostro caso abbiamo preferito la via "classica" anche per dare una occhiata al software ed eseguire una verifica di compilazione. Cominciamo a scaricare tutti i software necessari. L'IDE di sviluppo di Arduino, è disponibile gratuitamente dal sito omonimo (<http://arduino.cc/en/Main/Software>), la versione da installare è la 022 o successiva. Dal sito <http://code.google>.

[com/p/ardupilot-mega/downloads/list](http://code.google.com/p/ardupilot-mega/downloads/list) scarichiamo la versione più recente del sorgente di Arducopter `Arducopter.xxxxx.zip` e dal sito <http://code.google.com/p/ardupilot-mega/downloads/list> il software più recente della applicazione APM Mission Planner `"APM Mission Plannerxxxxx.zip"`.

Scompattate i due file nel disco C: rispettando le directory originali. Colleghiamo la presa USB del modulo APM al PC con un cavo USB. Apriamo l'IDE di Arduino, in Tools-> Board selezioniamo Arduino Mega (ATmega2560), in Tools-> Serial Port selezioniamo la porta COM alla quale è collegato APM ed infine in File -> Preferences nella Sketchbook location inseriamo la directory che contiene il software di APM. Per rendere attive le impostazioni è necessario chiudere e riaprire l'IDE. Aprite il file `"ArduPilotMega.pde"` compilate ed eseguite l'upload sul modulo APM. Se lo desiderate potete configurare il tipo

Fig. 1**Fig. 2****Fig. 3**

di multicottero ("QUAD") e di frame ("+") modificando i relativi parametri nel modulo APM_Config.h, oppure potete eseguire questa configurazione in seguito dalla applicazione APM Mission Planner. Nel caso preferiate scegliere la via dell'installazione diretta del firmware da APM Mission Planner, collegate il modulo APM alla porta USB come indicato sopra, aprite la applicazione, in Options impostate la porta COM e la velocità. Utilizzate la funzione Action -> Tools -> Upload Firmware, e selezionate la tipologia di modello. A questo punto è possibile eseguire la configurazione iniziale del modulo APM utilizzando la modalità Common Line Interface (CLI) dal terminale della APM Mission Planner. Accendete la trasmettente del radiocomando con APM collegato al PC con il cavo USB, ed aprite la applicazione APM Mission Planner. Per attivare la CLI è necessario portare l'interruttore a slitta sul modulo APM in posizione Setup, ovvero verso il connettore dei cavi dei servocomandi ed eseguire il Reset del modulo con l'apposito pulsante. Nel primo box in alto a destra selezionate "ArduCopter2", selezionate nel secondo box la porta COM associata al cavo USB e nel terzo la velocità di comunicazione a 115200 bps. Infine aprite la funzione "Terminal" della Mission Planner. L'interfaccia a linea di comando si presenta come una applicazione telnet, ed è caratterizzata da una sintassi di comandi organizzati in una gerarchia di menù.

Digitando "Help" al prompt della finestra Terminal otteniamo un menù di funzioni disponibili. Digitando uno dei comandi si entra nel sottomenù corrispondente, la lista delle funzioni disponibili è sempre visibile digitando "Help", digitando "Exit" si torna indietro di un livello.

Il primo livello offre le seguenti funzioni:

- "logs": Gestione del log on line,
- "setup": Configurazione iniziale,
- "test": Lista e verifica impostazioni.

Digitiamo "Setup" e poi "help" per ottenere le funzioni disponibili in Setup; ci appare la lista delle funzioni.

- erase: Quando viene installata una nuova versione di ArducopterMega la funzione azzerla la EEPROM,
- reset: Inizializza la EEPROM con i valori di default,

- **radio:** Determina e registra i valori limite delle escursioni di tutti i canali della ricevente RC,
- **motors:** Permette la calibrazione interattiva dei regolatori e dei motori. Entrati nella funzione, collegate la batteria e muovete gli stick della trasmettente RC in tutte le direzioni; ad ogni movimento deve corrispondere la rotazione del motore che corrisponde alla direzione dello stick attivato. Muovendo lo stick dell'elevatore verso l'alto deve corrispondere la rotazione del motore frontale, il movimento dello stick verso destra deve attivare il motore destro,
- **level:** Calibra gli accelerometri, richiede che il modulo IMU sia perfettamente orizzontale,
- **modes:** Imposta le modalità di volo assegnate a ciascuna posizione dell'interruttore CNTRL. Il movimento dello stick degli alettoni verso destra mostra la modalità successiva nell'elenco, il movimento verso sinistra quella precedente. Il tasto invio salva le modalità impostate.
- **frame:** Imposta la configurazione del multicottero (X, +, tri, Hex, y6),
- **current:** Abilita il sensore di corrente di AttoPilot (on, off, milliamper ora),
- **sonar:** Abilita il sonar. Il sensore si collega al connettore del tubo di Pitot che è un ingresso analogico (on, off),
- **compass:** Abilita il magnetometro (on, off),
- **declination:** Imposta il valore della declinazione magnetica locale,
- **show:** espone una lista di tutte le impostazioni.

La prima inizializzazione richiede obbligatoriamente la seguente sequenza di comandi, disponibili dal menù "Setup".

- "Reset", per caricare in EEPROM i valori iniziali di tutti i parametri; rispondere "Y" alla richiesta di conferma,
- Eseguire il reboot di APM con l'apposito pulsante,
- Impostate il tipo di frame con il comando "frame +",
- Ora deve essere calibrato l'IMU rispetto al riferimento orizzontale. Con una piccola bolla circolare appoggiata al modulo APM sistemiamo il quadricottero fino a

Fig. 4

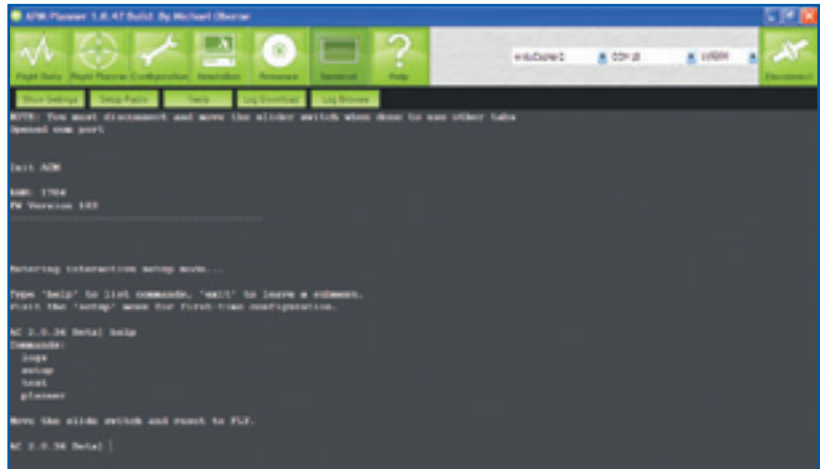


Fig. 5



quando la bolla ci indichi una orizzontalità perfetta,

- Dal menù "Setup" eseguiamo il comando "Level",
- "Radio", per calibrare i valori relativi alla corsa dei comandi del radiocomando, stick e interruttore; muoviamo gli stick della radio a fondo corsa in tutte le direzioni, spostiamo l'interruttore di controllo in tutte le posizioni, poi premiamo il tasto "Enter" per confermare i valori impostati,
- "Modes" questo comando ci permette di configurare le modalità di volo associate all'interruttore a tre posizioni sulla trasmettente; posizioniamo l'interruttore sulla prima posizione, e verifichiamo che sia impostato a "Stabilize", se non è così

Fig. 6



l'interruttore potrebbe essere invertito e "Stabilize" trovarsi nella posizione opposta dell'interruttore. Verificate e scegliete

come usare l'interruttore, eventualmente utilizzando il "reverse" sulla trasmettente. Posizionate l'interruttore sulla seconda po-

sizione, utilizzando lo stick degli alettoni ogni spostamento a fondo corsa verso destra ci permette di cambiare la modalità di volo associata; una volta impostata quella prescelta passiamo alla terza posizione e ripetiamo l'operazione. Al termine confermiamo il tutto con il tasto "Enter",

- Abilitate il magnetometro sempre dal menù "Setup" digitando "Compass on"; verificate digitando "Compass" lo stato del magnetometro,
- Se avete installato il sonar dobbiamo abilitarlo digitando "Sonar on"; verificate lo stato del sonar digitando "Sonar",
- Uscite dalla modalità CLI riportando la levetta dell'interruttore a slitta in posizione di volo, verso i DIP switch, e resettate APM con l'apposito pulsante,
- Ora calibrate i regolatori per memorizzare gli estremi della corsa



Fig. 7

Fig. 8

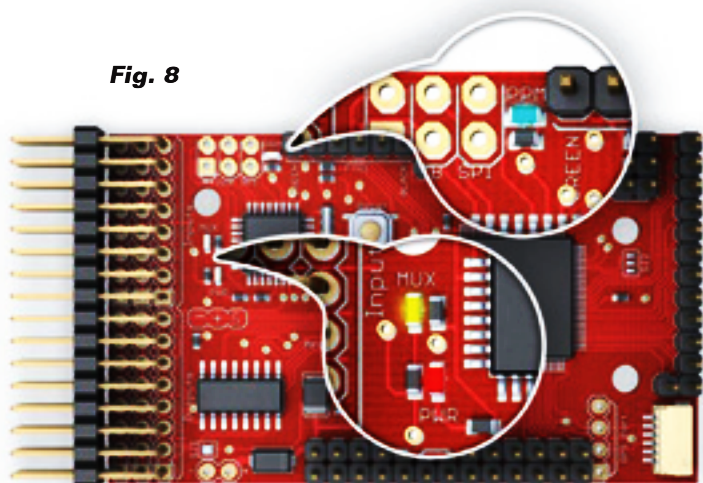
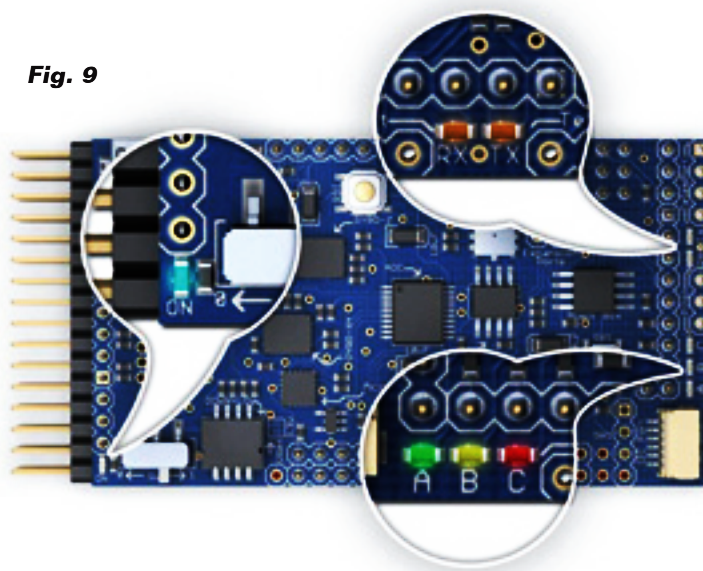


Fig. 9



degli stick del throttle. Verificate di non avere montato le eliche sui motori, in caso contrario smontatele. Verificate che la batteria di alimentazione non sia connessa. Sempre con la trasmittente accesa portate lo stick del throttle a fondo corsa (full throttle). Sentirete alcuni "bip". Poi collegate la batteria di alimentazione del modello. Abbassate lo stick del throttle fino al minimo. Attendete alcuni istanti e poi scollegate la batteria.

Dal menù "Test" è possibile accedere ad un elenco di funzioni che ci permettono di verificare il funzionamento dei vari componenti installati; ciascuna funzione mostra, una volta richiamata, i valori che vengono letti dai corrispondenti ingressi; per uscire dalla funzione premere il tasto "Enter".

- pwm: Mostra i valori pwn degli otto canali radio,
- radio: Mostra i valori degli otto canali radio in gradi x 100 o altri valori (Vedi il modulo software radio.pde),
- failsafe: (Per usi futuri),
- stabilize: Permette di verificare la stabilizzazione mostrando sul terminale i valori di debug (Fate attenzione perchè i motori possono entrare in funzione,
- gps: Mostra i dati GPS; per ave-

re valori significativi è necessario essere in un luogo dove il GPS possa ricevere i satelliti. Il GPS può richiedere alcuni minuti per fornire valori validi,

- adc: Mostra i valori letti dal convertitore analogico/digitale,
- imu: Mostra gli angoli calcolati dall'IMU,
- battery: Mostra il voltaggio della batteria,
- tune: (Per usi futuri),
- current: Mostra voltaggio e corrente letti da un sensore di corrente AttoPilot, se installato
- relay: Commuta il relè,
- waypoints: Stampa i dettagli della missione memorizzata in EEPROM,
- altitude: Mostra i valori in uscita dal sensore di pressione barometrica,
- sonar: Mostra i valori in uscita dal sonar (se collegato ed abilitato),
- compass: Mostra i valori di bussola in gradi (0 = Nord),
- xbee: Invia una sequenza di dati verso il modulo XBee, utile per prove di portata,
- eedump: Mappa della memoria EEPROM,
- rawgps: Mostra le stringhe native ricevute dal GPS,
- mission: Memorizza in EEPROM una missione di test (null, 'wp'). La opzione 'wp' porta il modello

quindici metri verso Nord e poi indietro al punto di partenza.

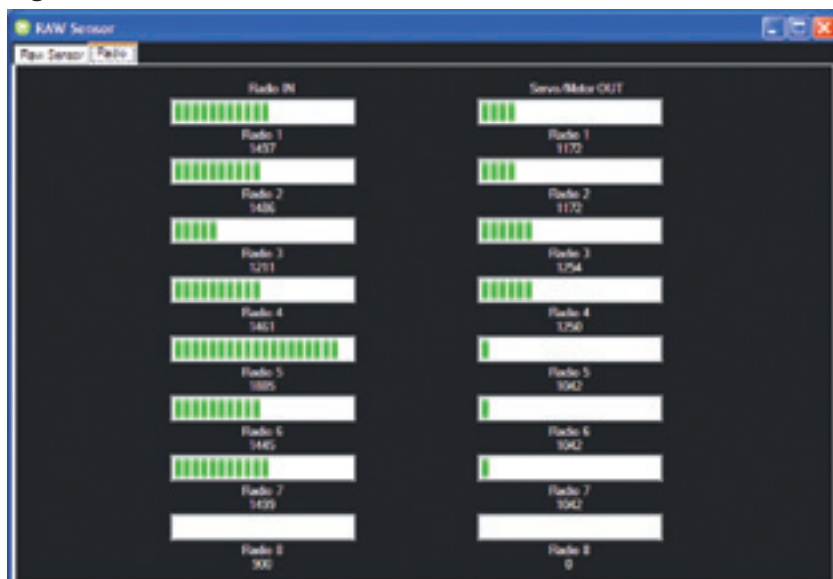
In questo momento dovreste essere collegati in modalità normale. In questa modalità possiamo utilizzare la telemetria XBee e la APM Mission Planner può essere collegata alla porta USB alla quale fa riferimento il modulo XBee. E' subito possibile verificare il funzionamento del modello controllando gli strumenti disponibili nel pannello "Flight Data" e "Gauges". In particolare, per analizzare il comportamento dei sensori che compongono l'IMU è disponibile dalla linguetta "Actions", sempre dal pannello "Flight Data" la funzione "Raw Sensor View".

