

Elettronica In

oltre l'elettronica

www.elettronica.in.it

Anno XXI - n. 192
Febbraio 2015

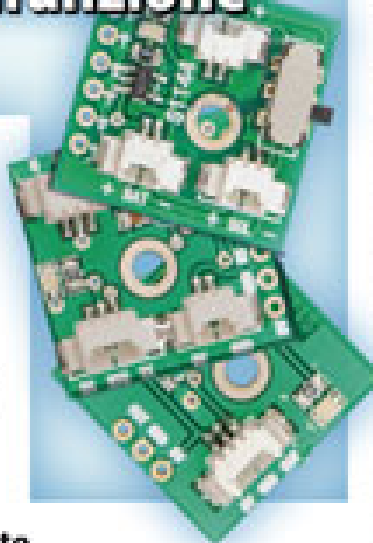
ArduDisplay
quando il
messaggio
diventa
interattivo

EMI:
Interfaccia
Flowcode



- Realizziamo un web server con RandA
- Scratch, le strutture logiche
- Nuove funzioni per la centrale di allarme
- È tempo di CeBIT

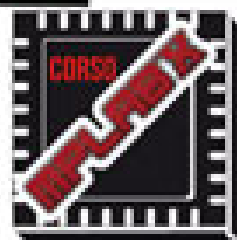
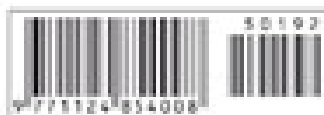
Breakout Board
multifunzione

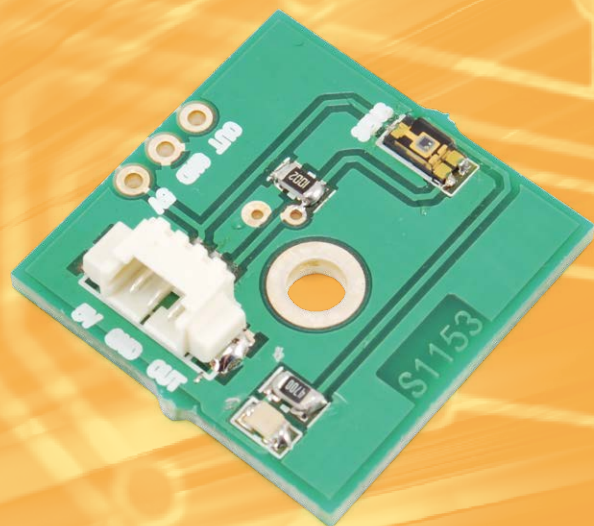


- Robot Spazzaneve, il firmware
- Current Sense Amplifier
- Linino ONE, ci hanno ristretto Arduino Yún
- Autoveicoli sempre più sicuri grazie all'elettronica



DM Board ICS
Telecontrollo GSM
universale





Proseguiamo la presentazione delle schede per prototipazione con tre nuove, piccole board dedicate alla carica delle Li-ion, alla connessione RS485 e al rilevamento del livello di illuminazione, ognuna basata su uno specifico componente SMD che potete già trovare cablato.

BREAKOUT BOARD MULTIFUNZIONE

di DAVIDE SCULLINO

Oggigiorno molti dei componenti integrati più interessanti sono disponibili solamente in versione a montaggio superficiale e in contenitori che, per esigenze di miniaturizzazione dei sistemi portatili, si presentano sempre più piccoli e difficili da saldare con le tradizionali attrezzature hobbistiche; è questa la ragione per cui abbiamo fatto una selezione di circuiti integrati e sensori che realizzano ciascuno una funzione utile in svariate applicazioni ma che sono in SMD e con essi abbiamo realizzato delle breakout board (che sono poi schede di prototipazione) ognuna delle quali rende accessibili le connessioni di uno di questi. Dopo aver descritto, nel fascicolo di Dicembre/Gennaio 2015, le prime tre schede, riprendiamo il discorso analizzandone altre tre:

una basata su un sensore di luce, una dotata di un integrato caricabatterie per un elemento agli ioni di litio ed un'ultima dedicata all'interfacciamento via RS485; gli integrati interessati sono, rispettivamente, il TEMENT6000, l'MCP73831T e l'ST485BDR. Analizziamo prima di tutto, individualmente, gli schemi elettrici delle singole breakout board.

SCHEDA CON SENSORE DI LUCE

Basata sul rilevatore TEMENT6000 della Vishay (www.vishay.com), questa breakout board è destinata a chi necessita di testare o integrare un sistema di rilevamento dell'illuminazione nell'ambiente. Il TEMENT6000 è un fototransistor NPN in silicio in esecuzione SMD, la cui disposizione dei contatti è in linea con quella adottata

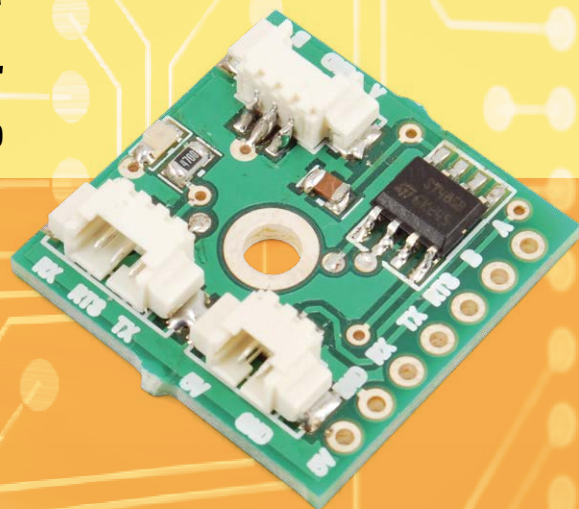
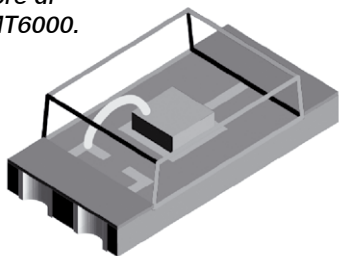


Fig. 1 - Aspetto del sensore di luce TEMT6000.



da molti componenti moderni: nessun piedino, ma contatti ad L che partono dal lato e proseguono sotto il corpo. A causa di ciò, la saldatura con sistemi tradizionali è difficoltosa e si procede (almeno in produzione industriale) con sistemi ad aria calda o ad infrarossi.

Il componente è conforme alla norma RoHS 2002/95/EC (e alla WEEE 2002/96/EC) per quanto riguarda l'assenza di piombo e si può quindi utilizzare nelle applicazioni in cui tale requisito è imposto (tutte quelle civili). Il fototransistor ha la giunzione base-emettitore affacciata ad una finestrella passivata sulla superficie superiore ed è sensibile allo spettro della luce visibile; la sensibilità è garantita entro un angolo di $\pm 60^\circ$ rispetto alla perpendicolare a detta superficie. Il funzionamento del fototransistor è semplice: polarizzandone la giunzione di collettore con una tensione -positiva nel caso dell'NPN- rispetto all'emettitore ed esponendo alla luce la base, l'energia della radiazione luminosa rompe i legami tra le cariche elettriche in eccesso e il reticolo cristallino del silicio drogato che compone il transistor, liberando le cariche elettriche, le quali danno origine a una corrente di base, amplificata dal campo elettrico tra collettore e base. Ne deriva una corrente di collettore, la cui intensità è direttamente proporzionale a quella dell'intensità luminosa, seppure la relazione non sia perfettamente lineare

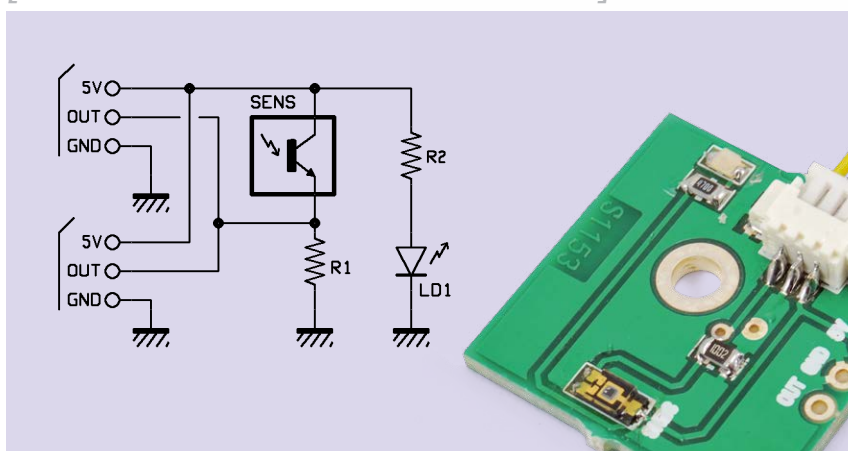
(ma ci va molto vicina). Per quanto riguarda le connessioni, nel TEMT6000 vengono resi disponibili i soli terminali di collettore ed emettitore; la base non è accessibile. Tale scelta è, per un sensore di illuminamento dell'ambiente, adatta allo scopo, in quanto non bisogna rilevare rapide variazioni di luce. Di norma nei fototransistor è utile collegare la base all'emettitore con una resistenza di valore relativamente elevato che accelera la ricombinazione delle cariche liberate nella giunzione base-emettitore dalla luce, solamente nelle applicazioni in cui il componente debba rilevare con precisione impulsi di luce; infatti in tal caso, quando la frequenza delle variazioni dell'intensità luminosa sulla giunzione è elevata, il tempo necessario alle cariche liberate dalla luce per ricombinarsi e lasciar interdire il fototransistor diventa comparabile con la durata degli impulsi e quindi il transistor non riesce più a seguire le variazioni stesse. Collegando la base all'emettitore con una resistenza, si crea un flusso di corrente che accelera la ricombinazione velocizzando la commutazione del fototransistor, anche se tale soluzione causa

una riduzione della sensibilità del componente, nel senso che a parità di intensità luminosa la corrente di collettore diminuisce. Per utilizzare il TEMT6000 bisogna sapere che, essendo un fototransistor, va polarizzato come un transistor usato da amplificatore (tipicamente a emettitore o a collettore comune); nel nostro circuito, SENS lo trovate configurato a collettore comune, il che significa prelevare ai capi della resistenza collegata tra l'emettitore e massa una differenza di potenziale che esprime l'intensità luminosa sulla base, ovvero l'illuminamento, espresso in lux.

La connessione a collettore comune consente la massima linearità del fototransistor, ovvero di avere una relazione praticamente lineare, come mostra la Fig. 2, tra l'illuminamento e la corrente di collettore e quindi quella di emettitore; siccome quest'ultima determina la tensione ai capi della resistenza di emettitore ($R1$, nello schema elettrico...) otteniamo una tensione d'uscita, tra i contatti OUT e GND che varia linearmente e in modo direttamente proporzionale rispetto all'illuminamento.

Nell'utilizzare il TEMT6000 bisogna ricordare le sue caratte-

[schema elettrico TEMT6000]



ristiche principali, che (a temperatura ambiente, tipicamente 25 °C...) sono una Vce massima (in assenza di luce) di 6 volt e una Vce inversa che non deve superare 1,5 V ed una massima corrente di collettore di 20 mA.

Nella nostra breakout board queste caratteristiche sono soddisfatte, in quanto la tensione di alimentazione è 5 volt e quella tra collettore ed emettitore è inferiore, dato che ai 5 volt va sottratta la caduta su R1.

La corrente di collettore Ic non raggiunge mai i 20 mA, dato che, essendo R1 da 10 kohm, non arriverebbe mai neppure a 0,5 mA. In condizioni di bassa illuminazione, OUT presenta circa 0,5 volt, ma con una torcia elettrica puntata sul sensore raggiunge quasi 5 volt.

Concludiamo l'analisi della scheda con il LED, che è stato inserito per segnalare la presenza dell'alimentazione a 5 volt.

Per offrire la massima versatilità, tutte le linee (alimentazione, tensione di uscita) sono disponibili sia in piazzole sui bordi del circuito stampato, sia su pin-strip per l'inserimento su altri PCB. Per quanto riguarda le applicazioni, la breakout board si presta a fare da rilevatore di

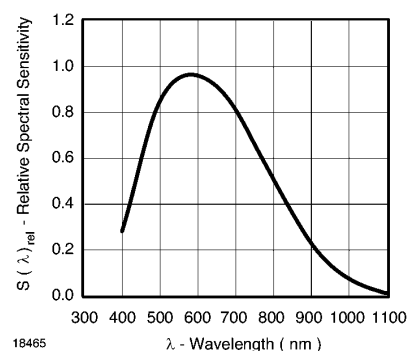
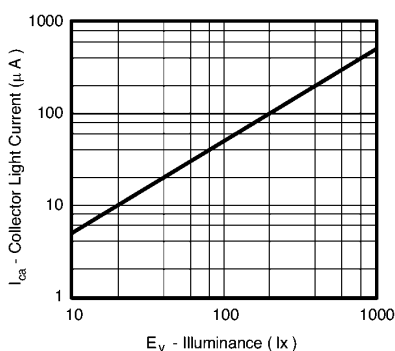


Fig. 2 - Curve che descrivono la corrente di collettore del TEMT6000 in funzione dell'illuminamento (a sinistra) e la risposta spettrale (a destra).

luce ambientale per il controllo automatico dell'illuminazione di un locale, ovvero della backlight (retroilluminazione) di display LCD, ma anche da sensore per realizzare un luxmetro dal quale non è richiesta una precisione elevatissima; può essere utile anche per governare l'accensione automatica delle luci in galleria negli autoveicoli.

SCHEDA CON RS485

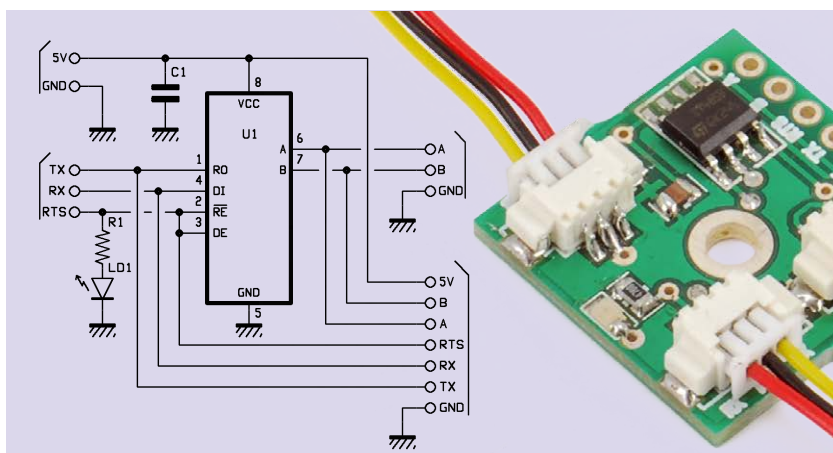
Passiamo all'analisi della seconda breakout board di questo articolo, ossia quella che realizza l'interfacciamento tra una linea seriale TTL (per esempio l'uscita di un UART discreto o integrato in un microcontrollore) ed un bus RS485, seriale anch'esso, però a linea bilanciata composta

da due fili rispetto a massa, invece che sbilanciata (un conduttore riferito a massa) come è nella classica seriale TTL.

La RS485 è una linea di comunicazione dati in simplex destinata ad ambienti elettricamente molto rumorosi e a coprire lunghe distanze: fino a 1,2 km. Le prestazioni in fatto di immunità al rumore elettrico e della conseguente possibilità di coprire grandi distanze, si devono al funzionamento a loop di corrente e all'ampiezza della tensione del segnale dati che caratterizzano l'RS485, i cui driver (stadi trasmettitori) producono sui due fili una tensione differenziale rispetto a massa, quindi a parità di tensione di alimentazione, l'ampiezza è doppia rispetto a quella ottenibile da una linea sbilanciata. Bilanciata significa che la linea vede, ad ogni impulso, due tensioni di pari ampiezza ma polarità opposta sui fili (A e B) della linea RS485, il che permette,

nel ricevitore, di utilizzare un amplificatore differenziale per sommare algebricamente i segnali abbattendo, in teoria, il disturbo. Infatti, collegando i due fili di

[schema elettrico RS485]



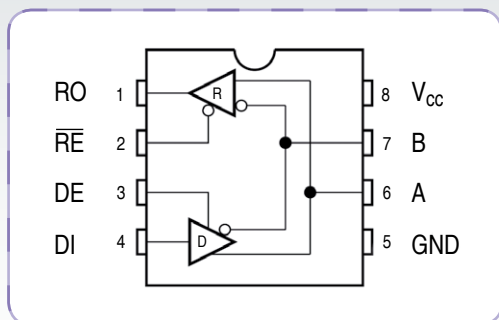


Fig. 3 - Piedinatura e connessioni interne dell'ST485DBR.

segnale uno all'ingresso invertente e l'altro al non-invertente di un differenziale (ad esempio un amplificatore operazionale configurato come comparatore di tensione) i segnali vengono sommati algebricamente, quindi quando il primo ingresso è negativo e il secondo positivo l'uscita è negativa (o a zero logico) mentre con la situazione opposta in uscita si ottiene il livello logico alto. Il rumore captato dai fili, che si suppone di pari ampiezza su A e B, teoricamente viene annullato, perché quello che entra dall'ingresso non-invertente viene compensato con quello che arriva sull'invertente.

L'RS485 è un bus in quanto è possibile collegare in parallelo ai fili della linea più dispositivi, l'ultimo (il più distante) dei quali deve essere terminato con un resistore da 120 ohm posto tra A

e B. Il data-rate consentito dalla RS485 è di ben 35 Mbit/s fino a 10 m di distanza e di 100 kbit/s a 1.200 m.

Nel circuito qui descritto, per la conversione TTL/RS485 e viceversa viene utilizzato un transceiver integrato ST485BDR (nel relativo schema elettrico è siglato U1) prodotto dalla ST e disponibile in contenitore SO-8 (una sorta di dip 4+4 piedini per SMD) o DIP; noi usiamo l'SO-8. Un transceiver è un componente che integra (Fig. 3) sia una sezione trasmittente (D) che una ricevente (R), nel nostro caso attivabili distintamente tramite appositi piedini.

Per l'esattezza, DE attiva, quando posta a livello logico alto (5 volt) la trasmittente mentre /RE (quando posta a zero logico, ossia massa) attiva lo stadio di ricezione; la logica di questi due ingressi è volutamente invertita in modo da comandarli con una sola linea per ottenere l'attivazione del ricevitore quando il trasmettitore è disabilitato (funzionamento in ricezione dati) ovvero l'attivazione della sezione trasmittente e la contemporanea disabilitazione della ricevente (funzionamento in ricezione). È possibile anche gestire separa-

tamente i due piedini per tenere sempre in funzione la ricevente anche quando l'ST485DBR trasmette, così da ottenere un feedback sulla condizione del trasmettitore; non conviene, invece, in una comunicazione, tenere attivo il trasmettitore quando si è in ricezione di dati da un'altra unità del bus, perché i due TX potrebbero danneggiarsi (anche se il componente della ST incorpora una protezione).

Nel nostro circuito, DE ed /RE sono gestiti insieme e pilotati dal segnale di controllo RTS della seriale (dell'UART); RX e TX di quest'ultima sono connessi, rispettivamente, ad RO (uscita dello stadio ricevente RS485) e a DI (input dello stadio trasmettente RS485).

I fili della linea RS485 corrispondono ai piedini A (piedino 6) e B (piedino 7) e fanno capo, sulla breakout board, ai contatti omonimi.

Per offrire la massima versatilità, tutte le linee (alimentazione, RX, TX, RTS della seriale TTL, A e B dell'RS485) sono disponibili sia in piazzole sui bordi del circuito stampato, sia su pin-strip per l'inserimento su altre schede. Completa il circuito, il solito LD1, che indica, accendendosi, la

[schema elettrico **CARICABATTERIA**]

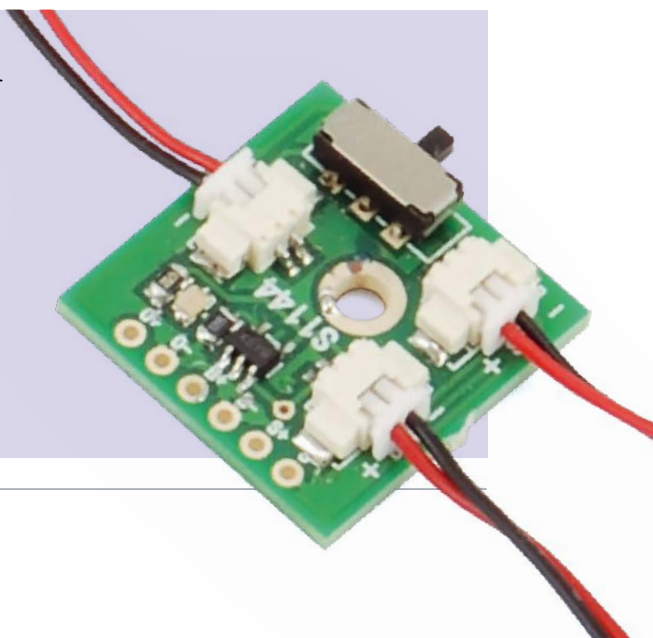
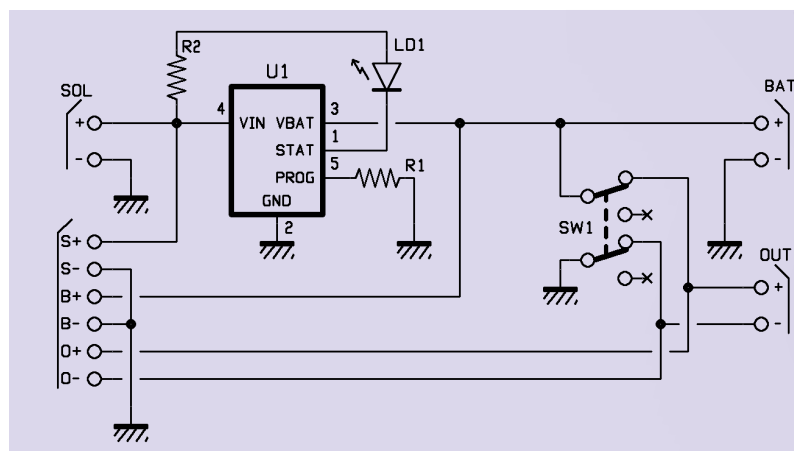


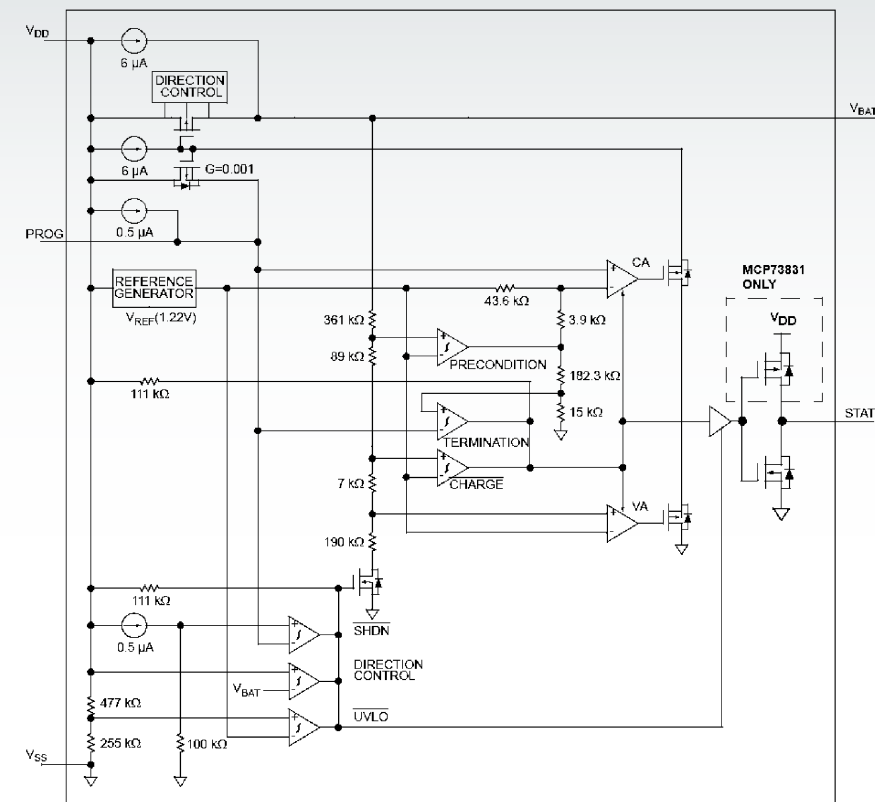
Fig. 4 - Schema a blocchi dell'MCP73831.

presenza dell'alimentazione (è a 5 volt perché l'integrato funziona ad alimentazione singola, appunto, di 5 volt); non manca un condensatore ceramico (C1) che filtra la linea dell'alimentazione. Prima di passare ad esaminare l'ultima breakout board riepiloghiamo le caratteristiche dell'ST485DBR, utili per impiegarlo al meglio: l'integrato accetta su A e B tensioni in modo comune (riferite a massa) comprese tra -7 e 12 V e a riposo, cioè con driver e receiver spenti, presenta un consumo di appena 300 μ A; il trasmettitore, quando è spento, mantiene in uscita uno stato d'alta impedenza, così da non caricare la linea.

Il tempo di propagazione, ossia il ritardo con cui un livello logico applicato a DI si presenta su A e B, ovvero quello con cui il livello tra A e B si presenta ad RO, è di appena 30 ns, il che spiega l'elevata velocità di comunicazione possibile. Al bus, ovvero sui fili A e B, si possono collegare insieme fino a 64 integrati dello stesso tipo, da attivare sempre uno solo alla volta in trasmissione. Lo stadio trasmettitore è provvisto di un limitatore di corrente che, intervenendo, mette l'uscita dello stesso in alta-impedenza; la limitazione è utile ad evitare danni in caso di cortocircuito e qualora venisse attivato, insieme, il trasmettitore di un altro integrato presente sul bus.

SCHEDA CARICABATTERIA

Eccoci giunti all'ultima breakout board: quella che incorpora il caricabatteria per singolo elemento Li-ion da 3,6-3,7 volt (4,2 V a piena carica); l'integrato su cui si basa è l'MCP73831T, un SMD incapsulato in un case SOT-23, prodotto dalla Microchip. La scheda accetta in ingresso (contatti SOL) tipicamente 5 volt,



che possono provenire sia da un comune alimentatore, sia da un pannello fotovoltaico miniatura, ma anche dal connettore USB di un computer; dai contatti SOL, la tensione di alimentazione raggiunge l'ingresso (VIN) del regolatore di carica MCP73831T (U1), il quale accetta all'ingresso tensioni dell'ordine dei 3,75÷6 volt e fornisce all'uscita la corrente occorrente alla carica di elementi agli ioni di litio o ai polimeri di litio (Li-Po) erogando fino a 550 milliampere. La batteria (da collegare ai contatti +/- BAT) può avere una capacità teoricamente illimitata, perché al massimo si carica in un tempo lunghissimo, tuttavia considerate che con una corrente di 550 mA un elemento da 550 mAh si carica in un'ora, uno da 1.100 mAh per la carica richiederà 2 ore, uno da 5,5 Ah necessiterà di 10 ore e via di seguito.

Nella breakout board l'integrato funziona nella configurazione canonica che vede il diodo luminoso LD1 alimentato (mediante

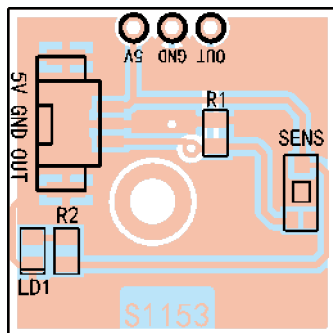
la resistenza R2 che ne limita la corrente) dall'uscita STAT, la quale si pone a livello logico basso durante la carica, mentre rimane a livello alto a fine carica; la stessa si porta ad alta impedenza (open) quando l'MCP73831T è in shutdown o all'uscita VBAT non risulta attualmente collegata alcuna batteria. Quest'ultima uscita (piedino 3) è la linea cui si connette la batteria.

L'integrato opera la carica a corrente e tensione costante. La corrente di carica (Ireg) viene impostata con la resistenza collegata al piedino 5 (R1 nel nostro caso), al cui valore è legata dalla relazione:

$$I_{reg} = 1.000/R$$

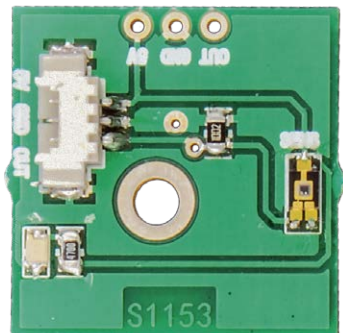
dove il valore della R risulta in ohm se la corrente Ireg è espressa in A. Ad esempio, con 4,7 kohm si ottiene una limitazione a 212 mA, mentre con una R di 2,2 kohm la corrente vale circa 454 mA. Se il piedino 5 viene aperto, l'integrato si porta a riposo ed as-

[piano di montaggio **TEMT6000**]



Elenco Componenti:

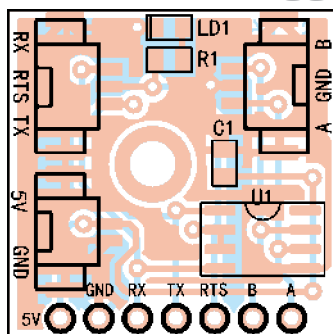
R1: 10 kohm (0805)
R2: 470 ohm (0805)
LD1: LED verde (0805)
SENS: TEMT6000



Varie:

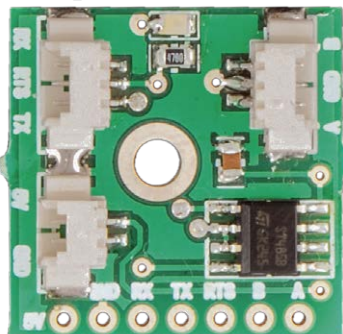
- Strip maschio 3 vie 90°
- Connettore JST 1.25mm 3 vie da CS
- Connettore JST 1.25mm 3 vie volante
- Circuito stampato

[piano di montaggio **RS485**]



Elenco Componenti:

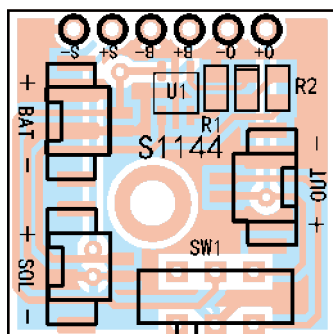
R1: 470 ohm (0805)
C1: 100 nF ceramico (0805)
LD1: LED verde (0805)
U1: ST485BDR



Varie:

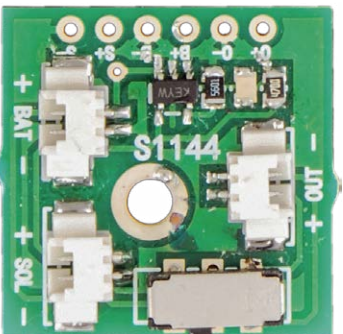
- Strip maschio 7 vie 90°
- Connettore JST 1.25mm 2 vie da CS
- Connettore JST 1.25mm 3 vie da CS (2 pz.)
- Connettore JST 1.25mm 2 vie volante
- Connettore JST 1.25mm 3 vie volante (2 pz.)
- Circuito stampato

[piano di montaggio **CARICABATTERIA**]



Elenco Componenti:

R1: 5,6 kohm (0805)
R2: 470 ohm (0805)
LD1: LED verde (0805)
SW1: Deviatore doppio a slitta 90°
U1: MCP73831T



Varie:

- Strip maschio 6 vie 90°
- Connettore JST 1.25mm 2 vie da CS (3 pz.)
- Connettore JST 1.25mm 2 vie volante (3 pz.)
- Circuito stampato

sorbe appena 2 μ A (shutdown); il piedino può quindi essere usato come enable.

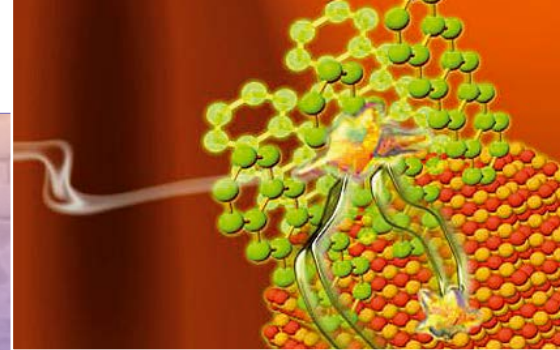
La carica si svolge in diverse fasi: una volta connessa la batteria all'uscita VBAT (piedino 1) se la tensione rilevata risulta minore di una certa soglia (tipicamente intorno al 70 % del valore di piena carica) viene avviato il pre-condizionamento ad un valore di corrente modesto, che serve a preparare la carica rapida; quest'ultima si avvia quando la tensione ai capi della batteria supera la predetta soglia e viene effettuata ad un valore di corrente stabilito dalla resistenza collegata al piedino 5.

Non appena la batteria raggiunge una tensione pari a quella di regolazione impostata nel chip, smette la carica rapida e inizia il mantenimento, che punta a stabilizzare la tensione acquisita a 4,2 V, 4,35 V, 4,4 V o 4,5 V a seconda della versione dell'integrato, che nel nostro progetto è la MCP73831T-2, la quale regola a 4,2 V. Le diverse tensioni di regolazione dell'integrato sono state volute dal costruttore per venire incontro alle esigenze di carica dei nuovi modelli di batterie al litio, che possono lavorare con tensioni anche superiori a 4 volt. A completare il circuito del caricabatteria, abbiamo inserito il deviatore a slitta SW1, che permette di sconnettere l'uscita OUT dalla batteria (e dall'OUT dell'MCP73831) quando si vuole togliere alimentazione al carico, ovvero fin quando la batteria stessa non è carica abbastanza. Come per le due schede appena descritte, è prevista la possibilità sia di accedere ai collegamenti sulle piazzole a bordo scheda, sia di montare la breakout board con un pin-strip (sullo stampato ci sono le apposite piazzole) su altri stampati.

Fotovoltaico oltre il 33,7%? È possibile!

La percentuale della luce solare convertita in elettricità nelle celle fotovoltaiche è teoricamente (limite di Shockley-Queisser) limitata al 33,7%; ciò perché non si riesce ad estrarre gli elettroni contenuti in una parte rilevante (detta spin-triplet) degli stati eccitati (eccitoni) gene-

rati dall'assorbimento della luce da parte del semiconduttore. Gli studiosi sono comunque sempre alla ricerca di soluzioni per estrarre questa elettricità "latente" e forse i ricercatori dell'Università di Cambridge (Regno Unito) sono sulla strada giusta. Uno dei loro più recenti ritrovati sperimentali è un nanocristallo ibrido a base di pentacene (un semiconduttore organico) e seleniuro di piombo (PbSe - un semiconduttore inorganico) che teoricamente convertirebbe il 100% della luce ricevuta



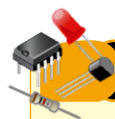
in energia elettrica. Il gruppo che conduce la ricerca sta sviluppando un rivestimento organico che potrebbe essere applicato alle celle solari a base di silicio, quelle cristalline, per intendersi, allo scopo di aumentarne l'efficienza di conversione.