VERIFICA DEL CORRETTO FUNZIONAMENTO DEL SOFTWARE DI CONTROLLO

Al termine dell'assemblaggio del quadricottero e prima del primo volo è indispensabile eseguire una serie di verifiche che permettano di valutare il corretto comportamento del modello. Per comprendere lo stato del sistema APM è opportuno dare una descrizione del significato dei led presenti sulle board.

Led presenti sulla basetta APM:
- Power (rosso): acceso quando il modulo è alimentato;

Fig. 10



- MUX (giallo): acceso quando l'interruttore CNTRL non è in modo Manual;
- PPM (Blu): lampeggia quando è in funzione il decoder PPM.

Led presenti sulla basetta IMU:

- LED A (Verde): stato dei motori, acceso quando i motori sono alimentati; lampeggia quando i motori sono disarmati in modo che non possano attivarsi anche se viene mosso lo stick del throttle;
- LED B (Giallo): non utilizzato;
- LED C (Rosso): stato del GPS, acceso quando il GPS fornisce dati validi, lampeggia quando il GPS è in attesa di segnali validi, spento quando il GPS non è collegato o ci sono problemi con la comunicazione seriale.

Attenzione: Se tutti i tre i led descritti lampeggiano in sequenza significa che vi trovate nella funzione di calibrazione dei regolatori dei motori. Per uscire da questa modalità scollegate e ricollegate la batteria.

Prima di iniziare i test smontate le eliche dai motori per evitare inconvenienti.

Le eliche collegate ai motori elettrici in rotazione possono provocare ferite anche serie.

Quindi fate attenzione.

La prima serie di test è finalizzata a verificare che i motori funzionino e

girino nel senso corretto e che il software li controlli correttamente sia nella modalità di stabilizzazione che nel volo pilotato. Per questo potete aiutarvi con la funzione "Flight Data"

-> "Actions" -> "Raw Sensor View"
-> "Radio"

- Attivate i motori spostando lo stick del timone verticale completamente a destra. Ora portate lo stick del throttle al 50% della corsa,
- Verificate che i motori girino nel senso corretto, altrimenti intervenite invertendo due dei cavi che collegano il motore al regolatore come descritto in precedenza,
- A mano, inclinate il quadricottero verso sinistra. Il motore di sinistra deve aumentare la velocità di rotazione mentre il motore di destra deve diminuirla,
- Inclinate il quadricottero verso destra. Il motore di destra deve aumentare la velocità di rotazione mentre il motore di sinistra deve diminuirla,
- Abbassate il muso (il motore frontale deve trovarsi più in basso di quello posteriore). Il motore anteriore deve aumentare la velocità di rotazione mentre quello posteriore deve diminuirla,
- Alzate il muso del quadricottero (il motore frontale deve trovarsi

più in alto di quello posteriore). Il motore posteriore deve aumentare la velocità di rotazione mentre il motore anteriore deve diminuirla,

- Ruotate il quadricottero in senso orario. I motori anteriore e posteriore devono aumentare la velocità di rotazione,
- Ruotate il quadricottero in senso antiorario. I motori destro e sinistro devono aumentare la velocità di rotazione,
- Muovete lo stick degli alettoni verso sinistra. Il motore destro deve aumentare la velocità di rotazione mentre il sinistro deve diminuirla,
- Muovete lo stick degli alettoni verso destra. Il motore sinistro deve aumentare la velocità di rotazione mentre il destro deve diminuirla,
- Muovete lo stick del timone orizzontale verso l'alto (a picchiare). Il motore posteriore deve aumentare la velocità di rotazione mentre il motore anteriore deve diminuirla,
- Muovete lo stick del timone orizzontale verso il basso (a cabrare). Il motore frontale deve aumentare la velocità di rotazione mentre il motore posteriore deve diminuirla,
- Muovete lo stick della deriva verso sinistra. I motori frontale e posteriore devono aumentare la velocità di rotazione,
- Muovete lo stick della deriva verso destra. I motori destro e sinistro devono aumentare la velocità di rotazione,
- Azzerate lo stick del throttle,
- Disattivate i motori spostando lo stick del timone verticale completamente a sinistra per qualche secondo.

PRIMO TEST IN VOLO

Se tutti i controlli precedenti sono stati superati correttamente è possibile montare le eliche e iniziare i

test a terra e poi in volo. Prima di attivare i motori assicuratevi che tutte le persone e gli oggetti si trovino a distanza di sicurezza dalle eliche.

Collegate la batteria LiPo ed aspettate finché i led rossi e verdi lampeggiano; il led rosso lampeggia fin quando non viene ricevuto un segnale valido dal GPS, il led verde lampeggia quando i motori sono disattivati. I motori vengono attivati tenendo lo stick della deriva tutto a destra per almeno due secondi, con lo stick del throttle al minimo.

Quando i motori sono attivati il led verde rimane sempre acceso.

La disattivazione dei motori avviene portando lo stick della deriva tutto a sinistra per almeno due secondi sempre con lo stick del throttle al minimo. Sia l'attivazione che la disattivazione dei motori richiede di tenere lo stick della deriva a fondo corsa per qualche secondo; questa è una forma di sicurezza per evitare di disattivare accidentalmente i motori durante il volo o attivarli durante il trasporto.

Assicuratevi che l'interruttore CNTRL sia in posizione "Stabilize". Muovete lentamente lo stick del throttle fin quando il quadricottero comincia ad alzarsi. Se avete l'impressione che si metta a girare o che non salga verticalmente portate lo stick del throttle al minimo e portate a terra il modello. Prendete il quadricottero in mano afferrandolo in un punto sicuro ed in modo da non essere colpiti dalle eliche. Date un pò di motore, giusto per fare girare le eliche.

Osservate se i motori si comportano correttamente quando muovete gli stick per avanzare, indietreggiare, traslare a destra e a sinistra.

Se il comportamento non è corretto i motivi possono essere la impostazione errata della configurazione "+" o "X", i motori collegati in modo errato o le

eliche montate non correttamente. Se invece sale regolarmente potreste notare un leggero movimento di rotazione sull'asse verticale; se non supera i trenta gradi si correggerà da solo col tempo. Non ci devono essere vibrazioni eccessive, in caso contrario potrebbe esserci qualche elica sbilanciata o un motore montato non perfettamente in asse. Il quadricottero deve tendere a rimanere fermo su un punto, e non vi dovrete trovare nella necessità di eseguire eccessive correzioni; se il quadricottero non si comporta correttamente non usate i trim per eseguire correzioni. Il problema, probabilmente risiede nel fatto che il modello non era perfettamente in "bolla" al momento della calibrazione oppure che il telaio del modello non è in squadra. Riportate a terra il modello e correggete le imperfezioni. Se tutto funziona correttamente potete iniziare a sperimentare le altre modalità di volo. Test della modalità di volo Loiter; stando in modo "Stabilize" portate il modello in un punto a piacere poi, con il modello stabile, passate alla modalità "Loiter".

Il modello deve eseguire dei cerchi di circa cinque metri di diametro ad altezza costante, in caso di vento i cerchi possono essere più ampi,

con vento forte il sistema può non essere in grado di compensare le deviazioni. In modo "Altitude hold" il modello rimane ad altezza costante mantenendola nell'ambito di qualche decina di centimetri, senza il sensore sonar l'ampiezza delle oscillazioni passa ad un paio di metri. In ogni caso è possibile pilotare il modello con gli stick oppure variane la altezza con il throttle.

Trovate la descrizione delle altre modalità di volo nel box dedicato a questo argomento. A questo punto è necessaria una importante avvertenza; un quadricottero vola in modo molto differente rispetto ad un aereo. Un aereo vola sostenuto dalla portanza delle ali, se il motore si ferma, il modello può ancora planare in sicurezza verso terra. Un quadricottero è sostenuto in aria unicamente dalla portanza delle eliche condotte dai motori.

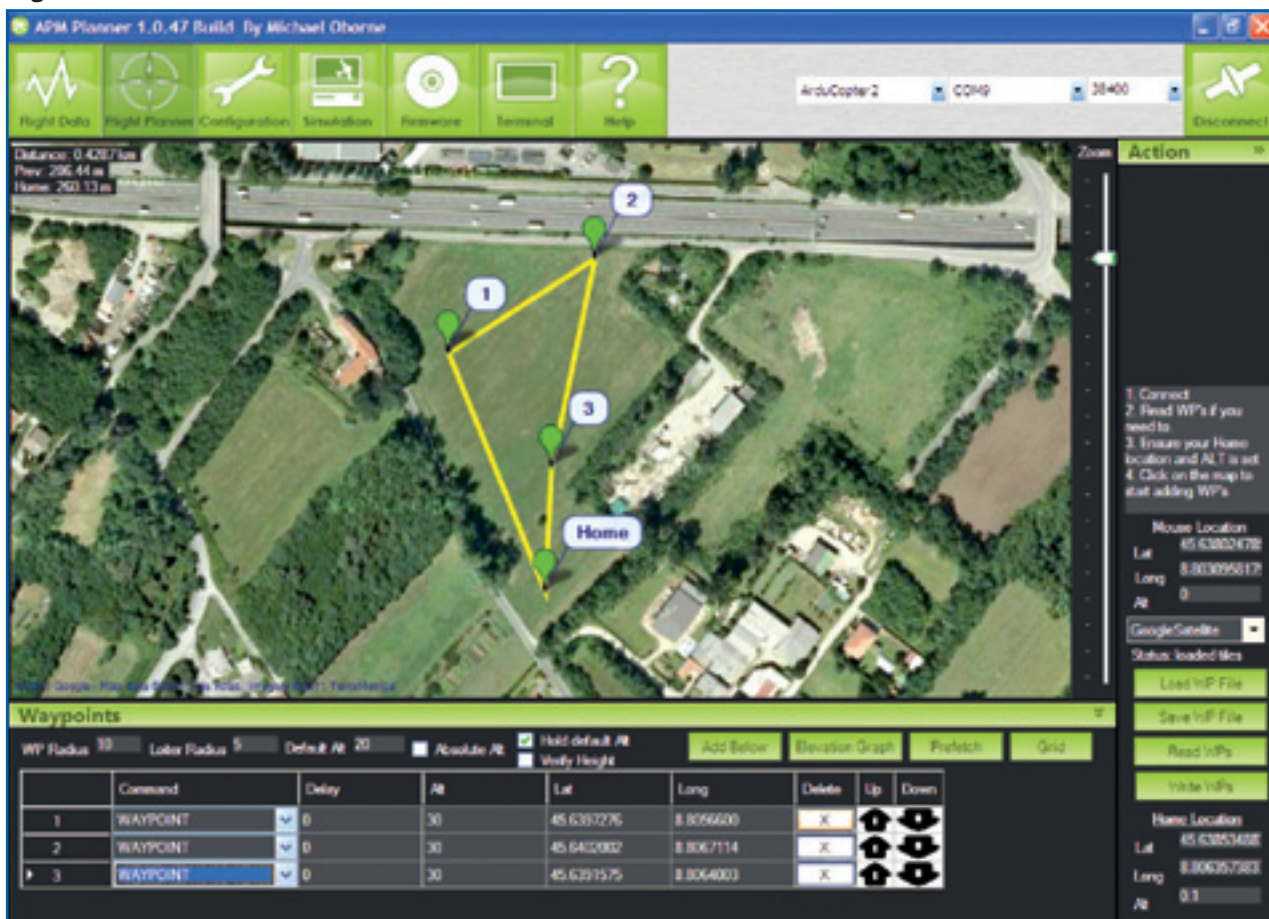
Se la batteria si esaurisce cessa il sostentamento del modello che precipita senza alcun preavviso. Durante i primi voli volate solo per brevi periodi e verificate la durata della batteria.

PIANIFICAZIONE MISSIONI CON LA APM MISSION PLANNER

Come già descritto negli articoli pre-



Fig. 11



cedenti una ulteriore grossa novità introdotta da questa generazione di autopiloti è l'applicazione di configurazione, calibrazione e gestione delle missioni APM Mission Planner che si appoggia al collegamento di telemetria sfruttandolo in modo bidirezionale. Il protocollo di comunicazione utilizzato è il Mavlink, un protocollo standard open source, in continua evoluzione per applicazioni e sistemi di controllo di navigazione a distanza in genere. Rispetto alla versione recensita nei numeri precedenti l'applicazione APM Mission Planner è stata notevolmente migliorata, sia nella ricchezza delle funzionalità che nella loro presentazione grafica. Alcune delle nuove funzionalità introdotte sono destinate specificatamente alla gestione dei modelli ad ala rotante. Riassumiamo le funzionalità della APM Mission Planner utili per i nostri scopi:

- Permette di caricare il firmware

su APM senza bisogno del compilatore,

- Offre una Ground Control Station con gli strumenti per il controllo del volo e la mappa che permette di tracciare il percorso seguito (GPS),
- Un pannello che simula lo schermo di un oscilloscopio per controllare il comportamento dei sensori,
- Una ricca funzionalità per pianificare le missioni,
- Una apposita funzione permette di configurare i parametri di calibrazione dei controllori PID,
- Un emulatore di terminale per configurare i parametri di APM in modalità Setup,
- La possibilità di salvare i log delle missioni per analizzarli in momenti successivi al volo,
- Molte altre funzionalità che si scoprono man mano e altre ancora in fase di continuo sviluppo.

Per utilizzare la Mission Planner in comunicazione on line accendiamo la trasmettente del radiocomando, diamo corrente al quadricottero, colleghiamo il modulo XBee al PC con il cavo USB ed apriamo la applicazione. Nel primo box in alto a destra selezioniamo "ArduCopter2", nel secondo la porta di comunicazione e nel terzo la velocità. Infine premiamo "Connect". Selezioniamo il pannello "Flight data". Se muoviamo il quadricottero orientandolo in diverse direzioni vedremo gli strumenti seguire i nostri movimenti, segno che la comunicazione è attiva e funzionante. Per pianificare una missione selezioniamo il pannello "Flight Planner"; questa funzione richiede Google Earth installato ed un collegamento attivo alla rete Internet. Con questa funzione possiamo descrivere una missione utilizzando una specie di linguaggio di scripting, salvare la missione su un file e ca-

ricarla nella EEPROM del modello. Come prima operazione impostiamo i valori di:

- WP radius: la distanza alla quale una "boa" viene riconosciuta come "raggiunta", 10 metri è un valore soddisfacente,
- Loiter Radius: il raggio del volo in circolo che viene eseguito una volta raggiunta la posizione Home in modalità RTL (Ritorno al punto di lancio), 5 metri, anche se un pò stretto è un valore che permette un buon controllo del volo,
- Default Alt: altezza di volo, per un buon controllo scegliete un valore di 15-20 metri,
- Absolute Alt: lasciando il box non selezionato la altezza precedente è considerata "altezza rispetto al terreno", selezionando il box (segno di spunta) l'altezza viene considerata al di sopra del livello del mare, che è molto più difficile da gestire. Noi abbiamo optato per il valore al di sopra del terreno,
- Il box Verify Height, se selezionato, informa APM Mission Planner di verificare l'altezza del terreno reale mantenendo l'altezza da questo anche in presenza di colline o rilievi. Non selezionare in caso di lancio del modello da un punto sopraelevato, come una collina.

Il modo più semplice di pianificare una missione è di individuare su Google Earth le coordinate del vostro punto "Home", copiarle nelle caselle in basso a sinistra e premere il tasto "Enter"; se dovesse apparire un errore che avvisa della mancanza di un valore di altezza valido significa che non è accettato il valore zero (probabilmente dovuto alla configurazione delle impostazioni internazionali del computer); in questo caso è sufficiente impostare un valore di 0.1 per superare il problema. La mappa si posizionerà nel punto indicato e

gli altri punti GPS (boe) potranno essere impostati semplicemente cliccando sulla mappa con il mouse. Ad ogni punto inserito corrisponde una riga di dettaglio nella parte inferiore del pannello. Su ogni riga il bottone "Delete" permette di eliminare un punto dal percorso, mentre è possibile "afferrare" un punto con la freccia del mouse e, con il pulsante destro tenuto premuto, spostarlo a piacere sulla mappa. Con il bottone "Write WPS" si carica la missione nella EEPROM del modello. Con il bottone "Read WPS" si legge la missione corrente dalla EEPROM, è possibile eseguire delle modifiche e ricaricarla. Utilizzando il bottone "Save WP File" si può salvare la missione su un file per un successivo utilizzo mentre con il bottone "Load WP File" si carica una missione precedentemente memorizzata su file. L'uso combinato di queste funzioni permette una grandissima flessibilità e facilità nella configurazione dei percorsi.

REGISTRATORE DEI DATI DI VOLO

Il registratore dei dati di volo è presente anche in ArduCopterMega, con le stesse modalità di funzionamento descritte negli articoli precedenti; riprendiamo brevemente le modalità di attivazione ed utilizzo. Il registratore dei dati di volo funziona dall'accensione del modello fino allo spegnimento; la dimensione massima del log è data dai 16Mb di memoria flash.

Per gestire il registratore dei dati di volo è necessario entrare in modo CLI e utilizzare il pannello "Terminal" della APM Mission Planner. Entrare nel menù "Logs" che offre le seguenti funzioni.

- "dump (n)": Dump log "n",
- "erase": Cancellazione di tutti i log,
- "enable (nome) or all": Abilita uno specifico log "nome" o tutti,
- "disable (nome) or all": Disabilita uno specifico log "nome" o tutti.

La prima operazione da effettuare per ottenere log validi è la pulitura e formattazione della memoria flash che si ottiene eseguendo la funzione "erase". Dopodiché, una volta generati i log, possono essere scaricati con la CLI o, più facilmente, scaricati, salvati e analizzati con le funzioni presenti in APM Mission Planner. Le funzioni di APM Mission Planner permettono di analizzare i dati di volo in forma tabellare o grafica e di generare file KML da visualizzare in Google Earth.

MODALITÀ DI VOLO

Ricordiamo che le modalità di volo sono associate alle posizioni dell'interruttore a tre posizioni CNTRL sulla trasmittente.

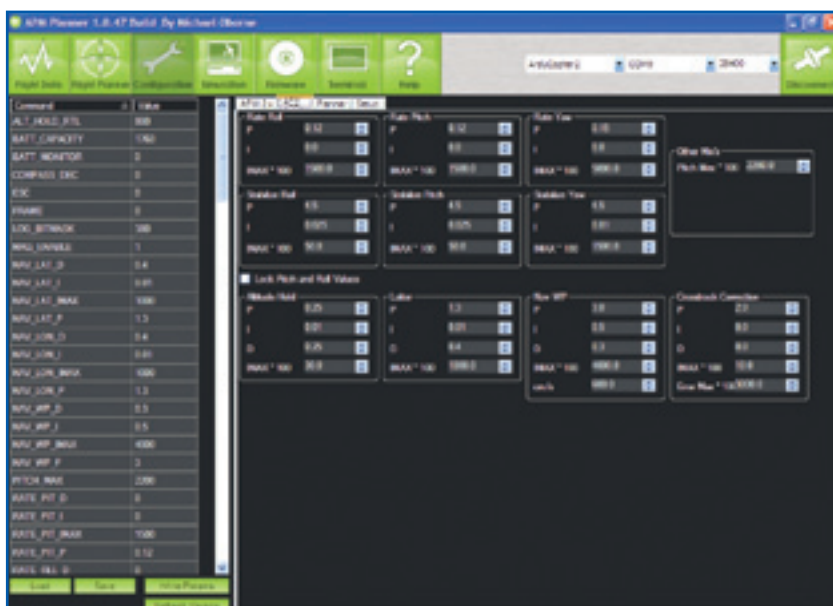
Le associazioni delle modalità di volo alle posizioni dell'interruttore vengono impostate in modalità CLI dal menù "Setup" utilizzando la funzione "modes". Posizionate l'interruttore nella posizione prescelta, poi scegliete la modalità di volo spostando lo stick degli alettoni a destra e a sinistra, al termine "Enter".

Acro. In questa modalità la velocità di rotazione sugli assi del quadricottero corrisponde alla velocità di movimento degli stick. Da utilizzare solo da piloti esperti.

Stabilize. Questa è la modalità di stabilizzazione di base. Gli stick determinano i movimenti di rollio e beccheggio fino ad un massimo di 45 gradi. Rilasciando gli stick il modello torna livellato. Tenete conto che anche il modello più bilanciato non può mantenere a lungo una posizione immobile. Il modello si sposta lentamente e richiede una leggera ma costante correzione da parte del pilota.

Simple. La modalità simple rappresenta il modo di volo maggiormente sicuro. E' una alternativa alla modalità Stabilize che considera come frontale la direzione di fronte al pilota, indipendentemente dall'orientamento del modello. Questa modalità è molto utile per recuperare un mo-

Fig. 12



dello che si allontana o che è molto alto con difficoltà a determinarne l'orientamento. L'angolo della direzione frontale viene impostato alla attivazione dei motori, per resettarlo disattivare e riattivare i motori.

Alt Hold. La modalità di mantenimento dell'altitudine equivale alla modalità Stabilize con in più il controllo automatico del throttle. E' comunque possibile variare l'altitudine con lo stick del throttle: portando lo stick completamente in alto il modello sale, totalmente in basso il modello scende.

Loiter. Loiter mantiene il modello in un volo circolare attorno ad un punto prefissato. Corrisponde alla modalità Stabilize con il controllo del rollio e beccheggio per eseguire la traiettoria circolare. Con una buona calibrazione dei PID il raggio della circonferenza è di circa cinque metri. Muovendo gli stick si può ovviamente sempre intervenire sul movimento del modello.

Guided. Il modello si dirige verso il punto indicato via Ground Control Station e, una volta raggiunto, entra in modo Loiter attorno al punto stesso.

Auto. In questa modalità il modello esegue la missione memorizzata in EEPROM. Raggiunto l'ultimo punto GPS il modello entra in modalità Loi-

ter attorno ad esso.

Rtl. Il modello torna verso il punto di decollo mantenendo l'altitudine. Il braccio frontale punta verso Home. In tutte le modalità di volo gli interventi del pilota sugli stick sono prioritari rispetto al sistema di controllo. Questo permette di ristabilizzare il modello in caso di comportamenti anomali con qualche leggero intervento sugli stick, in qualsiasi modalità.

Alcune precisazioni: Come già detto i motori si attivano spostando lo stick della deriva tutto a destra per qualche secondo, e si disattivano col movimento opposto.

Controllate con un colpetto di throttle di essere nella configurazione desiderata. La modalità Auto non si attiva con il throttle al minimo e il modello a terra, ed i motori non si metteranno in moto. I motori non si metteranno in moto nemmeno se si attiva la modalità Stabilize con il throttle al minimo.

COME FUNZIONANO I CONTROLLORI PID IN ARDUCOPTER MEGA

Negli articoli precedenti vi abbiamo descritto la architettura ed il modo di funzionamento dei controllori PID per la ottimizzazione dei sistemi di controllo della stabilizzazione della

navigazione. Date le diverse caratteristiche del volo di un multicottero rispetto ad un modello ad ala fissa, il software ArduCopterMega dispone di una architettura di controllori PID adatta anche a gestire le modalità di volo peculiari per questa configurazione, in particolare il controllo dell'imbardata (yaw), il controllo della posizione e della quota. Di seguito riassumiamo brevemente il significato dei parametri di ciascun gruppo di controllori e vi diamo una descrizione sul modo di intervenire di ciascun gruppo di controllori sul comportamento del multicottero. Il parametro P dei controllori PID è il coefficiente con il quale il controllore risponde direttamente all'errore. Ad esempio se il modello che dovrebbe essere stabilizzato si trova con un angolo di rollio di 25 gradi si avrebbe, con un P di 0,54 un errore E calcolato di $-2500 * 0,54$ che viene convertito in un valore PWM di $E * 0,1 = 135$; questo è il valore che viene inviato al regolatore del motore interessato per correggere il rollio. Il parametro D rappresenta la forza in diretta opposizione alla velocità del comportamento errato. Ad esempio se il modello ruota troppo velocemente su un asse il parametro D è il più appropriato per aggiustare questa tendenza.

La giusta calibrazione dei parametri P e D permettono di rendere il modello giustamente reattivo alle perturbazioni.

Come potete immaginare un valore troppo alto del parametro D otterrebbe l'effetto di una grossa sovra correzione. Se il vostro modello è lento molto probabilmente non è richiesto alcun parametro D.

Il parametro I consente di correggere un errore in un tempo più o meno lungo. Infatti il parametro I viene moltiplicato alla grandezza dell'errore per diminuirne la gravità allungando il tempo necessario a riportarlo entro i limiti prefissati iMax e iMin. I parametri dei controllori PID

si modificano utilizzando il pannello collegato alla linguetta "AP2" nella funzione "Configuration" di "APM Planner". "Write Params" permette di caricare i parametri modificati sul modello, anche in volo; attenzione! Inoltre i parametri possono essere salvati su file e richiamati.

SIGNIFICATO DEI PARAMETRI PID UTILIZZATI NEL SOFTWARE DI CONTROLLO E LORO IMPATTO SULLE PRESTAZIONI DEL MODELLO

La modalità Stabilize è intrinseca in ogni altra modalità tranne Acro. I parametri sono codificati per funzionare in modo simmetrico ad esempio agiscono allo stesso modo su un errore di rollio destro o sinistro.

- **STABILIZE_ROLL_P, STABILIZE_PITCH_P:** Controlla la potenza applicata ai motori per raggiungere gli angoli di rollio e beccheggio desiderati. Motori poco potenti richiedono valori di P più alti,
- **STABILIZE_ROLL_I, STABILIZE_PITCH_I:** Usati per la calibrazione fine, determinano in quanto tempo vengono raggiunti gli angoli massimi di rollio e beccheggio. Non dovrebbe essere mai necessario modificarli. Valori alti rendono più reattivo il controllo,
- **STABILIZE_ROLL_D, STABILIZE_PITCH_D:** Il parametro più fine da tarare. Rallenta le rotazioni del modello impedendogli di oscillare. Deve essere mantenuto il più basso possibile compatibilmente con la capacità del modello di ristabilizzarsi dopo manovre rapide,
- **STABILIZE_PITCH_IMAX:** Utilizzato per correggere il comportamento di modelli sbilanciati.

Questi parametri vengono utilizzati nel controllo della modalità Acro. Ricordiamo che questa modalità è ge-

stibile unicamente da piloti esperti.

- **ACRO_RATE_ROLL_P, ACRO_RATE_PITCH_P:** Controlla la potenza applicata ai motori per raggiungere gli angoli di rollio e beccheggio desiderati. Motori poco potenti richiedono valori di P più alti,
- **ACRO_RATE_ROLL_I, ACRO_RATE_PITCH_I:** Non utilizzati,
- **ACRO_RATE_D, ACRO_RATE_PITCH_D:** Non utilizzati,
- **STABILIZE_PITCH_IMAX:** Non utilizzato.

Il controllo dell'imbardata è preposto a controllare il mantenimento delle direzioni richieste. Se il modello tende a piroettare autonomamente non è possibile mantenere una direzione data. La calibrazione del parametro P permette di annullare la tendenza a piroettare.

- **YAW_P:** Controlla la potenza applicata ai motori per correggere la tendenza a piroettare,
- **Yaw_I:** Funziona come un trim per correggere modelli sbilanciati. Influenza il tempo necessario a raggiungere i valori massimi. Valori più alti comportano una maggiore reattività del controllo,
- **Yaw_D:** Utilizzato per rallentare rotazioni rapide sull'asse di imbardata. Valori troppo elevati tendono a "incagliare" il modello impedendogli di ruotare.

Il controllo di imbardata in modalità Acro (praticamente Manuale), è utilizzato per calibrare la velocità di rotazione sull'asse di imbardata. La calibrazione è in gradi al secondo.

- **YAW_P:** Controlla la potenza applicata ai motori per controllare la rotazione sull'asse di imbardata,
- **Yaw_I:** Non utilizzato,
- **Yaw_D:** Non utilizzato.

NAV_LOITER è utilizzato per con-

trollare il tasso di beccheggio in modalità Loiter. In pratica determina la velocità di esecuzione della traiettoria.

- **NAV_LOITER_P:** Valori alti aumentano l'angolo di beccheggio,
- **NAV_LOITER_I:** Permette di contrastare l'effetto del vento. Valori più alti permettono un maggiore contrasto del vento,
- **NAV_LOITER_D:** Rallenta il comportamento del modello, se troppo aggressivo,
- **NAV_LOITER_IMAX:** Sovracorrezione dell'angolo di beccheggio per contrastare il vento.

NAV_WP è utilizzato per controllare la velocità del modello verso i waypoints.

- **NAV_WP_P:** Valori più alti comportano un beccheggio maggiore e di conseguenza una velocità maggiore,
- **NAV_WP_I:** Permette di contrastare l'effetto del vento. Valori più alti permettono un maggiore contrasto del vento,
- **NAV_WP_D:** Rallenta la tendenza del modello ad accelerare troppo bruscamente,
- **NAV_WP_IMAX:** Sovracorrezione dell'angolo di beccheggio per contrastare il vento.

THROTTLE_BARO è utilizzato per mantenere una posizione fissa utilizzando le indicazioni del sensore barometrico di altitudine.

- **THROTTLE_BARO_P:** valori di P alti comportano una risposta dei motori più aggressiva,
- **THROTTLE_BARO_I:** Permette di adeguare il comando del throttle per compensare il consumo delle batterie,
- **THROTTLE_BARO_D:** Rallenta la tendenza del modello a salire o scendere troppo bruscamente,
- **THROTTLE_BARO_IMAX:** Massimo valore di correzione per il throttle.



VERSO NESSIE 2012

CONSIDERAZIONI SU UN INSEGUITORE DI LINEA A 128 BIT

di GIULIO VITALE

Nessie è il robot della classe "rescue", che abbiamo presentato nei numeri di febbraio e marzo di Elettronica In. Per questa tipologia di robot, l'elemento fondamentale per guidarne il movimento è il sistema di inseguimento di una linea nera su sfondo bianco; l'efficienza di questa funzione è determinante nello stabilire il livello delle prestazioni ottenibili durante le competizioni della Robocup Junior. Una esecuzione impeccabile del percorso guidato è il punto di partenza necessario per consentire al robot di affrontare, con una maggiore disponibilità di tempo, il resto delle difficoltà nell'evitare gli ostacoli, nel superare macerie e nel salvataggio delle vittime. Nella versione precedente di Nessie, presentata a Catania 2011, il sensore delegato a questa funzione impiegava tre dispositivi della Taos in grado di convertire l'intensità luminosa, misurata da un unico fotodiodo, all'interno di un singolo cono di visibilità, e trasformata in un segnale digitale a frequenza variabile. Visto lo scopo didattico di questa attività, all'ITCS Erasmo da Rotterdam di Bollate si sta cercando di far sviluppare agli studenti, sotto forma di percorsi di ricerca personale da presentare all'Esame di Stato, nuove alternative in grado di tentare il superamento dei limiti riscontrati nei modelli precedenti. L'idea qui presentata è stata, appunto, lo spunto del lavoro e delle prove che hanno consentito allo studente Simone Marani di conseguire il diploma di perito elettronico con il massimo dei voti, completando, con un'ottima presentazione del proprio percorso di ricerca, il quadro, già di per sé lusinghiero, di un esame all'altezza del curriculum scolastico.