REALIZZAZIONE PRATICA

Bene, descritti gli schemi elettrici e spiegato in che modo funzionano i sensori alla base delle tre breakout board di questo articolo, vediamo come costruire le singole schede; per prima cosa dovete preparare gli stampati di vostro interesse, seguendo le tracce pubblicate nel nostro sito web www.elettronica.in.it. Ottenuto il c.s. disponetevi i componenti iniziando dal sensore o dall'integrato del caso. Ricordiamo che per una buona saldatura è consigliabile operare con un saldatore da 20 o 25 watt a punta fine e filo di lega saldante del diametro di 0,5 mm o meno, stagnando le piazzole interessate e poi ricoprendole di fluxante per agevolare la saldatura dei terminali; conviene altresì servirsi di una lente d'ingrandimento, meglio se di quelle illuminate da tavolo, per verificare che i componenti siano centrati rispetto alle rispettive piazzole e che non vi siano baffi di stagno a creare cortocircuiti. La disposizione e l'orientamento dei componenti sono visibili nei disegni di montaggio e nelle foto dei prototipi che trovate in queste pagine. Per la breakout board contenente l'MCP73831, montate questo integrato aiutandovi con il fluxante per evitare che lo stagno unisca piedini vicini tra

loro (il passo è piccolissimo) e con una lente d'ingrandimento. Completate il circuito con le resistenze, il microdeviatore a slitta, i tre connettori e il LED in SMD (la tacca su di esso contrassegna il catodo). Per quella contenente il sensore si luce, ricordate che il TEMENT6000 va appoggiato sulle piazzole corrispondenti dopo averle stagnate e passate con del fluxante, quindi saldato scaldando lo stagno lateralmente fino a veder affondare il componente; se occorre, bisogna aggiungere stagno fino ad ottenere una saldatura affidabile che si propaga di lato. Dopo il TEMENT6000, montate i resistori, il connettore e il solito LED in SMD. Per la scheda RS485, stagnate l'integrato sulle rispettive piazzole spalmando del fluxante in gel per evitare di mettere in cortocircuito con lo stagno i contatti vicini e alternando la saldatura dei pin dai due lati; poi completate il circuito con il condensatore C1, il connettore, la resistenza e il LED, ricordando che il catodo di quest'ultimo è dal lato contrassegnato dalla tacca. Per tutte e tre le schede, considerate che il connettore SMD a tre vie è opzionale, perché potete prelevare le connessioni anche solo dalle piazzole previste allo scopo e posizionate sul bordo del circuito stampato.

Anzi, se prevedete di inserire le breakout board direttamente sul circuito stampato di un'altra apparecchiatura, vi conviene saldare nelle piazzole un pin trip a passo 2,54 mm: ciò semplificherà il collegamento e l'utilizzo. Tutte le schede si alimentano con tensione stabilizzata di 5 volt, quindi se le collegate a circuiti esistenti dovete accertarvi della disponibilità di un'alimentazione stabilizzata a livello TTL; le correnti richieste sono davvero minime e prelevabili in linea di massima da qualsiasi circuito o pila, ma anche pannello fotovoltaico in miniatura. ■



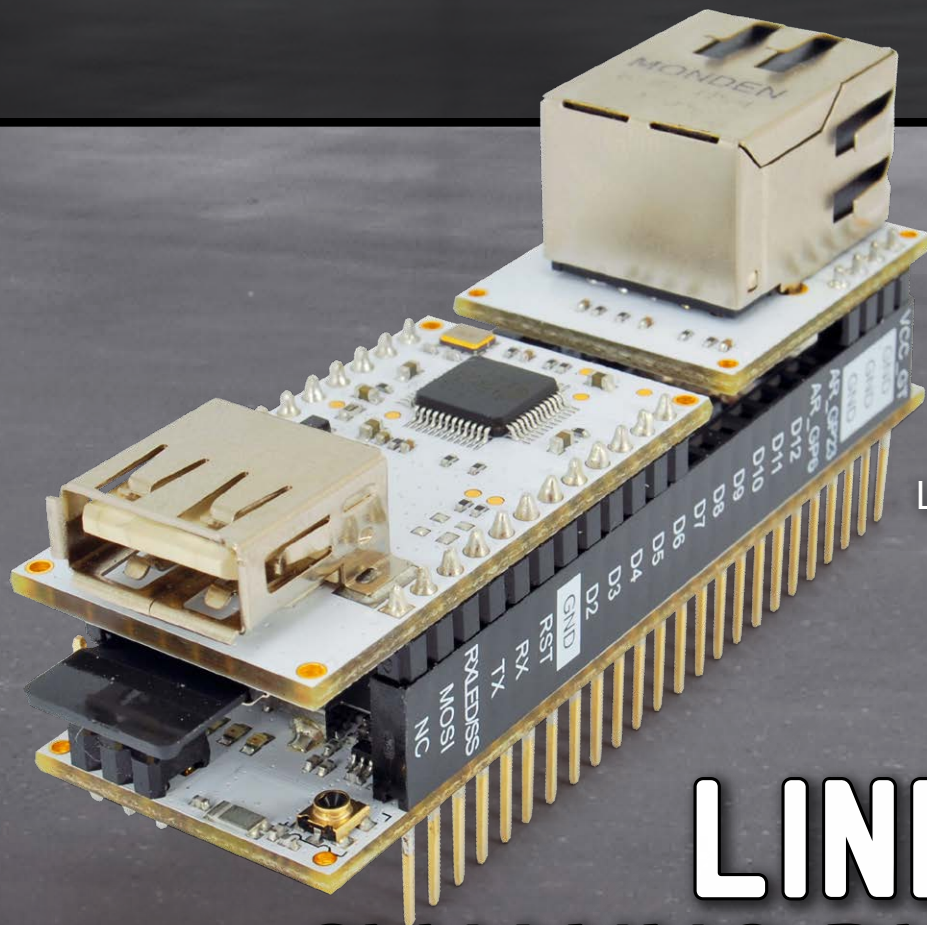
per il MATERIALE

Le schede vengono fornite assemblate e comprese di cavetto.

Possono essere acquistate in Italia presso Futura Elettronica.

- Sensore di luce Euro 11,00 (cod. 7100-BREAKOUT009)
- RS485 Euro 10,00 (cod. 7100-BREAKOUT007)
- Caricabatteria Euro 11,00 (cod. 7100-BREAKOUT004)

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



La società "Dog Hunter" produce una serie di schede in stile Arduino Yún, ovvero con due CPU in comunicazione tra di loro. La novità sta nelle dimensioni e nella modularità della soluzione, totalmente compatibile con l'originale Arduino Yún.

LININO ONE CI HANNO RISTRETTO ARDUINO YÚN

di MARCO MAGAGNIN

A livello macroscopico, nel campo dei sistemi di elaborazione elettronica abbiamo visto nascere i primi sistemi di calcolo come elaboratori mono programma stand-alone, qualcosa di simile, anche se in formato gigante, ai microcontrollori di oggi. Le esigenze di rendere questi giganti in grado di eseguire più elaborazioni in contemporanea e di servire più utenti, ha portato, grazie alla crescita di potenza dei processori hardware, alla loro miniaturizzazione ed allo sviluppo di sistemi operativi sempre più evoluti e di sistemi sempre più compatti, potenti e uniformati dal punto di vista degli standard. I prodotti di questo tipo più vicini alla nostra esperienza sono i microcomputer con a bordo GNU/Linux; Raspberry Pi tra le più diffuse. Sul piano professionale, invece, primeggiano mainframe di ogni dimensione fino ai giganteschi supercomputer usati in ambiente scientifico e militare. Nel frattempo anche i componenti più orientati alla gestione dell'hardware

sono evoluti fino a prendere l'aspetto dei moderni microcontrollori, piccoli, potenti e programmabili. Con un ritorno di fiamma tipico dei processi evolutivi, questi due filoni tecnologici, che hanno proceduto in parallelo, tornano a fondersi in dispositivi che incorporano entrambe le caratteristiche, di gestione dell'hardware e di gestione di molteplici processi in contemporanea. Stiamo parlando dei vari sistemi embedded, composti da una unità di elaborazione multi processo, governata da un sistema operativo, nella maggior parte dei casi GNU/Linux, affiancata da un microcontrollore, reale o simulato, finalizzato ad interfacciare le periferiche e le risorse hardware. Un esempio sul piano amatoriale è la scheda Arduino Yun, ma non possiamo certo non tenere conto di tablet e smartphone, quintessenza dell'evoluzione di questo filone tecnologico. Dispositivi in grado di comunicare con l'utente mediante interfacce grafiche evolute, o persino con la voce, e contem-

Fig. 1

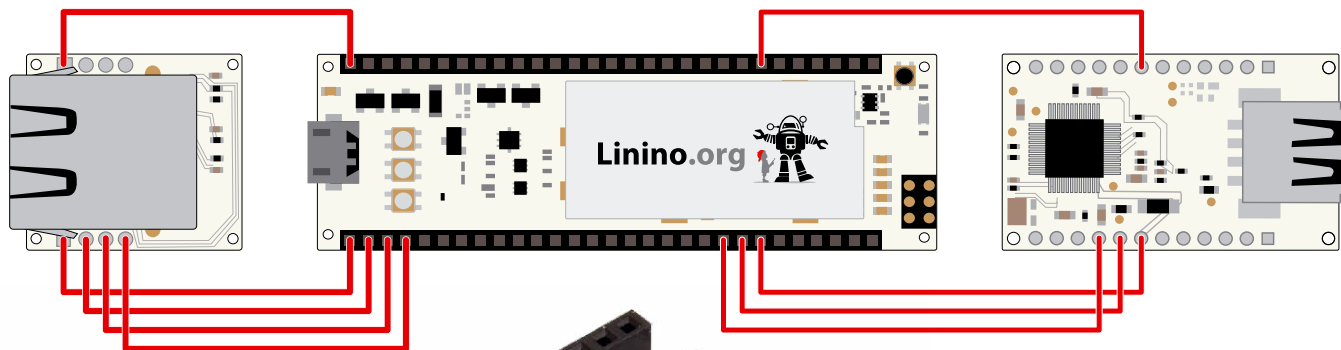
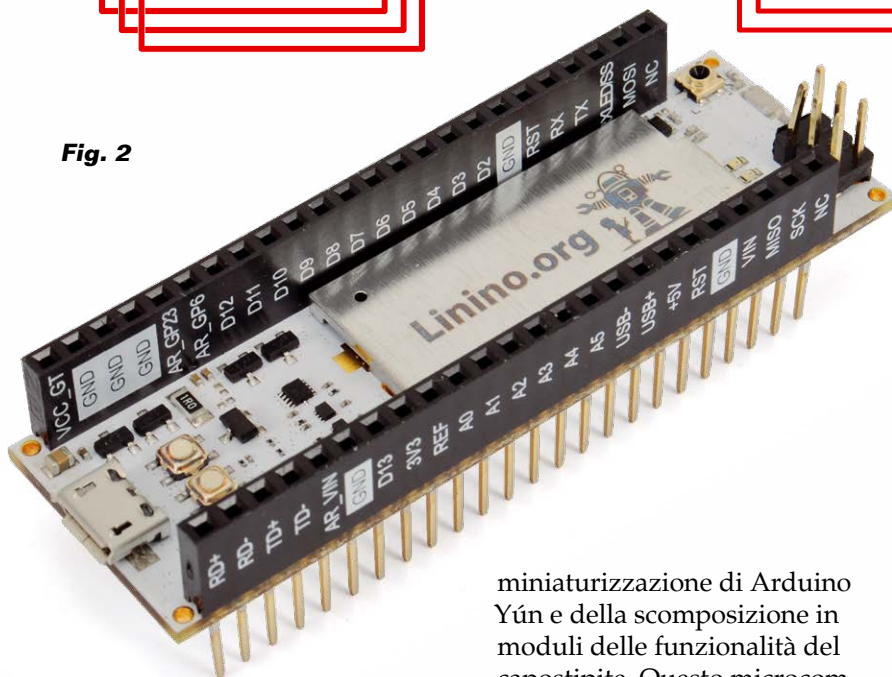


Fig. 2



poraneamente gestire e coordinare dispositivi di connessione alla rete internet, via cavo o WiFi, Bluetooth, GSM, oltre che sensori come accelerometri, GPS, microfoni, videocamere ecc. Sempre nel filone evolutivo generale, passi significativi sono stati fatti nel campo della miniaturizzazione e della modularizzazione di componenti e dispositivi. Dal grande al piccolo, al modulare e, a parità di dimensioni, sempre più potente. Un prodotto di quest'ultimo processo è il micro-computer che vi presentiamo in questo articolo. Si tratta di Linino ONE, un prodotto della Società dog hunter, che ha collaborato alla progettazione ed allo sviluppo di Arduino Yún. Ed infatti Linino ONE è il risultato della

miniaturizzazione di Arduino Yún e della scomposizione in moduli delle funzionalità del capostipite. Questo micro-computer è realizzato attorno agli stessi chip Atheros AR9331 e ATmega 32u4 ma organizzati in un layout molto più compatto. Inoltre i connettori Ethernet ed USB, assieme alla loro circuiteria di controllo, sono stati realizzati in moduli separati, per permettere di comporre il dispositivo più adatto ad ogni tipo di esigenza. In Fig. 1 vediamo lo schema di collegamento tra i due moduli aggiuntivi e la scheda base.

DESCRIZIONE GENERALE

La presenza del supporto nativo per la connettività WiFi e l'adozione del sistema operativo Linino, una distribuzione GNU/Linux derivata da OpenWrt, realizzata appositamente per equipaggiare i router di rete, ne fanno un dispositivo particolarmente adatto per connettere dispositivi fisici alla rete internet. La soluzione complessiva consiste di tre moduli, il modulo principale, con a bordo i chip delle due sezioni, il modulo WiFi e tutti i pin di collegamento, che riproducono quelli di una scheda Arduino, anche se disposti diversamente dal layout "classico". I pin di I/O e di utilizzo generale fanno capo a due connettori "passanti" collocati sui lati lunghi della schedina in formato "stick", visibile in Fig. 2. I connettori hanno terminali a passo 2,54mm, maschi sulla parte superiore della schedina e femmina sulla parte inferiore. La funzione di ciascun pin è serigrafata sui fianchi laterali di ciascun connettore, come visibile nella Fig. 3. Alcuni pin all'inizio ed alla fine dei connet-

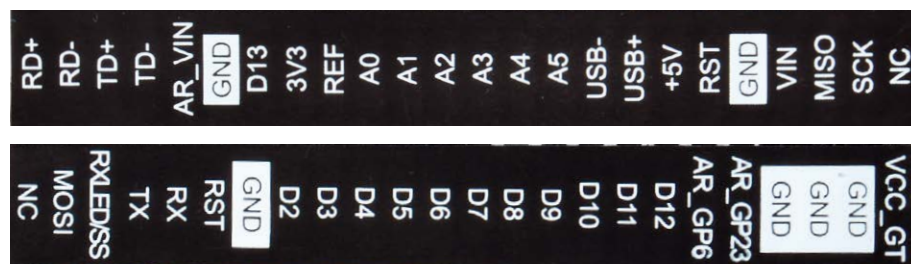


Fig. 3



Fig. 4

tori permettono l'alloggiamento dei moduli aggiuntivi. Allo stato attuale sono disponibili i moduli "dogRJ45" e "dogUSB". Il modulo "dogRJ45" visibile in Fig. 4, è stato progettato per poter dotare la scheda base di uno switch Fast Ethernet in standard IEEE 802.3, in grado di riconoscere la polarità dei cavi di rete e quindi adattarsi all'utilizzo di cavi di rete sia "normali" che "crossati". L'implementazione del modulo supporta il port mirroring, la gestione dei broadcast storm ed il controllo di flusso in full duplex. Anche il modulo "dogUSB", visibile in Fig. 5, è montato su una basetta stampata della stessa larghezza dello "stick" principale, con due file di connettori che vanno a collocarsi su quelli presenti sulla scheda base. Il modulo "dogUSB" porta a bordo un connettore USB totalmente compatibile con le specifiche Hub USB 2.0 e retro compatibile con le specifiche Hub USB 1.1. Supporta lo switching automa-

tico tra la modalità di alimentazione bus-power e self-power. A bordo del modulo "dogUSB" è presente anche un connettore per micro SD Card. Lo avremmo preferito collocato sulla scheda base, per l'uso che intendiamo farne in seguito, ma probabilmente dipende dall'Hub USB e di conseguenza non vi è alternativa possibile. Anche questo modulo è caratterizzato dalla dimensione in larghezza uguale a quella della scheda base e da due file di connettori per il suo collegamento alla scheda base. In Fig. 6 è visibile l'insieme delle tre schede assemblate tra di loro.

CARATTERISTICHE TECNICHE

Dal punto di vista architetturale la scheda Linino ONE è l'unione di un microcontrollore basato sul chip ATmega32u4 (lo stesso di Arduino Leonardo) con un processore Atheros AR9331. Praticamente una scheda simile ad Arduino Yún, direte voi. La risposta è affermativa, ma il layout della soluzione Linino One è molto più compatto e la realizzazione è modulare. Come già anticipato i moduli Ethernet e USB/SDCard sono utilizzabili solo se necessario. Questa rivisitazione in termini di minia-

turizzazione e modularizzazione rafforza la vocazione di aprire le classiche funzionalità Arduino verso il mondo esterno e l'Internet delle cose. Infatti, non a caso la scheda principale mostra in bella vista un modulo WiFi, con tanto di connettore per collegare un'antenna esterna. Sempre sulla scheda principale sono inoltre presenti i pin di I/O del microcontrollore ATmega 32u4: 20 pin di I/O, sette dei quali possono essere usati come uscite PWM, e 12 come ingressi analogici, una porta micro USB, un connettore ICSP, un pulsante di reset. Gli altri due pulsanti fanno capo alla sezione gestita dal processore Atheros AR9331.

Per ciascuna "sezione" della scheda riportiamo le caratteristiche tecniche ed elettriche (Tabella 1).

Una particolare attenzione è richiesta nell'alimentazione della scheda che non prevede un regolatore di tensione e che, di conseguenza richiede tassativamente una tensione di alimenta-

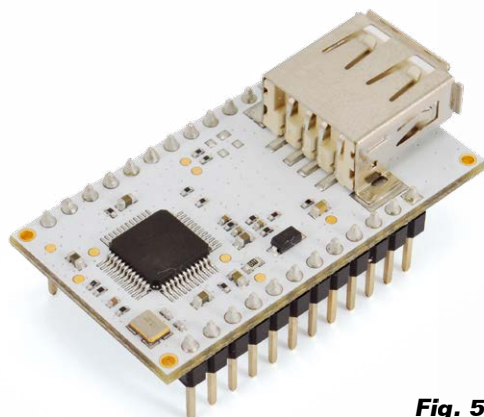


Fig. 5

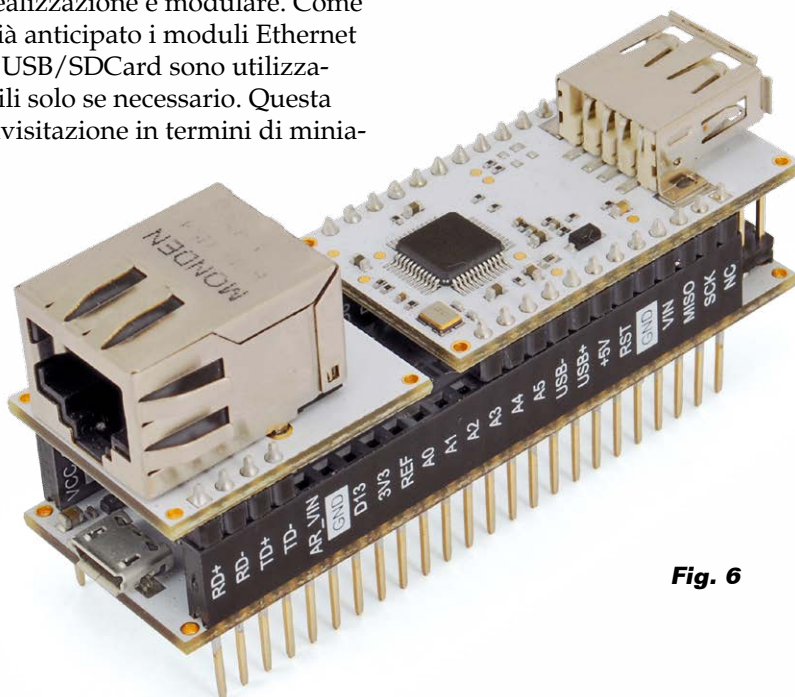


Fig. 6

AVR microcontroller

Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage	5V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (of which 4 KB used by bootloader)
SRAM	2.5 KB
EEPROM	1 KB
Clock Speed	16 MHz

Tabella 1**Linux microprocessor**

Processor	Atheros AR9331
Architecture	MIPS @400MHz
Operating Voltage	3.3V
Ethernet	IEEE 802.3 10/100Mbit/s
WiFi	IEEE 802.11b/g/n
USB Type-A	2.0 Host/Device
Card Reader	Micro-SD only up to 32 GB
RAM	64 MB DDR2
Flash Memory	16 MB

Tabella 2

zione di 5V. Pertanto è necessario controllare l'abitudine di utilizzare gli alimentatori a 9 e 12V. In questo compito siamo aiutati dal connettore in formato micro USB, lo stesso dei telefonini. Gli alimentatori con uscite USB garantiscono di fatto la tensione a 5V. Il rischio rimane nell'alimentare la scheda tramite il pin VIN, in questo caso bisogna porre particolare attenzione alla tensione che si applica. I pin coinvolti nell'alimentazione sono gli stessi di una scheda Arduino Leonardo, anche se con delle differenze nelle specifiche e nell'utilizzo:

- VIN – In una scheda Arduino classica questo pin è direttamente collegato all'alimentatore esterno, prima del regolatore di tensione. Può essere utilizzato sia per prelevare tensione per alimentare gli shield collegati, sia per alimentare direttamente la scheda, prelevando tensione dallo shield. Questo rende possibile alimentare gli shield con ten-

sioni maggiori rispetto ai 5V necessari al processore ATmega 32u4. Tutto questo non è più vero nel caso di Linino One. Siccome non è previsto il regolatore di tensione, su questo pin sono presenti i 5V dell'alimentazione stabilizzata in ingresso. Pertanto non è possibile alimentare da questo pin shield che richiedono tensioni di alimentazione superiori. Anche nel caso si voglia alimentare la scheda utilizzando questo pin, bisogna fare attenzione a non fornire una tensione maggiore di 5V stabilizzati. Differentemente si procurano danni irreparabili alla scheda stessa.

- 5V – Riporta la tensione di alimentazione della scheda. La tensione può provenire dall'ingresso POWER, dal pin VIN o dalla presa USB;
- 3V3 – Uscita a 3.3 volt generata dal regolatore interno della scheda. La corrente massima fornita è di 50 mA;

- GND – Pin di massa;
- IOREF – Sulle schede classiche permette di verificare la tensione di riferimento alla quale operano i pin. Sulla scheda Linino One è fissa a 5V.

MEMORIA

Vediamo la distribuzione dei banchi di memoria sui due processori presenti sulla scheda. Il microcontrollore ATmega 32u4 ha 32 KB di memoria nei quali vengono memorizzati gli sketch (in realtà 4 KB sono occupati dal bootloader). Inoltre sono presenti 2.5 KB di memoria SRAM ed 1 KB di memoria EEPROM, che può essere scritta e letta dallo sketch Arduino utilizzando la libreria EEPROM.

I banchi di memoria non fanno parte del SoC AR9331, ma sono realizzate con chip dedicati collegati esternamente. Dal lato del processore AR9331 la scheda Linino One ha 64 MB di memoria RAM DDR2 e 16 MB di memoria flash. La memoria flash è pre-caricata in fabbrica con la distribuzione GNU/Linux Linino, basata sulla distribuzione OpenWRT. Questa immagine permette di riportare alla configurazione iniziale il sistema operativo Linino utilizzando il pulsante "WLAN RST", come descritto in seguito. È una funzionalità molto utile che permette di sperimentare l'utilizzo di Linino, caricando pacchetti aggiuntivi e modificando pacchetti e configurazioni, con la tranquillità di poter sempre "tornare indietro" in caso di situazioni "bloccanti". È possibile, anzi caldamente raccomandato, l'utilizzo di una micro SDCard esterna per espandere la memoria di massa assegnata a Linino, assai limitata. Potremo anche utilizzare memorie esterne purché compatibili con lo standard USB 2.0, come chiavette e dischi fissi.

Nel seguito dell'articolo vedremo come superare il problema del limite di memoria dedicata al file system di Linino.

PULSANTI DI RESET

Abbiamo accennato all'utilizzo del pulsante "WLAN RST" per riportare la configurazione di Linino alla versione originale caricata in fabbrica. Ne approfittiamo per descrivere il funzionamento dei pulsanti di reset. Ne sono presenti ben tre, collocati dietro la presa micro USB di alimentazione. Una nota al riguardo: con il modulo Ethernet installato è pressoché impossibile utilizzare i pulsanti. In caso di "bisogno" è necessario spegnere la scheda, rimuovere il modulo Ethernet, riaccendere la scheda ed operare sui pulsanti desiderati. Nella Fig. 7 è visibile il posizionamento sulla scheda e la descrizione dei tre pulsanti. Descriviamo le funzionalità associate a ciascuno:

- 32U4 Reset button - Questo pulsante permette il reset del microcontrollore ATmega 32U4 in modo da far ripartire lo sketch dall'inizio;
- Linino Reboot button - La pressione del pulsante "Linino Reboot button" permette di eseguire il reboot del sistema operativo Linino. Chiaramente tutti i dati in memoria saranno persi e tutti i processi attivi al

momento saranno terminati;

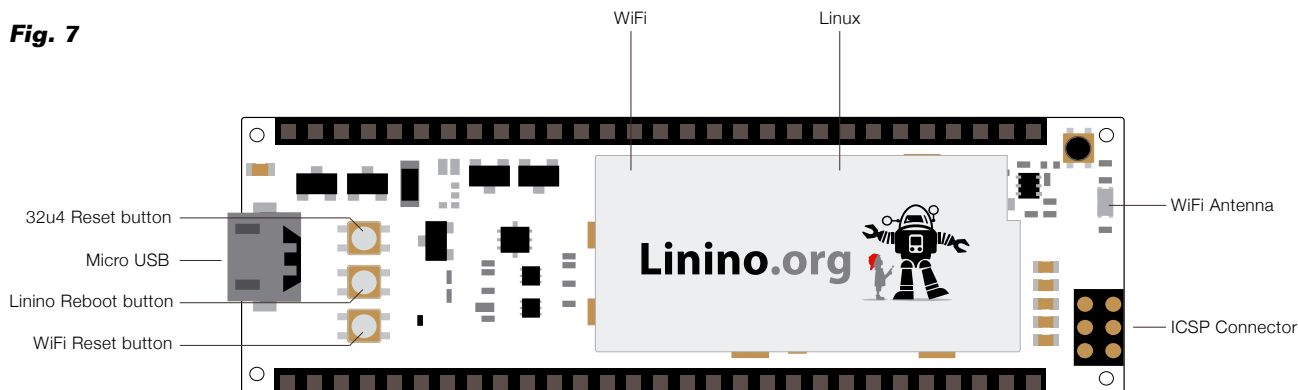
- WiFi Reset button - Il pulsante "WiFi Reset button" ha una doppia funzione. La pressione normale permette di riportare il modulo WiFi alla configurazione di fabbrica, cioè configurato come Access Point con IP statico 192.168.240.1 con nome SSID "Linino One-XXXXXXXXXXXX", dove le dodici "X" rappresentano il MAC address del modulo WiFi della scheda. Per resettare il modulo WiFi è necessario tenere premuto il pulsante per almeno cinque secondi ... ma non molti di più. Il reset del modulo WiFi comporta anche il reboot di Linino. Abbiamo detto di non tenere premuto il pulsante molto oltre i cinque secondi in quanto il pulsante ha una doppia funzione. Tenuto premuto per più di 30 secondi provoca il ripristino del sistema operativo Linino con la versione originale, ovvero quella caricata in fabbrica. Una buona opzione per permettere di recuperare eventuali operazioni maldestre. Questo lascia la libertà di sperimentare senza il timore di trovarsi in situazioni senza via d'uscita. Più semplice di così. Chiaramente il prezzo da pagare è di perdere tutte le configurazioni e gli eventuali aggiornamenti eseguiti su Linino.

INGRESSI E USCITE "ARDUINO STYLE"

Nella scheda Linino ONE non è previsto accedere direttamente ai pin di I/O del processore Atheros AR9331. La scheda espone invece i pin di I/O presenti sui connettori tipici delle schede Arduino Leonardo. Ciascuno dei venti pin di I/O digitali, collegati al microcontrollore ATmega 32U4 può essere utilizzato sia come ingresso che come uscita utilizzando le istruzioni pinMode(), digitalWrite() e digitalRead(). Tutti i pin lavorano a 5V.

Ogni pin può fornire in uscita o sopportare in ingresso una corrente massima di 40mA. Ciascun pin può essere configurato con una resistenza interna di pull-up di 20-50 kOhm, disabilitata come configurazione predefinita. La configurazione dei pin è riportata in serigrafia sui lati dei connettori che li espongono come visibile nella foto di Fig. 3. Potete notare che la "geometria" dei connettori non corrisponde a quella della classiche schede simil Arduino, pertanto eventuali shield dovranno essere adattati. In compenso si guadagna molto in compattezza e riduzione degli ingombri. Come nel caso delle schede Arduino classiche, molti pin sono multiplexati e presentano funzioni

Fig. 7



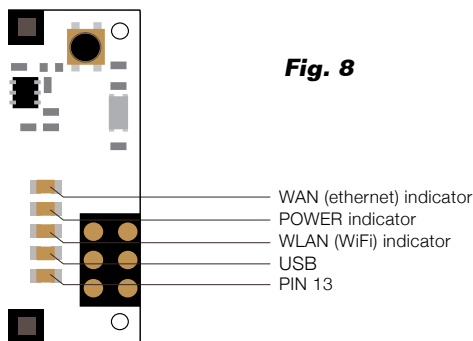


Fig. 8

multiple che possono essere usati in alternativa:

- **Serial: pin 0 (RX) e 1 (TX).** Utilizzati nella comunicazione seriale rispettivamente come pin di ricezione (RX) e trasmissione (TX). Sulla scheda Linino One la comunicazione seriale attestata ai pin 0 e 1 è utilizzata per comunicare con il processore AR9331, in quanto le uscite seriali hardware del microcontrollore ATmega 32U4 e del processore AR9331 sono collegate insieme e costituiscono il ponte di comunicazione tra i due processori. Come normale per i sistemi GNU/Linux la porta seriale è dedicata alla console a linea di comando per l'accesso al sistema. Per utilizzare una comunicazione seriale con dispositivi esterni è necessario utilizzare la classe Serial1 e, se necessario la libreria SoftSerial;
- **TWI: pin2 (SDA) e 3 (SCL).** Permettono di gestire la comunicazione i2C (TWI) utilizzando la libreria Wire;
- **Interrupt esterni: pin 3 (interrupt 0), pin 2 (interrupt 1), pin 0 (interrupt 2), pin 1 (interrupt 3) e pin 7 (interrupt 4).** Questi pin possono essere configurati per intercettare un evento esterno. L'evento può essere innescato da un livello basso del pin, da un fronte di salita, da un fronte di discesa o da una variazione di valore. Maggiori dettagli disponibili

nelle istruzioni sull'utilizzo della funzione attachInterrupt(). In realtà non è raccomandabile utilizzare i pin 0 e 1 per riconoscere eventi esterni, in quanto questi pin sono collegati alla porta seriale collegata al processore GNU/Linux.

Il pin 7 è anch'esso collegato al processore AR9331 ed in futuro è previsto il suo utilizzo come segnale di handshake. È bene porre attenzione nell'utilizzare questi pin per intercettare eventi esterni in modo da evitare possibili conflitti;

- **PWM: pin 3, 5, 6, 9, 10, 11, e 13.** Questi pin possono essere utilizzati come PWM con 8-bit di risoluzione in output utilizzando la funzione analogWrite();
- **SPI (MISO, MOSI, SCK).** Questi pin supportano la comunicazione SPI utilizzando la libreria omonima. I pin SPI sono collegati anche ai pin gpio del processore AR9331, sul software del quale è parimenti implementata l'interfaccia SPI. Questo significa che il microcontrollore ATmega32u4 ed il processore AR9331 possono comunicare tra loro utilizzando anche il protocollo SPI;
- **Analog Inputs: Pin A0 - A5, A6 - A11 (corrispondono ai pin digitali 4, 6, 8, 9, 10, e 12).** La scheda Linino One ha 12 ingressi analogici, che comunque possono essere utilizzati anche come ingressi/uscite digitali. I pin da A0 ad A5 sono indicati nella serigrafia; gli ingressi da A6 ad A11 corrispondono rispettivamente ai pin digitali 4, 6, 8, 9, 10, e 12. Ciascun pin permette una risoluzione di 10 bit (1024 valori differenti). In configurazione predefinita gli ingressi analogici utilizzano il riferimento di tensione di 5V (1024 intervalli tra 0 e 5V). È

comunque possibile modificare il livello superiore dell'intervallo di tensione oggetto di misura utilizzando il pin AREF e la funzione analogReference();

- **AREF.** A questo pin è possibile connettere una sorgente di tensione da utilizzare come riferimento per gli ingressi analogici. Per la configurazione si utilizza la funzione analogReference().

Una nota sui LED presenti sulla scheda e visibili in Fig. 8:

- **WAN** - Indica la presenza di una connessione di rete Ethernet attiva;
- **POWER** - Indica la presenza di alimentazione;
- **WLAN (WiFi)** - WiFi attivo
- **USB** - indica attività sul connettore USB;
- **Pin 13** - Riporta lo stato del pin 13 – a scheda nuova lampeggia grazie al programma Blink caricato all'origine.

Ciò che non appare alla vista, e che distingue la scheda Linino One così come la scheda Yún è l'anima "software" interna alla scheda, che permette ad uno sketch Arduino di comunicare e cooperare con programmi e API disponibili nella sezione GNU/Linux, realizzando un dispositivo dalle notevoli potenzialità di comunicazione sul network, con la facilità di utilizzo tipica di Arduino. Nella sezione GNU/Linux si possono richiamare sia comandi di shell come cURL, per comunicare sul network, sia veri e propri programmi personalizzati che possiamo scrivere come script di shell o utilizzando il linguaggio python.

ARCHITETTURA INTERNA

In Fig. 9 è visibile lo schema dell'architettura generale della scheda. Sono chiaramente

visibili le due sezioni: quella in stile "Arduino" e quella controllata da "Linino" con le periferiche assegnate a ciascuna, ed il "ponte" di collegamento indicato in figura come "Bridge". Questa feature è quella che permette ai due ambienti di operare congiuntamente e costituire una innovativa soluzione Linux embedded in grado di portare nel mondo esterno tutte le funzionalità del microcontrollore ATmega 32u4, finora confinate al solo ambito locale. A dir il vero nel passato sono state presentate soluzioni di connettività basate su shield Ethernet, WiFi, Bluetooth, ecc. ma tutte dovevano fare i conti con la vocazione "microcontrollore" dei dispositivi e con le limitate risorse di memoria a disposizione.

Era pressoché impossibile gestire diversi utenti e pagine HTML complesse, per non parlare della gestione delle concorrenze tra accessi web e gestione di sensori e periferiche di I/O. Con questa architettura tutto è superato e ciascuna sezione si occupa al meglio di ciascuna funzionalità. La disponibilità di librerie e ambienti di sviluppo adatti a gestire il "ponte" di comunicazione integrati nell'IDE e sviluppati nella logica di semplificazione propria della "filosofia" Arduino fanno il resto, dando la possibilità di sviluppare soluzioni, anche complesse, con la consueta semplicità e linearità.