La nuova soluzione qui presentata prevede l'utilizzo del sensore TSL1401R-LF, visibi-

Iniziamo il percorso verso la RoboCup Junior 2012 aggiornando il progetto che ha partecipato alla precedente edizione. In questo articolo viene illustrato l'impiego di un nuovo sensore di linea in grado di offrire prestazioni decisamente superiori.



SI: impulso di inizio del periodo d'integrazione
CLK: clock per la lettura dei pixel
AO: output analogico
VDD: alimentazione 3-5,5V

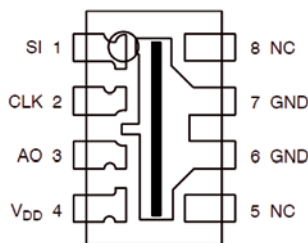


Fig. 1 - Pinout del dispositivo TSL1401R-LF

le nella Fig. 1, composto da 128 fotodiodi disposti in modo lineare, prodotto ancora della Taos e fornito da Parallax, con la sigla TSL1401-DB, in un modulo corredato da un'ottica che permette la scansione di un segmento di spazio di lunghezza pari alla distanza tra il tratto inquadrato e l'obiettivo.

Il sensore Taos 1401R-LF: principio di funzionamento

Nella Fig. 2 è illustrato lo schema a blocchi di principio del sensore. Come si nota dalla Fig. 2 il sensore è costituito da una sequenza di 128 fotodiodi, ognuno dei quali è condizionato da un circuito di conversione della fotocorrente in tensione tramite un integratore. La tensione integrata, dopo essere stata trasferita in un opportuno condensatore di mantenimento, è resa disponibile sull'uscita analogica AO grazie a un sistema d'indirizzamento ottenuto da un registro a scorrimento scandito con un clock la cui frequenza è compresa in un intervallo da 5 a 8000 kHz. Il principio di funzionamento del sistema di misura della luminosità è semplice e si basa su un meccanismo a tre fasi della durata complessiva di 129 fasi di clock, a cui va aggiunto un tempo (tqt) di trasferimento della carica:

- la prima, detta di “reset”, della durata di 18 periodi di clock, serve per scaricare i condensatori d'integrazione e predisporli per la fase successiva di accumulo della carica prodotta dal fotodiodo;

- la seconda, detta di “integrazione”, dura per i rimanenti 111 periodi di clock, e rappresenta la fase in cui la foto-corrente, prodotta dall'esposizione del fotodiodo alla luce incidente, viene integrata e la relativa carica prodotta accumulata nel circuito d'integrazione;
- la terza, detta di “holding”, prevede il trasferimento simultaneo della carica accumulata nei 128 circuiti d'integrazione, associati ad ognuno dei fotodiodi, ai rispettivi condensatori di mantenimento, e dura per il tempo di trasferimento “tqt”.

Guardando più da vicino il blocco di condizionamento di ognuno dei pixel (Fig. 3), si notano i due condensatori d'integrazione e

Fig. 3 - Dettaglio del circuito di condizionamento di ognuno dei 128 pixel.

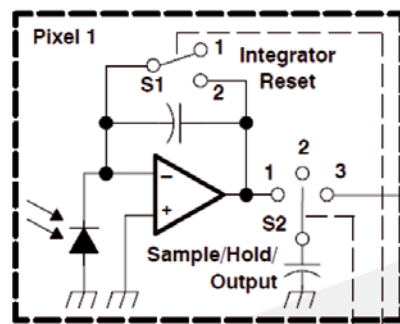
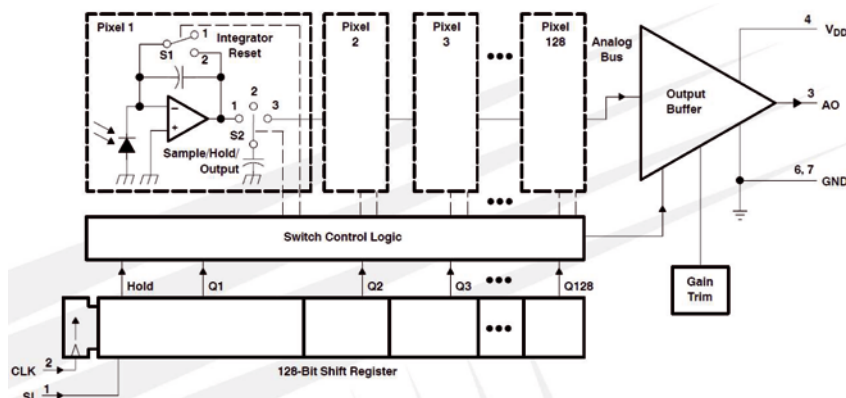


Fig. 2 - Schema a blocchi di principio del sensore TAOS1401R-LF



di mantenimento; il primo accumula la carica generata dal fotodiodo, il secondo la ricopia e la mantiene durante la fase di scansione e di cattura della carica successiva. Il sensore, in effetti, compie due operazioni simultaneamente sovrapponendo la fase di integrazione e acquisizione di una nuova misura con quella di lettura, tramite scansione, della carica accumulata nel ciclo precedente.

Ogni ciclo inizia sul fronte di salita del segnale SI (Start Integration) che commuta al primo ciclo di clock lo switch S2 in posizione 2, per isolare il condensatore di holding, e lo switch S1 in posizione 2, per eliminare la carica accumulata, contenuta nel condensatore d'integrazione. Questa operazione è effettuata in parallelo su tutti i 128 fotodiodi. Sul fronte di SI, viene anche caricato il primo “1” nel registro a scorrimento per preparare la lettura del valore mantenuto. Questa fase di “reset” si conclude alla fine del diciottesimo ciclo di clock con lo switch S1 in posizione 2. Il ciclo

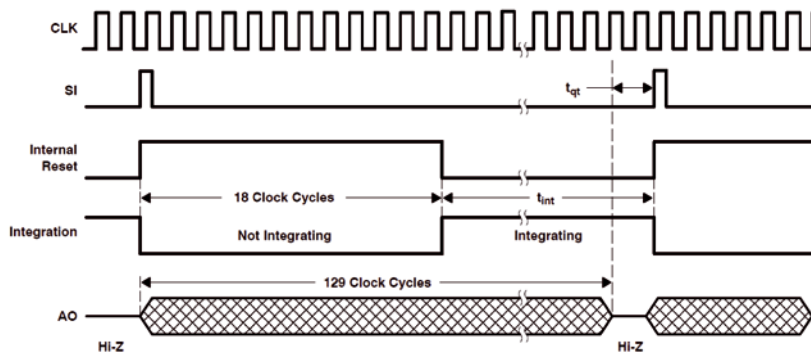


Fig. 4 - Diagramma temporale che illustra la sequenza delle fasi di scansione

t_{int} : tempo effettivo di integrazione della foto corrente; $111 \times t_{clk}$.

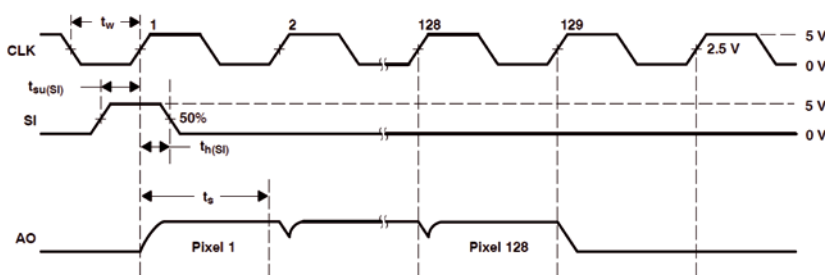


Fig. 5 - Diagramma temporale che illustra la sequenza operativa di lettura del valore di tensione accumulato nella fase di integrazione precedente

$t_{su}(SI)$: tempo di setup del segnale SI della durata minima di 20 ns

$t_{h}(SI)$: tempo di hold del segnale SI pari a zero

t_{qt} : tempo di trasferimento della carica integrata verso di condensatori di holding; minimo 20 μ s

t_s : tempo di assestamento del segnale analogico, della durata minima di 120 ns

prosegue con la fase d'integrazione, durante la quale si effettua la lettura del nuovo valore di luminosità per i rimanenti 111 periodi di clock; in questo lasso di tempo lo switch S1 è mantenuto in posizione 1.

Intanto, in parallelo, per tutta la durata del ciclo completo di 128 periodi di clock consecutivi, su ogni fronte di salita, l'uno caricato nel registro a scorrimento fa commutare, serialmente, per ognuno dei 128 pixel, lo switch S2 in posizione 3, per la durata di un periodo di clock, in modo tale da esporre il condensatore di mantenimento all'amplificatore d'uscita e consentire che la tensione ai suoi capi sia leggibile su pin AO (Analog Output); in questo modo, si rende simultanea la lettura della tensione, dovuta alle cariche accumulate per effetto dell'esposizione alla radiazione luminosa, nei condensatori di holding durante la scansione precedente, con l'acquisizione del valore corrente nei condensatori d'integrazione.

La sequenza temporale delle operazioni descritte è raffigurata in Fig. 4.

Sono necessari 129 cicli di clock per svuotare il registro e per misurare l'ultimo pixel scandito. Terminata una scansione di 129 fasi di clock, è necessario rispettare il tempo " t_{qt} " prima d'iniziarne un'altra, riattivando ancora il segnale SI. Nel tempo " t_{qt} " avviene il trasferimento simultaneo della carica integrata durante il tempo di esposizione dei fotodiodi, dai condensatori d'integrazione a quelli di mantenimento, con lo switch S2 in posizione 1. Naturalmente, la prima scansione in assoluto non è significativa, in quanto misura una carica accumulata in modo del tutto casuale.

In Fig. 5 si può vedere, invece, il diagramma temporale operativo simultaneo in cui si nota la relazione tra il segnale d'inizio della fase d'integrazione "SI" e il segnale analogico in uscita "AO".

Caratteristiche tecniche e prestazioni ottenibili dal sensore Taos 1401R-LF

Guardando la Fig. 6, si nota che il sensore ha una curva di sensibilità fortemente centrata nella zona estrema delle lunghezze d'onda visibili, al limite dell'infrarosso; pertanto l'illuminatore può essere fatto con i più economici led rossi ad alta efficienza.

La tensione d'uscita V_o è legata alla radiazione luminosa incidente tramite la seguente relazione:

$$V_o = V_{drk} + (R_e)(E_e)(t_{int})$$

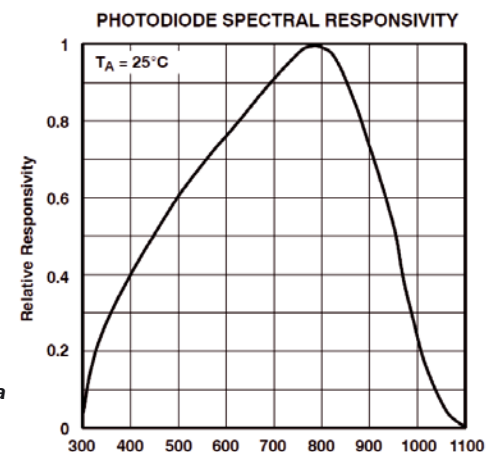
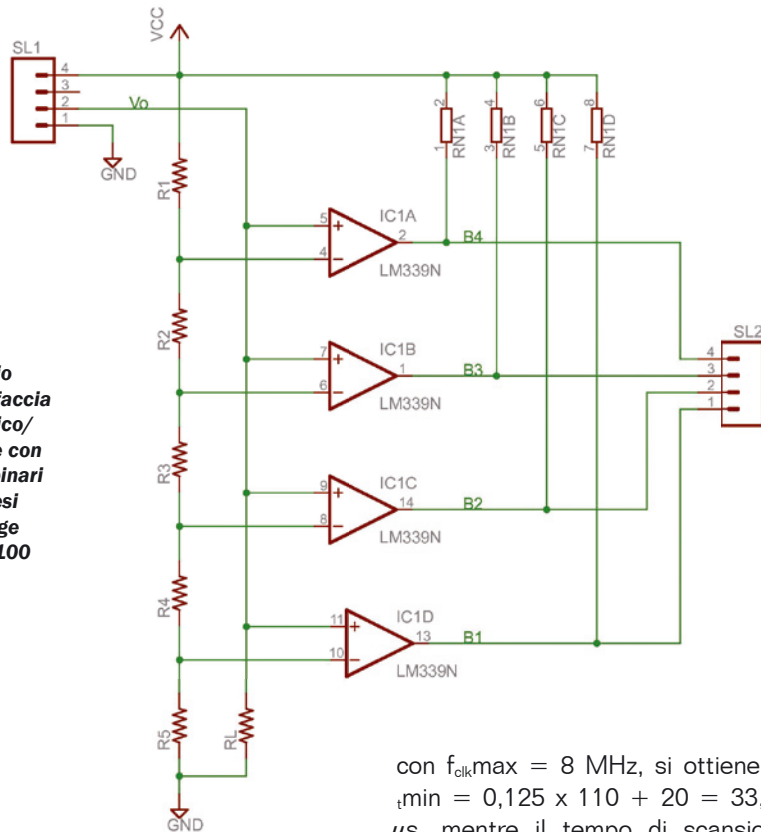


Fig. 6
Curva di risposta spettrale dei fotodiodi.

Fig. 7
Esempio
d'interfaccia
analogico/
digitale con
valori binari
compresi
nel range
000 ÷ 100



Dove:

V_{drk} è la tensione d'uscita in condizioni di buio: compresa tra 0 e 0,2 , con valore tipico di 0,1[V];

R_e è la responsività dei fotodiodi per una data lunghezza d'onda: compresa tra 25 e 45, con valore tipico di 35 [V/(μ J/cm²)];

E_e è la densità di potenza della radiazione incidente [μ W/cm²];

t_{int} è il tempo d'integrazione visto in precedenza.

(condizioni di test: $f_{clock} = 1$ MHz, $V_{DD} = 5$ V, $T_A = 25^\circ\text{C}$, $\lambda_p = 640$ nm, $t_{int} = 5$ ms, $R_L = 330 \Omega$, $E_e = 11 \mu\text{W/cm}^2$)

In queste condizioni di test, una superficie bianca uniforme produce una tensione d'uscita V_o media tipica di 2 V, misurata mediando su tutti i 128 pixel, con un valore massimo $V_o \text{ max} = 2,4$ V. L'amplificatore d'uscita ha una tensione di saturazione tipica di $V_o \text{ sat} = 4,8$ V.

Il tempo d'integrazione minimo è vincolato dalla massima frequenza lecita ed è calcolabile come:

$$t_{int \text{ min}} = (1/f_{clk \text{ max}}) \times (n - 18) \text{ pixel} + 20 \mu\text{s}$$

con $f_{clk \text{ max}} = 8$ MHz, si ottiene $t_{in \text{ min}} = 0,125 \times 110 + 20 = 33,75 \mu\text{s}$, mentre il tempo di scansione complessivo minimo, inclusa la fase di "reset", è pari a $0,125 \times 129 + 20 = 36,125 \mu\text{s}$.