Soffermiamoci un momento sulla soluzione "Bridge". I processori ATmega 32u4 e Atheros AR9331 sono collegati mediante porta seriale. Dal lato Arduino la libreria "Bridge" permette di realizzare il "ponte" verso l'ambiente Linino, utilizzando la porta seriale Serial1, che non può essere utilizzata per altri scopi. Semplificando molto, la libreria

"Bridge" invia sulla porta seriale dei "comandi" di "console" a Linino. I "comandi" permettono di attivare dal lato Linino i processi in grado di fare da ponte con i servizi e le applicazioni "GNU/Linux". Dal lato Linino la seriale è assegnata, al momento del boot all'interfaccia CLI (Command Line Interface). In questo modo tutte le stringhe di comandi inviati dalla libreria Bridge sono interpretati da "Linino" come comandi di "shell", che attivano programmi in python che si incaricano di realizzare le diverse modalità di interfaccia disponibili. In questo modo è possibile instaurare la comunicazione tra la sezione Arduino e la sezione Linino simulando una emulazione di terminale remoto (Console), come shell per la gestione di comandi e processi remoti:

- come ponte per leggere e scrivere file sulla SDCard di Linino dagli sketch Arduino;
- come ponte di comunicazione di URL HTTP per gestire dallo sketch richieste passate dal web server uHTTPd di Linino (micro Hypertext Transfer Protocol Daemon);
- in diverse altre modalità.

Sempre dal lato Linino sono

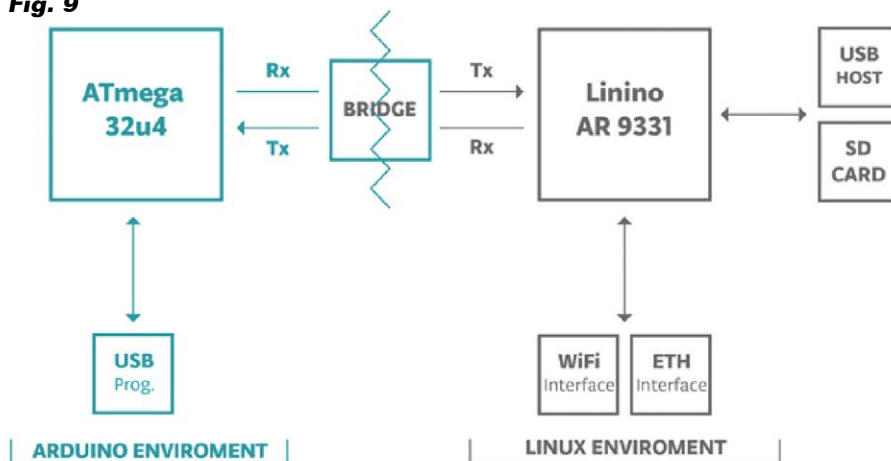
disponibili diversi strumenti per utilizzare applicativamente questa tipologia di comunicazione. Citiamo il framework REST (Representational state transfer) che permette di interfacciare i pin di I/O lato Arduino mediante URL HTTP.

Questa soluzione permette agli sketch Arduino di comunicare con le applicazioni in esecuzione nel sistema Linino e viceversa. La ricetta di hardware e software, ben integrati ed armonizzati tra loro ne fanno una piattaforma semplice da utilizzare per realizzare sistemi amatoriali e didattici di automazione domestica, controllo remoto e di robotica, oltre a poter essere utilizzata anche come Access Point o router per una soluzione integrata di automazione domotica, o di qualunque piccolo ambiente da tenere controllato a distanza.

COMUNICAZIONE

Linino One offre molteplici modalità di comunicare con altri dispositivi di varia natura, come altri microcontrollori, moduli embedded e PC. Linino One può controllare direttamente la porta USB, allo stesso modo della scheda Leonardo, e utilizzando le apposite librerie, è possibile

Fig. 9



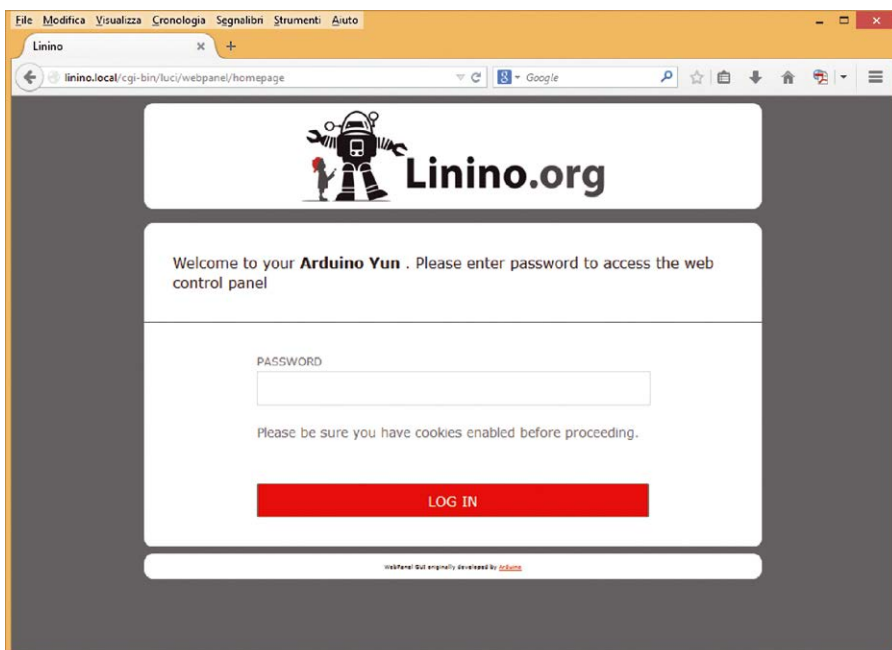


Fig. 10

unità di memoria aggiuntive, tastiere e webcam. Per utilizzare questi dispositivi e per altri scopi è possibile installare pacchetti software aggiuntivi utilizzando il gestore di pacchetti opkg (Open Package manager).

ACCENDIAMO LININO ONE

Dunque, colleghiamo un cavo di rete alla presa Ethernet di Linino One, ovviamente dopo aver installato il modulo “dogRJ45” nel suo alloggiamento, e diamo alimentazione collegando un alimentatore a 5V con presa micro USB (quella per la ricarica dei te-

interfaciare la scheda ad un computer simulando una tastiera od un mouse. Ovviamente può essere utilizzata anche in modalità di emulazione seriale via USB CDC (Communications Device Class). In questa modalità la scheda Linino One appare al dispositivo collegato come una porta COM virtuale.

Linino One può comunicare in modalità USB 2.0 full speed, utilizzando driver USB COM standard. La libreria Software Serial permette di utilizzare ciascuna coppia di pin digitali per la comunicazione seriale, ad esclusione, ovviamente dei pin 0 e 1. Il controllore 32U4 supporta anche la comunicazione su bus I2C / TWI ed SPI. Per queste modalità di comunicazione si utilizzano le librerie standard Wire ed SPI. Le interfacce Ethernet e WiFi sono esposte direttamente dal processore AR9331. Per configurare queste interfacce si utilizza il software di configurazione network incluso nell'applicazione di configurazione web LuCI (vedi libro già citato). L'interfaccia USB può anche essere utilizzata per collegare device esterni da utilizzare con Linino, come

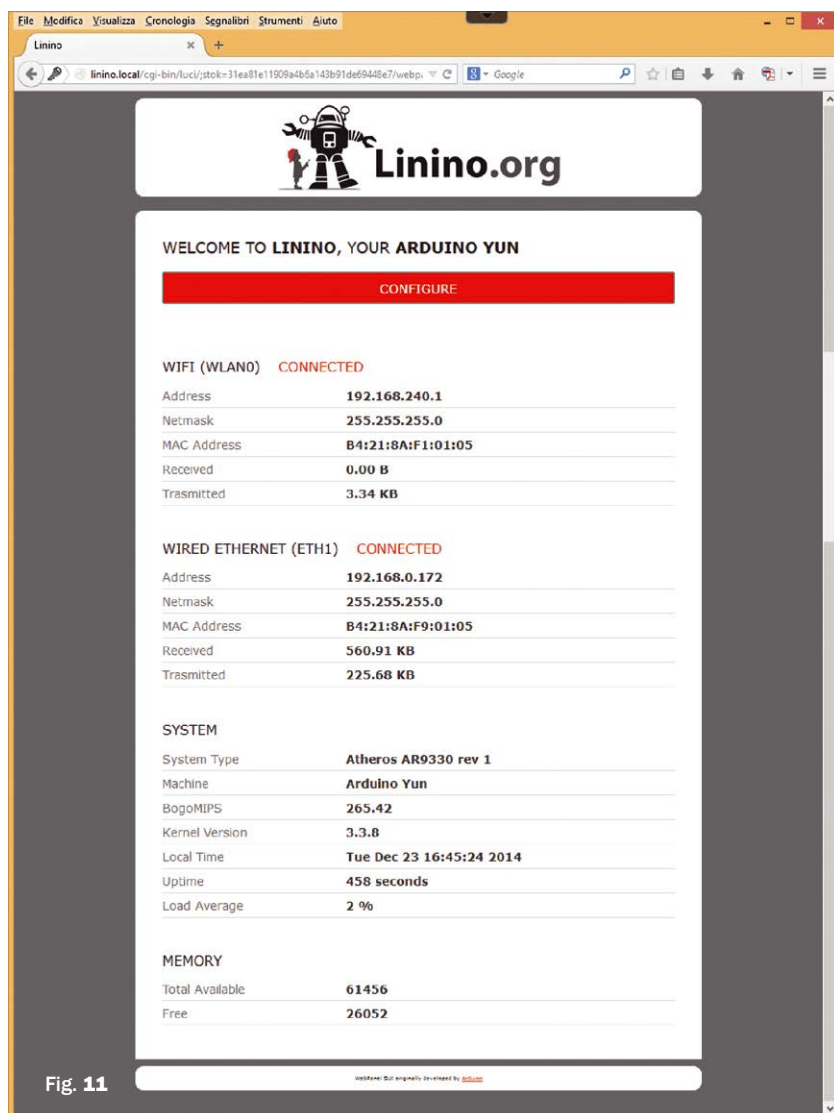


Fig. 11

lefonini) al connettore micro USB al centro della scheda. Il LED ON si accende di un bel verde e poco dopo il LED WAN diventa giallo. Apriamo un browser e digitiamo l'indirizzo:

<http://linino.local>

Otteniamo la pagina visibile in **Fig. 10**.

Diamo la password "doghunter" (è la password dell'utente "root" predefinito) e vediamo cosa succede. Otteniamo la pagina visibile in **Fig. 11** che ci mostra la configurazione delle interfacce di rete attive. Già questa pagina ci dice che le interfacce di rete sono preconfigurate ed un web server è attivo con un'applicazione già installata per configurarne il funzionamento lato GNU/Linux.

A proposito, avete notato che per il primo accesso abbiamo usato un nome (linino.local) e non un indirizzo IP?. Questo significa che non è necessario dare la caccia all'indirizzo IP assegnato a Linino One dal server DHCP della nostra rete, ma possiamo accedere alla scheda direttamente con il suo nome host, in quanto la distribuzione GNU/Linux di Linino One, orientata alla connettività, è dotata di un server DNS interno. Sempre nella **Fig. 11** vediamo anche qual'è l'indirizzo IP assegnato alla scheda lato WAN e la configurazione della sezione WiFi, già attiva e preconfigurata come access point con un IP statico. Questo significa che possiamo collegarci a Linino One anche in WiFi, semplicemente connettendoci alla rete con SSID "Linino-<mac-address-della-scheda>". Niente monitor, niente tastiera, niente mouse, volendo niente cavo di rete. Niente male per facilitare la vita a chi è alle prime armi. Continuiamo a curiosare. Premiamo il grosso pulsante

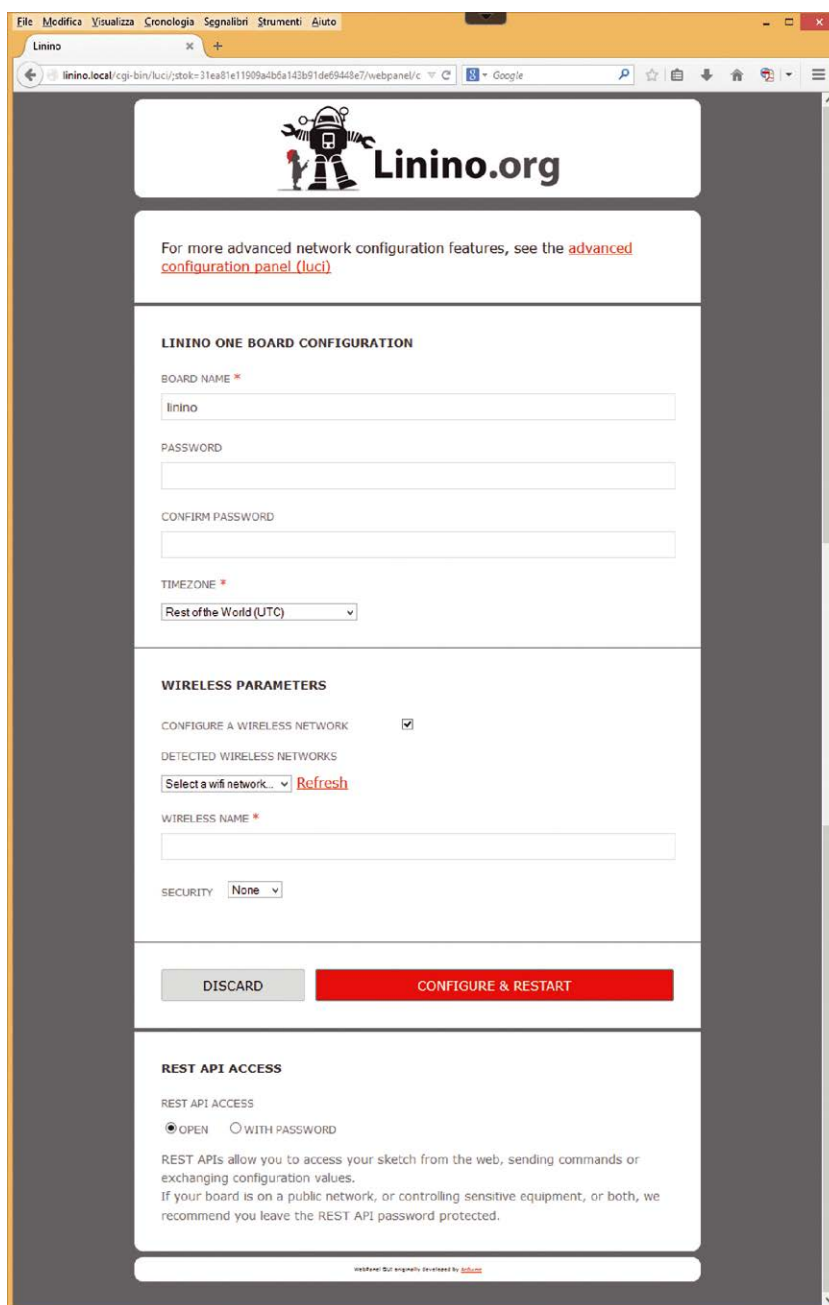


Fig. 12

"Configure" in alto a destra nella pagina ed otteniamo un'ulteriore pagina di configurazione (**Fig. 12**) con un ulteriore link che attira la nostra attenzione, in alto a destra. Prima però ne approfittiamo per configurare la Time Zone corretta (Europe/Rome), e selezionare il check box "Open" nella sezione "Rest API Access"; ci verrà utile per gli utilizzi futuri. Bene, clicchiamo sul link "advanced configuration panel" ed entriamo

in un vero e proprio ambiente di amministrazione, visibile in **Fig. 13**. Dalle pagine di menù, in alto, possiamo accedere alle diverse sezioni di amministrazione del sistema Linino. Se, per curiosità, andiamo alla sezione "Network" e "Wifi", all'interno della pagina clicchiamo sul pulsante "Edit" e scorriamo la pagina un po' verso il basso, troviamo la sezione che ci permette di configurare il modu-

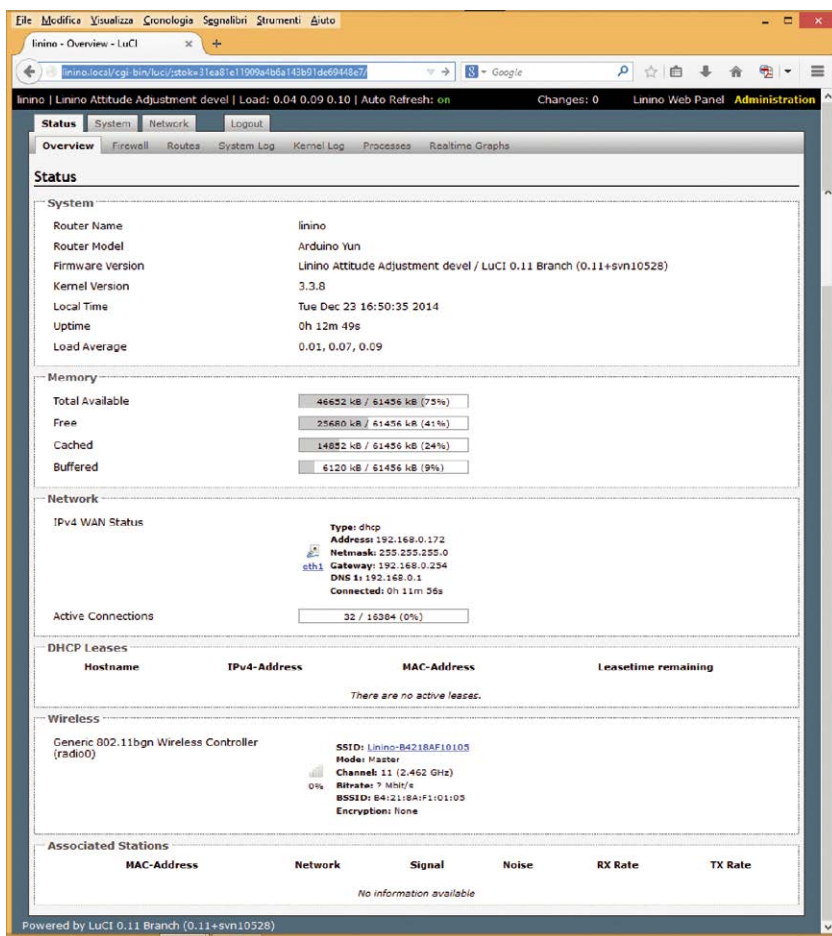


Fig. 13

la pagina in Fig. 11), con l'utente "root" e password "doghunter". Otteniamo i risultati visibili in Fig. 15 e Fig. 16, che ci dimostrano che anche il server SSH è attivo e funzionante "out of the box".

PROGRAMMAZIONE

Linino One si programma utilizzando l'IDE di Arduino in quanto il chip 32u4 viene precaricato con un bootloader in grado di colloquiare direttamente con l'IDE stesso, facilitando il compito di programmazione, caratteristica comune delle schede della famiglia Arduino. Si può programmare direttamente il chip utilizzando un apposito programmatore collegato al connettore ICSP (In-Circuit Serial Programming). Abbiamo già anticipato che possiamo scaricare l'IDE di sviluppo per Arduino YÚN dal sito di

lo WiFi della scheda nelle diverse modalità (Access Point, Client, Ad Hoc, ecc). Le altre sezioni di configurazione ci permettono di impostare regole di sicurezza accessi (Firewall), regole di Routing, DHCP e DNS (Fig. 14).

In definitiva, ci troviamo di fronte ad un vero e proprio dispositivo di connettività che ci permetterà di risolvere una vasta gamma di applicazioni, di esigenze di elaborazione e di configurazione di rete. Un'ultima curiosità, proviamo a connetterci alla scheda utilizzando i client SSH Putty e WinSCP (per chi volesse saperne di più su tutto quanto detto finora, rimandiamo agli articoli introduttivi su Raspberry Pi ed al libro "Il mio primo Linux embedded"). Ci connettiamo con il nome "host" o l'indirizzo assegnato dal DHCP della nostra rete (lo vediamo nel-

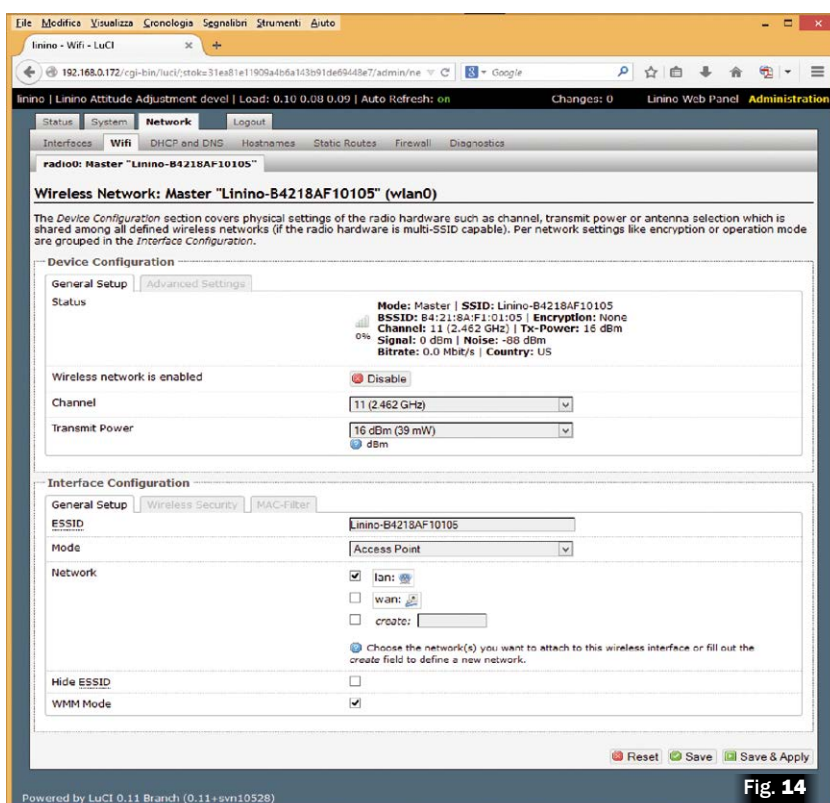


Fig. 14



Arduino. È necessario scaricare il file dall'indirizzo <http://arduino.cc/download.php?f=/arduino-1.5.8-windows.zip> o versione successiva, se esiste. Fate attenzione, la versione 1.0.6 non è adatta a compilare ed eseguire l'upload degli sketch per Linino One (o l'equivalente Arduino YÚN). Per installare il pacchetto dell'IDE è sufficiente scompattare il file in una cartella opportuna.

Creiamo un collegamento del programma arduino.exe sul desktop e clicchiamoci sopra due volte. Si aprirà la consueta interfaccia dell'IDE con alcune novità. In Fig. 17 vediamo la configurazione della porta di comunicazione tra IDE e scheda Linino One, in realtà ci viene presentato l'indirizzo IP della scheda. Lo selezioniamo e attiviamo il canale di comunicazione, senza necessità di collegare cavi USB e installare driver. Teniamo presente che questo metodo funziona, ovviamente, anche se la scheda è collegata in modalità wireless. Anche questo rappresenta una grossa innovazione rispetto al passato, infatti in questo modo possiamo programmare e configurare Linino One anche se la scheda si trova installata in un luogo recondito o montata sopra un dispositivo mobile, come, per esempio, un robot. Fatta la scelta della porta di comunicazione

non ci resta che configurare il tipo di dispositivo, nel nostro caso "Arduino YÚN", in "Strumenti">"Board". Il resto corrisponde a ciò che abbiamo sempre fatto con le altre schede Arduino. Potrebbe capitare che, con tutto correttamente acceso e funzionante, non si veda la scheda nell'elenco della funzione "Port" del menu strumenti all'interno dell'IDE. In questo caso sapere che la connessione avviene utilizzando la porta 5353 UDP della connessione TCP/IP ci può aiutare. La causa del malfunzionamento può essere un firewall, per esempio quello dell'antivirus, che blocca l'utilizzo della porta, oppure una configurazione degli indirizzi di rete che non permette al PC di vedere la scheda.

SPOSTIAMO IL FILE SYSTEM SULLA SD CARD

Terminiamo questa presentazione della scheda Linino One con un'applicazione pratica di utilizzo generale. Partiamo dal problema. Sulla scheda Linino One il sistema operativo Linino è ospitato nella memoria flash di 16 MB, dei quali ne occupa circa 9. Questa situazione lascia poco spazio per l'installazione di nuovi pacchetti e applicazioni. Come già previsto per Arduino YÚN, è possibile installare una SDCard dove configurare le cartelle dati

Fig. 15

da utilizzare per le pagine web e per altri dati aggiuntivi. In questo esercizio trasferiamo completamente la partizione di root del file system di Linino sulla scheda SDCard. Dopo questa operazione tutti gli aggiornamenti e le nuove installazioni di pacchetti verranno eseguite sulla SDCard. Nelle normali distribuzioni GNU/Linux l'operazione che permette di posizionare il file system di root su una partizione diversa dall'originale richiede una certa conoscenza di amministrazione del sistema operativo. In questo specifico caso possiamo approfittare di un script per l'automazione di questa operazione disponibile sul sito [arduino.cc](http://arduino.cc/en/Tutorial/ExpandingYunDiskSpace), all'indirizzo <http://arduino.cc/en/Tutorial/ExpandingYunDiskSpace>, in quanto originariamente realizzato per Arduino YÚN. Lo script è fornito in forma di sketch da caricare sul micro controllo-

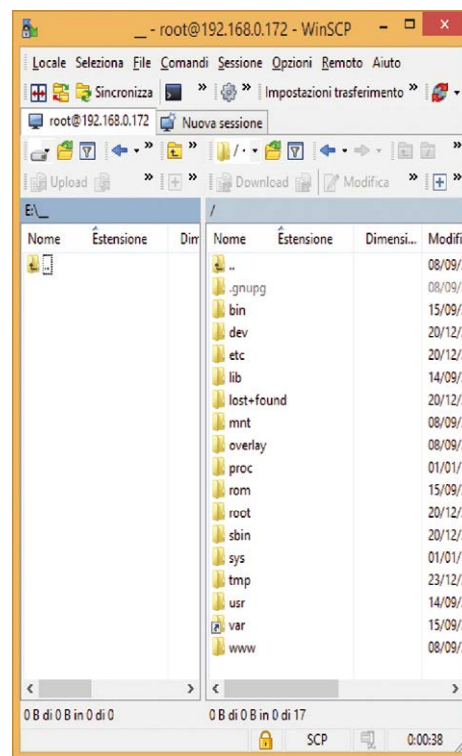


Fig. 16

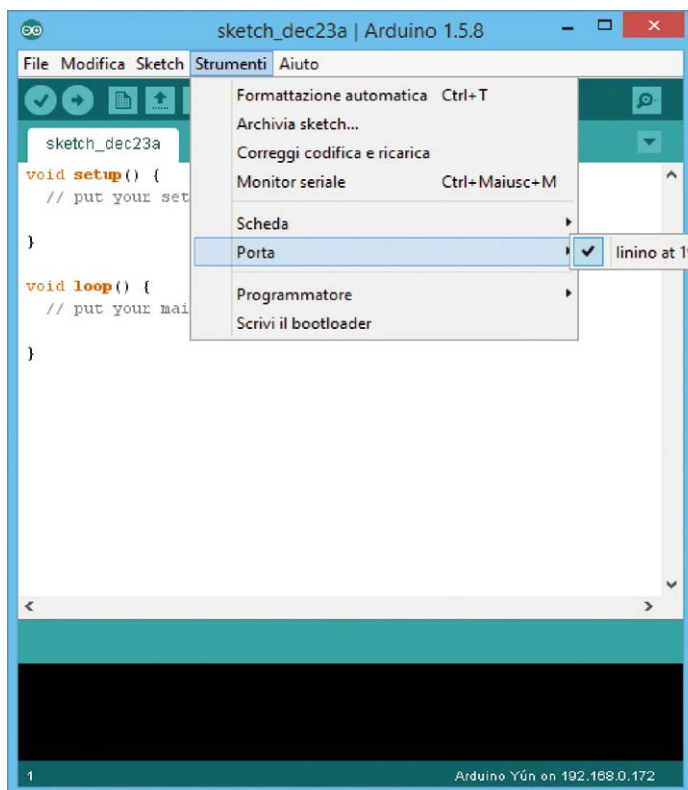


Fig. 17

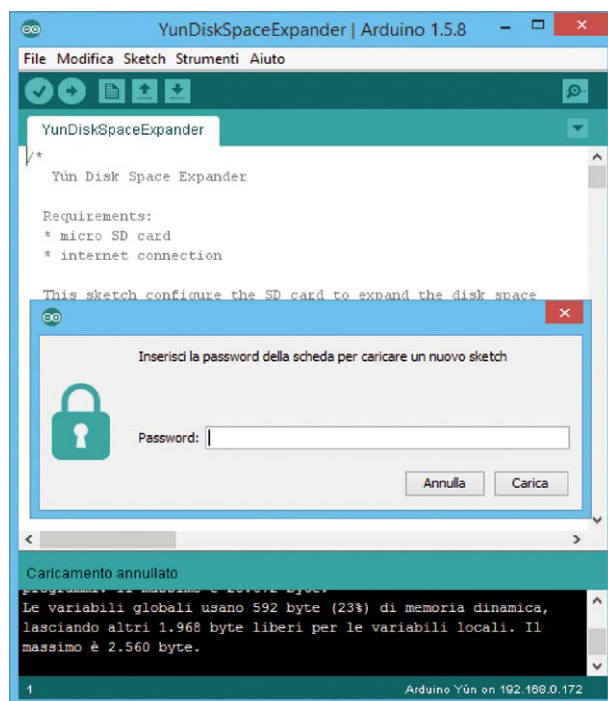


Fig. 18

re 32u4 della scheda Linino ed utilizza la libreria Bridge per configurare la memoria flash e la SDCard. Possiamo scaricare lo sketch dall'indirizzo <http://arduino.cc/en/uploads/Tutorial/YunDiskSpaceExpander.zip>. Per ottenere il risultato basta scompattare lo sketch in una cartella con lo stesso nome del file ".ino", inserire la SDCard nell'apposito slot, caricare lo sketch mediante l'IDE di Arduino (Fig. 18), aprire il Serial Monitor e seguire le indicazioni fornite dallo sketch stesso. Lo sketch divide lo spazio sulla SDCard in due partizioni. La prima partizione viene formattata in formato FAT32 e verrà utilizzata per la parte dati. La seconda partizione verrà formattata in modalità "ext4" e su questa verrà "copiato" il file system di "root". Alla fine del processo otterremo una scheda Linino con le funzionalità invariate ma con molto più "spazio" per i dati e le applicazioni. Per eseguire il processo, togliete l'alimentazione alla

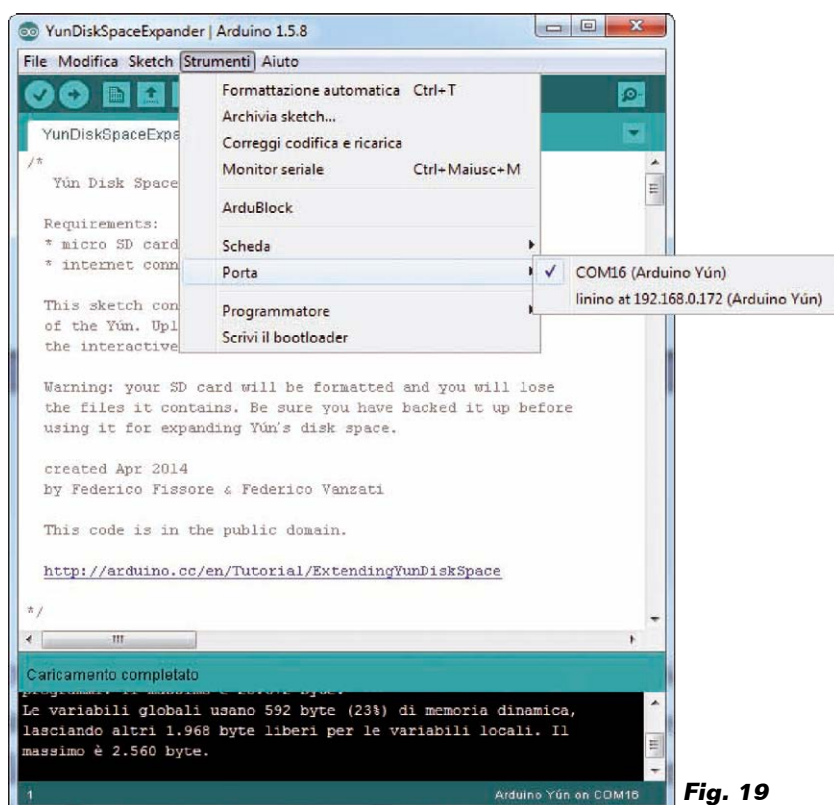


Fig. 19

scheda Linino One, montate sulla scheda principale i due moduli aggiuntivi "dogRJ45" e "dogUSB". Inserite una SDCard nell'apposito slot (noi abbiamo scelto un modello da 8 GB). Collegate la scheda alla rete locale con un cavo di rete in quanto il processo di trasferimento della partizione di root richiede l'installazione di pacchetti aggiuntivi. Poi collegate la porta microUsb ad una porta USB del vostro PC. In questo modo la scheda verrà vista come un device seriale virtuale e non attraverso la connessione di rete. Accendete il tutto, scaricate il file compresso dall'indirizzo citato in precedenza, scompattatelo e, dopo aver aperto l'IDE di Arduino, copiatelo all'interno. Salvatelo con un nome, verificate che nel menu "**Strumenti>>Board**" la scheda sia impostata come "Arduino Yún" e che la porta sia impostata come porta COMx, come visibile in Fig. 19. Ora possiamo compilare ed eseguire l'upload dello sketch. Aprite il Monitor Seriale, selezionate "New Line" nel menu a tendina ed impostate la velocità a 115200 baud. Appena aperto il Monitor Seriale verrete avvertiti di verificare che non vi siano collegati sulla scheda Linino One memorie di massa diverse dalla SDCard. Confermate con "yes" e proseguite con il tasto "Enter" (Fig. 20). Il processo prosegue con l'installazione del software necessario a partizionare e formattare la SDCard. Prima viene richiesto di verificare che Linino One sia connesso alla rete internet, per poter scaricare il software necessario. Verificate, rispondete "yes" e confermate con il tasto enter (Fig. 21). Il passo successivo consiste nella partizione e formattazione della SDCard. Ancora una volta rispondete con "yes" e confermate con "Enter" (Fig. 22). Viene richiesto

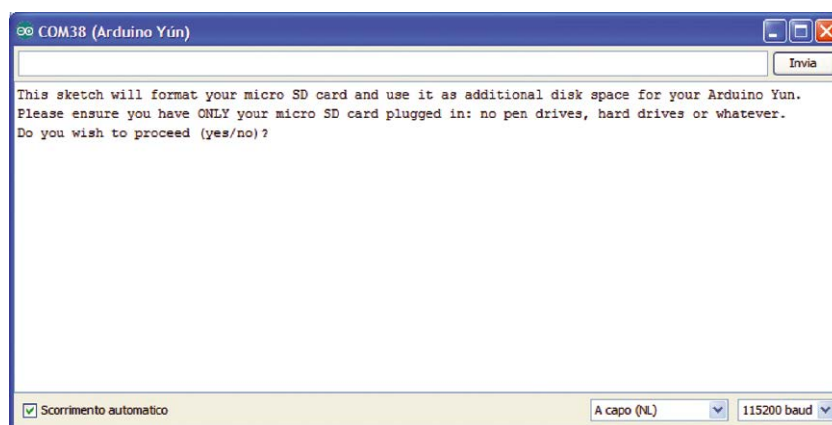


Fig. 20

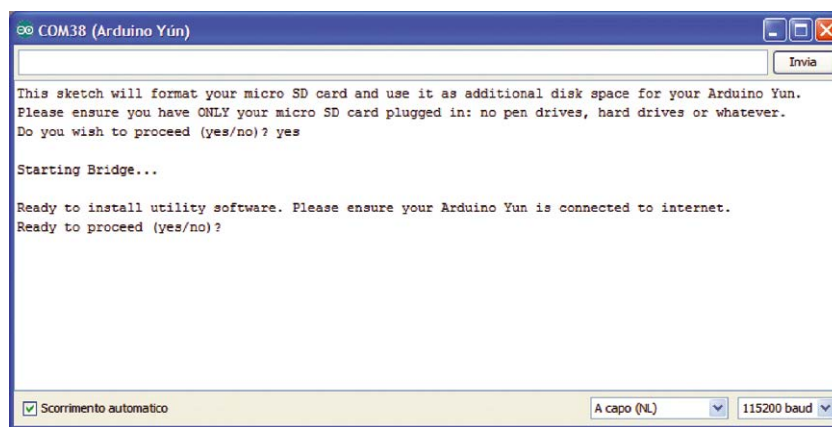


Fig. 21

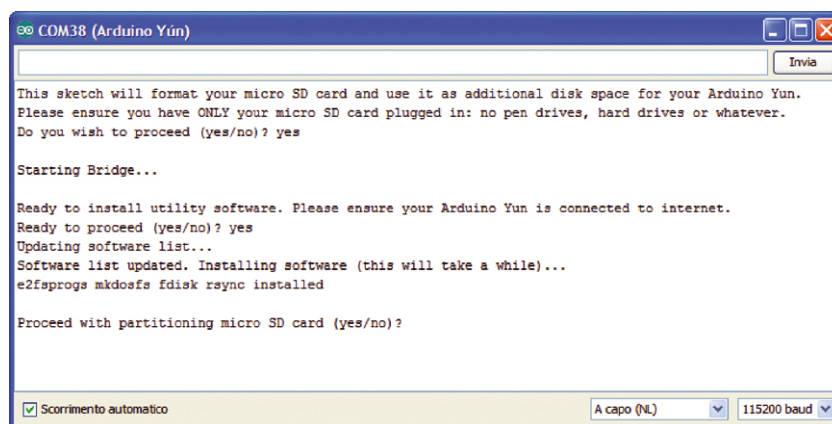


Fig. 22

di specificare la dimensione della partizione dati sulla SDCard, il resto dello spazio disponibile verrà dedicato al file system di root. Per la nostra SDCard di 8 GB, per non fare torto a nessuno, abbiamo scelto di dedicare 4 Gb alla partizione dati e 4 GB alla partizione di root (Fig. 23). Dopo

aver partizionato e formattato le partizioni sulla SDCard tutti i file memorizzati nella memoria flash vengono copiati nella partizione di root della SDCard, con il risultato finale visibile in Fig. 24. Per rendere effettive le modifiche è necessario riavviare la scheda Linino One. Con qualche diffi-

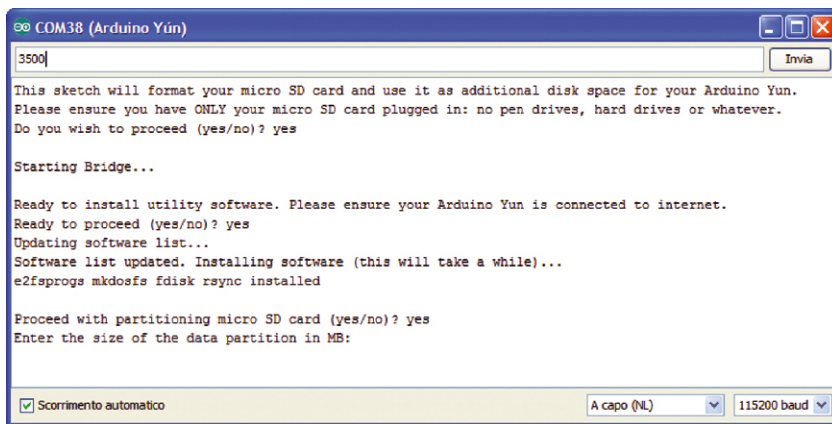


Fig. 23

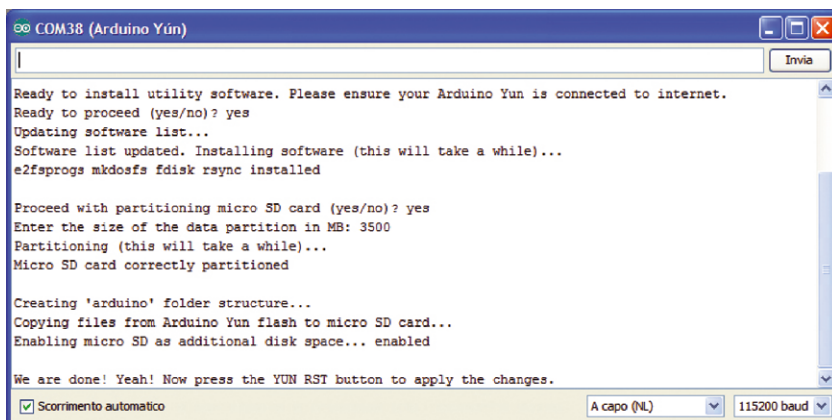


Fig. 24

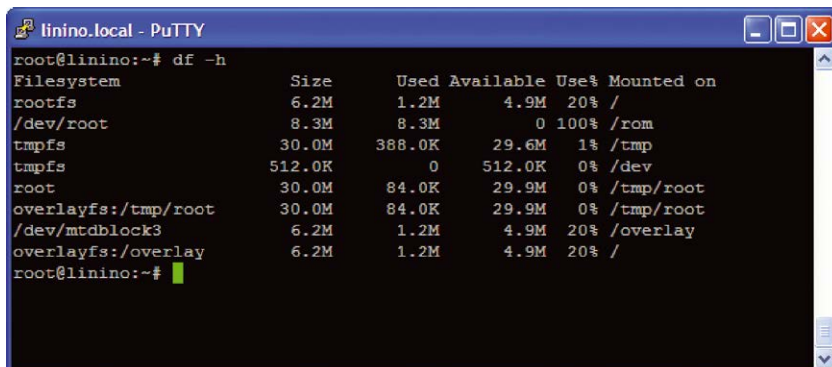


Fig. 25

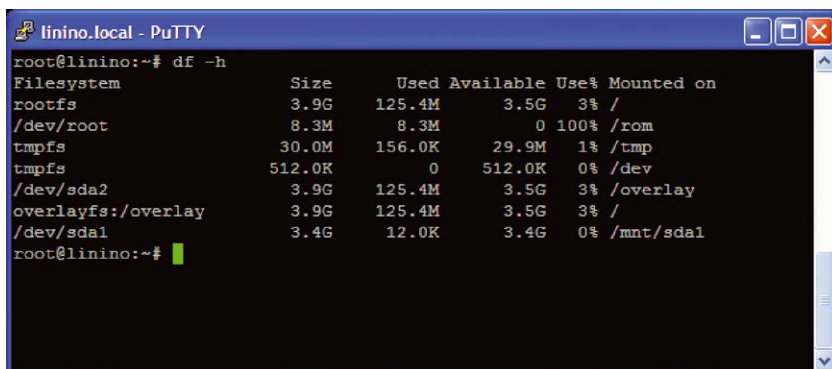


Fig. 26

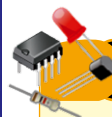
coltà, premete il pulsante Reset, guardando la scheda dal lato del connettore Ethernet, il primo a sinistra, quello vicino ai due regolatori. Sul monitor seriale vedrete scorrere i messaggi di console relativi al processo di boot della scheda. Al termine del processo di boot da una finestra SSH, digitate il comando:

`df -h`

Il comando permette di visualizzare l'occupazione delle memoria di massa presente sulla SDCard.

Nel nostro caso, utilizzando la connessione remota via Putty abbiamo ottenuto gli output visibili nelle Fig. 25 e Fig. 26, rispettivamente prima e dopo il reboot:

Nel "mondo" OpenWrt l'operazione che abbiamo eseguito viene identificata come "extroot". Ovviamente questa operazione è perfettamente replicabile sulle schede Arduino Yún. Nel prossimo appuntamento utilizzeremo questa configurazione per realizzare un progetto al quale stiamo pensando da tempo. Ma non vogliamo rovinarvi la sorpresa. ■



per il MATERIALE

La board Linino ONE (cod. 7300-LININOONE) è disponibile al prezzo di 62,00 Euro mentre la confezione che contiene sia la scheda RJ45 che la scheda USB è disponibile al prezzo di 29,00 Euro (cod. 7300-LININOAD-DONS). I prezzi si intendono IVA compresa.

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



È TEMPO DI CeBIT

CeBIT si ripropone ogni anno come il più importante evento mondiale dell'economia digitale. Nella cornice di un intreccio unico tra momento espositivo e momento congressuale, importanti gruppi mondiali, piccole e medie imprese e startup presentano i temi più importanti dell'industria della Information Technology: big data, applicazioni cloud, mobile, social business, sicurezza informatica o esempi applicativi dell'area Industria 4.0.

..... di ARSENIO SPADONI

I temi della prossima edizione del CeBIT di Hannover (16/03/2015 – 20/03/2015) sono stati presentati a Milano durante una Conferenza stampa alla quale hanno partecipato **Hartwig von Saß**, Capo Ufficio Stampa CeBIT, e il Dott. **Andrea Ardizzone**, Segretario Generale Assintel.

Hartwig von Saß ha sottolineato come il CeBIT – a partire dalle ultime due edizioni – si stia focalizzando sempre di più sui temi **dell'IT e della digitalizzazione**, lasciando ad altri eventi gli aspetti legati al mondo consumer.

Anche per l'edizione 2015, dunque, il CeBIT di Hannover si ripropone come il più importante evento mondiale dell'economia digitale. Quest'anno

il CeBIT si propone con il tema conduttore della **d!conomy**, con la presenza della Cina come Paese Partner, con un programma internazionale di conferenze e di contenuti su tutti i temi più importanti relativi a IT e digitalizzazione. Come massima piattaforma mondiale dell'economia digitale, il salone riunisce gli importanti trend di mercato dei big data, del cloud, del mobile, del social e della security trattandoli in tutta la loro ampiezza e profondità.

D!conomy: IL TEMA CONDUTTORE

Al centro dell'attenzione del CeBIT 2015 ci saranno la vertiginosa crescita dell'influenza della Information Technology in tutti gli ambiti dell'economia e



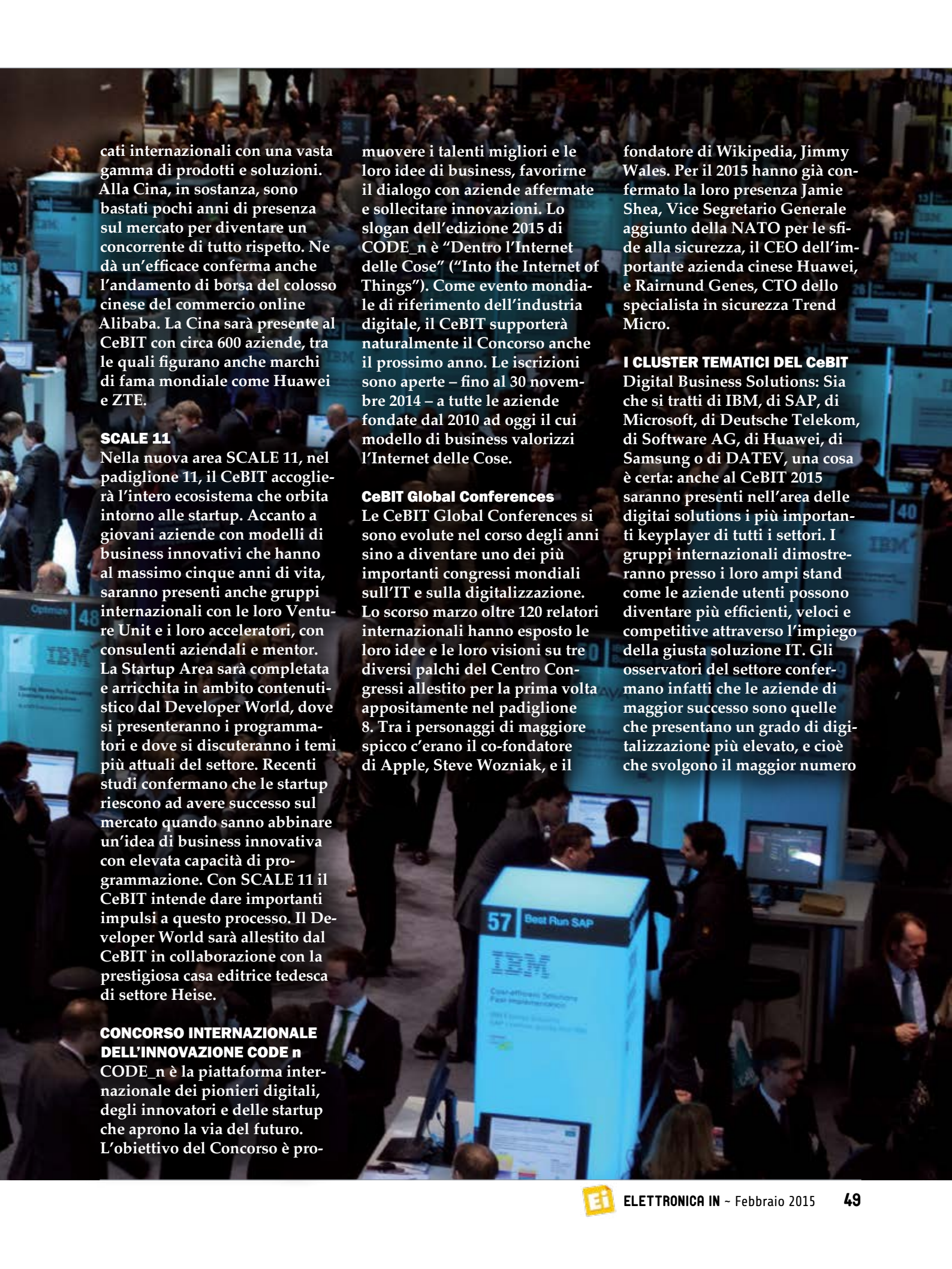
della società e l'importante ruolo di motore dell'innovazione che da essa ne deriva. "L'industria dell'IT, e noi come organizzatori, abbiamo scelto per il CeBIT 2015 il tema conduttore della d!conomy", spiega Oliver Frese, del CdA di Deutsche Messe. "La tendenza alla digitalizzazione investe sempre più incondizionatamente la sfera della vita professionale e sociale. L'IT trasforma radicalmente i modelli di business esistenti e allo

stesso tempo fa nascere modelli completamente nuovi. Riesce a rivoluzionare interi settori dell'industria con sorprendente rapidità. E la d!conomy tiene conto di questa evoluzione. Gli importanti trend tecnologici che hanno caratterizzato il settore dell'IT negli scorsi anni con il loro rapido sviluppo - big data, cloud computing, mobile, social e security - si innestano ora l'uno nell'altro con un profondo impatto sia sull'economia che sulla società in generale. Questa convergenza porterà a innovazioni fondamentali, che autorizzano già a parlare dell'inizio

di una nuova era dell'industria - l'era digitale."

PAESE PARTNER: CINA

La Cina, Paese Partner del CeBIT 2015, ha registrato negli ultimi anni un dinamico sviluppo proprio nel settore dell'IT. Presentatasi inizialmente sul mercato soprattutto come Paese subfornitore di componenti hardware, solo in un secondo tempo la nazione più popolata del Pianeta ha iniziato a produrre hardware di qualità competitivo a livello internazionale. E oggi un numero crescente di gruppi cinesi che operano nell'IT conquista i mer-



cati internazionali con una vasta gamma di prodotti e soluzioni. Alla Cina, in sostanza, sono bastati pochi anni di presenza sul mercato per diventare un concorrente di tutto rispetto. Ne dà un'efficace conferma anche l'andamento di borsa del colosso cinese del commercio online Alibaba. La Cina sarà presente al CeBIT con circa 600 aziende, tra le quali figurano anche marchi di fama mondiale come Huawei e ZTE.

SCALE 11

Nella nuova area SCALE 11, nel padiglione 11, il CeBIT accoglierà l'intero ecosistema che orbita intorno alle startup. Accanto a giovani aziende con modelli di business innovativi che hanno al massimo cinque anni di vita, saranno presenti anche gruppi internazionali con le loro Venture Unit e i loro acceleratori, con consulenti aziendali e mentor. La Startup Area sarà completata e arricchita in ambito contenutistico dal Developer World, dove si presenteranno i programmatori e dove si discuteranno i temi più attuali del settore. Recenti studi confermano che le startup riescono ad avere successo sul mercato quando sanno abbinare un'idea di business innovativa con elevata capacità di programmazione. Con SCALE 11 il CeBIT intende dare importanti impulsi a questo processo. Il Developer World sarà allestito dal CeBIT in collaborazione con la prestigiosa casa editrice tedesca di settore Heise.

CONCORSO INTERNAZIONALE DELL'INNOVAZIONE CODE_n

CODE_n è la piattaforma internazionale dei pionieri digitali, degli innovatori e delle startup che aprono la via del futuro. L'obiettivo del Concorso è pro-

muovere i talenti migliori e le loro idee di business, favorirne il dialogo con aziende affermate e sollecitare innovazioni. Lo slogan dell'edizione 2015 di CODE_n è "Dentro l'Internet delle Cose" ("Into the Internet of Things"). Come evento mondiale di riferimento dell'industria digitale, il CeBIT supporterà naturalmente il Concorso anche il prossimo anno. Le iscrizioni sono aperte – fino al 30 novembre 2014 – a tutte le aziende fondate dal 2010 ad oggi il cui modello di business valorizzi l'Internet delle Cose.

CeBIT Global Conferences

Le CeBIT Global Conferences si sono evolute nel corso degli anni sino a diventare uno dei più importanti congressi mondiali sull'IT e sulla digitalizzazione. Lo scorso marzo oltre 120 relatori internazionali hanno esposto le loro idee e le loro visioni su tre diversi palchi del Centro Congressi allestito per la prima volta appositamente nel padiglione 8. Tra i personaggi di maggiore spicco c'erano il co-fondatore di Apple, Steve Wozniak, e il

fondatore di Wikipedia, Jimmy Wales. Per il 2015 hanno già confermato la loro presenza Jamie Shea, Vice Segretario Generale aggiunto della NATO per le sfide alla sicurezza, il CEO dell'importante azienda cinese Huawei, e Raimund Genes, CTO dello specialista in sicurezza Trend Micro.

I CLUSTER TEMATICI DEL CeBIT

Digital Business Solutions: Sia che si tratti di IBM, di SAP, di Microsoft, di Deutsche Telekom, di Software AG, di Huawei, di Samsung o di DATEV, una cosa è certa: anche al CeBIT 2015 saranno presenti nell'area delle digital solutions i più importanti keyplayer di tutti i settori. I gruppi internazionali dimostreranno presso i loro ampi stand come le aziende utenti possono diventare più efficienti, veloci e competitive attraverso l'impiego della giusta soluzione IT. Gli osservatori del settore confermano infatti che le aziende di maggior successo sono quelle che presentano un grado di digitalizzazione più elevato, e cioè che svolgono il maggior numero



possibile di processi con l'utilizzo di soluzioni IT.

ECM, Input/Output Solutions: Questo cluster offre una panoramica completa di soluzioni, tendenze e sviluppi relativi a raccolta, gestione, memorizzazione, archiviazione e fornitura di contenuti e documenti in ambito aziendale per supportare al meglio i processi di business. Verranno inoltre presentate anche soluzioni per applicazioni Office e Workplace o Helpdesk. Con il cluster ECM, Input/Output Solutions, il CeBIT è la sola manifestazione a livello mondiale che presenti soluzioni per documenti in ambito aziendale e loro collegamento sistematico in rete, oltre all'integrazione in altri processi aziendali digitali, dall'inizio alla fine: dall'ufficio postale alla elaborazione in ECM e Mailing Solutions, alle soluzioni in uscita.

ERP, CRM, BI & Commerce Solutions: Il grado di digitalizzazione è uno dei fattori decisivi per il successo di un'azienda. Chi voglia digitalizzare e ottimizzare processi e pianificazione delle risorse, troverà in questo cluster tematico il software adeguato, spesso mirato sulle specifiche esigenze di ogni singolo settore. Smart Factory e Big Data in particolare fanno dell'Enterprise Resource Planning e delle Com-

merce Solutions temi ancora più importanti. I sistemi per la pianificazione delle risorse – siano esse finanze, giacenze di magazzino, personale o materie prime – sono in effetti la spina dorsale di ogni azienda di successo. Al centro dell'attenzione del cluster ci saranno in particolare applicazioni per piccole e medie imprese. L'offerta espositiva sarà espressamente arricchita da una nuova lounge per le PMI. Nel padiglione 5 gli espositori presenteranno anche soluzioni per la digitalizzazione del Customer Relationship Management (CRM) e soluzioni di Social Media Marketing. Saranno proposte inoltre soluzioni di Business Intelligence (BI), che, nel contesto della Customer Analytics, aiutano a capire meglio i desideri della clientela e a tenerne conto nello sviluppo dei prodotti o nella pianificazione del marketing. Anche CRM, BI e Enterprise 2.0 sono influenzate dai Big Data: si rilevano in effetti sempre più dati – nell'ottica del business tali dati sono però rilevanti solo se analizzati con programmi intelligenti. Gli esperti ne parleranno, e discuteranno inoltre di problematiche attuali, di nuovi esempi applicativi e di tendenze, nella cornice di un forum che si prospetta molto interessante.

Business Security: Nel padiglione 6,

il CeBIT allestirà una piattaforma unica nel suo genere per le esigenze di sicurezza delle aziende digitali – esigenze che spaziano dalla cybersicurezza alla sicurezza dei dispositivi mobili, alla protezione dei dati. Saranno presenti importanti leader di mercato e pionieri dell'innovazione del settore della sicurezza informatica. Già durante l'edizione 2014 oltre 500 aziende hanno presentato soluzioni per una risposta concreta alle crescenti sfide che si pongono in questo ambito. In concreto si tratta di: soluzioni antispy e antivirus, sicurezza degli endpoint & delle applicazioni, sicurezza di rete e firewall, sicurezza della comunicazione & sicurezza dei dispositivi mobili, sicurezza del cloud e cybersicurezza, privacy, prevenzione della perdita di dati, recupero dati, autenticazione, crittazione.

Public Sector Parc: Nel padiglione 7 sarà accolta la fiera leader per l'utilizzo dell'IT nel settore pubblico. L'offerta esposta coprirà le esigenze dell'intera categoria delle pubbliche amministrazioni – dall'UE ai singoli Stati, alle Regioni, ai Comuni. Saranno proposte soluzioni per l'egovernment, soluzioni per la pubblica amministrazione, sistemi di geoinformazione, soluzioni per il settore della sanità pubblica, soluzioni per gli agglomerati urbani. Presso gli stand degli espositori o nella cornice del Forum si potranno toccare con mano anche le tendenze originate dalla digitalizzazione della pubblica amministrazione: ad esempio l'impiego di identità sicure nelle applicazioni di e-government, infrastrutture digitali che abbracciano l'intero campo delle autorità (ad esempio gestione dell'informazione federale), cloud computing per il settore pubblico (GCloud), eParticipation, gestione documentale/

atti elettronici, social media o servizi mobili per i cittadini.

Research & Innovation: Il CeBIT è da sempre la più importante piattaforma internazionale per la ricerca e lo sviluppo nel settore dell'IT. Molte università e istituti di ricerca internazionali si avvaranno pertanto del padiglione 9 del quartiere fieristico per fare vedere da vicino il loro lavoro. Al centro dell'attenzione ci saranno temi come tecnologie 3D, intelligenza ambientale, intelligenza artificiale, big data, cybersicurezza, istruzione, elearning, sistemi embedded, interazione uomo-macchina, Industria 4.0, sourcing dell'innovazione, mondi in rete, visioni urbane, realtà virtuale/aumentata.

IT enables: IT e digitalizzazione sono sempre più alla base dell'innovazione negli ambiti più diversi. L'area IT enables presenta progetti "faro" nati dall'incontro tra aziende IT e settore utente. Al centro dell'attenzione ci saranno prodotti concreti e momenti di confronto tra esperti in merito a esempi di best practice. Verranno trattati in particolare temi relativi ai settori energetico, della produzione, automotive, mobilità & reti, agricoltura & industria alimentare, banche & finanze, commercio al dettaglio & sanità.

DatacenterDynamics@CeBIT: Il boom dei big data e dei servizi cloud sta generando una vertiginosa crescita dei data center a livello mondiale. La cooperazione tra l'organizzatore di fama internazionale DatacenterDynamics e il CeBIT riunirà le loro competenze sul fronte convegnistica, espositivo e del networking. Con lo slogan "The Business of Data Centres" verrà data visibilità all'intera catena del

valore dei centri dati. In occasione della conferenza, esperti di tutti i settori saranno invitati a intervenire su vari temi. Sul palco ci saranno opinion leader di fama internazionale che spiegheranno quale ruolo i moderni centri dati possono avere in un'azienda dei nostri giorni anche con riguardo all'applicazione delle tecnologie cloud. Il programma tratterà quattro temi centrali: Design e strategia, IT e reti, Ambiente critico, Sourcing & Delivery.

Communications & Networks: In questo cluster si troveranno tutte le tecnologie relative al tema della comunicazione professionale: da telefonia classica a telefonia IP, a SIP (Session Initiation Protocol). L'attenzione sarà rivolta sia alle soluzioni in banda larga e alle tecnologie di rete sia all'infrastruttura delle telecomunicazioni e ai servizi. Nel padiglione 13, che ospiterà il settore, i visitatori troveranno nuovi prodotti e soluzioni su comunicazione unificata, soluzioni in banda larga, business communication, dispositivi

mobili & relativi servizi, reti, casa intelligente, telematica & navigazione, soluzioni M2M.

Planet Reseller: Planet Reseller è il centro del commercio di settore al CeBIT. Su circa 16.000 metri quadrati saranno presenti oltre 250 tra produttori, distributori e cooperative d'acquisto di 40 diversi Paesi. Il "Planet" offre ai visitatori specializzati la possibilità di vedere in anteprima nuovissime tecnologie come gli ultimissimi smartphone, tablet, laptop, desktop-computer e relativi accessori.

Global Sourcing: Global Sourcing, la piattaforma per i produttori di accessori, componenti e dispositivi, è il più grande mercato europeo per gli OEM (Original Equipment Manufacturer), gli ODM (Original Design Manufacturer) e le grandi marche. Buyer e partner commerciali di tutto il mondo incontrano qui espositori che intendono valorizzare il CeBIT come via di accesso al grande mercato europeo. ■



Piattaforma per sviluppare applicazioni di telecontrollo e teleallarme via cellulare, si programma facilmente grazie a un ambiente di programmazione visuale a blocchi. Prima puntata.

DM BOARD ICS: IL TELECONTROLLO GSM UNIVERSALE

di DENIS DE GRANDIS E MARCO SCIPIONI

La diffusione degli smartphone sta portando ad una crescita esponenziale dell'offerta di sistemi gestibili a distanza utilizzando non solo la connessione cellulare, ma anche i link wireless di cui sono equipaggiati i moderni telefonini, primi fra tutti Bluetooth e WiFi. Con gli smartphone è possibile, ad esempio, controllare l'accensione della caldaia per il riscaldamento, oppure comandare l'apertura del cancello automatico di casa o, ancora, ricevere messaggi di allarme dal nostro sistema antifurto.

Normalmente gli apparati per il controllo a distanza tramite telefono cellulare, oltre ad essere abbastanza costosi, utilizzano hardware e software proprietari e possono svolgere solo una determinata funzione. Anche se i telecontrolli sono ormai ampiamente diffusi nelle nostre case e nei luoghi più disparati, risulta difficile per la gran parte

degli sperimentatori elettronici riuscire a realizzare degli apparati in grado di dialogare con i più moderni sistemi di comunicazione, in quanto a chi si accinge a farlo sono richieste competenze molto elevate sia dal punto di vista della progettazione hardware e software, sia per quel che riguarda la costruzione del circuito stampato, che sovente deve essere multistrato o almeno a doppia faccia con fori metallizzati, per contenere le dimensioni. Per consentire anche agli hobbisti di praticare con le applicazioni controllate da smartphone, abbiamo quindi pensato di risolvere il problema principale: quello della progettazione. La soluzione che abbiamo trovato è stata creare una piattaforma di prototipazione che permetta di realizzare svariate applicazioni basate su cellulare.

Questa piattaforma si chiama DM Board, ed è un progetto che permette anche agli sperimentatori di



Fig. 1

realizzare un qualsiasi dispositivo che possa essere comandato a distanza tramite smartphone. DM Board ICS è un progetto open source che permette di controllare e comandare ciò che desideriamo, ovunque e nel modo che vogliamo; l'idea si basa sulla realizzazione di una scheda (DM Board ICS) dotata di una serie di periferiche, programmabili in un ambiente grafico. Il software di sviluppo creato allo scopo si chiama DM Design ed è di utilizzo pratico, veloce ed intuitivo: per programmare le applicazioni non richiede di conoscere particolari linguaggi di programmazione, perché con semplici blocchi predefiniti consente di controllare tutte le funzioni della scheda. Ciò, semplicemente unendo blocchi software ognuno dei quali gestisce una singola funzione della scheda. La programmazione viene quindi supportata attraverso l'ambiente

di sviluppo integrato DM Design, che oltre alla creazione del firmware permette la programmazione del microcontrollore presente nella scheda. Inoltre, da un apposito store online (DM Store) accessibile direttamente da DM Design (www.dmboard.it/) è possibile scaricare dei programmi già realizzati in formato sorgente, da poter modificare e/o inserire direttamente nella DM Board ICS. Il DMStore è uno spazio sul sito, dove tutti gli utenti possono inserire i loro progetti condividendoli con la community; così ognuno può scaricare il programma che più si adatta alle proprie esigenze, modificarlo o utilizzarlo immediatamente sulla propria DM Board ICS. Grazie alla condivisione tra utenti, per la completa gestione della DM Board ICS potrebbe non essere necessario scrivere una riga di codice, in quanto, semplicemente seguendo delle procedure guidate, DM Design è in grado

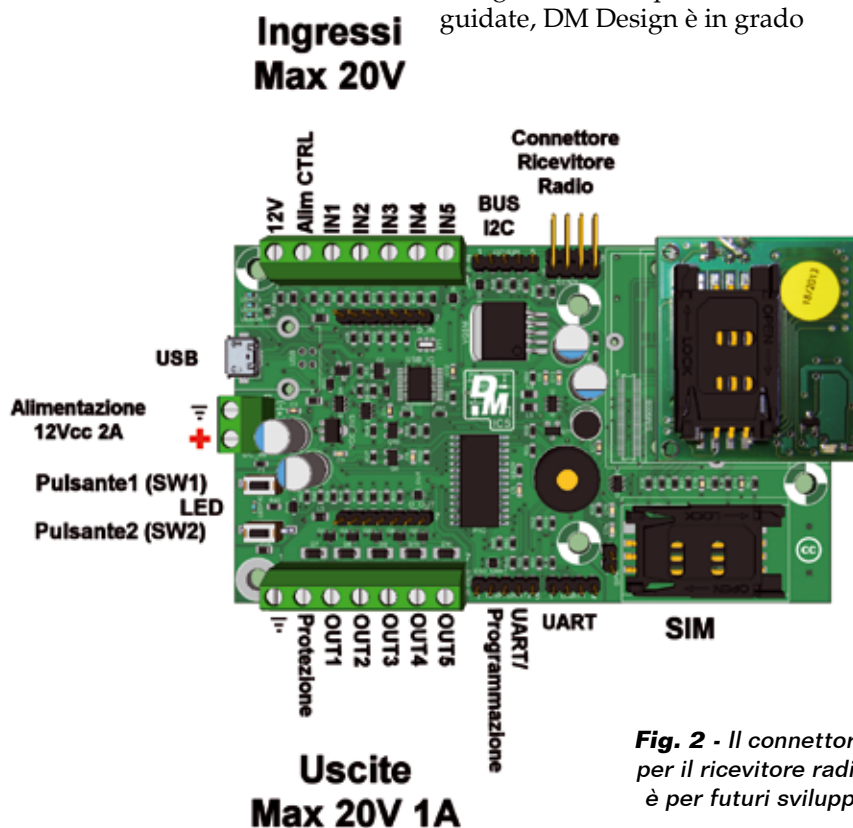


Fig. 2 - Il connettore per il ricevitore radio è per futuri sviluppi.

di generare automaticamente il codice da scaricare nella scheda. Il progetto DM Board consta della scheda DM Board ICS e da un software di sviluppo del firmware per la gestione dell'hardware, ossia DM Design. In queste pagine analizziamo l'hardware della DM Board ICS e l'utilizzo di DM Design, per terminare con la realizzazione di un'applicazione pratica: un allarme filare che utilizza semplici contatti magnetici, oppure, attraverso dei convertitori filare/wireless, un allarme senza fili che utilizza i più comuni sensori radio; l'allarme dovrà segnalare, mediante una sirena e tramite l'invio di un SMS, una tentata effrazione o un'intrusione nell'ambiente protetto.

DM Board ICS

Diamo subito uno sguardo alla demoboard, rappresentata in **Fig. 2**. La scheda è dotata di 5 ingressi e altrettante uscite; sia gli ingressi che le uscite possono funzionare a 3,3 V o 12 Vcc a seconda del punto di connessione. Le uscite sono realizzate attraverso degli stadi open drain in grado di pilotare direttamente dei relé. A bordo scheda è inoltre presente un buzzer che si può utilizzare per generare segnalazioni acustiche. Sono altresì montati due pulsanti e un LED gestibili direttamente dal firmware caricato nel microcontrollore della scheda. Il controllo a distanza della DM Board ICS è affidato ad un modulo GSM provvisto di microfono per l'ascolto ambientale e di uscita audio per le chiamate vocali. Per la comunicazione con il mondo esterno, la scheda dispone di una porta USB 2.0 (in funzione device) un link I²C-Bus e, inoltre, due UART condivisi con il modulo GSM e con la porta USB. Infine DM Board ICS contiene un circuito per la rilevazione della

tensione di alimentazione, il che permette di tenere sotto controllo lo stato di carica di un'eventuale batteria per alimentare la scheda in caso di blackout o nelle applicazioni dove manca la tensione di rete. Infatti la DM Board ICS è progettata per essere alimentata a 12 Vcc attraverso una batteria, ma può tranquillamente funzionare anche a tensioni inferiori, purché continue.

L'hardware è stato progettato per funzionare a basso consumo, assorbendo appena 20 µA, grazie alla specifica modalità prevista dal microcontrollore, il quale rimane comunque in "ascolto" di eventuali interrupt che, quando giunti, lo riportano nella modalità normale. Tutte le periferiche on board prevedono dei componenti per la protezione contro le scariche elettrostatiche.

SCHEMA ELETTRICO

Iniziamo ora ad analizzare nel dettaglio le singole parti del circuito elettronico, partendo dallo stadio di alimentazione, che è stato realizzato con un regolatore Microchip a basso consumo (ha una corrente di quiescenza di circa 2 µA) che è in grado di erogare in uscita una tensione stabilizzata di 3,3 volt ed una corrente di 250 mA. La tensione in uscita dal regolatore (siglato, nello schema elettrico, VCC3V3) è utilizzata per alimentare il microcontrollore e la porta USB.