Il tempo d'integrazione massimo è, invece, legato alla costante di tempo del circuito integratore; pertanto, affinché la misura si mantenga accurata, esso non deve superare i 100 ms e la relativa f_{clk} può essere così calcolata: $100 \times 10^{-3} = (1/f_{clk \text{ min}}) \times 111 + 0,02 \times 10^{-3} \Rightarrow f_{clk \text{ min}} = 1,11$ kHz.

Il tempo d'integrazione, compreso tra i due valori calcolati, determina quindi il tempo di esposizione.

In fase operativa, Nessie 2012 avrà il problema di dover modificare questo tempo, in quanto il valore di E_e varia al variare delle condizioni d'illuminamento. Avendo a disposizione un esposimetro che consenta il calcolo in tempo reale della densità di potenza luminosa che impatta il sensore e fissando un valore massimo per V_o , sarebbe possibile calcolare con precisione il tempo d'integrazione. Ma avendo a disposizione un microprocessore che gestisce il sensore è più semplice implementare un algoritmo "adattativo" che regola la frequenza f_{clk} con il seguente criterio:

- se "per ogni pixel misurato $\Rightarrow V_o \text{ sat} > V_o > V_o \text{ max}$ " occorre aumentare la frequenza f_{clk} per ridurre il tempo d'esposizione;
- se "per ogni pixel misurato $\Rightarrow V_o < V_o \text{ minima desiderata}$ " occorre diminuire la frequenza f_{clk} per aumentare il tempo d'esposizione;
- naturalmente, rimanendo nei limiti imposti per f_{clk} .

Come determinare il tempo di conversione di V_o ?

Per calcolare di quanto tempo dispone il circuito di conversione da analogico a digitale, è necessario tenere in conto il tempo di assestamento " t_s " (setting timing) del segnale analogico d'uscita che dal datasheet risulta di 120 ns max. Tenendo in conto che il segnale d'uscita viene mantenuto per un intero periodo di clock, il convertitore A/D deve avere un tempo di "setup" massimo

$$t_{su \text{ max}} = (1/f_{clk}) - t_s$$

Come scegliere la struttura del convertitore analogico/digitale?

Se ci si accontenta di due soli livelli, un semplicissimo trigger di Schmitt è sufficiente a stabilire un criterio per individuare una linea nera su sfondo bianco.

Se si vuole qualche bit di dettaglio in più si può usare il circuito che abbiamo sperimentato all'Erasmo da Rotterdam, che prevede l'utilizzo di un comune dispositivo, come il LM339, che contiene quattro comparatori integrati e, pertanto può essere usato per avere quattro livelli di comparazione più lo zero, che possono essere convertiti in binario come "000" ÷ "100".

In questa configurazione, considerando che l'integrato LM339 ha un tempo di risposta compreso nel range da 300 a 1300 ns, in funzione dell'ampiezza del segnale differenziale, si può prendere un valore intermedio di 500 ns, assumendo una differenza minima di 50 mV tra i due morsetti d'ingresso. Con queste

assunzioni, un periodo di clock minimo, che garantisca il "setup" completo, può essere assunto pari a circa 700 ns; con una frequenza corrispondente di circa 1400 kHz. Riducendo questo tempo di "setup", è possibile estendere la frequenza fino a 8000 kHz, oppure aumentare ulteriormente il numero di bit della rappresentazione, ma queste diventano considerazioni economiche relative al rapporto costo/prestazioni che si desidera.

L'interfaccia tra il sensore e il BUS

Nessie ha una struttura hardware basata su FPGA e nella versione 2 userà come CPU il Microblaze 8.1, che ha la possibilità di essere configurato come master su un BUS standard AXI4-lite, compatibile con la famiglia dei core ARM. Utilizzando il wizard fornito con il sistema di sviluppo EDK 13.1, è possibile creare un template VHDL che implementa il modulo wrapper che incapsula la logica d'interfaccia verso la periferica definita dall'utente, eliminando tutti i problemi relativi alle temporizzazioni specifiche del BUS AXI e offrendo una metodologia d'interconnessione semplificata (vedi Fig. 8).

All'utente rimane il compito di realizzare soltanto la logica d'interfaccia verso la propria periferica "custom", senza dover tenere in conto le specifiche del BUS AXI ma utilizzando una infrastruttura intuitiva, organizzata secondo un approccio classico del tipo "memory mapped" ma con la logica di decodifica già implementata nel modulo "wrapper". Per avere un controllo semplificato, si è deciso di rendere visibile il sensore, dal punto di vista software, come un banco di tre registri da 32 bit ciascuno; due sono di controllo in scrittura e uno di stato in lettura. Per facilitare il test, tutti sono separatamente scrivibili e rileggibili.

In Fig. 9 si può vedere lo schema a blocchi generale dell'interfaccia digitale da noi realizzata per controllare il sensore Taos1401. Come si può notare nella figura, la sequenza di scansione può essere iniziata dal microprocessore di controllo semplicemente ponendo alto il bit "StartScansione", mentre il controllo dell'esposizione si effettua impostando il campo a 16 bit "ClkDivisore" che consente di dividere il clock di base, impostato alla massima frequenza di 1,4 MHz. Aumentan-

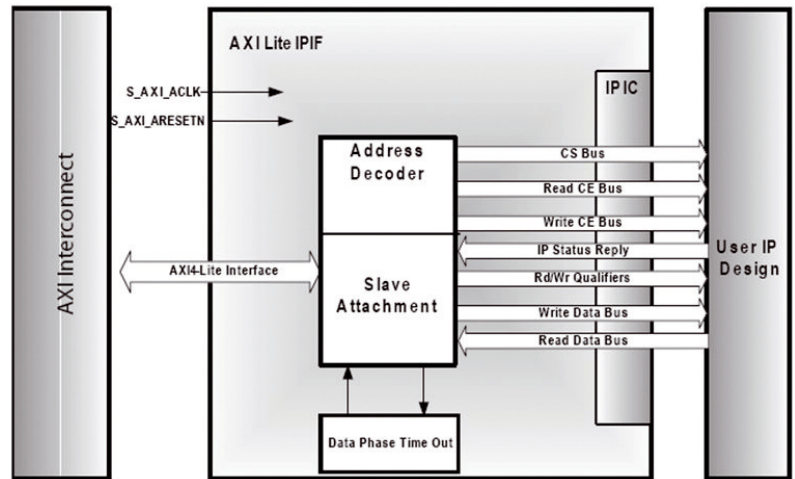
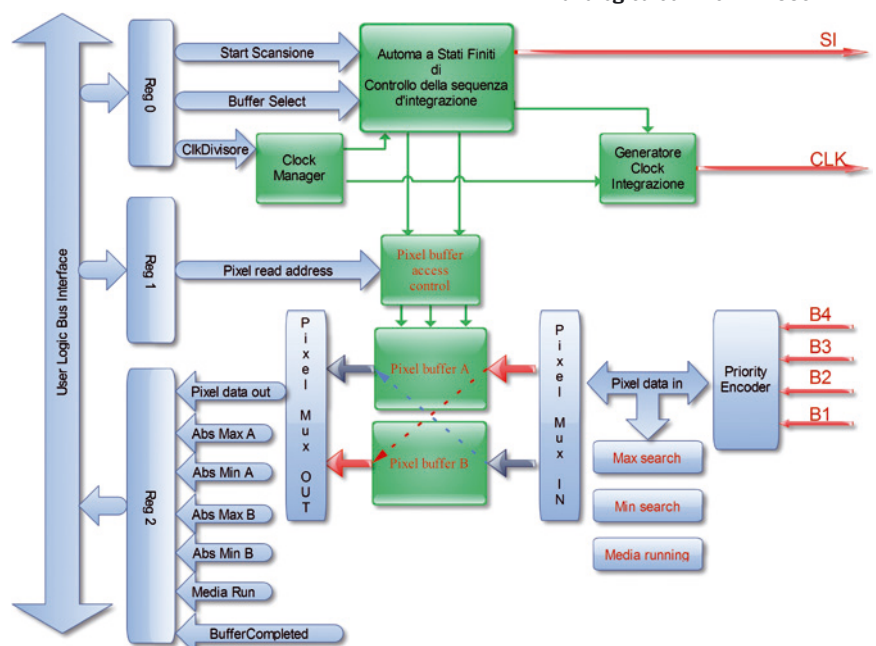


Fig. 8 – Template d'interfaccia della logica d'utente per interconnettere una periferica "custom" al BUS AXI4-lite del "software core" Microblaze, disponibile con le FPGA Spartan della Xilinx.

do il valore del divisore si allunga il tempo di esposizione.

Abbiamo realizzato la lettura dei pixel con una tecnica detta di "buffer swapping" che impiega un doppio buffer di memoria per i 128 pixel; in questo modo si ottiene il massimo della velocità di scansione quando occorre catturare una sequenza continua di trame successive. Il sistema funziona, infatti, con un criterio di lettura/scrittura simultaneo: quando la CPU legge il buffer A, la logica di scansione carica il buffer B; terminata una scansione si effettua lo "swapping" dei buffer, in modo tale che il buffer appena letto sia disponibile per una nuova scrittura e il buffer appena scritto sia leggibile. Operando così, il tempo speso per l'implementazione dell'algoritmo di controllo del robot va "in overlap"

Fig. 9 – Schema a blocchi della "user logic" implementata per il controllo della interfaccia tra il sensore e il microprocessore SI: Start integration CLK: Clock di scansione, la cui frequenza determina il tempo di esposizione B4 ÷ B1: 4 bit che indicano i livelli rilevati dalla interfaccia analogica con il c.i. LM339.

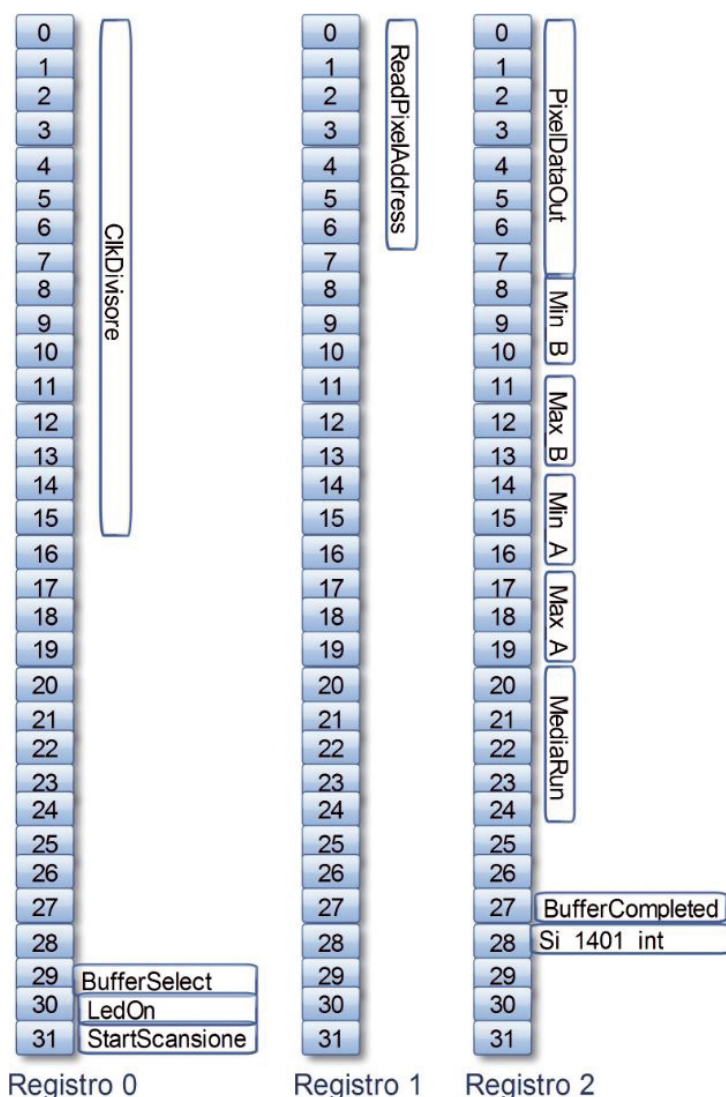


con la scansione dei 128 pixel; quest'ultima costituisce l'operazione più onerosa delle due e rischia di mantenere bloccata la CPU. Si può pensare a questa tecnica come a una sorta di pipeline a due stadi.

La sequenza dettagliata d'impiego del sensore è la seguente:

- 1) Si seleziona il bufferA e si dà il segnale di "StartScansione" (scrittura di un "1" seguito da uno "0")
- 2) Si attende fino a che **buffer_completed** acquista il valore "1"
- 3) Il primo buffer è inutilizzabile!
- 4) Si seleziona il bufferA e si dà il segnale di "StartScansione" (scrittura di un "1" seguito da uno "0")
- 5) Si attende fino a che **buffer_comple-**

Fig. 10 - Struttura dei registri d'interfaccia tra CPU e logica di controllo del sensore. Nel registro 2 si notano due bit di stato per la gestione dell'evento di conclusione di una scansione: "BufferCompleted" per il polling "SI_1401_int" per la gestione di un eventuale interrupt.



- ted acquista il valore "1"
- 6) Si seleziona il bufferB e si dà il segnale di "StartScansione" (scrittura di un "1" seguito da uno "0")
- 7) Il **bufferA** è disponibile in lettura al "main" per essere elaborato
- 8) Il "main" completa le operazioni di elaborazione del buffer acquisito fino a che non è necessaria la nuova lettura
- 9) Per leggere il nuovo buffer acquisito, si attende fino a che **buffer_completed** acquista il valore "1"
- 10) Si seleziona il bufferA e si dà il segnale di "StartScansione" (scrittura di un "1" seguito da uno "0")
- 11) Il **bufferB** è disponibile in lettura al "main" per essere elaborato
- 12) Il "main" completa le operazioni fino a che non è necessaria la nuova lettura ripartendo dal passo 5.

Dalla sequenza descritta si nota che il segnale di "StartScansione" deve essere dato solo se **non** in corso una ulteriore scansione, precedentemente schedulata, per evitare che avvenga una sovrapposizione, e ciò è rilevabile dalla lettura del bit di stato "BufferCompleted" che risulta nullo per tutta la durata di scansione del sensore. Va osservato che la frequenza di lettura dei fotodiodi è asimmetrica rispetto alla lettura dei bit da parte della CPU, in quanto, la prima, è legata al tempo di esposizione, e può essere anche dell'ordine dei millisecondi, mentre la seconda è vincolata solo dal tempo di esecuzione del software e dipende dal clock del microprocessore, che è dell'ordine delle decine di nanosecondi.

Nella **Fig. 10** è visibile la struttura dei tre registri d'interfaccia e il relativo nome dei singoli bit.

Come si può rilevare dalle **Figg. 9 e 10**, la "user logic" di controllo del sensore fornisce anche qualche ulteriore informazione sintetica sul contenuto dei buffer che può essere raccolta durante la scansione stessa; in particolare sono calcolati i valori minimi e massimi di entrambi i buffer durante la scansione stessa, per avere disponibile la lettura di due valori in sequenza, e la media del contenuto del solo buffer corrente. Questi dati possono risultare utili se si vuole implementare un algoritmo adattativo della gestione dell'espo-



Fig. 11 - Il modulo TSL1401-DB realizzato e fornito da Parallax.

sizione per regolare il tempo d'integrazione in funzione delle condizioni d'illuminamento.