La DM Board ICS è dotata anche di un secondo regolatore di tensione, siglato VGSM, utilizzato per l'alimentazione del modulo GSM: in questo caso è del tipo a uscita variabile, la cui tensione è determinata dai valori delle resistenze R1 ed R2, secondo la relazione:

$$R1 = R2 \cdot \left(\frac{V_{out}}{1,240} - 1 \right)$$

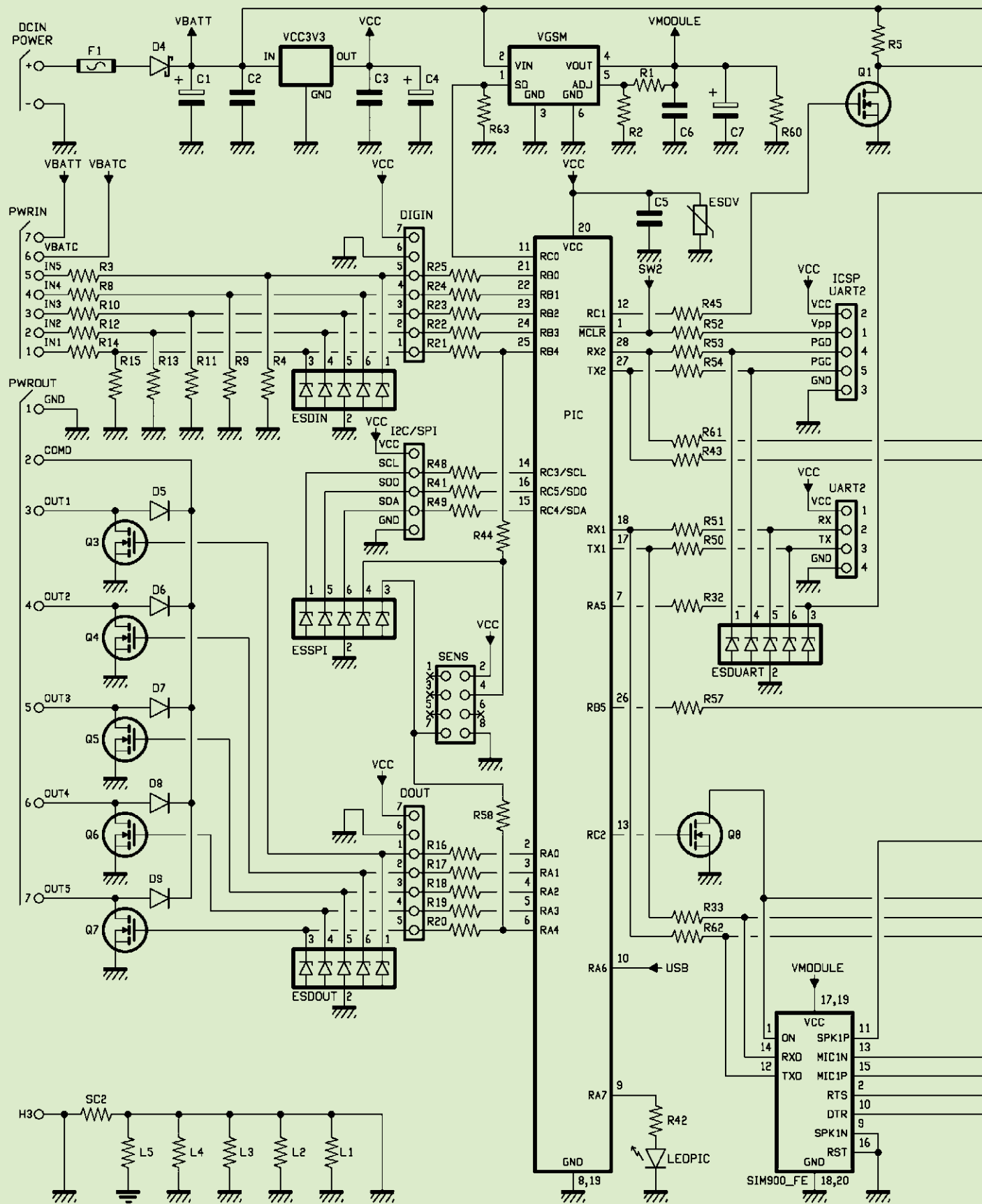
Volendo una tensione di uscita di circa 4,5V e impostando R2 a 39 kohm, la R1 risulterà del valore:

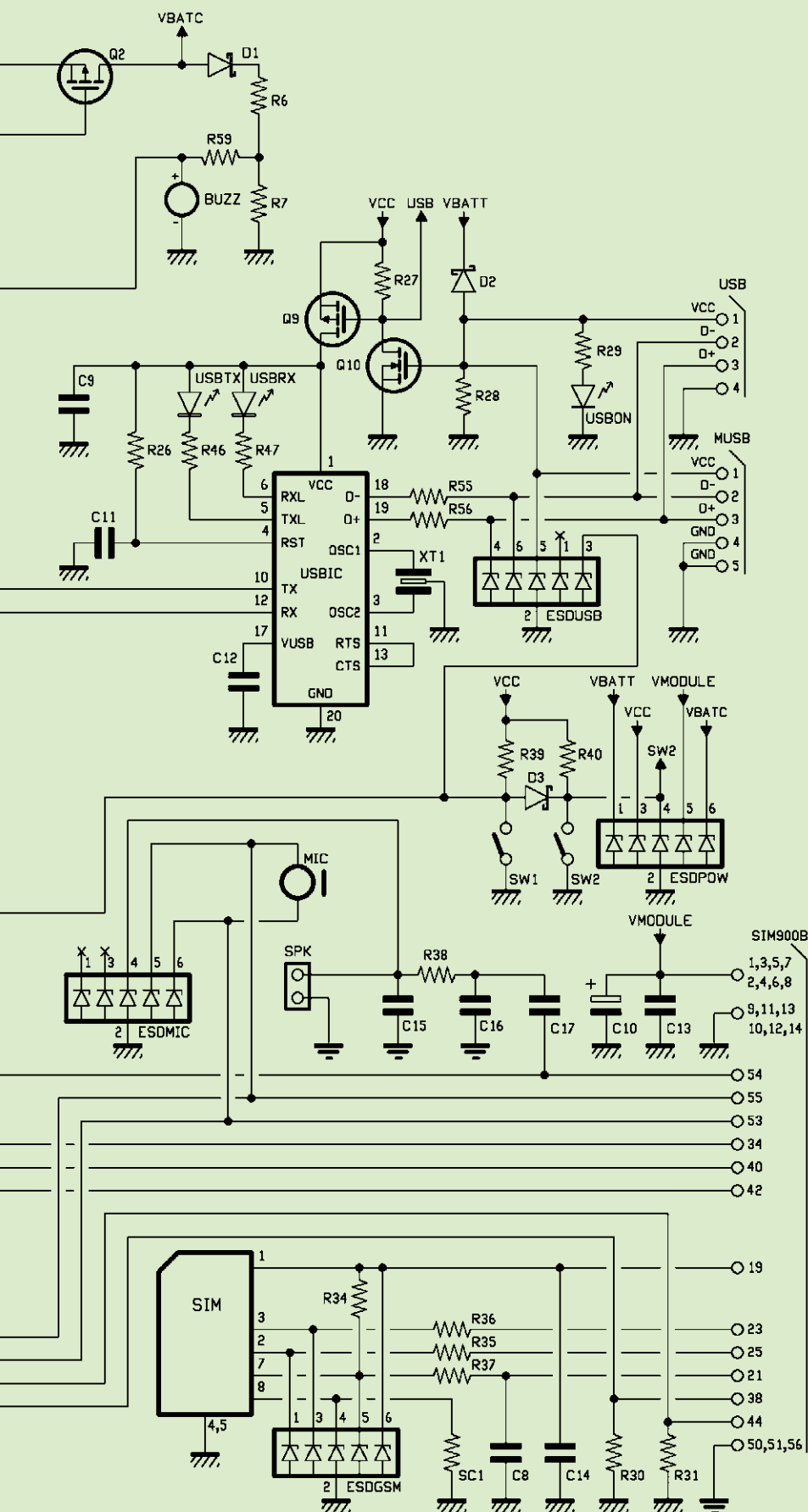
$$R1 = 39000 \cdot \left(\frac{4,5}{1,240} - 1 \right) = 102532$$

Scegliamo quindi R1 pari a 100 kohm, che è il valore standard più prossimo nella serie E12 e nella E24. Nello schema, la resistenza R60 ha lo scopo di garantire un carico anche con il modulo GSM spento, allo scopo di stabilizzare meglio l'uscita del regolatore anche in caso di basse correnti richieste in uscita.

Per ridurre i consumi, è possibile spegnere il regolatore quando non viene utilizzato il modulo GSM: a ciò provvede il piedino 11 del microcontrollore, che agisce sull'SD (ShutDown) del regolatore VGSM.

La DM Board ICS è governata da un microcontrollore microchip 18F26K22 con tecnologia XLP (eXtreme Low Power) che permette alla scheda di funzionare in basso consumo (circa 20 µA) mantenendo tutti i registri e la memoria attivi e rimanendo in ascolto di eventuali interrupt in arrivo dalle periferiche presenti nella scheda. Al microcontrollore sono connessi direttamente gli ingressi e le uscite a bassa tensione a 3,3V, il LED ed il buzzer. La porta I²C/SPI del microcontrollore è protetta da un array di diodi TVS (Transient Voltage Suppressor) contro le scariche elettrostatiche che potrebbero derivare dai collegamenti elettrici con l'esterno. Questi particolari diodi sono simili agli Zener, ovvero entrano in conduzione nel momento in cui la tensione ai loro capi supera una certa soglia; tuttavia, a differenza degli Zener, i TVS hanno tempi di intervento molto brevi e quindi risultano adatti a proteggere i circuiti dalle





scariche elettrostatiche e in generale da sovratensioni molto brevi o con fronti di salita ripidi.

INGRESSI A 12V

Il connettore degli ingressi (PWRIN) presente nella DM Board ICS consente la ricezione di cinque segnali di comando attraverso la connessione con altrettante linee. Ogni stadio d'ingresso è formato da un partitore resistivo che consente di rendere compatibili gli ingressi a 12V con i pin del microcontrollore, i quali accettano una tensione massima di 3,3V. Grazie ai valori scelti, applicando la regola del partitore resistivo, una tensione d'ingresso di 12 Vcc viene ridotta, all'uscita del partitore, a:

$$V_{out} = \frac{12 * 2200}{2200 + 10000} \approx 2.2V$$

In assenza di tensione, il partitore di tensione d'ingresso pone a 0V l'ingresso del microcontrollore. La tensione da fornire a ciascun ingresso può provenire dall'esterno, oppure, nel caso di comando da parte di contatti puliti, può essere quella prelevabile dal contatto 7 della morsettiere PWRIN e fatta passare, appunto, dai contatti. Nella stessa PWRIN c'è un contatto chiamato VBATC che corrisponde a una linea controllabile via software dal microcontrollore, utile in caso si voglia realizzare un programma a basso consumo. Anche in questo circuito sono presenti i TVS per la protezione contro le scariche elettrostatiche. Volendo realizzare un allarme con ingressi a contatto normalmente chiuso, dovremo connettere i pin d'ingresso con il morsetto che fornisce i +12V, come in Fig. 3.

USCITE DI POTENZA

Lo stadio di uscita della nostra scheda è realizzato attraverso

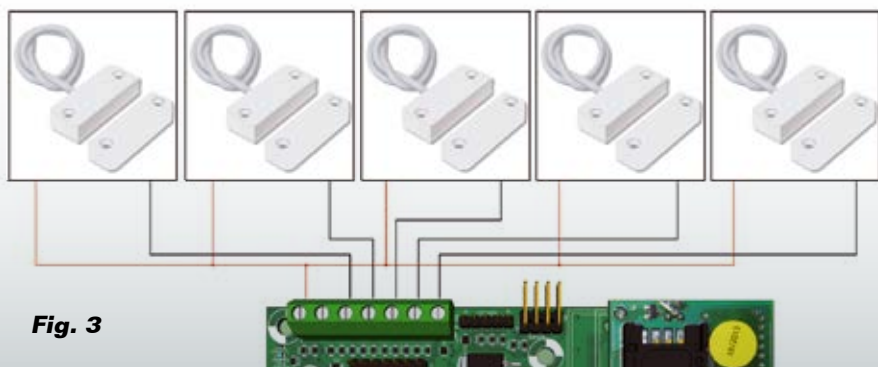


Fig. 3

dei MOSFET enhancement-mode a canale N, in configurazione open drain. Nel circuito sono inseriti dei diodi di protezione dalle extratensioni generate dai carichi induttivi in aggiunta a quelli contenuti nei MOSFET; tutti i diodi hanno un capo in comune collegato al COMD della morsettiera PWROUT, il quale va portato allo stesso potenziale di alimentazione dei carichi. Questa soluzione consente di alimentare gli utilizzatori a tensione diversa da quella di alimentazione della scheda, ma vale solo se tutti gli utilizzatori sono alimentati dalla stessa linea; in caso contrario, COM non va utilizzato ma resta la protezione implementata dai diodi interni ai MOSFET. Tornando al nostro allarme, avremo la necessità di connettere ad un'uscita di PWROUT un relé, per il comando di una sirena; il collegamento avverrà come mostrato nella **Fig. 4**, nella quale il positivo di alimentazione della bobina è collegato al +12V d'ingresso (DCIN POWER) ed al morsetto COMD.

PORTE DI COMUNICAZIONE

L'interfaccia USB della DM Board ICS è una USB 2.0 di tipo device, implementata grazie all'integrato siglato, nello schema elettrico, USBIC; viene utilizzata per la programmazione, ma può essere utilizzata anche per il trasferimento di dati dalla scheda al PC e viceversa. L'integrato USBIC è un converter TTL/USB

bidirezionale, pilotato dalle linee TX2 ed RX2 del microcontrollore (facenti capo al secondo UART disponibile nel PIC); il componente è un integrato Microchip MCP2200. Questo integrato ha un consumo in modalità attiva di circa 10 mA; essendo la DM Board ICS una scheda a basso consumo, si è deciso di accendere lo stadio USB solo se viene connesso un PC alla DM Board ICS. A ciò provvedono i due transistor siglati Q9 e Q10, che si attivano solo se tra i contatti 1 e 4 del connettore USB è presente una tensione di 5V (valore tipico di alimentazione di una porta USB); attraverso D2 (è uno Schottky, che permette di ridurre la caduta diretta ad appena 0,3V) la tensione giunge anche all'ingresso del regolatore VCC3V3, in modo da alimentare la scheda anche in as-

senza dell'alimentazione principale in arrivo da DCIN POWER. Il diodo D4 isola il circuito di questo ingresso quando la scheda funziona con la tensione dell'USB. Notate che in questa condizione gli ingressi funzionano solo se gli viene fornita una tensione esterna, ovvero se si ridimensionano i relativi partitori resistivi; le uscite di potenza funzioneranno solo se alimentate da una tensione esterna. I LED montati nell'interfaccia USB segnalano la connessione al PC (USBON), dati in trasmissione (USBTX) o in ricezione (USBRX). Anche in questo stadio sono stati inseriti dei TVS per la protezione contro le scariche elettrostatiche: precisamente, sono stati collocati sul lato USB, dato che i disturbi possono giungere dai cavi o dall'USB del computer.

MODULO GSM

La connessione alla rete GSM è affidata a un modulo cellulare GSM/GPRS; per aumentare la versatilità della scheda, è stato deciso di predisporre l'hardware per scegliere tra il modulo SIMCOM SIM900B (che è quello predefinito) che si salda sul

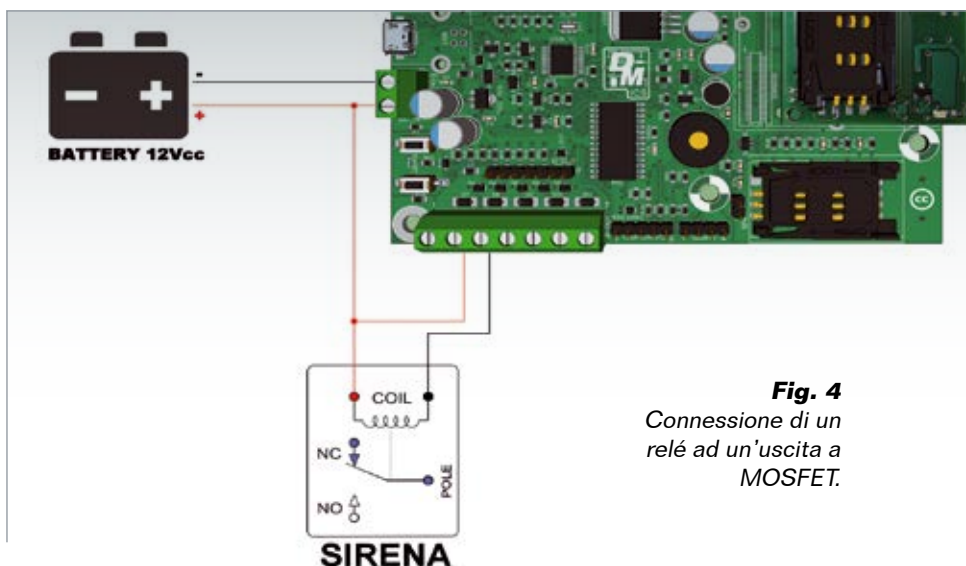
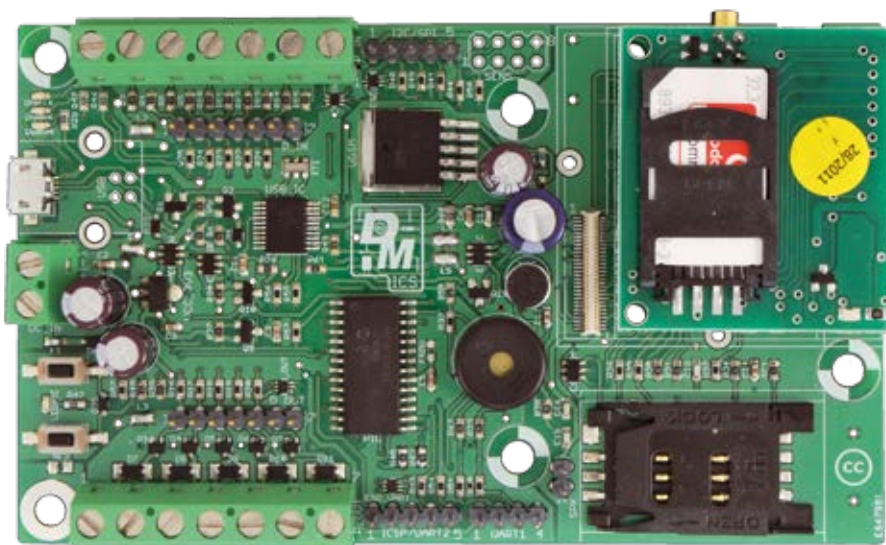
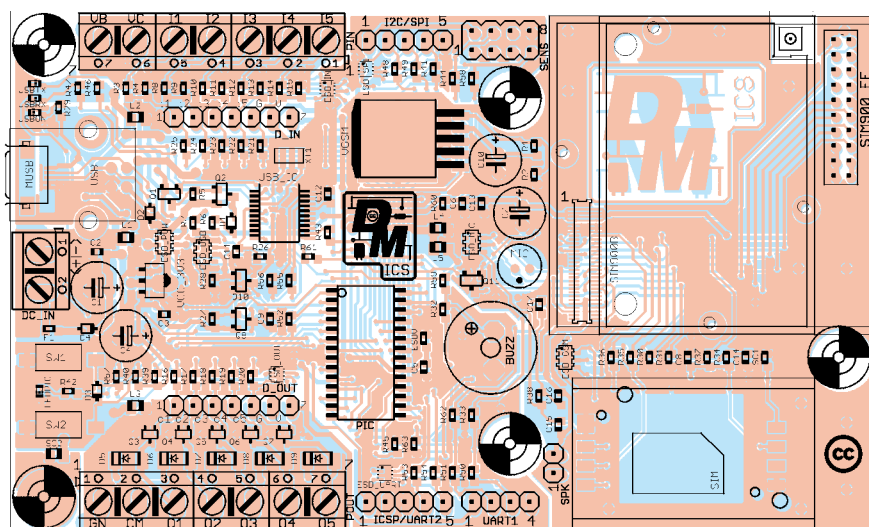


Fig. 4
Connessione di un relé ad un'uscita a MOSFET.

Elenco Componenti:

R1: 100 kohm (0603)
 R2: 39 kohm (0603)
 R3, R5, R6: 10 kohm (0603)
 R4, R7, R9, R11, R13, R15, R26:
 2,2 kohm (0603)
 R8, R10, R12, R14: 10 kohm (0603)
 R16÷R25: 22 ohm (0603)
 R27: 10 kohm (0603)
 R28: 1 kohm (0603)
 R29: 330 ohm (0603)
 R30, R31: 1 kohm (0603)
 R32: 22 ohm (0603)
 R33: 1 kohm (0603)
 R34: 10 kohm (0603)
 R35÷R37: 22 ohm (0603)
 R36: 22 ohm (0603)
 R37: 22 ohm (0603)
 R38÷R40: 10 ohm (0603)
 R41: 22 ohm (0603)
 R42: 330 ohm (0603)
 R43: 1 kohm (0603)
 R44: 22 ohm (0603)
 R45: 10 kohm (0603)
 R46, R47: 330 ohm (0603)
 R48÷R58: 22 ohm (0603)
 R59: 1 kohm (0603)
 R60: 330 ohm (0603)
 R61, R62: 1 kohm (0603)
 R63: 10 kohm (0603)
 C1: 100 µF 16 VL elettrolitico
 C2, C3: 100 nF ceramico (0603)
 C4: 100 µF 16 VL elettrolitico
 C5, C6: 100 nF ceramico (0603)
 C7: 330 µF 16 VL elettrolitico
 C8: 22 pF ceramico (0603)
 C9: 100 nF ceramico (0603)
 C10: 330 µF 16 VL elettrolitico
 C11: 100 nF ceramico (0603)
 C12: 470 nF ceramico (0603)
 C13: 10 pF ceramico (0603)
 C14: 220 nF ceramico (0603)
 C15: 33 pF ceramico (0603)
 C16: 33 pF ceramico (0603)
 C17: 1 µF ceramico (0603)
 D1÷D4: BAT20JFILM (SOD-323)
 D5÷D9: MM5D4148T3G (SOD-323)
 USBTX: LED rosso (0603)
 USBRX: LED giallo (0603)
 USBON: LED verde (0603)
 LEDPIC: LED verde (0603)
 L1÷L5: 0 ohm (0603)
 SC1: 0 ohm (0603)
 SC2: 0 ohm (0603)
 XT1: CSTCE12M0G15L99-R0
 Q1: 2N7002 (SOT-323)



Q2: FDN338P (SOT-23)
 Q3÷Q7: DMG1012UW-7 (SOT-323)
 Q8: 2N7002 (SOT-323)
 Q9: FDN338P (SOT-23)
 Q10: 2N7002 (SOT-323)
 BUZZ: PS1240P02BT
 F1: 0603SFF250F/32-2 (0603)
 VCC3V3: MCP1702T-3302E/MB
 VGSM: MIC29302WU
 USBIC: MCP2200-I/SS
 PIC: PIC18F26K22-I/SS (MF1175)
 SIM900_FE: Modulo GSM FT900M
 ESDIN: PESD5V0L5UV,125 (SOT-666)
 ESDOUT: PESD5V0L5UV,125 (SOT-666)
 ESDSPI: PESD5V0L5UV,125 (SOT-666)
 ESDUART: PESD5V0L5UV,125 (SOT-666)
 ESDMIC: PESD15VS5UD (SOT-457)
 ESDUSB: PESD15VS5UD (SOT-457)

ESDPOW: PESD15VS5UD (SOT-457)
 ESDGSM: PESD15VS5UD (SOT-457)
 USB: Connettore Micor-USB
 SW1: Pulsante (FSMSM)
 SW2: Pulsante (FSMSM)
 SIM: Connettore SIM

Varie:

- Morsetto 2 poli
- Morsetto 6 poli (2 pz.)
- Strip maschio 2 vie
- Strip maschio 4 vie
- Strip maschio 5 vie (2 pz.)
- Strip maschio 7 vie (2 pz.)
- Strip maschio 4x2 vie
- Connettore SQW-110-01-L-D
- Connettore BB530-06001-20R
- Circuito stampato

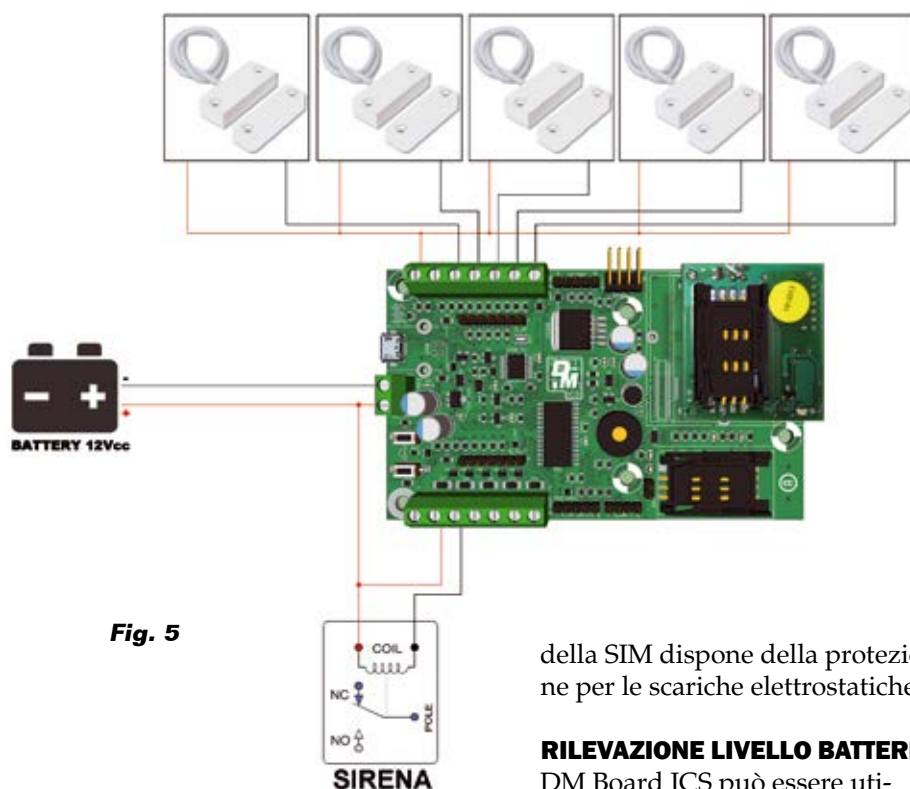


Fig. 5

PCB, e il più pratico FT900M di Futura Elettronica, che è una piccola scheda con su montato un SIM900 già corredato dei componenti che gli occorrono per funzionare nel migliore dei modi. In quest'ultimo caso, la presa dell'antenna è sulla scheda del modulo e non va usata quella prevista sulla DM Board ICS. Ricordate che i moduli possono essere montati uno in alternativa all'altro, altrimenti si incorre in guasti anche abbastanza rilevanti. Entrambi i GSM vengono gestiti nativamente dal software di sviluppo DM Design. Nel circuito, il modulo GSM può essere messo in standby e acceso dal microcontrollore attraverso il MOSFET Q8. Entrambi i moduli sono in grado di gestire autonomamente il microfono (già amplificato) a loro collegato e l'altoparlante (non amplificato). Il connettore per la SIM viene utilizzato solo dal SIMCOM SIM900B in quanto il modulo FT900M di Futura Elettronica è già fornito di connettore porta SIM. Anche il connettore

della SIM dispone della protezione per le scariche elettrostatiche.

RILEVAZIONE LIVELLO BATTERIA

DM Board ICS può essere utilizzata per applicazioni a basso consumo e quindi per sistemi alimentati a batterie. Risulta quindi fondamentale monitorare il livello di carica della batteria di alimentazione; per fare ciò si è deciso di realizzare un circuito per la lettura della tensione d'ingresso della scheda.

La tensione di alimentazione è naturalmente troppo elevata per essere letta direttamente dai convertitori analogico/digitale del microcontrollore e quindi deve essere ridotta; pertanto si è utilizzato un partitore come quello utilizzato per gli ingressi. Se il partitore fosse costantemente connesso all'alimentazione avremmo un consumo continuo di circa 1mA, valore troppo elevato per la realizzazione di sistemi a batterie.

Sono stati quindi inseriti i transistor Q1 e Q2, per consentire l'accensione del circuito di rilevazione della carica della batteria solo quando richiesto.

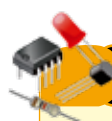
Il diodo D1 è stato inserito per evitare dei conflitti con la RA5 del microcontrollore, in quanto la

linea VAL_BATT viene condivisa con il buzzer BUZZ.

ALLARME BASATO SULLA DM Board ICS

Ora che abbiamo analizzato il funzionamento dell'hardware della DM Board ICS, possiamo riassumere come connettere i vari sensori e la sirena alla DM Board ICS per realizzare il nostro allarme GSM. Come si può vedere dallo schema di collegamento mostrato nella Fig. 5, i sensori a contatto e il relé per comandare la sirena possono essere collegati direttamente alla DM Board ICS. Per convertire il nostro circuito in un allarme wireless è possibile connettere le uscite di un qualsiasi ricevitore radio per sensori wireless a contatto pulito al posto dei singoli contatti magnetici visibili in Fig. 5.

Nel prossimo fascicolo di Elettronica In andremo ad analizzare la logica di funzionamento del firmware della DM Board ICS ed inizieremo a strutturare il software da inserire nella DM Board ICS per realizzare il nostro allarme con invio di SMS.



per il MATERIALE

La board è disponibile montata al prezzo di 134,00 Euro (cod. 7302-DMBOARDICS). La scheda non comprende i seguenti prodotti disponibili separatamente: modem cellulare montato cod. FT900M, Euro 49,00; antenna GSM ministilo cod. ANTSMAGSM, Euro 8,00; Cavo adattatore cod. CVANT-SMA, Euro 8,00. I prezzi sono 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>

UTILIZZIAMO RANDA

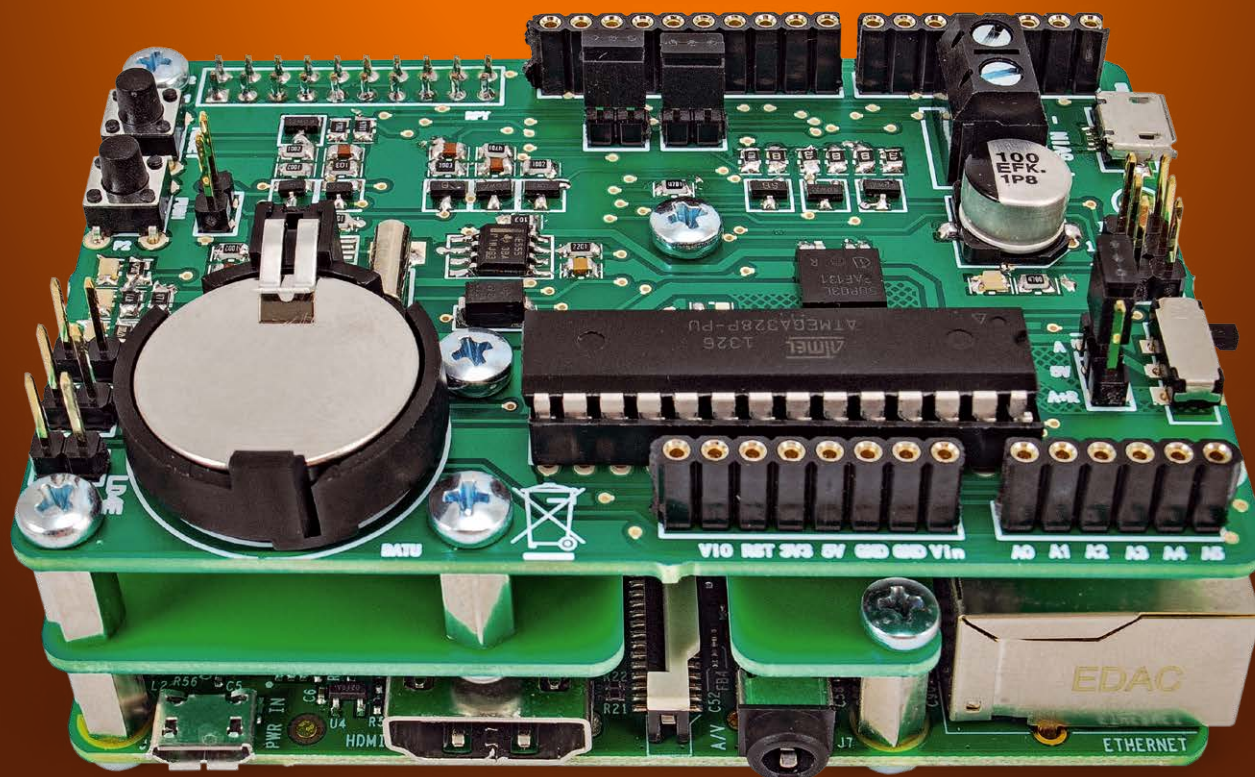
di DANIELE DENARO

Installiamo il software, verifichiamo l'operatività della scheda e realizziamo un Web Server, concludendo con un esercizio di programmazione.