Il modulo TSL1401-DB della Parallax

Il modulo da noi impiegato (Fig. 11) è fornito come "daughter board" della scheda madre del kit di valutazione TSL1401-EVM, offerto da Parallax/Taos, ma può essere utilizzato come circuito d'interfaccia per qualunque sistema di elaborazione digitale. TSL1401-DB abbinata il sensore della Taos con un obiettivo ottico di lunghezza focale di 7,9 mm e di apertura pari a f2.4; l'angolo d'inquadratura è di 53° e riesce a mettere a fuoco da 2,5 mm all'infinito. Questo particolare valore dell'angolo consente di avere la lunghezza "L" del segmento inquadrato circa uguale alla distanza "D" dell'oggetto stesso dall'obiettivo (vedi Fig. 12). Quindi, supponendo di montare su Nessie 2012 il modulo in modo da avere una distanza tra l'obiettivo e il terreno di 40 mm, la larghezza della linea scandita sarà

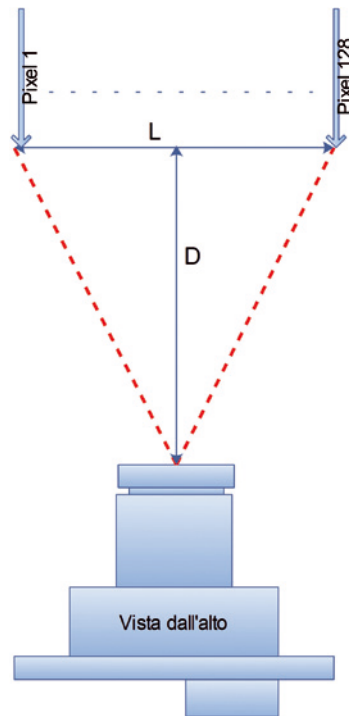


Fig. 12 - L'angolo di apertura dell'obiettivo montato sul modulo TSL1401-DB consente una lunghezza della linea di scansione pari, circa, alla distanza dell'oggetto.

anch'essa di circa 40 mm; ciò implica una risoluzione di $40/128 = 0,3125$ mm (ovvero di $312,5 \mu\text{m}$). Niente male!

Un esempio pratico di ciò che si può ottenere

Nelle quattro figure della pagina precedente (Fig. 13, 14, 15, 16) si può vedere l'uscita del modulo ponendo l'obiettivo a circa 40 mm dalla normale linea utilizzata nelle competizioni della Robocup Junior per tracciare il percorso dei robot "rescue", e normalmente realizzata con un nastro isolante nero di circa 14 mm di larghezza, disponendo il modulo TSL1401-DB in modo che la linea di scansione del sensore sia, rispettivamente, ortogonale, inclinata di circa 30° e parallela al tracciato.

Nella prima foto in alto è riprodotto il tracciato di prova in cui, oltre al nastro, sono state disegnate alcune linee di spessore diverso con un pennarello dotato di una punta di 0.7 mm, per avere qualche elemento di confronto sul grado di dettaglio ottenibile con il sensore.

Nella foto successiva, si vede la scansione dei pixel con il sensore ortogonale al percorso; qui si può osservare la dimensione rilevata dei vari segmenti, imponendo una soglia mediana sul valore di luminosità misurato. Le linee più sottili misurano da due a quattro pixel, la scansione

Fig. 13

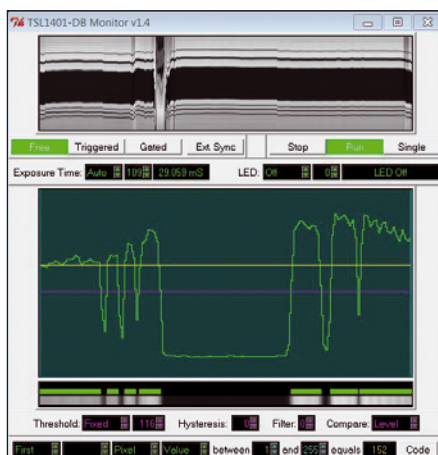
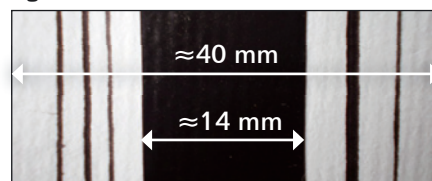


Fig. 14

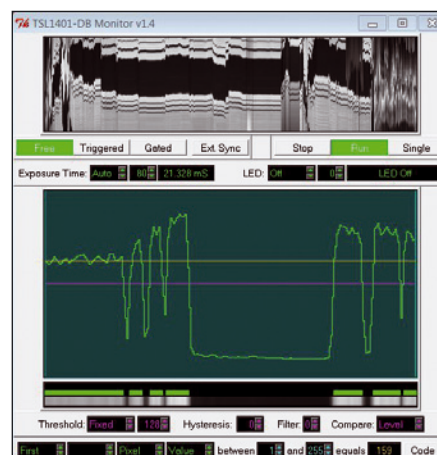


Fig. 15

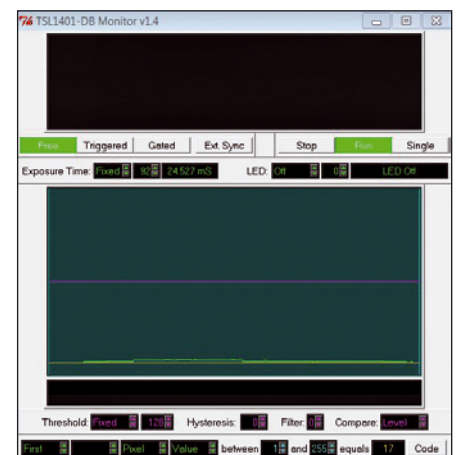


Fig. 16

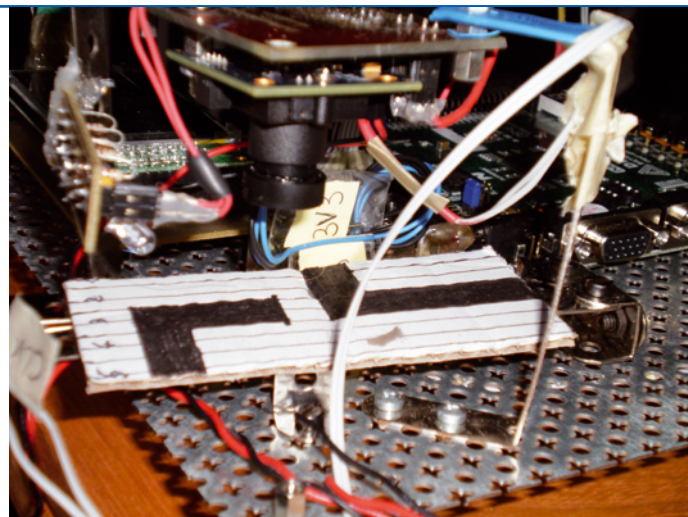
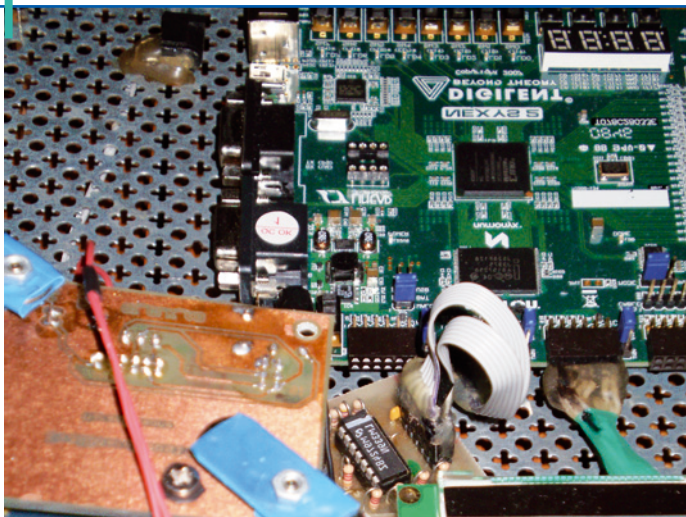


Fig. 17

Fig. 18

Fig. 17, 18, 19 – Particolari del banco di prova del sensore realizzato dallo studente Simone Marani.



Fig. 19

del nastro centrale è lunga 46 pixel, la linea più spessa sulla destra misura 4 pixel. Nonostante l'approssimazione casereccia nel posizionamento del sensore, si vede la buona precisione del sensore; infatti, 46 pxl per 0,3125 mm/pxl è uguale a 14,375 mm. Anche i tratti di penna da due a quattro pixel sono significativi, visto lo spessore 0.7 della punta di scrittura del pennarello.

Nella penultima foto, anche qui in modo piuttosto approssimativo, il sensore è stato ruotato di circa 30° e la linea nera centrale, che riproduce il nastro, risulta lunga pressappoco 53 pixel, ottenendo una lunghezza misurata di 16,5625 mm; abbastanza vicina al valore teorico di $14/\cos 30^\circ$.

(NB: le prove qui indicate sono state fatte utilizzando la scheda completa di valutazione TSL1401-EVM – USB, della Parallax, che scandisce il sensore con un convertitore analogico/digitale a otto bit e può, interpolando i pixel, raddoppiare la risoluzione del sensore fino a 256 valori.)

Conclusioni

In sostanza, dalle prove che abbiamo fatto, e che qui proponiamo, l'impressione che abbiamo avuto è che, rispetto ai soliti sensori spot, fatti con due o più fotodiodi o fototransistor isolati, questa categoria di sensori introduce una maggiore quantità d'informazione che, se ben sfruttata dal software, dovrebbe garantire una più elevata accuratezza nel sistema di guida dei robot "rescue".

In effetti essi rappresentano un passo intermedio verso l'impiego di una telecamera ma, avendo soltanto vettori unidimensionali da trattare, risultano più semplici da manipolare e più vicini al livello di conoscenze e abilità di una scuola media superiore. Comunque, mai

dire mai. In un prossimo numero di Elettro- nica In spiegheremo più dettagliatamente il modo in cui Nessie 2012 impiegherà questo sensore e verranno analizzati più da vicino gli algoritmi relativi alla cinematica dell'inse- guimento della linea e alle problematiche di auto adattamento al variare delle condizioni d'illuminamento, senza trascurare gli aspetti di filtraggio e il rilevamento dei bordi.

Il Laboratorio Permanente di Robotica Di- dattica dell'Erasmus da Rotterdam, cerca d'interpretare il senso della Robocup Junior coinvolgendo quanto più è possibile i propri ragazzi in una sfida continua verso il miglio- ramento. Ad ogni partecipazione annuale alla manifestazione nazionale è stato proposto un nuovo modello, cercando di superare i li- miti del modello precedente. Il contributo dei ragazzi è sempre stato importante, non solo per l'entusiasmo e per le idee profuse ma anche per verificare quanti dei suggerimenti proposti dal docente sono stati recepiti e a quale livello di profondità sono giunti.

Tutte le nuove modifiche che Nessie 2012 subirà, sono il frutto delle proposte di alcune tesine che i ragazzi che hanno partecipato quest'anno hanno portato all'Esame di Sta- to come proprio percorso di ricerca, alcune delle quali potranno essere oggetto di nuo- vi articoli, se Elettro nica In li vorrà ospitare. Per questo articolo, ricordo che le interfac- ce verso la scheda di controllo e il relativo banco di test per verificare la fattibilità del progetto, sono il frutto del lavoro di Simone Marani che si è diplomato quest'anno e a cui, qualunque sia l'ambito in cui vorrà impiegare la sua intelligenza, la sua tenacia e la sua di- sponibilità, faccio i miei più sinceri auguri di un percorso di vita ricco delle soddisfazioni che merita. □



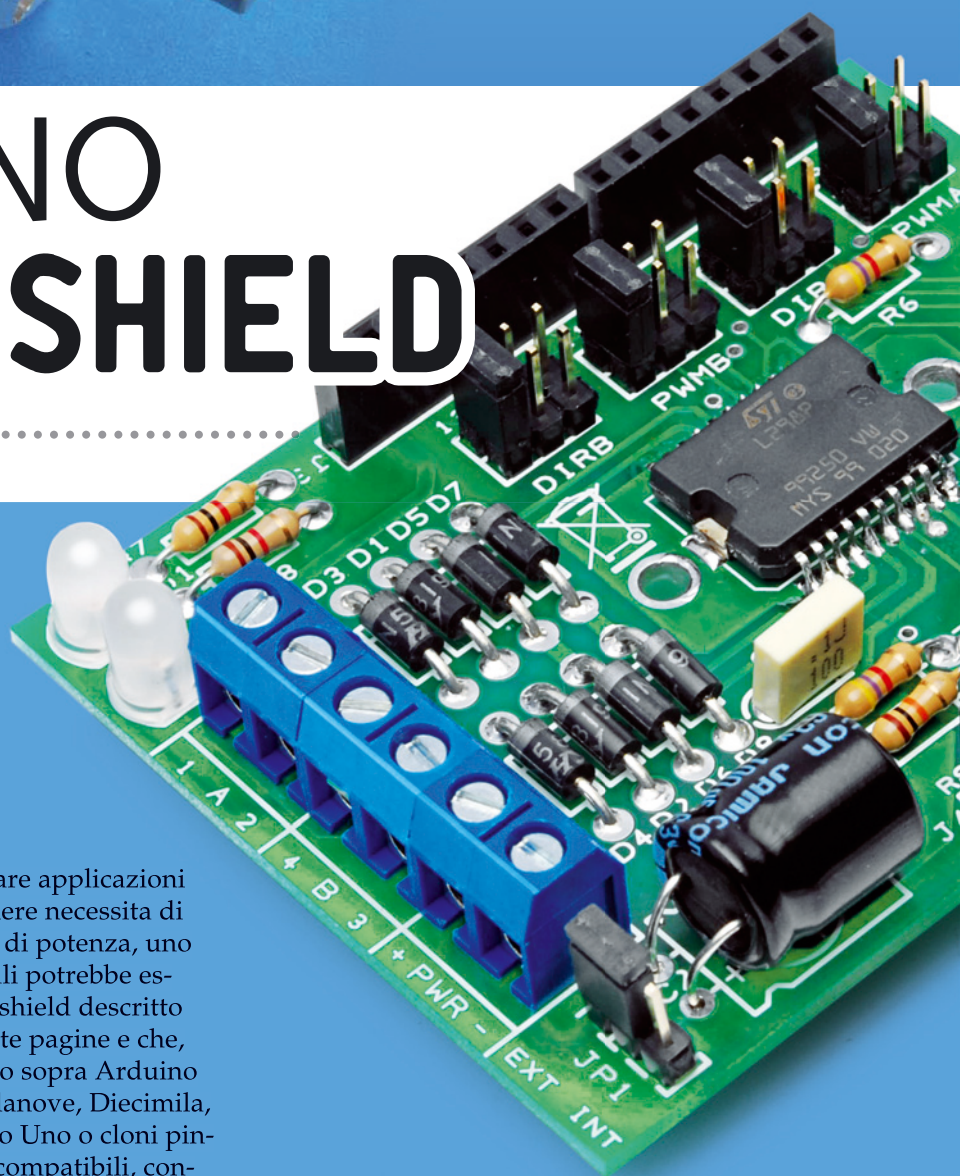
Montato sopra la popolare scheda a microcontrollore, permette di comandare due motori indipendentemente, gestendo sia la velocità, sia il verso di rotazione.

ARDUINO MOTOR SHIELD

di BORIS LANDONI

Uno degli impieghi cui sovente viene destinato Arduino, è il controllo di robot o di radiomodelli; in questi casi le linee di I/O digitali sono destinate a comandare l'attivazione di motori e motoriduttori, però possono riuscire nell'intento solamente se dotate di stadi amplificatori di corrente o line-driver, dato che la corrente da essi erogabile o assorbibile è dell'ordine dei 40 mA. Ecco quindi che chi intende

realizzare applicazioni del genere necessita di circuiti di potenza, uno dei quali potrebbe essere lo shield descritto in queste pagine e che, montato sopra Arduino Duemilanove, Diecimila, Arduino Uno o cloni pin-to-pin compatibili, con-



sente la gestione di due motori a spazzole in continua. La scheda che vi proponiamo è basata sul doppio driver a ponte L298P, in grado di pilotare sia motori DC che passo-passo; dispone di una morsettiera per il collegamento di due motori DC o un motore passo-passo bipolare e di pin strip maschio/femmina che consentono l'inserimento in cascata di altre schede. L'alimentazione per i motori può essere fornita dall'esterno ($7 \div 46 V_{cc}$) o prelevata da Arduino, tenendo però presente che in questo caso non è possibile richiedere più di 2 ampere. Ciascuno dei due canali d'uscita dello shield è in grado di fornire una corrente massima di 2 A, il che basta a pilotare due motori in continua da 2 A ciascuno o un motore passo-passo bipolare con un assorbimento di 2 A per fase.

SCHEMA ELETTRICO

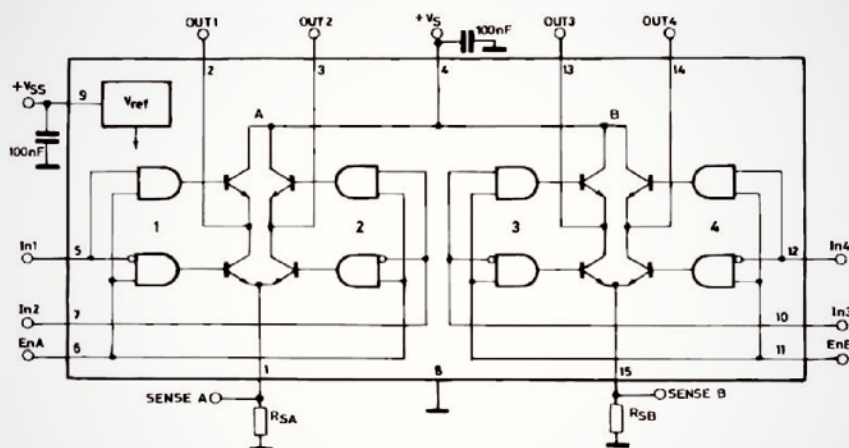
Per comprendere le potenzialità del nostro shield facciamo riferimento allo schema che ne mostra il circuito elettronico: tutto è imperniato sul driver L298P, un doppio driver a ponte ad H

a BJT interamente gestibile mediante i piedini di controllo EN (enable) e IN (direzione). Ogni sezione di questo componente è formata da un ponte a transistor bipolari NPN ed ogni ponte è formato da due metà in cui i BJT sono connessi a simmetria quasi complementare, ossia il transistor collegato al positivo è connesso con il collettore e quello inferiore si collega a massa con l'emettitore; dunque, l'uscita è prelevata fra l'emettitore del transistor connesso al positivo e quella del transistor collegato a massa. Le basi dei transistor di ogni mezzo ponte vengono pilotate distintamente da segnali logici ricavati da porte AND che hanno un ingresso comune e due input distinti e collegati agli ingressi IN dell'intergato; l'ingresso comune è l'EN e attiva il ponte in modo da fargli ricevere i segnali logici dovuti all'impostazione degli IN. Questa configurazione consente sia di sfruttare l'integrato per comandare quattro motori a spazzole in un solo verso, quindi di usare ogni mezzo ponte distintamente, sia di usare un intero ponte in

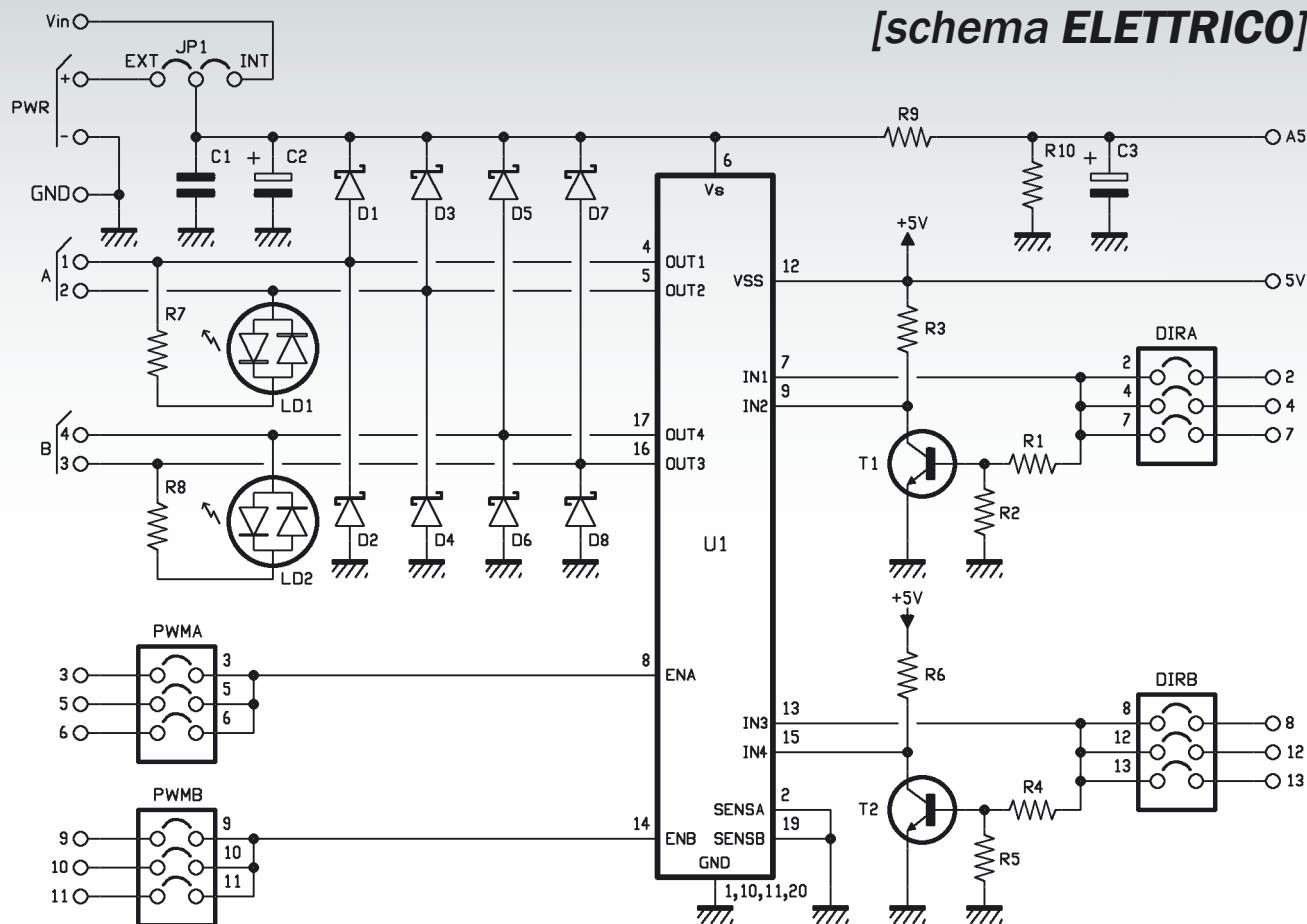
modo combinato (fornendo agli ingressi di comando IN livelli logici opposti) per gestire la rotazione di un motore in un verso e nell'altro.

Per meglio comprendere il funzionamento di ogni singolo ponte, prendiamo in considerazione il primo, che ha per uscite i piedini 2 e 3 (OUT1 e OUT2) e partiamo dalla condizione in cui il rispettivo EN (ENA, piedino 8) sia a zero logico: in questo caso le basi di entrambi i rami del ponte sono a livello basso ed i transistor sono tutti in interdizione; le uscite si trovano in una condizione di alta impedenza, nel senso che non conducono in alcuna direzione e non erogano alcuna corrente.

Ora poniamo ENA a livello logico alto, applicandogli un potenziale di almeno 5 volt: le uscite delle quattro AND del rispettivo ponte ora assumeranno una condizione che dipenderà da quella degli ingressi di comando IN1 (piedino 7) e IN2 (piedino 9); applicando a ciascuna di esse lo stato logico alto, il rispettivo mezzo ponte presenterà all'uscita lo stato logico alto. Ad esempio, ponendo il pin 7 ad uno logico, il transistor collegato al positivo condurrà e determinerà in uscita (OUT1) il livello alto di tensione, mentre quello connesso a massa risulterà interdetto (perché la AND che lo comanda ha l'ingresso collegato ad IN1 negato). In questa condizione, se lasciamo IN2 a zero logico il rispettivo mezzo ponte rimane con il transistor superiore interdetto e quello inferiore in conduzione, mentre se portiamo anche IN2 a livello logico alto, l'uscita OUT2 assumerà anch'essa il livello alto, ovvero il transistor superiore sarà in conduzione e quello inferiore in interdizione. Invertendo le cose, ossia lascian-



L'integrato L298, sia in versione per montaggio comune, sia nella SMD, contiene un doppio ponte ad H formato da transistor bipolari NPN.



do IN1 a zero e ponendo IN2 allo stato logico uno, OUT1 presenterà il livello basso e OUT2 assumerà l'1 logico. Da quanto esposto, deriva che il comando del ponte 1 avviene secondo la Tabella 1; per il secondo ponte il discorso è analogo. Inoltre, vediamo che per controllare un motore a spazzole facendone invertire il senso di rotazione dobbiamo collegarne i capi tra OUT1 e OUT2, quindi applicare, per ogni verso, una coppia di livelli logici in opposizione agli input di comando IN1 e IN2; invece, per comandare un motore in una sola direzione ci basta mezzo ponte e dobbiamo applicare il livello logico alto all'ingresso di comando corrispondente (in tal caso il motore va collegato tra l'OUT1 o 2 e massa).

Da questo modo di funzionamento possiamo dedurre che controllando un motore a spazzole collegato tra le uscite di uno dei ponti, possiamo farlo girare in un verso o nell'altro semplicemente applicando tensioni discordi agli ingressi di comando, mentre per frenarlo dobbiamo semplicemente porre a zero logico i predetti ingressi: in tal caso le uscite del ponte cortocircuitano a massa i capi del motore, annullando la corrente che il motore stesso tenderebbe a produrre girando a vuoto, quindi caricando ed esercitando un'azione frenante. Bene, spiegato come funziona l'L298P, possiamo vedere in che modo è stato impiegato nel nostro shield: l'integrato funziona nella classica configurazione, alimentato tra il piedino 6 (posi-

tivo) e i pin 1, 10, 11, 20 (massa) e viene gestito con le linee di enable e con quelle di comando da Arduino. Le uscite sono state portate a due morsettiere e dotate ognuna di un LED bicolore a due terminali (dotato di resistenza serie per limitarne la corrente) che quindi si illumina di rosso per indicare una polarità e di verde per segnalare la polarità opposta; più esattamente, per il primo ponte, quando è positiva OUT2 il diodo s'illumina di verde, mentre il colore rosso corrisponde ad avere l'uscita OUT1 positiva rispetto alla OUT2. Per quanto riguarda il secondo ponte, il led bicolore diventa verde quando l'uscita OUT4 è positiva rispetto alla OUT3 e rosso nel caso contrario. Ciascuna delle uscite è dotata di diodi Schottky di protezione,

Tabella 1

IN1	IN2	ENA	OUT1	OUT2
0	0	0	alta impedenza	alta impedenza
1	0	0	alta impedenza	alta impedenza
0	1	0	alta impedenza	alta impedenza
1	1	0	alta impedenza	alta impedenza
0	0	1	0	0
1	0	1	1	0
0	1	1	0	1
1	1	1	1	1

Tabella di verità del funzionamento di uno dei ponti dell'L298; per le uscite, 0 logico significa che il transistor inferiore chiude a massa l'uscita corrispondente.

uno collegato al positivo e l'altro al negativo di alimentazione e disposti in modo che nel normale funzionamento siano interdetti; i diodi conducono solo se, per effetto del carico induttivo quale è il motore elettrico, alle uscite tornano impulsi di tensione inversa, cosa che accade tipicamente al distacco dell'alimentazione o all'inversione di polarità, ovvero se si manda il piedino di enable a zero logico (in tal caso i transistor non conducono ed isolano il motore, con la conseguenza che si innescano sovratensioni che senza diodi si scaricherebbe-

ro sui transistor danneggiandoli). Per quanto riguarda il comando dell'L298P, abbiamo previsto di gestire le linee IN1 e IN2 con i piedini 2, 4 e 7 (I/O digitali) di Arduino e le IN3 e IN4 con 8, 12, 13 di Arduino (ancora I/O digitali); siccome abbiamo deciso di realizzare un comando non di tipo on/off o avanti/indietro, ma con regolazione della velocità, pilotiamo gli ingressi ENA ed ENB con dei segnali PWM. Allo scopo, abbiamo previsto di connettere tali input alle linee di Arduino che permettono di restituire un segnale

PWM, ossia i contatti 3, 5 e 6 per ENA e 9, 10, 11 per ENB. Applicando agli enable un segnale PWM, otteniamo l'accensione e lo spegnimento del ponte alla frequenza dell'onda rettangolare di comando e per i periodi dettati dal duty-cycle del PWM stesso; ciò consente di ottenere la regolazione della tensione media applicata ai motori e quindi del regime di rotazione, mantenendo una coppia elevata anche a basso numero di giri.

Notate che per le linee di enable abbiamo previsto dei jumper che ci servono a decidere localmente quale uscita PWM di Arduino debba comandarle; i ponticelli vanno chiusi uno alla volta per ogni singolo enable; i jumper servono perché se insieme al nostro shield ne montiamo altri a sandwich, può essere che qualcuno di essi impieghi uno degli I/O PWM e che quindi sia necessario passare il controllo ad un'altra.

Torniamo adesso al controllo di direzione, operato mediante



Nel futuro del volo c'è l'elettrico

in fibra di carbonio e composito. L'aereo, il cui completamento è previsto per il 2035, verrà mosso

Non solo a bordo delle auto: la propulsione elettrica si appresta a salire sugli aerei per portarci dove oggi ci portano i convenzionali turbofan e turboelica. Ce lo dice EADS, la multinazionale europea del settore aerospaziale cui appartiene il consorzio Airbus, che sta lavorando alla realizzazione di Voltair: si tratta di un velivolo a breve raggio, da 50/70 posti, con ali ad assetto variabile (che si adatta, in funzione delle diverse fasi del volo) fusoliera

da motori elettrici alimentati da batterie realizzate con materiali superconduttori, poco pesanti e capaci di immagazzinare molta energia a parità di peso. Il gruppo delle batterie alimenterà i motori, che muoveranno due eliche controrotanti propulsive poste sulla coda dell'aereo. Il progetto VoltAir guarda allo sviluppo dell'aviazione del futuro, basata su aerei a basso impatto ambientale e capaci di fare a meno del petrolio.