Se vi siete persi gli articoli precedenti, riassumiamo brevemente cosa è RandA: si tratta di una scheda che permette l'integrazione fisica e funzionale tra Raspberry Pi e Arduino, consentendo di utilizzare l'ampia offerta di shield per Arduino coniugando le enormi potenzialità di Raspberry Pi. RandA, inoltre, include

un'intelligente gestione dell'alimentazione ed un orologio (in Fig. 1a vedete lo schema funzionale di RandA, mentre in Fig. 1b trovate comandi e connessioni). Per governare la scheda abbiamo previsto un set di software che prende posto su una SD-Card da inserire nell'apposito slot di Raspberry Pi; quindi, la prima cosa da fare

dopo aver realizzato RandA ed averla accorpata a Raspberry Pi, è installare nella SD il software necessario al funzionamento del sistema. Infatti, senza di esso l'hardware dello switch elettronico, dell'orologio RTC e la stessa Arduino non potrebbero collaborare con Raspberry Pi. Inoltre nel software è compresa anche quella parte da installare



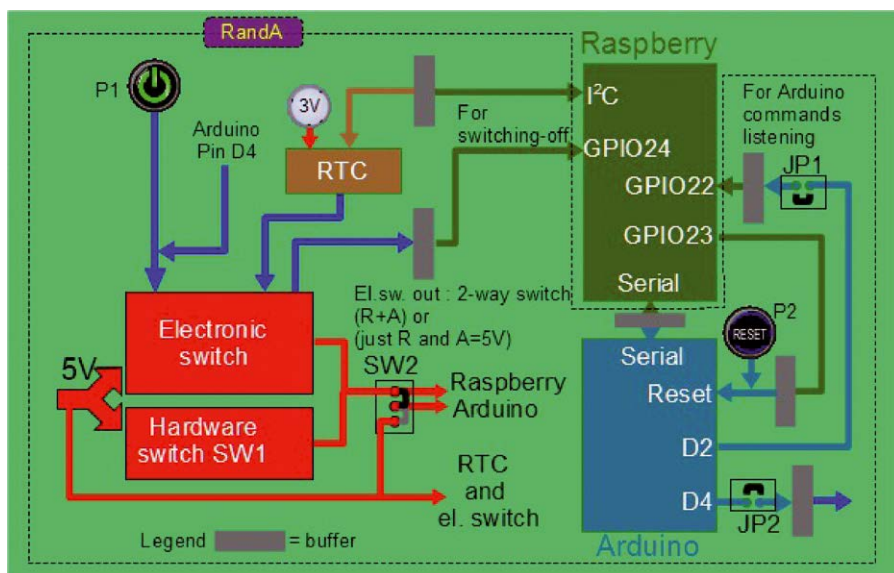


Fig. 1a - Schema a blocchi di RandA.

sul vostro PC per modificare l'IDE di Arduino in modo da utilizzare le porte remote e fare

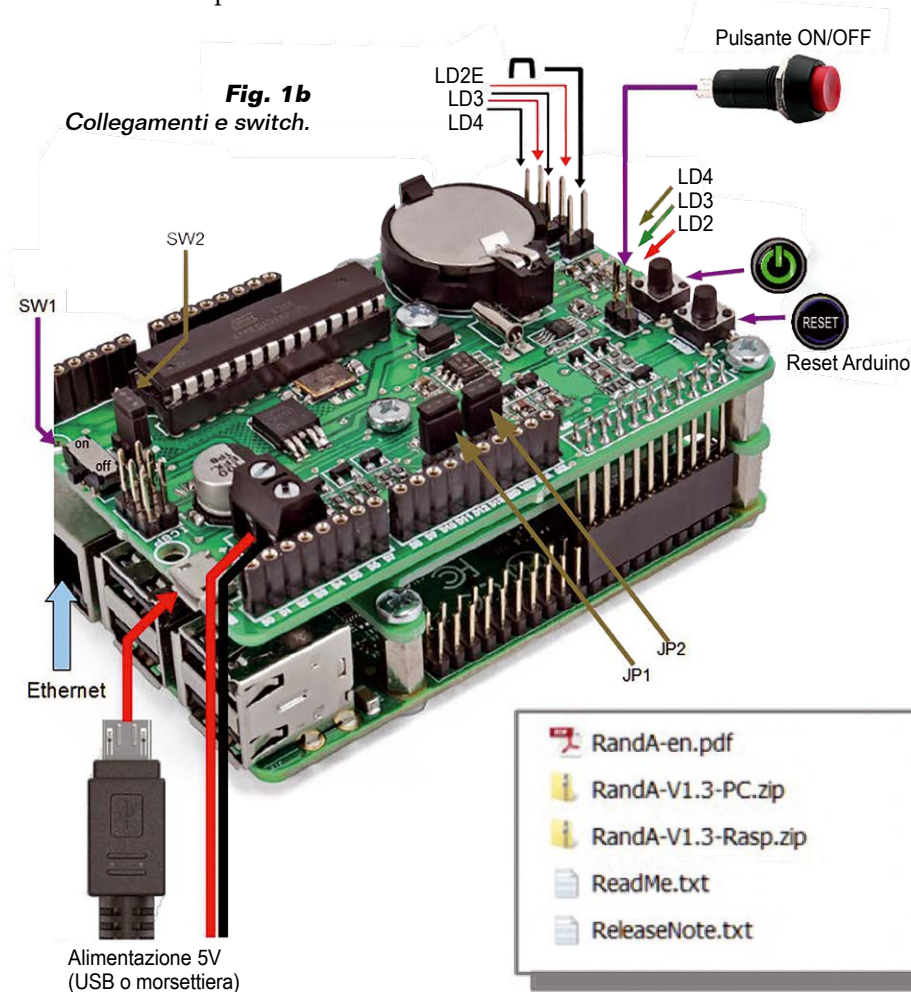


Fig. 1b

Collegamenti e switch.

l'uploading di sketch su RandA collegato in rete. Questa parte, in realtà dovete installarla anche se avete la SD-Card già configurata acquistabile su www.futurashop.it. In questo caso l'archivio con i file di installazione lo trovate nella directory `/home/pi/RandA`.

Il software è scaricabile dal nostro sito insieme ai file del progetto; è sotto la forma di archivio compresso (zip) e contiene a sua volta due archivi: uno per il vostro PC ed uno per la Raspberry Pi su cui è montata RandA. Inoltre contiene anche un manuale in Inglese (Fig. 2). La procedura da seguire per l'installazione del software è la seguente:

1. estraete l'archivio per PC, e da questo estraete a loro volta i file che andranno collocati nella cartella "lib" dell'IDE di Arduino (utilizzate solo la versione 1.0.5);
2. prima, per sicurezza, rinominate i tre file che sostituite. L'IDE di Arduino risulterà ora già modificato; completate l'operazione estraendo la libreria RAComm dalla cartella "libraries" e collocandola nella cartella "libraries" dell'IDE modificato;
3. estraete l'archivio "RandAinstall.tar.gz" ed il file "RandAinstall.sh" dall'archivio per Raspberry Pi.

A questo punto dovete operare su Raspberry Pi. Poiché le



Fig. 2 - Software di installazione "RandA-V1.3.zip".

installazioni Raspbian (ricordiamo che il software è pensato per questo ambiente operativo) sono configurate con il server SSH già attivo, possiamo collegarci a Raspberry Pi via LAN con un software che preveda questo protocollo; in questo articolo ipotizziamo che utilizziate MobaXterm, che contiene anche un X window server ed una comoda finestra FTP oltre alla console Linux. Per questa ragione, identificato l'indirizzo LAN di Arduino (possiamo identificarlo tramite il router o tramite delle utility che fanno lo scanning della rete come "Advanced Ip scanner"), apriamo una sessione SSH con esso (Fig. 3). Vediamo subito che accanto alla console c'è la finestra FTP. A questo punto non ci resta che fare il downloading dei due file su Raspberry Pi: li possiamo mettere nella cartella `/home/pi`. Prima di avviare lo script `RandAinstall.sh` bisogna renderlo eseguibile con il comando:

```
sudo chmod 777 RandAinstall.sh
```

Ora non rimane che lanciare lo script, mediante il comando (vale supponendo di essere nel percorso `/home/pi`):

```
./RandAinstall.sh
```

ed il gioco è fatto! Ricordiamo che, siccome alla fine lo script installa Arduino IDE per Linux e "codeblocks" dal sito dei software per Raspbian, dovete avere attiva la connessione Internet. Nel caso non aveste il collegamento Internet, potete installare "codeblocks" ed Arduino IDE in secondo tempo, con i comandi:

```
sudo get-apt install codeblock
sudo get-apt install arduino
```

Però, dopo l'installazione di Ar-

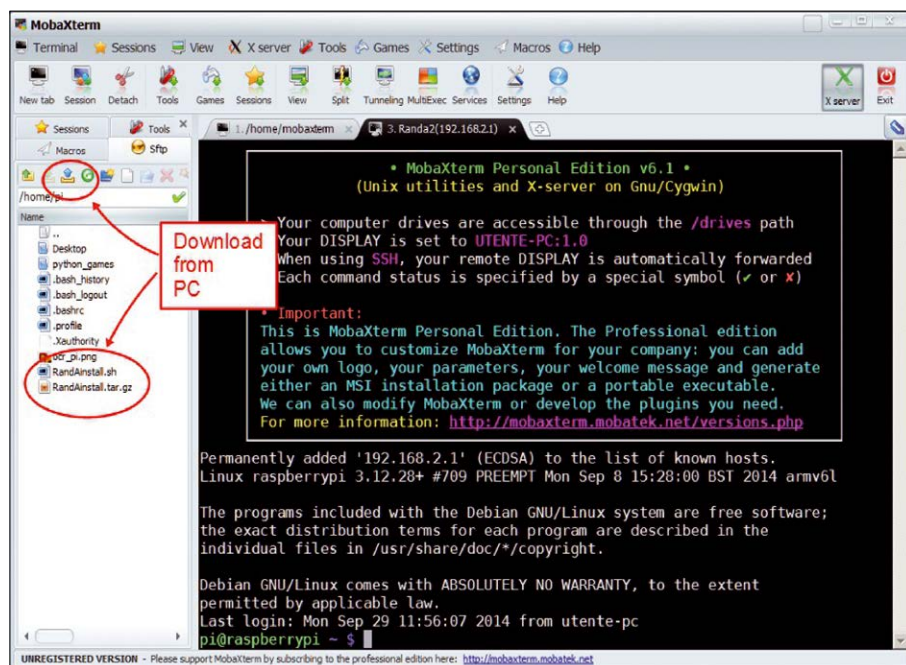


Fig. 3 - Console MobaXterm.

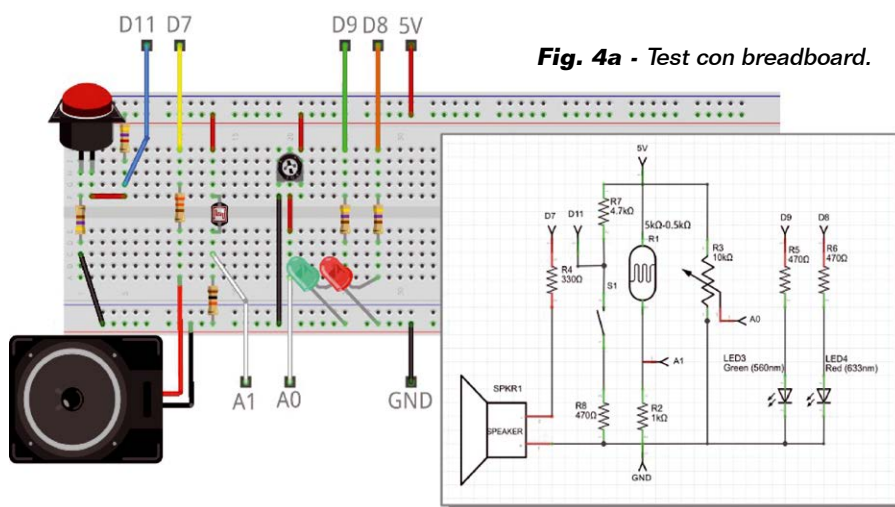


Fig. 4a - Test con breadboard.

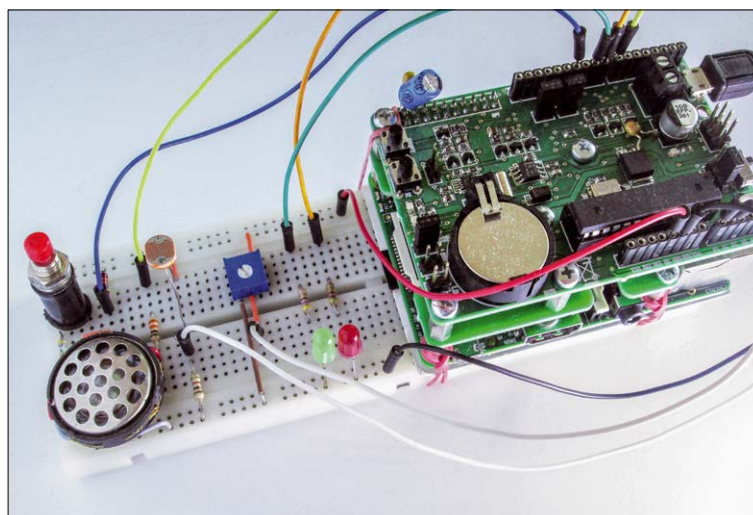


Fig. 4b
Foto
breadboard.

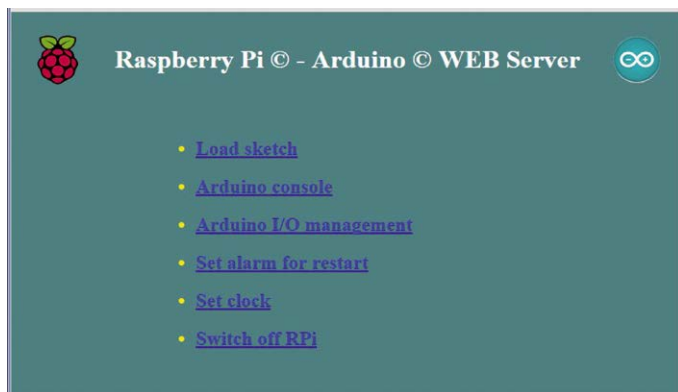


Fig. 5 - Menu generale di Arduino Web Server.

duino dovete modificare manualmente l'IDE anche su Raspberry Pi facendo riferimento alle tre righe che trovate nello script di installazione nella sezione relativa alla modifica di Arduino IDE.

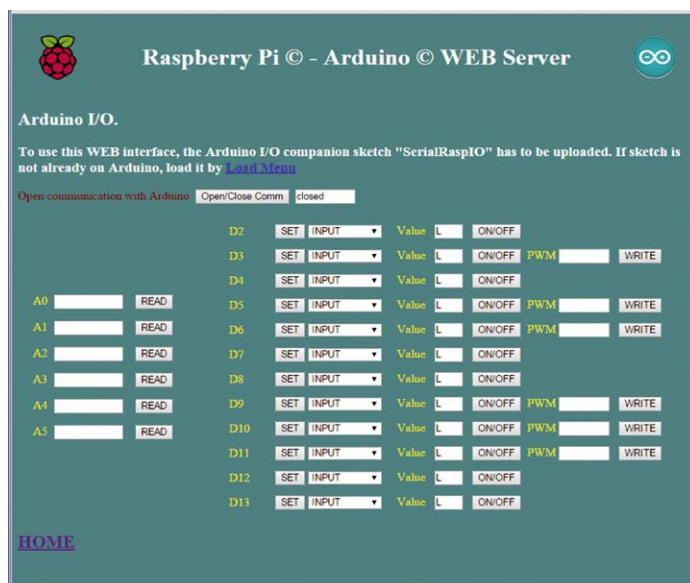
TEST DEL SISTEMA

Facciamo ora delle prove per verificare l'operatività di RandA e prendere confidenza con questo sistema. Innanzitutto effettuiamo il riavvio del sistema per rendere operativo il software; lo facciamo impartendo il comando:

```
"sudo reboot"
```

Il sistema si spegne (la sessione SSH si chiude) e si riavvia. Ora dovrete vedere il LED giallo spegnersi dopo un po': infatti esso rimane acceso per la durata dello startup e il suo spegnimento significa che il sistema è pronto ad operare. Apriamo di nuovo la sessione SSH e dovremmo vedere anche il nuovo messaggio di benvenuto, con l'elenco dei comandi specifici per RandA. Facendo il "refresh" del riquadro *sftp* (tasto destro del mouse), vedremo anche le nuove cartelle create in */home/pi*. Possiamo provare la collaborazione Raspberry-Arduino utilizzando una breadboard come in **Fig. 4**. Usiamo il comando "ArduIO -h" così da avere l'help. Ora possia-

Fig. 6
Schermata I/O.



mo provare ad accendere il LED rosso impartendo in sequenza i comandi:

```
ArduIO -set 8 out
ArduIO -wrd 8 1
```

Se non avete a disposizione la breadboard in **Fig. 4**, potete comunque testare l'I/O di Arduino utilizzando il LED standard (pin 13). Per cui dovete fare: `ArduIO -set 13 out` e `ArduIO -wrd 13 1`. ArduIO utilizza lo sketch *SerialRasp*. Se non lo trova su Arduino lo installa, ma va lanciato di nuovo. Lo sketch utilizzato si trova anche in forma di "sorgente" nella sotto-cartella "schetch4cmd" di */home/pi/bin*. Per spegnerlo, bisogna impartire il comando:

```
ArduIO -wrd 8 0
```

Analogamente, possiamo operare per il LED verde (pin 9); anche questo potremmo volerlo accendere a intensità variabile con il PWM (ma solo il 9); in questo caso il comando è (per avere $50/255 = 1/5$ dell'intensità):

```
ArduIO -wra 9 50
```

Leggiamo il valore della fotore-

sistenza su A1 (o del partitore di tensione su A0) con il comando:

```
ArduIO -rda 1
```

Mandiamo un'onda quadra alla frequenza di 600 Hz su D7 con il comando:

```
ArduIO -pou 7 600
```

Per bloccare l'onda, impartiamo:

```
ArduIO -pou 7 0
```

Naturalmente la potenza del suono è irrisoria, dato il pilotaggio diretto di Arduino; nel caso volessimo avere un suono più deciso, occorre un buffer od un collegamento ad un amplificatore. Infine possiamo leggere la durata di un impulso (dal valore 1 al valore 0) su D11, con il comando:

```
ArduIO -pin 11 0
```

Nel caso volessimo misurare la durata del valore 0, avremmo dovuto scrivere:

```
ArduIO -pui 11 1
```

Se usate il pulsante, ricordate che

Fig. 7 - Load sketch.

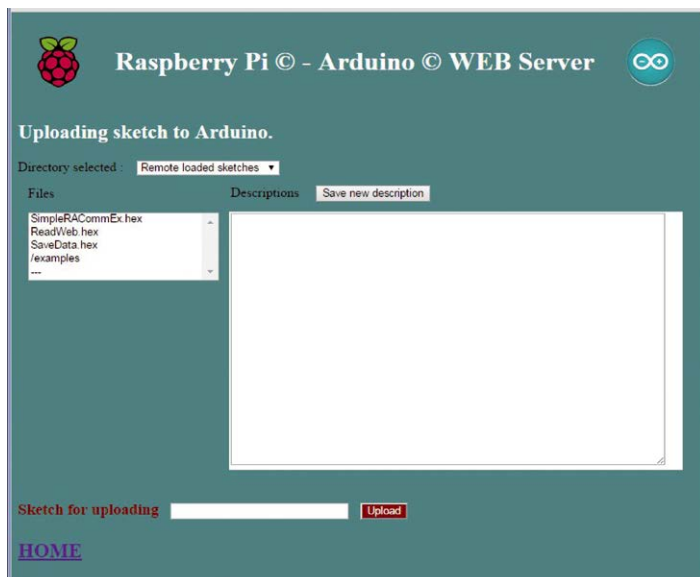
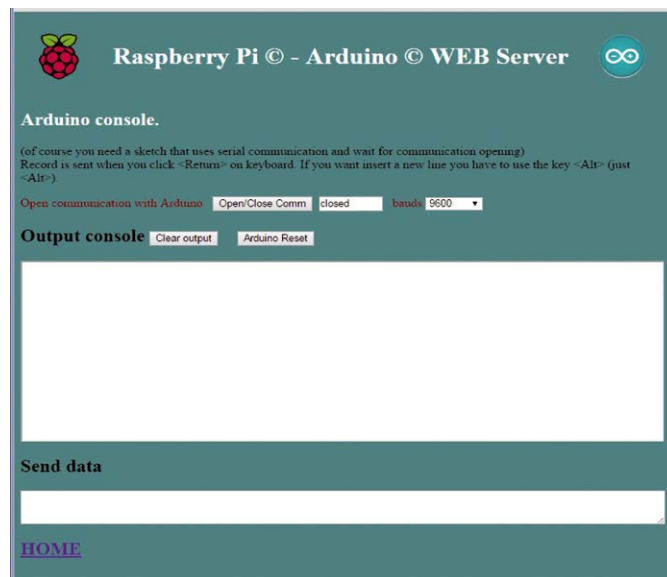


Fig. 8 - Arduino console.



il timeout è di un secondo. Infine possiamo usare il pulsante come input digitale:

```
ArduIO -rdd 11
```

In realtà potreste dialogare con Arduino utilizzando direttamente la porta seriale:

```
echo "WD13=1" > /dev/ttyS0
```

Questo comando accende il LED di Arduino (pin 13), se su Arduino risulta caricato lo sketch "SerialRasp" (quello utilizzato dal comando ArduIO); infatti corrisponde a quello che fa il comando "ArduIO -wrd 13 1". Il protocollo utilizzato da SerialRasp è molto semplice ed è descritto nel sorgente dello sketch.

WEB SERVER

Proviamo ora l'accesso tramite browser. Se operate in LAN, inserite l'indirizzo numerico di RandA, nel vostro browser preferito, e vi apparirà la pagina di partenza del server. Se avete appena acceso RandA, abbiate un po' di pazienza prima di accedere all'indirizzo; attendete una decina di secondi dopo che il LED giallo dello startup si è

spento, per dare modo al server web di inicializzarsi. Cliccate su "RPI&Arduino management" e vi apparirà il menu generale di Fig. 5. Potete provare ad utilizzare la stessa breadboard per testare il colloquio tramite web: allo scopo, cliccate su "Arduino IO management" e vi apparirà la schermata in Fig. 6.

Se avete già testato la breadboard con i comandi ArduIO, su Arduino è rimasto lo sketch che gestisce i pin, che è sostanzialmente lo stesso utilizzato da questa applicazione. Altrimenti dovete prima caricarlo utilizzando la pagina "Load sketch"; in questa schermata, la prima cosa da fare è aprire il colloquio tramite porta seriale (attendete finché non appare open).

Poi potete utilizzare i vari pulsanti per configurare e modificare o leggere i valori: ad esempio, per leggere il valore di illuminazione rilevato dalla fotoresistenza, cliccate sul bottone <READ> in corrispondenza di A1. Torniamo al menu e passiamo alla pagina "Load sketch" (Fig. 7): in essa potete vedere che ci sono due riquadri principali, uno con un elenco di sketch ed uno che contiene le descrizione che

voi potete inserire o modificare direttamente; in realtà gli elenchi sono due, commutabili tramite la selezione della lista a scorrimento. Infatti una cartella contiene gli sketch creati con l'IDE remoto ed uno quelli creati con l'IDE locale (su Raspberry Pi).

Selezioniamo lo sketch "Test-Serial.hex" dentro la cartella "/examples": ne apparirà la descrizione. Ora clicchiamo sul tasto <Upload>: dovrebbe apparire il messaggio di uploading effettuato con successo (se qualcosa non va per il verso giusto, fate un reset o fate di nuovo un upload). A questo punto non ci resta che utilizzarlo.

Ora torniamo al menu e selezioniamo la voce "Arduino console" (Fig. 8); per prima cosa verifichiamo che la velocità selezionata sia 9.600 baud e quindi apriamo la porta seriale. Attendiamo fino a che non risulta "open" e clicchiamo sul pulsante <Arduino Reset> per far ripartire lo sketch. Ora possiamo modificare il tempo di blinking inserendo un valore in millisecondi (>100) nel riquadro "Send data" e premendo invio. Ricordatevi di chiudere la seriale quando abbandonate la pagina. Poi potremmo andare alla pagina

“Set clock” e inserire il corretto valore per l’orologio. Lo possiamo prendere automaticamente da quello di sistema; infatti se Raspberry Pi è collegata in Internet, dovrebbe avere la data aggiornata. Quindi potremmo utilizzare la pagina “Set alarm for restart” per fissare un orario di riaccensione automatica, e quindi spegnere RandA tramite la pagina “Switch off RPi” che lancia uno shutdown e chiude l’alimentazione.

Randa DIVENTA UN ARDUINO POTENZIATO

Supponiamo che abbiate acquistato anche la scheda SD già configurata: allora l’installazione si riduce alla modifica dell’IDE Arduino che già utilizzate sul vostro PC. Questa operazione è stata descritta all’inizio dell’ar-

ticolo. L’archivio di installazione lo trovate come backup nella cartella `/home/pi/RandA`; dovete solo copiarvi l’archivio sul PC, per esempio utilizzando la finestra FTP della sessione MobaXterm, e procedere come già detto, ma solo per la parte che attiene al software per PC.

Una volta sostituiti i file, lanciate l’IDE di Arduino sul vostro PC. Potrà sembrare metterci un po’ di più di prima perchè deve scandagliare la rete per trovare porte remote RandA. Appena aperto, potete verificare che tra le porte disponibili ne compare anche una remota, contenente un indirizzo di rete, per esempio `//192.168.1.8/Arduino`: selezionatela.

Per fare una prova potete utilizzare il classico Blink oppure un

vostro sketch. Tenete presente che qualunque sketch voi carichiate su RandA, sarà comunque tenuto in copia nella cartella dedicata allo sviluppo con IDE remoto. Gli sketch saranno però solo in forma eseguibile (.hex), ma avranno lo stesso nome del sorgente.

Proviamo ad utilizzare lo sketch “TestSerial” che abbiamo già utilizzato con il Web sever. Il sorgente lo trovate nella cartella “examples-notRAComm” della libreria RAComm. In questa cartella abbiamo messo esempi da utilizzare con RandA che non fanno uso propriamente della libreria.

Se non è già stato caricato precedentemente, facciamo l’upload dello sketch attraverso la porta remota. Ora, per utilizzarlo ed interagire tramite la porta seriale, apriamo semplicemente la console dell’IDE: proprio come faremmo se avessimo Arduino collegato su USB.

Conviene comunque fare un reset, perchè è bene tener presente che ora non c’è più l’automatismo del reset alla apertura della console. Il reset potete farlo manualmente, oppure da remoto utilizzando il comando Linux “ResetRandA”, o, ancora, tramite il Web Server dalla pagina “Arduino console” (Fig. 9); in questo caso non c’è bisogno di aprire la connessione seriale.

Proviamo ad utilizzare un esempio con l’uso della libreria che dialoga con Raspberry Pi. Lo sketch più semplice contenuto negli esempi della libreria è “BasicRACommExample”. Prima di fare l’uploading conviene aprire una sessione MobaXterm (se non già aperta), perchè, usando la libreria, la console dell’IDE è inutilizzabile in quanto la porta seriale è gestita dal programma Linux che dialoga con Arduino

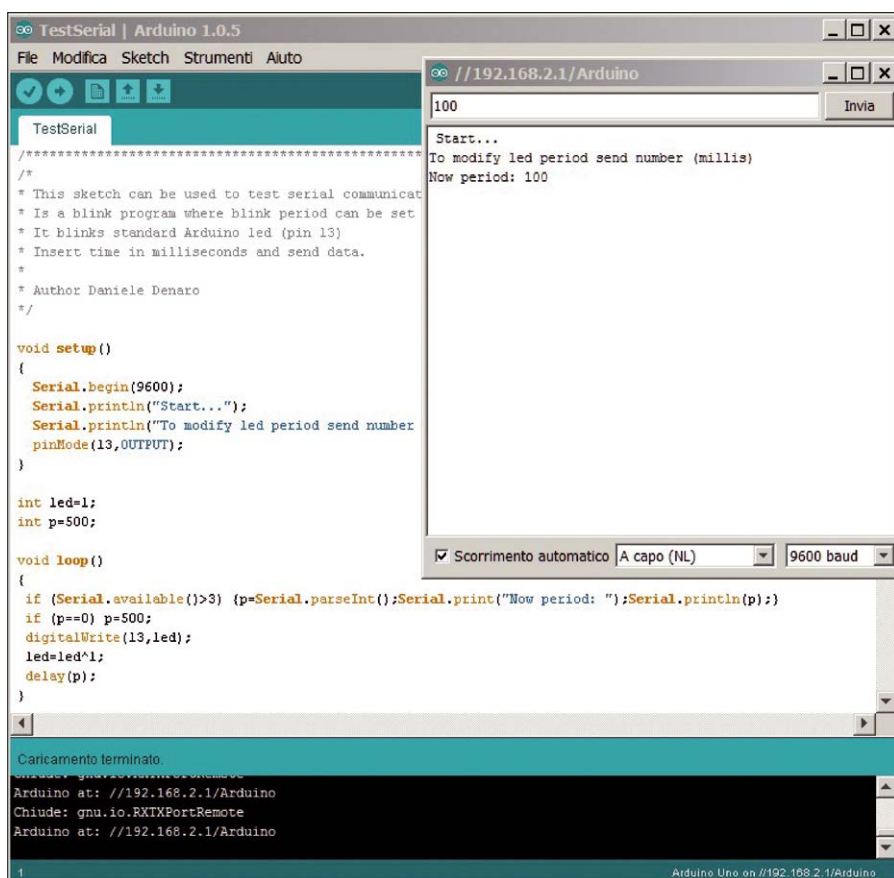


Fig. 9 - Lo sketch TestSerial da IDE Arduino.

per eseguire i suoi comandi. Per cui per visualizzare dei contenuti è necessario scriverli su un Xterminal aperto per l'occasione (Fig. 10).

Caricato lo sketch, questo dovrebbe aprire una finestra Xterminal e visualizzare il timestamp chiesto a Raspberry Pi. Dopo qualche secondo la finestra si chiude e lo sketch fa lampeggiare il LED tante volte quante sono le ore correnti. La lettura dell'orologio avviene ad ogni reset. Il reset può essere effettuato manualmente o per via remota tramite il comando "ResetRandA" o tramite Web Server.

Nella cartella degli esempi sono contenuti altri sketch più complessi: uno di essi salva su file alcuni dati di sensori analogici e digitali ogni ora, mentre un altro scarica da un sito web dati meteorologici e li salva su un file.

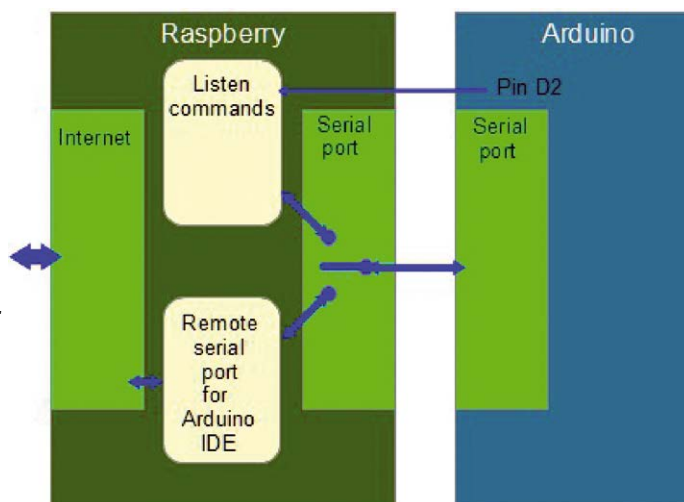
Infine nella cartella esempi c'è uno sketch che utilizza la modalità "Arduino sempre acceso". In questo sketch, il valore su un pin analogico decide se accendere Raspberry Pi -se non fosse già acceso- per poi scrivere su un file di log l'evento e spegnere di nuovo Raspberry Pi nel caso lo avesse trovato spento. Con questo sketch, bisogna quindi porre il deviatore SW2 sulla modalità "Arduino sempre acceso" ed avere il ponticello sul JP2 (che collega lo switch con il pin D4 di Arduino).

Per la sua sperimentazione potete utilizzare la breadboard con la fotoresistenza come illustrato all'inizio dell'articolo: vedrete che, oscurandola, Raspberry Pi viene accesa (se non lo è già) e parte la registrazione dell'evento.

PROGRAMMIAMO RandA

Se in ambito Arduino la programmazione avviene tramite il ben conosciuto ambiente IDE, in ambito

Fig. 10
Porta seriale
al servizio di
programmi diversi.



Raspberry Pi la scelta è più ampia: intanto potete utilizzare i comandi già predisposti e catalogati in `/home/pi/bin`. Ricordiamo, comunque, che per muovervi nel "file system" di Raspberry Pi, potete utilizzare il riquadro "Sftp" della sessione MobaXterm, o, meglio

ancora, lanciare l'interfaccia grafica del file manager "pcmanfm" che noi abbiamo rinominato "explorer".

Più in generale, per programmare, potete utilizzare il linguaggio di script che meglio conoscete oppure un completo linguaggio

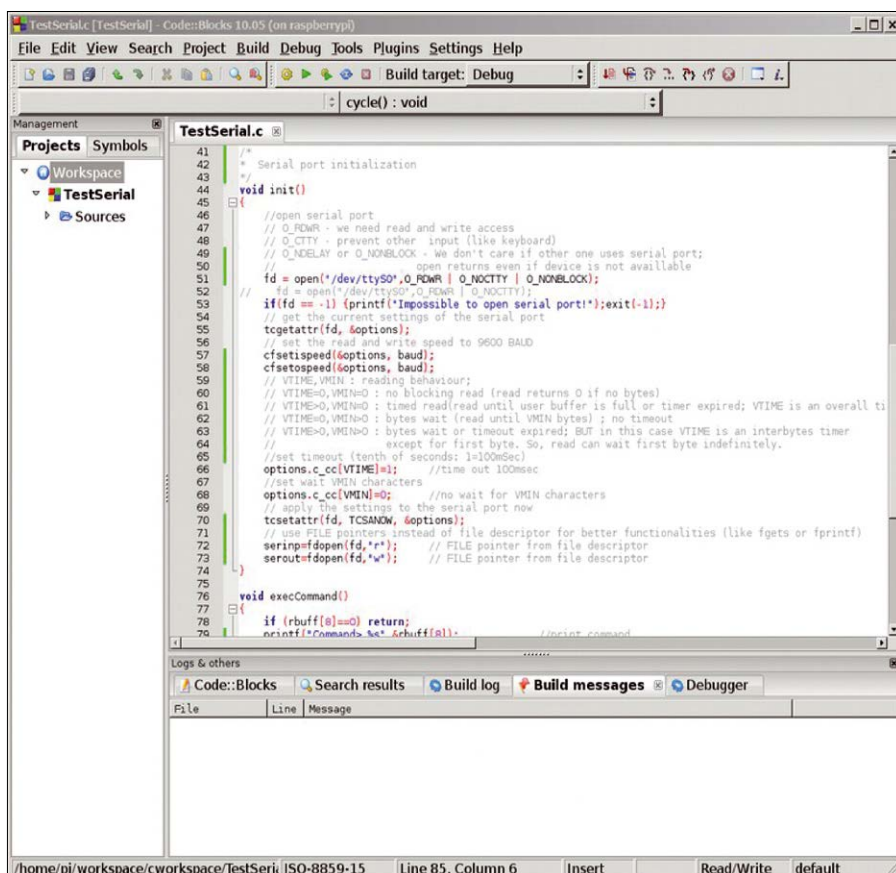
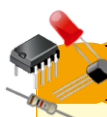


Fig. 11 - Codeblocks.



per il MATERIALE

Questo progetto può essere facilmente realizzato da chiunque abbia un minimo di esperienza nel montaggio manuale di componenti SMD e disponga dell'attrezzatura necessaria. La board RandA è comunque disponibile già montata e collaudata (comprese le minuterie) al prezzo di 39,00 Euro IVA compresa. È disponibile anche la board Raspberry PI Tip B+ (cod. 8111284RS) al prezzo di 36,00 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>

di programmazione come C++ (o Java). Trovate due esempi di script in `/home/pi/bin/examples`: uno di essi fa uso di bash e l'altro di Python. Proviamo ora a fare un programmino in C ed allo scopo lanciamo codeblocks. Tenete presente che la partenza potrebbe essere un po' lunga e soprattutto senza segnali di "work in progress"; inoltre state attenti che, se qualche finestra, che attende una convalida, fosse rimasta sotto le altre, codeblocks potrebbe non reagire. Alla prima partenza, codeblocks chiede qua-

le compilatore utilizzare tra quelli trovati: selezionate il primo, che è Gnu GCC compiler, come compilatore predefinito. Una volta aperto codeblocks, potete cercare il progetto esempio molto elementare che si trova in `/home/pi/workspace/cworkspace/TestSerial/TestSerial.cbp`. In codeblocks, i progetti si aprono facendo riferimento ai file ".cbp". La finestra codeblocks è costituita da diversi settori, tra cui quelli principali dell'elenco dei progetti aperti con la loro struttura e quello dell'editor dei sorgenti. L'output sulla console viene invece mostrato aprendo automaticamente una finestra Xterminal.

Il programma TestSerial.c (da non confondere con lo sketch Arduino TestSerial visto precedentemente) è il sorgente catalogato nel progetto ed è un esempio di come si apre una porta seriale per comunicare con Arduino. Il programma apre la porta seriale, legge i record che Arduino spedisce e li visualizza sulla console Linux; per visualizzarlo, basta selezionarlo nel riquadro del progetto con un doppio clic. Per utilizzare il programma bisogna avere uno sketch che spedisca record a Raspberry Pi.