IN1 e 2 ed IN3 e 4: come potete vedere, per comandare un ponte è sufficiente una linea digitale, in quanto per com'è fatto il circuito, un ingresso si trova sempre al livello logico opposto rispetto all'altro. Con riferimento al primo ponte, se applichiamo lo stato alto ad una delle linee 2, 4, 7, IN1 dell'L298P si trova ad 1 logico e inoltre T1 va in saturazione (perché polarizzato attraverso R1) e pone a livello basso IN2; viceversa, ponendo a zero una delle linee 2, 4, 7, IN1 è a livello basso e IN2, a causa dell'interdizione di T1, allo stato alto, assicurato dalla resistenza di collettore R3. Analogo discorso va applicato al secondo ponte, quello che ha per ingressi IN3 ed IN4. I motori vanno collegati alle morsettiere A e B; quanto all'alimentazione dello shield, per scegliere la fonte è stato previsto il jumper JP1.

Per prelevare la tensione da Arduino (arriva dai contatti Vin e GND) bisogna chiudere il ponticello tra INT e la linea centrale, mentre per prendere alimentazione dall'esterno occorre fare ponticello su EXT; in quest'ultimo caso lo shield va alimentato dalla morsettiera PWR e con Arduino viene condivisa la sola massa.

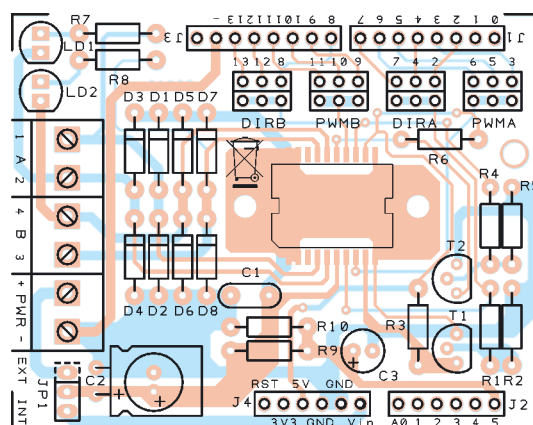
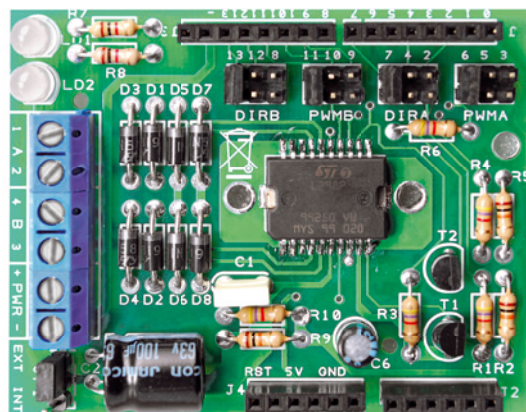
Il nostro shield prevede un collegamento con l'ingresso analogico A5 di Arduino (attraverso le due resistenze R9 ed R10) utile per misurare la tensione di alimentazione; i valori del partitore permettono di far giungere sull'A5 un potenziale pari ad 1/11 di quello presente sul piedino 6 dell'U1. Rammentate che su questo ingresso non può essere applicata una tensione superiore a 5 V, ma tale condizione è sempre rispettata, dato che la tensione d'alimentazione, anche se esterna, deve restare compresa tra 7 e 46 V.

Elenco Componenti:

R1, R6: 4,7 kohm
R2: 10 kohm
R3, R4: 4,7 kohm
R5: 10 kohm
R7, R8: 1 kohm
R9: 100 kohm
R10: 10 kohm
C1: 100 nF 100 VL
poliestere
C2: 100 µF 63 VL
elettrolitico
C3: 22 µF 35 VL
elettrolitico
U1: L298P
LD1: Led 5 mm R/V
(L-57EGW)
LD2: Led 5 mm R/V
(L-57EGW)
D1 ÷ D8: 1N5819
T1, T2: BC547

Varie:

- Morsetto a 2 poli (3 pz.)
- Strip maschio 3 poli (9 pz.)
- Jumper (5 pz.)
- Strip M/F 6 poli (2 pz.)
- Strip M/F 8 poli (2 pz.)
- Circuito stampato

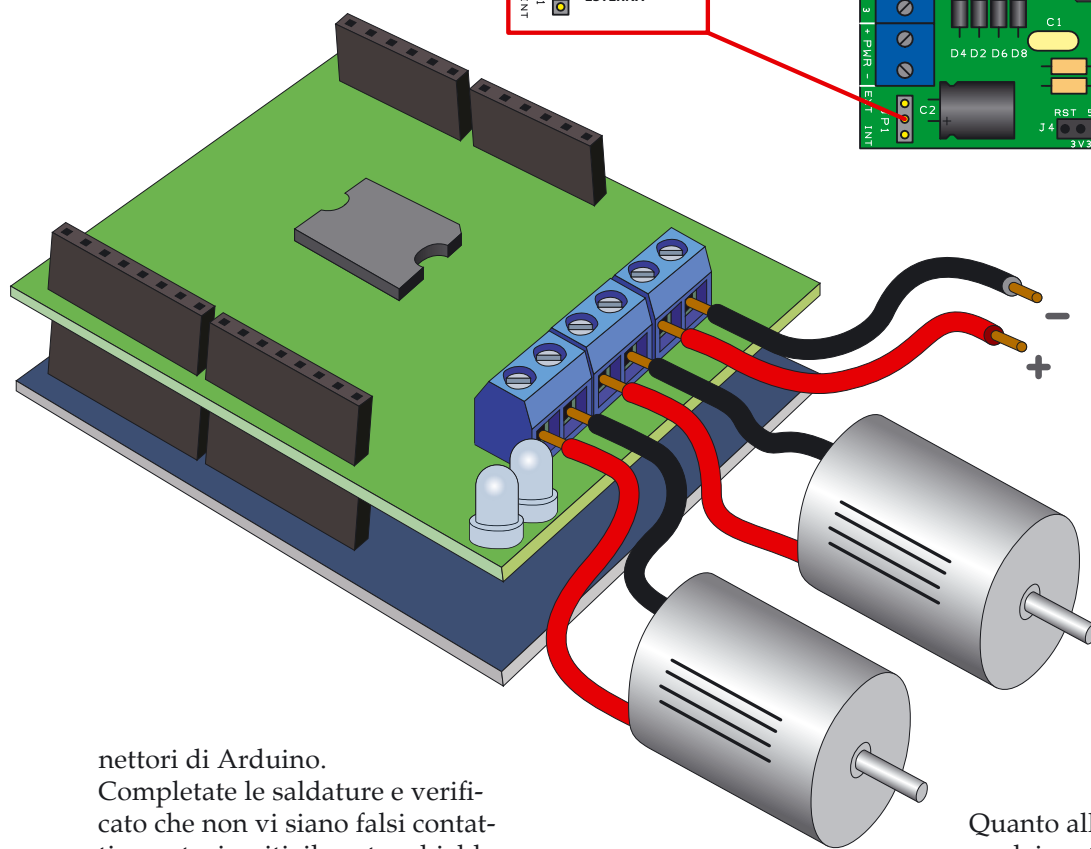
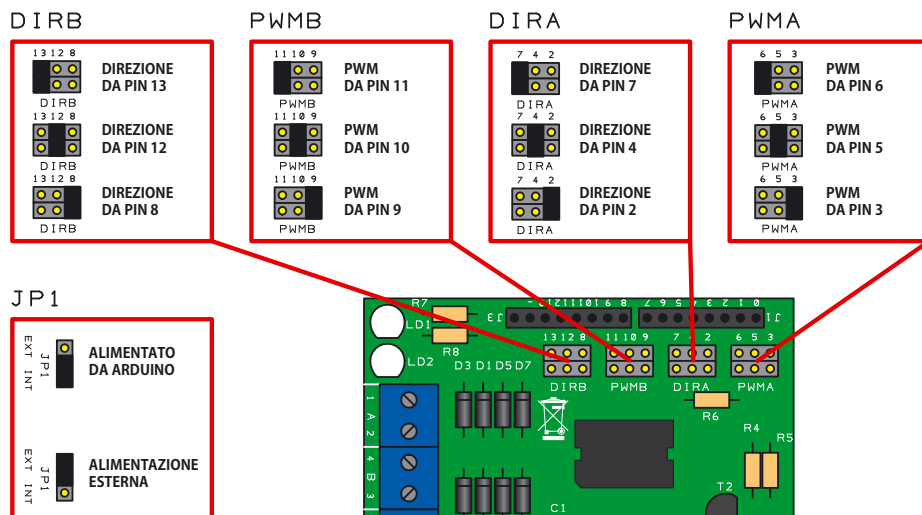


REALIZZAZIONE PRATICA

Lo shield è di facile realizzazione; l'unico componente che richiede un po' d'attenzione è il chip SMD L298P, che dovete saldare partendo dai pin posti agli angoli e dopo aver centrato bene il componente nelle rispettive piazzole (ricordate di posizionare l'angolo smussato verso C1). Terminata la saldatura dei piedini, stagnate bene anche le due alette laterali del chip, che permetteranno di smaltire attraverso le zone ramate dello stampato il calore prodotto. Per saldare l'L298P e il resto dei componenti, utilizzate un saldatore da non più di 25 watt e del filo di lega saldante da 0,75 mm massimo. Sistemato l'integrato, inserite e saldate le resistenze, i diodi (ricordando che il catodo è il terminale in corrispondenza della

fascetta), i condensatori (prestando attenzione alla polarità degli elettrolitici) e poi i transistor e i LED bicolore, per i quali dovete aver cura di posizionare il lato smussato dalla parte delle morsettiere; quindi inserite e saldate le morsettiere e i pin strip a passo 2,54 mm con cui realizzerete i jumper: ne servono due file da tre per ciascuno dei DIRB, PWMB, DIRA, PWMA e una sola fila da tre per JP1. Completate il montaggio con i connettori S.I.L. a passo 2,54 mm femmina a pin lunghi (almeno 20 mm): ne occorrono due da otto contatti per J1 e J3 ed altrettanti, ma da sei contatti l'uno, per J2 e J4. Noterete che introducendo i connettori femmina coi terminali lunghi nei rispettivi fori dello stampato, i pin permetteranno allo shield di introdursi nei con-

Impostazione dei jumper riguardanti il verso di rotazione dei motori dei due canali (funzionamento a ponte) e la gestione dell'ingresso PWM, nonché la scelta dell'alimentazione per lo shield.



Schema di collegamento dei motori (nel caso di comando a ponte con inversione del verso di rotazione) e dell'alimentazione esterna (prelevata dall'apposita morsettiera).

nettori di Arduino. Completate le saldature e verificato che non vi siano falsi contatti o cortocircuiti, il vostro shield è pronto per essere configurato e utilizzato. Se servisse, sull'integrato L298P può essere applicato un eventuale dissipatore di calore da 18 °C/W (ad esempio l'ML26 per TO-220, sagomato ad U) che può essere fissato alla scheda con due viti sfruttando gli appositi fori.

USARE LO SHIELD

La prima cosa da fare prima di

utilizzare lo shield è definire mediante i jumper DIRB, PWMB, DIRA, PWMA quali pin di Arduino dovranno comandare la direzione (DIR) e la velocità (PWM) dei due canali A e B. La Fig. 1 mostra come impostare i jumper in modo da assegnare il contatto 13 di Arduino a DIRB, il contatto 11 a PWMB, il 7 a DIRA ed il 6 a PWMA.

Quanto alla fonte di alimentazione dei motori, chiudete con un jumper EXT sul centrale se optate per l'alimentazione esterna (che avrete cura di applicare alla morsettiera + e - PWR, rispettando rigidamente la polarità indicata, dato che altrimenti i diodi Schottky cortocircuiteranno l'alimentatore) o su INT se preferite far alimentare lo shield da Arduino. Ad esempio, la Fig. 2 mostra l'impostazione nel caso di alimentazione esterna.

Listato 1 - Lo sketch per Arduino

```
/*
10-5-10
Pete Dokter
SparkFun Electronics
*/

int pwm_a = 3; //PWM control for motor outputs 1 and 2 is on digital pin 3 (or 5,6)
int pwm_b = 9; //PWM control for motor outputs 3 and 4 is on digital pin 9 (or 10,11)
int dir_a = 2; //direction control for motor outputs 1 and 2 is on digital pin 2 (or 4,7)
int dir_b = 8; //direction control for motor outputs 3 and 4 is on digital pin 8 (or 12,13)

void setup()
{
  pinMode(pwm_a, OUTPUT); //Set control pins to be outputs
  pinMode(pwm_b, OUTPUT);
  pinMode(dir_a, OUTPUT);
  pinMode(dir_b, OUTPUT);

  analogWrite(pwm_a, 100); //set both motors to run at (100/255 = 39)% duty cycle (slow)
  analogWrite(pwm_b, 100);
}

void loop()
{
  digitalWrite(dir_a, LOW); //Set motor direction, 1 low, 2 high
  digitalWrite(dir_b, LOW); //Set motor direction, 3 high, 4 low

  delay(1000);

  analogWrite(pwm_a, 255); //set both motors to run at 100% duty cycle (fast)
  analogWrite(pwm_b, 255);

  delay(1000);

  digitalWrite(dir_a, HIGH); //Reverse motor direction, 1 high, 2 low
  digitalWrite(dir_b, HIGH); //Reverse motor direction, 3 low, 4 high

  delay(1000);

  analogWrite(pwm_a, 100); //set both motors to run at (100/255 = 39)% duty cycle
  analogWrite(pwm_b, 100);

  delay(1000);
}
```

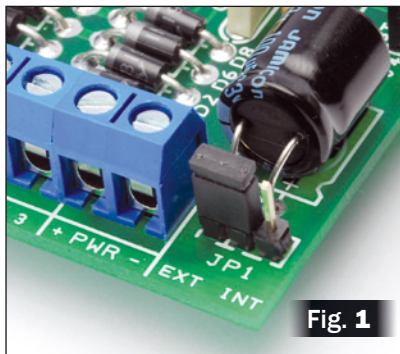


Fig. 1

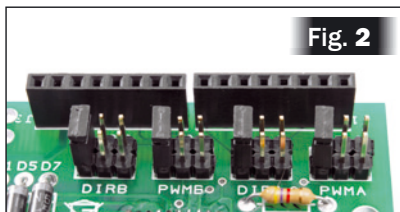
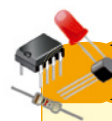


Fig. 2

Terminate le impostazioni è possibile montare lo shield sulla scheda Arduino e provare ad azionare i motori.

Nel **Listato 1** trovate lo sketch che vi permetterà di comandare i motori con Arduino, utilizzando per la gestione della direzione del canale A la linea digitale 2 e per quella del canale B la linea digitale facente capo al pin 8; il PWM per il canale A arriverà dal contatto 3 e quello per il canale B, dal 9. Sapete quindi come regolarvi per la chiusura dei jumper, nel senso che DIRA andrà chiuso su 2, DIRB su 8, PWMA su 3 e PWMB su 9. Buon lavoro!



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato e lo sketch sono scaricabili gratuitamente dal sito della rivista (www.elettronica.in.it).

Il motor shield è anche disponibile in scatola di montaggio (cod. MOTORSHIELD_FE) al prezzo di Euro 18,00 IVA compresa.

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