Naturalmente codeblocks, come tutti gli ambienti di programma-

zione, permette di procedere in modalità "debug" passo passo o definendo dei "break-point". Tramite la voce "Debugging windows" è possibile aprire una finestra per "watches" (lettura dei contenuti delle variabili) ed altro. I commenti presenti permettono di descrivere la configurazione basilare della porta seriale così da porre le basi per un programma di collaborazione ed utilizzo di Arduino. Sul sito Internet di codeblocks (www.codeblocks.org) troverete il manuale ed altre informazioni. Nella cartella `/home/pi/workspace/cworkspace` sono contenuti anche tutti i programmi che hanno la loro versione eseguibile in `/home/pi/bin` (ma anche quelli utilizzati in /etc). Se poi volete integrare il Web Server con vostre pagine, le potete porre in `/home/apache-tomcat-7.0.47/webapps/ROOT`. Una web-application (file .war) invece la potete caricare tramite il "manager" di Tomcat (username e password: *tomcat*). Infine, se volete utilizzare la strada degli script CGI, potete collocarli in `/home/apache-tomcat-7.0.47/webapps/ROOT/WEB-INF/cgi`, ma ricordatevi di referenziarli come `"http://...../cgi-bin/...."`. Nella cartella trovate già un paio di script di esempio. ■

RM ELETTRONICA

Forniture per hobbisti, scuole e industria

Distributore autorizzato:

**FUTURA
ELETTRONICA**

**ELSE
Kit**

**NUOVA
ELETTRONICA**

Il tuo punto
di riferimento
per l'elettronica
a ROMA

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



Con Arduino Yún
connesso a Internet e un
visualizzatore a matrice di
LED, visualizziamo del testo
inviato da uno smartphone
tramite WiFi.

ARDU DISPLAY, QUANDO IL DISPLAY DIVENTA INTERATTIVO

..... di LUCA BELLAN

Alla fine degli anni '90 del secolo scorso i display a LED hanno iniziato a sostituire o ad integrare le classiche insegne dei cinema e delle attività commerciali, non solo per il minor consumo, ma anche per la possibilità, che offrono, di comporre scritte e immagini a piacimento e di sostituirle e cambiarle in base alle esigenze del momento;

a ciò va aggiunta la possibilità di far scorrere ciò che viene rappresentato (text-scrolling), in modo da attirare i possibili clienti per strada comunicando programmazione, orari di apertura, eventuali promozioni nei negozi, le news e qualsiasi altra informazione utile. Se alle origini i display promozionali e quelli che vediamo nelle aree pubbliche veniva-



Fig. 1 - Time Square (New York) dopo l'avvento dei display a LED.

no programmati localmente e magari riproducevano le scritte e le immagini inviate da un computer sul posto, oggi, nell'epoca dell'IoT (Internet of Things), è possibile governare questi visualizzatori da remoto, tramite WiFi o Internet, sostituendo testi e immagini o effetti senza dover essere sul posto.

Il progetto qui proposto -che abbiamo battezzato ArduDisplay- rientra in quest'ottica, perché permetterà alle persone di connettersi a una rete WiFi con il proprio dispositivo mobile (uno smartphone, per esempio), digitare un messaggio attraverso una pagina web e vederlo visualizzato su un display a

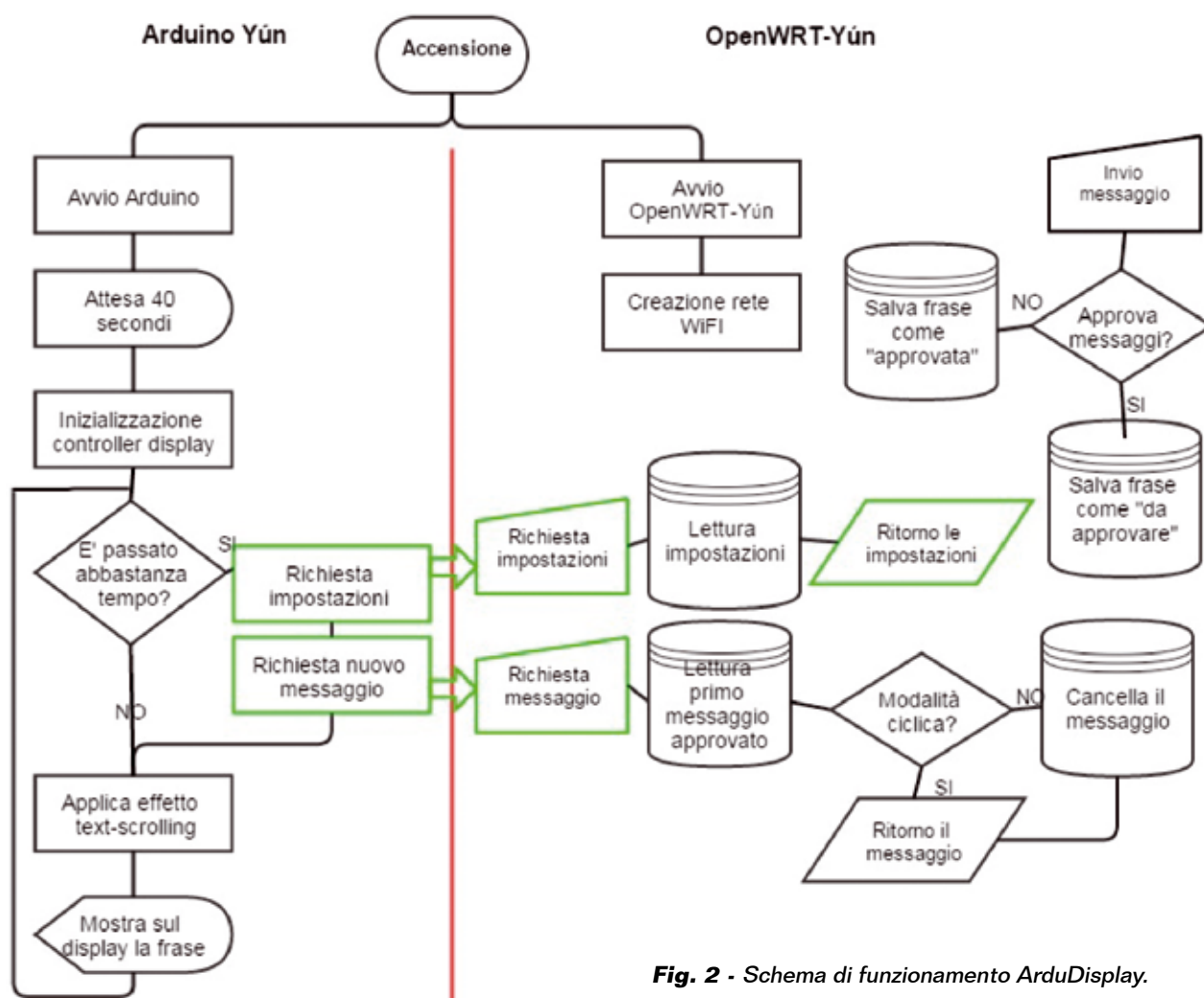


Fig. 2 - Schema di funzionamento ArduDisplay.

LED che potremo posizionare in uno stand in fiera, nel luogo di un concerto, in un locale aperto al pubblico, in un negozio ecc. Possiamo usare lo smartphone di fronte al nostro negozio o all'edificio dove è installato il display e usarlo come "telecomando" per scriverci i messaggi.

La nostra applicazione dispone di un pannello di controllo (sempre raggiungibile dalla Rete) che ci consentirà di approvare i messaggi ricevuti dal sistema remoto prima di mandarli sul display, impostare il tempo di visualizzazione di ogni singolo messaggio, la velocità di scorrimento del testo e di scegliere un messaggio predefinito da visualizzare in mancanza di messaggi ricevuti dal sistema.

È anche possibile decidere di non filtrare i messaggi, ossia inviarli alla coda di visualizzazione e visualizzarli senza doverli approvare prima. I messaggi ricevuti dal sistema verranno cancellati una volta mostrati sul display, ma potremo anche scegliere di visualizzarli ciclicamente: dal primo ricevuto fino all'ultimo per poi ricominciare dal primo.

IL NOSTRO SISTEMA

Per realizzare ArduDisplay avremo bisogno di:

- Arduino Yún;
- da uno a quattro display a LED Sure 3208 (dipende da quanto vogliamo sia lungo il nostro display);
- uno shield di connessione tra Arduino e i display, oppure dei jumper maschio-maschio, secondo lo schema di Fig. 5;
- una microSD (da 512 MB o più).

Abbiamo scelto di realizzare questo progetto utilizzando Arduino Yún perchè questa scheda è una soluzione completa per inter-

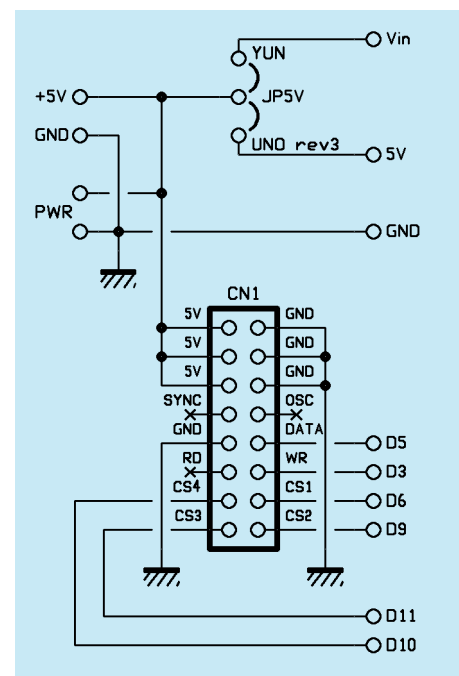
facciare al web i propri circuiti con la semplicità che contraddistingue Arduino. La connettività Internet si ottiene tramite WiFi, grazie al modulo integrato nella Yún e alla distribuzione Linux chiamata OpenWrt-Yun (tratta dalla più famosa OpenWrt) residente su microSD, che gira sul processore a 400 MHz dedicato, provvisto di 64 MB di RAM.

Questa dotazione permette di trasformare il nostro Arduino in un web server equipaggiato con PHP5 e database MySQL.

Il codice sorgente di Arduino si occuperà quindi di chiedere i messaggi a PHP (tramite la libreria Bridge) e visualizzarli sul display, creando anche l'effetto di text-scrolling, mentre il Web Server consentirà agli utenti di inviare messaggi e a noi di gestirli (approvarli o scartarli).

I messaggi e le impostazioni (ad esempio il tempo di permanenza della frase) saranno salvati in un database MySQL.

I file PHP e le pagine web sa-



ranno memorizzati sulla scheda microSD, mentre la rete WiFi cui si conatteranno gli utenti sarà creata direttamente da Arduino. Il display a matrice di LED è prodotto dalla Sure Electronics, ed è il modello P432X8LEDMATRIX (acquistabile da Futura Elettronica, www.futurashop.it): dispone di quattro matrici 8x8 formate ognuna da 64 LED rossi da 3 mm a intensità regolabile ed è basato sul controller HT1632C della Holtek.

Potremo collegare fino a quattro display in cascata grazie al cavo IDC a 16 pin previsto allo scopo; sono inoltre presenti due morsetti con tensione di uscita 5 V, nel caso in cui la corrente erogata dal nostro Arduino non bastasse ad alimentare tutti i display. L'interfaccia di comunicazione è a bus seriale.

Il nostro progetto vedrà impiegati, per fini di semplicità, due soli di questi display, tuttavia cambiando le impostazioni nella parte iniziale dello sketch, potete tranquillamente realizzare sistemi più estesi, fino a quattro moduli.

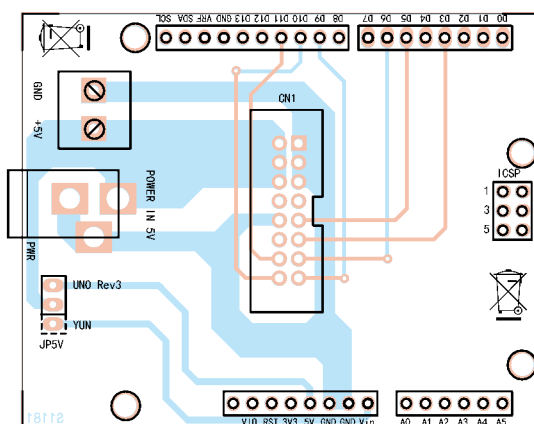
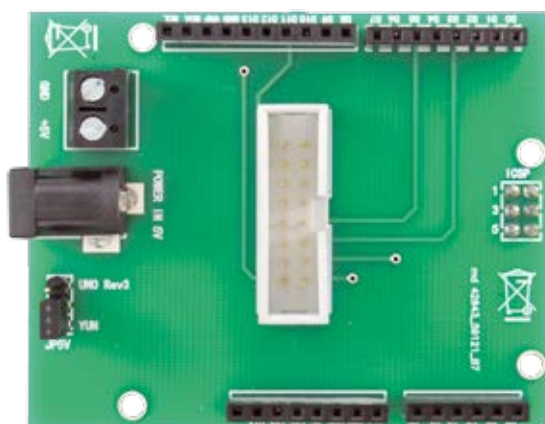
Ogni display, con tutti i LED accesi alla massima luminosità,



Fig. 3
Struttura
delle cartelle
e dei file
all'interno
della
microSD.

Elenco Componenti:

- Connettore vaschetta 16 poli da CS
- Strip maschio 3 poli
- Strip maschio/femmina 6 vie
- Strip maschio/femmina 8 vie (2 pz.)
- Strip maschio/femmina 3x2 vie
- Strip maschio/femmina 10 vie
- Plug alimentazione
- Morsetto 2 poli
- Jumper
- Circuito stampato



richiede una corrente di 0,36 A; siccome utilizzeremo la potenza predefinita (all'incirca la metà) e non terremo tutti i LED accesi per mostrare del testo, potremo alimentare il tutto con i pin 5V e GND di Arduino (alimentato a sua volta dal connettore Micro-USB).

Come abbiamo detto in precedenza, la microSD ospiterà i file PHP e le pagine web che consentiranno agli utenti di inviare i messaggi, a noi di gestirli e ad Arduino di prelevarli per poi visualizzarli sul display. Il collegamento tra il primo display della serie e Arduino Yún verrà effettuato con uno shield che permetterà di inserire direttamente il cavo IDC su di essa: trovate schema elettrico e piano di montaggio in queste pagine.

Ospitando solo connettori e un jumper, il montaggio dello shield è semplice al punto che non servono particolari indicazioni. Una volta che l'avrete realizzato, chiudete il ponticello JP5V tra la piazzola centrale e la Yún.

CONFIGURAZIONE DI ARDUINO YÚN

Vediamo come preparare la nostra piattaforma in modo da ottenere come risultato finale un Arduino Web Server capace di gestire pagine PHP, memorizzare i dati in un database MySQL e creare una rete WiFi stand-alone che si chiamerà "ArduDisplay" e non richiederà l'inserimento di una password per accedervi. Inoltre, quando ci conatteremo alla rete "ArduDisplay" e digiteremo l'indirizzo "http://scrivi.

mi" nel browser, verremo reindirizzati alla pagina per l'invio dei messaggi al display.

Come prima operazione scarichiamo dal nostro sito web www.elettronica.in.it l'archivio ZIP contenente lo sketch di Arduino, le librerie necessarie al funzionamento del display e i file da posizionare sulla microSD. Una volta scaricato e decompresso l'archivio ZIP otterremo, oltre allo sketch di Arduino e alla cartella HT1632, una cartella di nome "arduino" che dovrà essere posizionata nella root della nostra microSD: la cartella contiene l'installazione del database di ArduDisplay e tutti i file PHP che analizzeremo più avanti. Poi inseriamo la microSD nella slot dell'Arduino.

A meno che il vostro Arduino Yún sia nuovo, vi consigliamo di resettare completamente la distribuzione OpenWrt ospitata al suo interno: verranno così cancellate eventuali installazioni o configurazioni che potrebbero interferire con il nostro progetto.

Per resettare la OpenWrt-Yún occorre premere il pulsante di reset WiFi (situato a fianco della USB-A) per almeno 30 secondi: Arduino ripristinerà tutte le impostazioni originali, come se fosse stato appena acquistato; questa operazione rimuoverà, tra le altre cose, tutti i file installati e le impostazioni di rete. Tale procedura vi sarà utile se, nel realizzare l'applicazione, qualcosa andrà male.

Dopo aver effettuato il reset, o dopo aver acceso il vostro nuovo Arduino Yún, noterete una rete WiFi di nome "Arduino Yún-XXXXXXXXXXXX" (dove le X rappresentano dei caratteri alfanumerici che variano per ogni Arduino): connettiamoci e digitiamo "http://arduino.local" oppure "http://192.168.240.1"

per accedere al pannello di controllo, inseriamo la password (quella predefinita è "arduino"), facciamo clic su "Configure" e scegliamo una rete WiFi preesistente cui connetterci, perché avremo bisogno di un accesso a Internet; digitiamo la password, nel caso in cui sia richiesta, e facciamo clic sul pulsante "Configure & Restart".

Il nostro Arduino si riavvierà connettendosi alla rete WiFi da noi impostata; a questo punto dobbiamo trovare l'IP che gli è stato assegnato dal nostro router (ad esempio: 192.168.0.6) perché ci servirà per lanciare i comandi tramite SSH nel prossimo step. L'indirizzo IP è consultabile dalla pagina di configurazione del nostro router (solitamente accessibile dal browser con l'indirizzo 192.168.0.1 oppure 192.168.1.1) nella sezione "Dispositivi connessi".

I prossimi step saranno eseguiti tutti da riga di comando; dovremo quindi collegarci in SSH al nostro Arduino utilizzando PuTTY (scaricabile gratuitamente da <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>). Dopo averlo scaricato e avviato, inseriamo nel campo "Host Name" l'indirizzo IP di Arduino e facciamo clic sul pulsante "Open"; eseguiamo quindi l'accesso con l'utente "root" e la password che abbiamo impostato al nostro Arduino (la stessa che utilizziamo per connetterci al pannello di controllo). Per installare PHP5 lanciamo i seguenti due comandi in successione:

```
opkg update
opkg install php5 php5-cgi
```

Editiamo il file di configurazione di uHTTPd (il web server integrato in OpenWrt-Yún) per includere PHP:

```
nano /etc/config/uhttpd
```

Decommentiamo la riga:
list interpreter ".php=/usr/bin/php-cgi"

Aggiungiamo la seguente riga per fare riconoscere a uHTTPd i file con estensione ".php":

```
option index_page "index.php"
```

Chiudiamo e salviamo il file (CTRL+X, Y e Invio). Infine riavviamo il Web Server uHTTPd per fargli leggere le nuove impostazioni sul file di configurazione:

```
/etc/init.d/uhttpd restart
```

Una volta installato PHP5 e configurato uHTTPd, possiamo passare a installare MySQL lanciando il seguente comando:

```
opkg install libpthread libncurses
libreadline mysql-server
```

Apriamo il file di configurazione di MySQL:

```
nano /etc/my.cnf
```

Popoliamo le variabili "datadir" e "tmpdir" con i seguenti valori:

```
datadir      = /srv/mysql
tmpdir       = /tmp
```

Chiudiamo il file salvando (CTRL+X, Y e Invio) e lanciamo i seguenti comandi in sequenza.

```
mkdir -p /srv/mysql
mysql_install_db --force
/etc/init.d/mysqld start
/etc/init.d/mysqld enable
mysqladmin -u root password 'admin'
```

In questo modo avremo creato un MySQL server, associandogli l'utente "root" e la password "admin"; ora dovremo installare

il modulo MySQL per PHP; per farlo lanciamo questo comando:

```
opkg install php5-mod-mysql
```

Apriamo adesso il file di configurazione di PHP, con il comando:

```
nano /etc/php.ini
```

e decommentiamo la seguente riga:

```
extension=mysql.so
```

Verifichiamo inoltre che i parametri del seguente blocco di codice siano compilati così:

```
[MySQL]
mysql.allow_local_infile = On
mysql.allow_persistent = On
mysql.cache_size = 2000
mysql.max_persistent = -1
mysql.max_links = -1
mysql.default_port = 3306
mysql.default_socket = /tmp/run/
mysqld.sock
mysql.default_host = 127.0.0.1
mysql.default_user = root
mysql.default_password = admin
mysql.connect_timeout = 60
mysql.trace_mode = Off
```

Salviamo e chiudiamo (CTRL+X, Y e Invio).

Dopo aver creato il server MySQL dobbiamo popolarlo con il nostro database e le nostre

```
mysql> use ardudisplay;
Database changed
mysql> show tables;
+-----+
| Tables_in_ardudisplay |
+-----+
| frasi                  |
| impostazioni           |
+-----+
2 rows in set (0.00 sec)
```

Fig. 4 - Tabelle presenti nel database ArduDisplay.

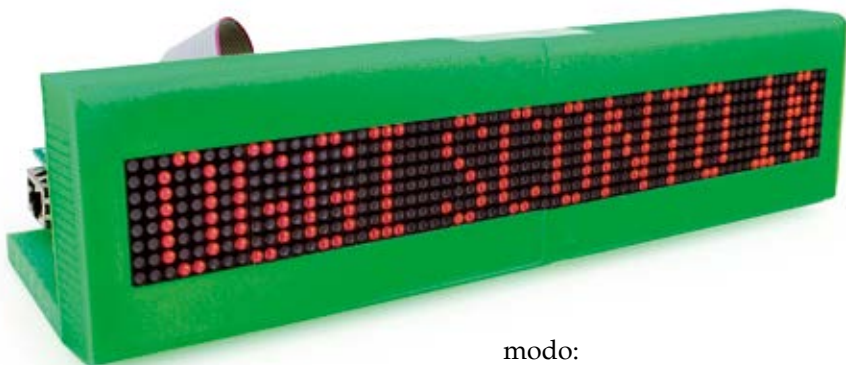


tabelle. Spostiamoci quindi nella cartella dove troveremo l'installazione del nostro database:

```
cd /www/sd/
```

Apriamo la shell MySQL con il comando:

```
mysql -u root -p
```

e inseriamo la password del database. Successivamente lanciamo il seguente comando SQL per creare il database:

```
source ardudisplay.sql;
```

Per verificare che tutto sia andato a buon fine lanciamo questi due comandi:

```
use ardudisplay;
show tables;
```

Dovremmo vedere comparire la tabella "frasi" e la tabella "impostazioni" come mostrato nella Fig. 4.

Usciamo quindi dalla shell di MySQL:

```
exit
```

Apriamo il file di configurazione che verrà utilizzato dai nostri file PHP per connettersi al database:

```
nano Common/db_class.php
```

Assicuriamoci che esso contenga i parametri compilati in questo

modo:

```
$this->mysql_server = "localhost";
$this->mysql_username = "root";
$this->mysql_pass = "admin";
$this->database_name = "ardudisplay";
```

Salviamo e chiudiamo (CTRL+X, Y e Invio).

Ora che abbiamo trasformato il nostro Arduino in un web server con PHP e database MySQL, dobbiamo fare in modo che, quando l'utente digiti l'indirizzo di Arduino sul browser, non venga reindirizzato al pannello di controllo, bensì alla pagina per inviare i messaggi; per fare questo spostiamoci nella cartella dove risiedono i file del web server con questo comando:

```
cd /www/
```

Rinominiamo il file che ci reindirizza al pannello di controllo con questi due comandi:

```
cp index.html OLDindex.html
rm index.html
```

Creiamo il file PHP che effettuerà il redirect corretto:

```
nano index.php
```

Scriviamo al suo interno il seguente codice:

```
<?php
header("Location: sd/index.php ");
?>
```

Salviamo e chiudiamo (CTRL+X, Y e Invio).

Dopo aver svuotato la cache del

browser, provando ad inserire l'indirizzo IP di Arduino verremo reindirizzati alla pagina di invio messaggi.

Arrivati a questo punto, dobbiamo fare in modo che Arduino crei la sua rete WiFi stand-alone, quindi teniamo premuto il tasto di reset WiFi per almeno 5 secondi (ma meno di 30, altrimenti perderemo tutto il lavoro fatto fino ad ora): verrà così ricreata la rete iniziale "Arduino Yún-XXXXXXXXXXXX". Connettiamoci a questa rete e andiamo sul pannello di controllo (<http://192.168.240.1/cgi-bin/luci/webpanel/homepage>), facciamo poi clic su "advanced configuration panel (luci)", utilizzando il menu principale spostiamoci nella sezione "Network" e successivamente nella sottosezione "Hostnames".

In questa sezione aggiungeremo una regola che trasformerà il nostro indirizzo IP in "http://scrivi.mi": facciamo clic sul pulsante Add, inseriamo "scrivi.mi" nel campo Hostname e selezioniamo "--custom --" nel campo IP address, per poi scrivere "192.168.240.1"; infine facciamo clic su "Save & Apply".

A questo punto gli indirizzi che avremo saranno i seguenti:

- <http://scrivi.mi> per inviare un messaggio al display;
- <http://scrivi.mi/sd/admin> per gestire i messaggi ricevuti e altre impostazioni (questa sezione verrà analizzata più avanti);
- <http://scrivi.mi/cgi-bin/luci/webpanel/homepage> per connettersi al pannello di controllo di Arduino.

L'ultima operazione che ci rimane da effettuare è il cambio dell'SSID della rete WiFi di Arduino: connettiamoci al pannello di controllo avanzato (luci) utilizzando il nuovo indirizzo, scorriamo la

pagina fino alla sezione "Wireless" e facciamo clic sul nome della nostra rete.

Digitiamo quindi "ArduDisplay" nel campo ESSID e facciamo clic su "Save & Apply", attendiamo infine il riavvio di Arduino per poi connetterci alla rete "ArduDisplay".

ANALISI DEI FILE PHP

Passiamo ora ad analizzare le funzioni dei singoli file PHP che abbiamo posizionato sulla scheda microSD.

Nella cartella "www", che sarà la cartella principale del nostro sito, troviamo il file index.php: questo file verrà caricato subito dopo aver digitato "<http://scrivi.mi>" e contiene la casella di testo per inviare un messaggio al dispositivo. Il messaggio sarà ricevuto dal file `invia_frase.php` che si occuperà di scriverlo nel database con uno stato 0 (da approvare) nel caso in cui avessimo impostato "Approva messaggi" a "Sì" nel pannello di controllo, altrimenti 1 (approvato), ciò significa che il nostro messaggio verrà direttamente posto nella coda di visualizzazione del display senza il bisogno di essere approvato.

Troviamo poi il file `ritorna_frase.php`, che si occuperà di restituire una nuova frase ogni volta che verrà interpellato da Arduino tramite l'oggetto `Process`; nel caso in cui la modalità "Ciclo frasi" sia disattivata, viene estratta la prima frase in coda di visualizzazione (con stato 1), altrimenti verrà comunicata la frase predefinita che potrà essere cambiata in ogni momento dal pannello di controllo. Ogni volta che sarà estratta una frase da mostrare, questa verrà immediatamente cancellata dal database.

Se invece è attiva la modalità "Ciclo frasi", la frase estratta non verrà cancellata, poichè in questa

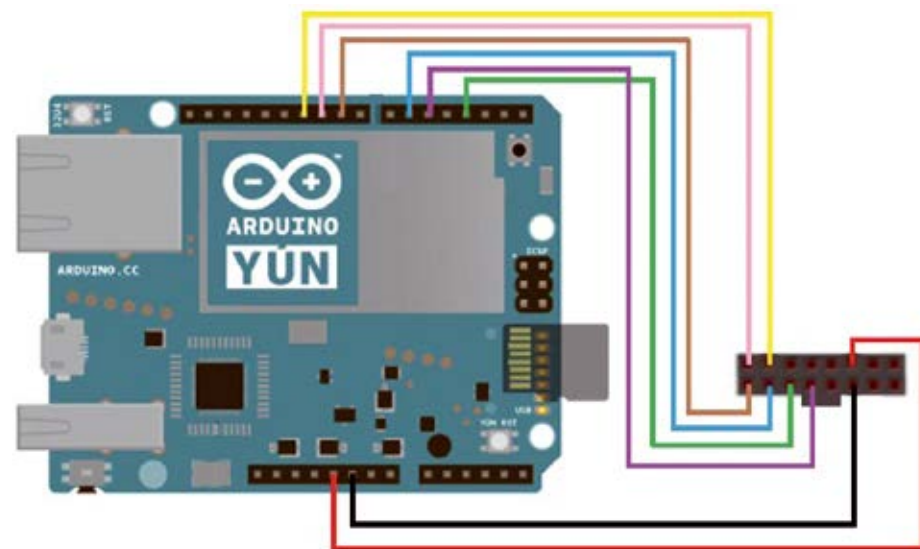


Fig. 5 - Collegamento tra il primo display e Arduino mediante jumper: seguitelo se intendete risparmiarvi la costruzione dello shield.

modalità verranno visualizzate sul display tutte le frasi ricevute e, arrivati all'ultima, il file PHP ricomincerà ad estrarle dalla prima, includendo la frase predefinita.

Il file `ritorna_impostazioni.php` viene richiamato da Arduino assieme a `ritorna_frase.php` e si occupa di trasmettere ad Arduino la durata (in secondi) di una frase sul display e la velocità di scroll (anch'esse impostabili dal pannello di controllo che vedremo più avanti).

Spostandoci nella cartella "admin", troviamo il file `index.php` che contiene una semplice casella di testo per effettuare l'accesso al pannello di controllo (tramite il file `login.php`); la password predefinita per accedere al pannello di controllo è "admin", che si potrà modificare una volta effettuato il login.

La pagina `admin.php` ci permette di effettuare molteplici operazioni: nel caso in cui sia attiva la modalità "Approva frasi" nel box di sinistra vedremo i messaggi da approvare, nel box centrale i messaggi in coda di visualizzazione, mentre nel box di destra possiamo modificare alcune

impostazioni di ArduDisplay:

- **Password admin:** è la password per accedere al pannello di controllo;
- **Durata frase display:** è il tempo di permanenza (in secondi) di una frase sul display a LED;
- **Velocità scroll:** è la velocità dell'effetto text-scrolling;
- **Approva frasi:** se impostata a "Sì", le frasi dovranno essere approvate prima di andare in coda di visualizzazione;
- **Ciclo frasi:** le frasi, una volta inviate al display, non vengono cancellate, ma vengono conservate poichè una volta arrivato a mostrare l'ultima, Arduino ricomincerà da capo;
- **Frase default:** è la frase che verrà mostrata sul display nel caso non ci fossero messaggi da mostrare.

Facendo clic sul pulsante "Logout" verrà richiamato il file `logout.php` che ci permetterà di uscire dal pannello di controllo di ArduDisplay.

La cartella "Common" invece, contiene la libreria javascript che ci permette di effettuare le chiamate ai file PHP (jQuery), il foglio di stile CSS, le immagini

Listato 1

```
#include <Process.h>
#include <HT1632.h>
#include <font_8x4.h>
byte numero_display=2;
byte wr =3;
byte data = 5;
byte cs1 = 6;
byte cs2 = 9;
byte cs3 = 0;
byte cs4 = 0;
int larghezzaTesto;
unsigned char* fraseCorrente =(unsigned char*) "";
unsigned char* fraseVecchia =(unsigned char*) "";
int translazione=0;
byte tempoDisplay;
int tempoDelay;
String id_frase="-1";
int iterazioni;
int cont=0;
void setup() {
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);
  Bridge.begin();
  digitalWrite(13, LOW);
  switch(numero_display){
    case 1:
      HT1632.begin(cs1,wr,data);
      break;
    case 2:
      HT1632.begin(cs1,cs2,wr,data);
      break;
    case 3:
      HT1632.begin(cs1,cs2,cs3,wr,data);
      break;
    case 4:
      HT1632.begin(cs1,cs2,cs3,cs4,wr,data);
      break;
    default:
      break;
  }
  for(int i=0;i<numero_display;i++){
    HT1632.renderTarget(i);
    HT1632.clear();
    HT1632.render();
  }
  delay(40000);
}

void loop() {
  if(cont==0){
    for(int i=0;i<numero_display;i++){
      HT1632.renderTarget(i);
      HT1632.clear();
      HT1632.render();
    }
    caricaImpostazioni();
    cambiaFrase();
  }
  controllaDisplay();
  cont=(cont+1)%iterazioni;
}

void controllaDisplay(){
  for(int i=0;i<numero_display;i++){
    HT1632.renderTarget(i);
    HT1632.clear();
    HT1632.drawText(fraseCorrente, OUT_SIZE * (numero_display-i)-translazione, 0, FONT_8X4, FONT_8X4_END, FONT_8X4_HEIGHT);
    HT1632.render();
  }
  translazione = (translazione+1)%(larghezzaTesto + OUT_SIZE * numero_display);
  delay(tempoDelay);
}

void caricaImpostazioni(){
```

dei pulsanti per approvare o cancellare un messaggio ricevuto e il file `db_class.php` che abbiamo analizzato nel precedente paragrafo e contiene la classe PHP e i dati per connetterci al database MySQL.

LO SKETCH PER ARDUINO

Nell'archivio ZIP che avete scaricato troverete anche un file denominato "ArduDisplay_v1.ino" e una cartella chiamata "HT1632". Il display da noi utilizzato richiede una libreria, rappresentata dalla cartella sopracitata. Per poterla utilizzare è necessario copiare l'intera cartella nella directory libraries di Arduino (solitamente situata in "Documents\Arduino\libraries"). Ora possiamo connettere il nostro Arduino al computer con il cavo USB e aprire il file con estensione ".ino" in Arduino IDE. Controllate che la porta COM e il modello di Arduino indicati dall'IDE siano corretti; se è così, potete procedere a compilare e caricare il codice.

Passiamo ora ad analizzare il codice presente nel file facendo riferimento al **Listato 1**.

Per prima cosa vengono incluse le librerie: la prima gestisce la comunicazione tra i due processori presenti sulla scheda, mentre le altre due sono necessarie per il funzionamento del display (la terza, in particolare, associa ad ogni carattere la sua rappresentazione grafica).

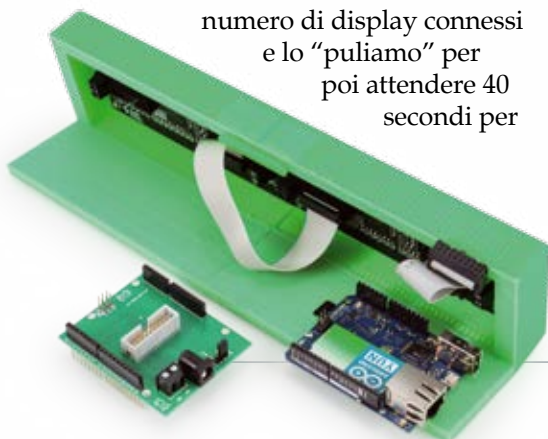
Troviamo poi il numero dei display e la dichiarazione dei pin necessari per il suo controllo, seguita dalla dichiarazione delle variabili:

- *larghezzaTesto* è la larghezza in pixel della frase da mostrare;
- *fraseCorrente* e *fraseVecchia* indicano rispettivamente la frase appena ricevuta e quella precedente, questo è utilizza-

to per decidere se resettare o meno la posizione del testo al cambio della scritta (nel caso in cui la frase non cambi, la frase non viene spostata di nuovo all'origine, dando così un senso di continuità);

- *traslazione* è il numero di pixel di cui è attualmente tralata la frase rispetto all'origine;
- *tempoDisplay* contiene la durata della frase in secondi;
- *tempoDelay* indica il tempo in millisecondi da aspettare tra un'esecuzione del loop e l'altra (un tempo minore implica una maggiore velocità di scorrimento);
- *id_frase* è l'id del database della frase mostrata, ed è necessario per implementare la funzionalità "ciclo frasi";
- *iterazioni* indica il numero di volte che il ciclo loop deve essere eseguito prima di cambiare una frase; esso dipende da *tempoDisplay* e *tempoDelay* (per esempio, una frase dev'essere mostrata per 10 secondi con un tempo tra un ciclo e l'altro di 100 millisecondi, questa durerà ovvero 100 cicli);
- *cont* tiene traccia di quante iterazioni sono state effettivamente eseguite.

Nel setup accendiamo un LED che sarà spento non appena il bridge tra i due processori di cui è dotata la scheda sarà stato stabilito; poi inizializziamo lo schermo in maniera diversa a seconda del numero di display connessi e lo "puliamo" per poi attendere 40 secondi per



Listato 1

```
Process p;
String temp;
p.begin("/usr/bin/php-cgi");
p.addParameter("-q");
p.addParameter("/mnt/sdal/arduino/www/ritorna_impostazioni.php");
p.run();
while (p.available()>0) {
    char c = p.read();
    if(c!='|'){
        temp += c;
    } else {
        tempoDisplay=temp.toInt();
        temp="";
    }
}
tempoDelay=temp.toInt();
iterazioni=tempoDisplay*1000/tempoDelay;
}

void cambiaFrase(){
    Process p;
    String temp="";
    p.begin("/usr/bin/php-cgi");
    p.addParameter("-q");
    p.addParameter("/mnt/sdal/arduino/www/ritorna_frase.php");
    p.addParameter(id_frase);
    p.run();
    while (p.available()>0) {
        char c = p.read();
        if(c!='|'){
            temp += c;
        } else {
            id_frase=temp;
            temp="";
        }
    }
    fraseVecchia=fraseCorrente;
    unsigned char b[temp.length()+1];
    temp.getBytes(b,temp.length()+1);
    fraseCorrente = b;
    if (fraseCorrente!=fraseVecchia){
        traslazione=0;
        larghezzaTesto = HT1632.getTextWidth(fraseCorrente, FONT_8X4_END,
        FONT_8X4_HEIGHT);
    }
}
```

l'avvio della rete WiFi e del Web Server di Arduino.

Nel ciclo loop controlliamo se la variabile *cont* sia uguale a 0: questo evento si verifica solo all'avvio del programma o quando una frase è stata mostrata per il numero necessario di cicli; nel caso in cui l'evenienza sia vera, spegniamo il display per il tempo necessario a cambiare frase. Vengono quindi richiamate le funzioni *caricaImpostazioni* e *cambiaFrase*.

Si passa poi alla gestione vera e propria del display: viene renderrizzata la frase su ogni schermo con una data traslazione (specifica di ogni display) e poi si procede ad incrementarla, even-

tualmente azzerandola nel caso in cui la frase abbia percorso tutti gli schermi; infine si attende per un periodo di tempo che dipende dalla velocità di scorrimento (delay).

L'ultima istruzione del loop incrementa la variabile *cont* azzerandola nel caso in cui siano state effettuate le iterazioni necessarie. Le funzioni *caricaImpostazioni* e *cambiaFrase* richiamano, utilizzando un oggetto di tipo *Process*, un file PHP presente sulla OpenWrt-Yún, e gestiscono l'output che ricevono sotto forma di flusso seriale di caratteri. La funzione *caricaImpostazioni* riceverà un testo del tipo "tempoDisplay | tempoDelay"

quindi si occuperà di estrarre dalla stringa ricevuta i due parametri e di assegnarli, convertendoli, alle relative variabili; *cambia-Frase* invece riceve una stream del tipo "id_frase | messaggio". Mentre è possibile assegnare direttamente l'id della frase alla sua variabile, per rendere comprensibile alle librerie del display il messaggio appena ricevuto è necessario convertirlo dal tipo String a Byte con la funzione *getBytes*. Procediamo infine a controllare se la frase nuova è uguale a quella vecchia, altrimenti annulliamo il vettore di traslazione, e concludiamo calcolando la larghezza in pixel della nuova frase. Se volessimo implementare un numero di display diverso da quello d'esempio è sufficiente modificare la variabile *numero_display* e assegnare il pin corretto alle variabili *cs1*, *cs2*, *cs3* e *cs4*.

COLLEGAMENTO DEI DISPLAY

Dopo aver configurato Arduino

e caricato lo sketch, passiamo al collegamento tra i display e la scheda; possiamo effettuare il collegamento in due modi: la maniera più veloce è senz'altro utilizzare lo shield dedicato, che potrete autocostruire seguendo lo schema elettrico e il piano di montaggio (e la relativa lista componenti); l'alternativa consiste nel cablare il circuito collegando il primo display ad Arduino utilizzando il cavo IDC mediante gli appositi cavetti chiamati jumper per collegamenti su breadboard. Lo schema che dovete seguire per il cablaggio è quello mostrato nella Fig. 5, ricordando che i jumper devono realizzare i collegamenti tra Arduino Yún e il primo display, secondo quanto riportato nella Tabella 1. Una volta collegato il primo display ad Arduino, possiamo collegare in cascata il secondo (e gli eventuali altri) utilizzando i cavi IDC. Indipendentemente dalla modalità di collegamento scelta, non dimentichiamoci di settare l'indirizzo di ogni display (da 1 a 4) agendo sui dip switch posti sul retro: l'indirizzo è determinante a ricomporre correttamente i messaggi sul display.

IL PROGETTO FINITO

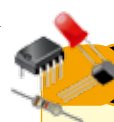
Arrivati a questo punto potremo posizionare il nostro ArduDisplay, alimentarlo (ad esempio tramite Micro-USB) e attendere circa un minuto per l'inizializza-

Tabella 1

PIN Arduino	PIN Display
3	WR
5	DATA
6	CS1
9	CS2
10	CS3
11	CS4
5V	+5V
GND	GND

zione del software e la creazione della rete WiFi. Se tutto è andato a buon fine, Arduino, non trovando nessuna frase da mostrare, mostrerà la frase predefinita. Come visibile in Fig. 6: possiamo connetterci con qualsiasi dispositivo alla rete "ArduDisplay", digitare sul browser "<http://scrivi.mi>" e inviare un messaggio di prova.

Ora spostiamoci nella pagina "<http://scrivi.mi/sd/admin>", eseguiamo l'accesso con la password predefinita "admin" e confermiamo il messaggio appena ricevuto: dopo pochi secondi dovremo vederlo sul display.



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Il master del circuito stampato può essere scaricato dal sito della rivista (www.elettronica.in) così come il file .stl per la stampa in 3D del contenitore e lo sketch per Arduino. Ciascuna matrice di LED costa 22,00 Euro (cod. 7719-P432X8LEDMATRIX) mentre la board Arduino Yún costa 62,90 Euro. Tutti i prezzi si intendono IVA compresa.

Il materiale va richiesto a:

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

Tel: 0331-799775 • Fax: 0331-792287

<http://www.futurashop.it>



Fig. 6 - Test di ArduDisplay.



CURRENT SENSE AMPLIFIER

Due soluzioni alternative
per misurare la corrente tramite
ADC esterno. Quale sarà la migliore?

dell'ing VINCENZO MENDOLA

Per misurare i valori di corrente in circuiti elettrici ed elettronici esistono svariate soluzioni, che possono essere divise in due grandi categorie: sistemi ad accoppiamento galvanico e sistemi isolati galvanicamente; nel primo caso il circuito si avvale di un resistore di shunt ai cui capi si misura la tensione per determinare, tramite la Legge di Ohm, la corrente che vi fluisce. Nel secondo caso, preferibile quando è necessario disaccoppiare il circuito di misura vero e proprio da quello in cui scorre la corrente da misurare, ci si avvale di metodiche di accoppiamento alternative come quella ottica (non lineare) o magnetica. In queste pagine vi proponia-

mo due esempi circuitali di misura tramite resistore di shunt, sistema utilizzabile quando non ci siano controindicazioni a mettere in comune la massa del circuito di misura con quella del circuito sottoposto a misura, cioè nella maggior parte dei casi comunemente incontrati dall'hobbista.

I due schemi corrispondenti sono basati su altrettanti circuiti integrati dedicati al current-sensing, prodotti uno dalla Linear Technology (LT6105) e l'altro dalla Texas Instruments (INA225). Abbiamo voluto proporre due dispositivi per farvi valutare due differenti approcci alla misura. Entrambi sono disponibili in package MSOP a 8

G0	G1	guadagno
GND	GND	25
GND	Vcc	50
Vcc	GND	100
Vcc	Vcc	200

Tabella 1

pin ed entrambi utilizzano una resistenza di sense la cui caduta di tensione (proporzionale alla corrente che l'attraversa), viene amplificata in modo da fornire in uscita una tensione, la quale, acquisita da un ADC (ad esempio quello all'interno di un microcontrollore), fornisce il valore della corrente che si vuole misurare.

A parte questi punti in comune, i due integrati hanno proprie prerogative: il Linear Technology LT6105 (<http://www.linear.com/docs/25764>) permette di monitorare la corrente in maniera unidirezionale, mentre l'INA225 (<http://www.ti.com/lit/gpn/ina225>) è bidirezionale, cioè rileva correnti in entrambi i versi; per garantire il rilevamento di correnti nei due versi, è necessario fornire una tensione di riferimento (reference voltage) che assicura la dinamica d'uscita richiesta per entrambi i versi della corrente, la cui ampiezza massima non deve essere necessariamente simmetrica (si veda il datasheet alla sezione "Bidirectional Operation" per ulteriori dettagli).

Il guadagno dell'integrato della Linear Technology può essere impostato a piacere tramite i va-

lori di alcune resistenze, mentre il current sense amplifier della Texas Instruments consente di scegliere solo tra quattro valori di guadagno predefiniti (25, 50, 100, 200), selezionabili tramite il livello logico dei due terminali G0 e G1 (si veda la **Tabella 1**).

La limitazione di poter impostare solo quattro valori è largamente compensata dall'elevata accuratezza del guadagno ($\pm 0,3\%$ massimo) contro l'1% massimo dell'LT6105; comunque l'accuratezza complessiva della misura non dipende esclusivamente da questo valore, ma ad essa concorrono molti altri fattori, tra cui la tolleranza della resistenza di sense, la sua stabilità termica, gli errori introdotti dall'ADC, ecc. Per quanto riguarda la tensione di alimentazione, in entrambi i casi è ammesso un ampio range: da 2,85 a 36 V per l'LT6105 e da 2,7 a 36V per l'INA225.

Quelli impiegati nei circuiti descritti in questo articolo, sono solo alcuni dei numerosi current sense amplifier presenti sul mercato; li abbiamo scelti per via di una serie di considerazioni, tra cui il costo, la disponibilità e la reperibilità, le caratteristiche ed il fatto di essere facilmente interfacciabili a tutti i microcontrollori dotati di ADC.

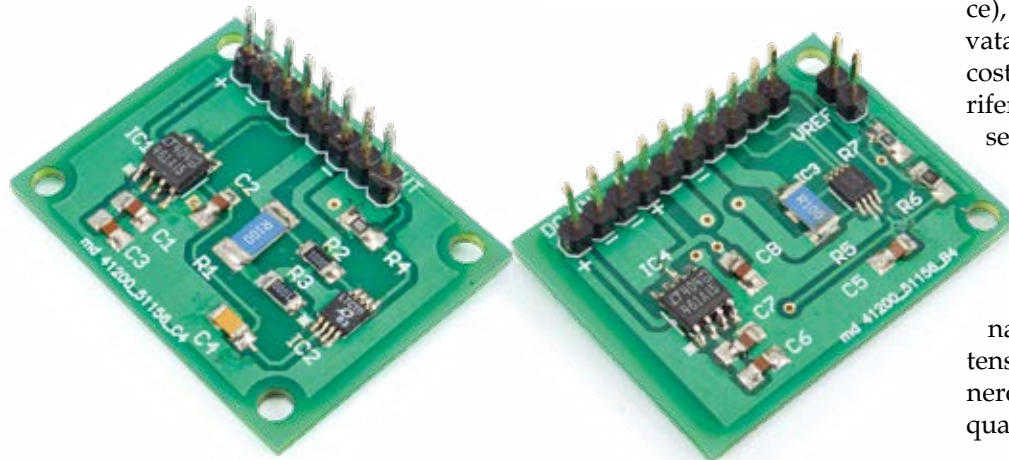
Dato che è proprio la tensione di riferimento dell'ADC il

parametro che più può influire nell'accuratezza della misura, dando per scontato che si stanno utilizzando resistenze di ottima qualità con elevata precisione (non inferiore all'1%) e con basso coefficiente di temperatura, abbiamo deciso di inserire in entrambi i circuiti un LT1461 (www.linear.com/docs/1342), che è un generatore di tensione di riferimento di elevata accuratezza (0,04% pari a $\pm 2\text{mV}$ per un'uscita di 5V) realizzato dalla Linear Technology e in grado di garantire in uscita una corrente superiore ai 50 mA, più che sufficiente nella maggior parte dei casi per alimentare anche il microcontrollore ed eventualmente anche il display LCD a questo connesso, se utilizzato.

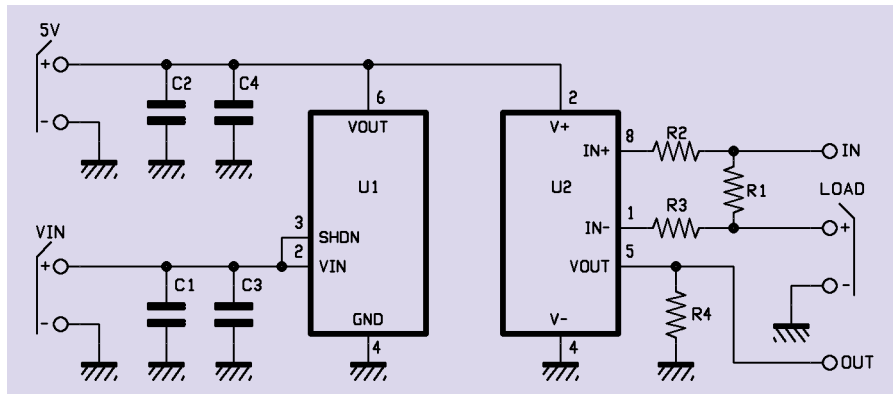
SCHEMA CON LT6105

Partiamo dall'analisi del primo circuito, ossia quello basato sull'integrato della Linear Technology. Innanzitutto notiamo il già citato riferimento di tensione, LT1461-5 (uscita a 5V), che accetta in ingresso tensioni fino a 20 V ed è caratterizzato da un basso drop-out (solamente 300 mV). I due condensatori da 1 μF (in ingresso) e da 2,2 μF (in uscita) devono essere necessariamente ceramici, in modo da garantire un bassissimo valore di ESR (Equivalent Series Resistance), permettendoci di avere l'elevata accuratezza dichiarata dal costruttore. Le connessioni del riferimento di precisione sono

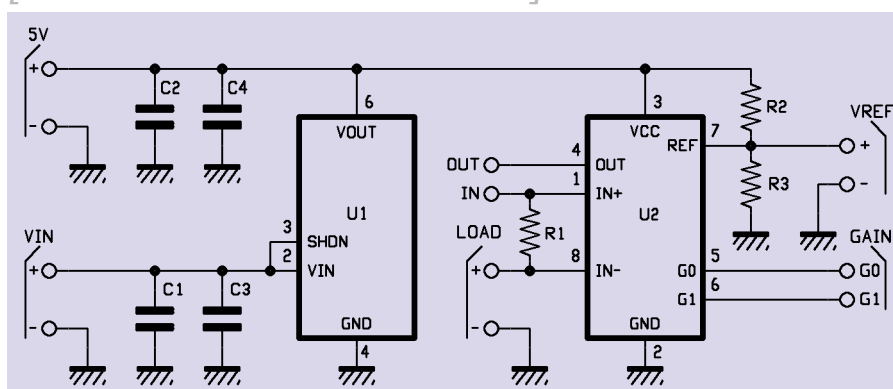
semplici e si limitano a quattro pin: tensione d'ingresso, tensione d'uscita, massa e shutdown; quest'ultima funzionalità non è utilizzata e quindi il terminale è connesso stabilmente alla tensione d'ingresso per mantenere l'LT1461 sempre "acceso" quando è presente tensione.



[schema elettrico LT6105]



[schema elettrico INA225]



Ricordate che, qualora decideste di utilizzare (come consigliato) l'LT1461 anche come riferimento di tensione dell'ADC, è necessario scegliere la versione che fornisce la tensione in uscita di valore adeguato al funzionamento del microcontrollore, al fine di evitare di danneggiare quest'ultimo; infatti non tutti i micro lavorano a 5 V, ma sono sempre più diffusi quelli che funzionano a tensioni inferiori, tipicamente 3,3 V.

L'uscita va ad alimentare l'LT6105, ma è anche disponibile sulle piazzole allineate a bordo del PCB per poter essere utilizzata come riferimento dell'ADC del microcontrollore o come alimentazione stessa del micro. Anche le connessioni dell'LT6105 rispecchiano quanto riportato dal data-sheet.

Soffermiamoci un attimo sulla resistenza di sense, il cui valore va scelto accuratamente in base alle necessità di misura: date le dimensioni e le caratteristiche del PCB, non è consigliabile misurare correnti di oltre 1 o 2 ampere, se non per brevissimi istanti, pena il surriscaldamento o dei danni al PCB stesso oppure ai componenti. Perciò consigliamo di scegliere una resistenza di 0,1 ohm da 1 o 2 watt e di non superare 1A di corrente che attraversa la scheda; con tale valore, la caduta di tensione ai capi della resistenza è pari a 100 mV, valore sufficientemente basso da non interferire eccessivamente nel funzionamento del circuito sotto misura. Ovviamente nulla vieta di utilizzare resistenze di valore differente, tenendo sempre conto del valore di corrente che si vuole effettivamente misurare. Nel circuito, avendo scelto un valore di Rsense pari a 0,1 ohm, abbiamo configurato il current sense amplifier per un valore del guadagno pari a 10 ($R3=R2=RIN=100$ ohm, $R4=1$

kohm, $AV=R4/RIN=10$).

Sul connettore a bordo del PCB troviamo, oltre alla tensione d'ingresso e alla tensione di riferimento di 5V, il terminale "IN" (cui collegare la sorgente applicata al carico), il terminale "LOAD" (cui connettere il carico) ed infine il terminale "OUT" che fornisce il valore di tensione proporzionale alla corrente che attraversa il carico. Ricordatevi di collegare correttamente i terminali "IN" e "LOAD" in quanto il circuito non è bidirezionale ed in caso di collegamento errato avreste in uscita un valore di tensione fisso a 0 V, indipendentemente dal valore di corrente che scorre in Rsense.

SCHEMA CON INA225

Per quanto riguarda il circuito basato sull'INA225, le considerazioni fatte in precedenza valgono ancora, con la differenza che in

questo caso il guadagno minimo impostabile è pari a 25 e viene selezionato tramite il livello logico sui due pin GS0 e GS1, come dettagliato nella **Tabella 1**. Il circuito è leggermente diverso da quello precedente, non essendo più necessarie le resistenze che nel caso dell'LT6105 fissano il valore del guadagno; sono state invece aggiunte R2 e R3, che impostano la tensione di riferimento necessaria per il funzionamento bidirezionale. Si ricorda che è possibile scegliere due valori di fondo scala differenti per le due direzioni della corrente (ad esempio, nel caso di una batteria, la corrente di scarica è diversa da quella di carica e abbiamo bisogno di maggior dinamica per misurare la prima), semplicemente scegliendo opportunamente queste due resistenze. In uscita, oltre alle connessioni previste per il

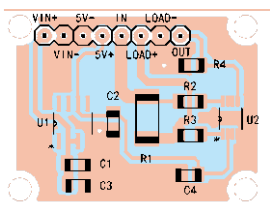
[piano di montaggio LT6105]

Elenco Componenti:

- R1: 0.01 ohm (1206)
 R2, R3: 100 ohm (0805)
 R4: 1 kohm (0805)
 C1: 1 μ F ceramico (0805)
 C2: 2,2 μ F ceramico (0805)
 C3, C4: 100 nF multistrato (0805)
 U1: LT1461DHS8-5#PBF
 U2: LT6105CMS8#PBF

Varie:

- Strip maschio 8 vie
- Circuito stampato



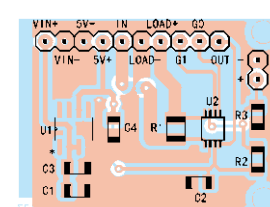
[piano di montaggio INA225]

Elenco Componenti:

- R1: 0.01 ohm (1206)
 R2, R3: 10 kohm (0805)
 C1: 1 μ F ceramico (0805)
 C2: 2,2 μ F ceramico (0805)
 C3, C4: 100 nF multistrato (0805)
 U1: LT1461DHS8-5#PBF
 U2: INA225AIDGKT

Varie:

- Strip maschio 2 vie
- Strip maschio 10 vie
- Circuito stampato



current sense amplifier della LT, sono presenti le due piazzole con i due contatti per la connessione ai GPIO che fissano il guadagno. Il jumper "VREF" permette di collegare il relativo pin dell'INA225 al potenziale di massa, permettendo di avere a disposizione tutta la dinamica possibile per la tensione d'uscita, qualora non servissero misure bidirezionali. Se utilizzate la scheda con il jumper inserito, Vout sarà compresa tra 0 e 5V (state alimentando l'INA225 con un LT1461-5), per cui, nel caso abbiate settato il minimo guadagno (25, ponendo GP0 e GP1 a massa), la massima corrente che potrete misurare, sarà di 2A; se invece volete effettuare delle misure bidirezionali di corrente, dovete togliere il jumper. In quest'ultimo caso, se avete scelto due valori uguali per R2 e R3, la tensione d'uscita con corrente nulla sarà pari a 2,5V, che scenderà fino a 0V per le tensioni "negative" e aumenterà fino a 5V per quelle positive. In pratica, per le correnti con polarità positiva (entranti nel terminale "IN" ed uscenti da "LOAD" del circuito) saranno comprese (con guadagno pari a 25) tra 0A (2,5V) e 1A (5V) mentre quelle negative (entranti nel terminale "LOAD" ed uscenti da "IN") saranno comprese tra 0A (2,5V) e -1A

(0V). Il codice da implementare per utilizzare queste breakout board con un microcontrollore è talmente semplice, che lo potrete realizzare per la vostra piattaforma di prototipazione senza alcuna difficoltà, tenendo a mente le poche accortezze evidenziate in precedenza.

REALIZZAZIONE PRATICA

Entrambe le schede sono previste sotto forma di breakout board per agevolarne l'integrazione in dispositivi e sistemi dove serva implementare la misura di corrente; allo scopo, ciascuna dispone di un pin strip a passo 2,54 mm. Per entrambe le unità è previsto l'uso di componentistica SMD, chi vorrà realizzarla, dovrà avere a disposizione, oltre ad una buona manualità ed esperienza in questo tipo di realizzazioni, l'attrezzatura necessaria, consistente in un saldatore di buona qualità (si consiglia una stazione saldatrice con stilo a bassa tensione, tipicamente 24V) dotato di punta di dimensione adatta alla tipologia di componenti da saldare, set di pinzette con caratteristiche adatte all'impiego con componenti a montaggio superficiale e flussante come ad esempio il tipo 951. Una volta in possesso del materiale necessario, si consiglia di iniziare a saldare i componenti

passivi, per poi passare a quello degli integrati. In questo caso l'uso del flussante vi aiuterà ad evitare di saldare erroneamente tra di loro i piedini vicini. Aiutatevi con la pinzetta per mantenere in posizione il dispositivo e iniziate a saldare due piedini diametralmente opposti per mantenere fermo l'integrato. Procedere poi a saldare i piedini rimanenti. Prestate attenzione a non scaldare eccessivamente gli integrati che potrebbero subire danni irreversibili durante il processo. Completato il montaggio della componentistica SMD, si possono infilare e saldare le file di pin-strip (8 punte per la basetta con l'LT6105 e 10 per l'altra).



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Gli integrati sono reperibili presso Futura Elettronica.
 Cod. 1350-LT6105CMS8 (Euro 4,50)
 Cod. 1350-INA225AIDGKT (Euro 5,60).
 Tutti i prezzi si intendono IVA compresa.

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

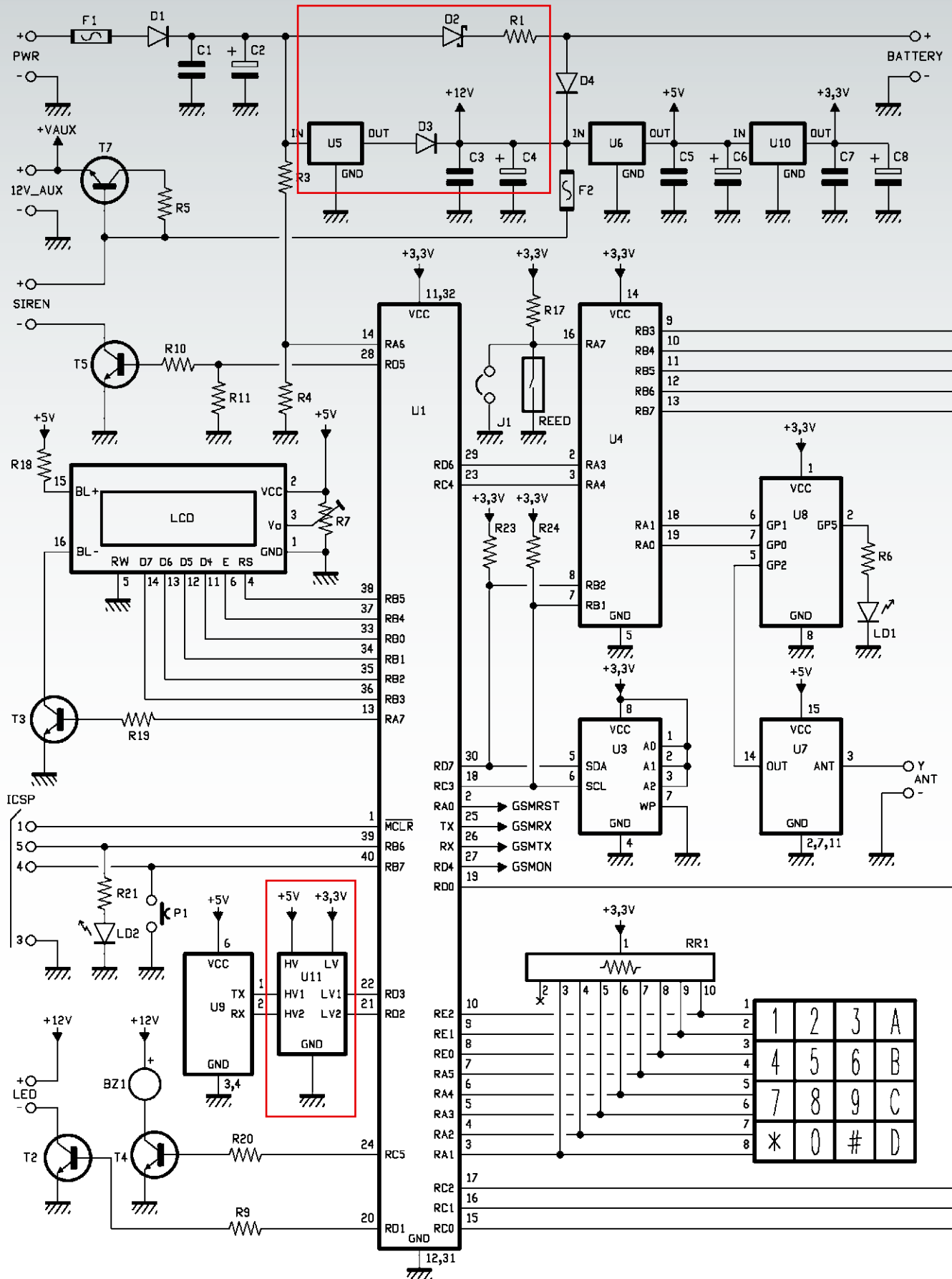
ELABORIAMO LA CENTRALE D'ALLARME A 24 ZONE

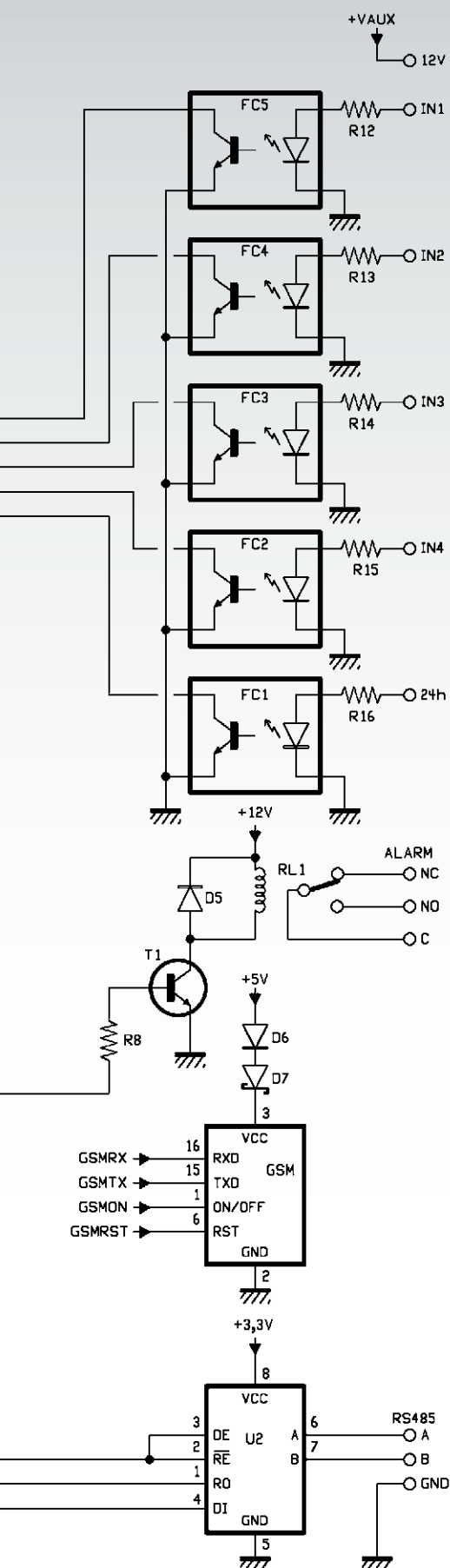
Aggiungiamo qualche funzione alla nostra centrale antifurto FT1015, per migliorarne l'utilizzabilità e le prestazioni.

di SANDRO TALAMINI

Sono passati ormai alcuni anni da quando abbiamo pubblicato (nei fascicoli n° 172 e n° 173) il progetto della nostra centrale d'allarme antifurto FT1015K, ma il progetto è seguito e apprezzato tuttora dal nostro pubblico, in mezzo al quale spiccano lettori che non perdono l'occasione di sperimentare le loro capacità di hackerare i dispositivi personalizzandoli e in qualche caso migliorandoli. Uno di questi è il Signor Sandro Talamini, che poco tempo fa ci ha sottoposto una sua revisione del progetto, che implementa nuove funzionalità a livello del firmware e qualche ritocco alla parte circuitale. Questo "remake" ci è sembrato interessante per gli spunti tecnici che offre ed abbiamo perciò deciso di pubblicarlo, a favore di tutti coloro che fossero alla ricerca di







La centrale d'allarme FT1015

24 zone parzializzabili
4 sensori per ogni zona wireless
Gestione di sensori wireless e via filo
Gestione codifiche MM53200, HT12 e Velleman
Ogni sensore della zona può essere gestito a piacere scegliendo tra perimetrale e volumetrico
Possibilità di memorizzare una ventina di telecomandi remoti
Possibilità di attivare la centralina anche da tastiera
Attivazione perimetrale o completa
Gestione zona 24h
Funzione Panic
Batteria tampone con autonomia di un giorno
Monitor alimentazione da rete attivabile/disattivabile
Uscita relé per eventuali accessori (ad esempio combinatori)
Spia di indicazione stato centralina
Log eventi
Contenitore provvisto di anti tamper, escludibile in caso di guasto
Comunicazione RS485 per monitor di stato in remoto
Predisposizione con uscita dedicata per la sirena
Uscita ausiliaria a 12V
Predisposizione per eventuale combinatore GSM
Configurazione via SMS
Accesso come installatore con codice dedicato

La centralina di cui proponiamo la modifica è del tipo a 24 zone, completa delle funzioni basilari di un qualsiasi sistema antifurto; è economica, semplice da usare e può essere abbinata a sensori sia a filo, sia wireless. Gli attuatori previsti sono una sirena, un'uscita a contatto pulito e, opzionalmente, un combinatore telefonico anche GSM. Inoltre, la nostra centralina ha un menu di configurazione chiaro e intuitivo, le cui funzioni più importanti di configurazione sono protette da password, lasciando per l'utente solo il cambio di codice utente o altre piccole funzioni non vitali per l'affidabilità del sistema. Per garantirne l'attività in caso di blackout, la centrale è provvista di batteria in tampone. Di seguito riportiamo le caratteristiche tecniche del sistema.



nuove soluzioni o che -avendo già montato e installato la centralina antifurto- volessero provare la variante. Prima di cedere la parola al nostro lettore, riassumiamo il progetto: innanzitutto riportiamo lo schema elettrico modificato dal Signor Talamini, evidenziando quali sono state le modifiche apportate; ciò vi permetterà di valutare quali varianti sono state apportate e perché, ovvero se vi interessano o preferite lasciare

lo schema elettrico com'è in origine. Riportiamo anche il nuovo firmware, che potrete altresì scaricare dal nostro sito Internet www.elettronica.in insieme agli altri file del progetto; questo firmware è quello del microcontrollore principale

Listato 1

```

***** MENU_USER *****
MENU_USER:
.
.
.
CASE 15 'TASTO D
    SELECT CASE RIGA_MENU
        CASE 1
            GOSUB MENU_21U 'Menu User-Attiva/Disattiva
            gosub menu_0
            esci=1
        CASE 2
            RIGA_MENU=1
            GOSUB MENU_22U1 'Menu User-Setup
        case 3
            RIGA_MENU=1
            gosub MENU_LOG 'Menu Log Eventi
        case 4
            RIGA_MENU=1
            gosub MENU_PARZIALIZZAZIONE 'Menu zone Parziali
        case 5
            RIGA_MENU=1
            gosub MENU_RITARDO 'Menu zone Ritardate
    .
    .
    .

***** M E N U _ R I T A R D O *****
*****
MENU_RITARDO:
    MENU=24
    if RIGA_MENU=1 or RIGA_MENU=4 or RIGA_MENU=7 or RIGA_MENU=10 or RIGA_MENU=13 or RIGA_MENU=16 or RIGA_MENU=19 or RIGA_MENU=22 THEN
        tmp=riga_menu
    ENDIF
    if RIGA_MENU=2 or RIGA_MENU=5 or RIGA_MENU=8 or RIGA_MENU=11 or RIGA_MENU=14 or RIGA_MENU=17 or RIGA_MENU=20 or RIGA_MENU=23 THEN
        tmp=riga_menu-1
    ENDIF
    if RIGA_MENU=3 or RIGA_MENU=6 or RIGA_MENU=9 or RIGA_MENU=12 or RIGA_MENU=15 or RIGA_MENU=18 or RIGA_MENU=21 or RIGA_MENU=24 THEN
        tmp=riga_menu-2
    ENDIF
    lcdout $FE, 1 'cancella display
    lcdout $FE, $02, "*** RITARDO ON/OFF ***"
    FOR I=1 TO 3
        SELECT CASE I
            CASE 1
                LCDOUT $FE, $C0, " Zona "
            CASE 2
                lcdout $FE, $94, " Zona "
            CASE 3
                lcdout $FE, $D4, " Zona "
        END SELECT
        if tmp<10 then
            lcdout "0",dec tmp
        else
            lcdout dec tmp
        endif
        ZONA_TMP=tmp
        gosub CTRL_ZONE_RITARDATE
        IF ZONA_RIT_ON=1 THEN
            lcdout " Ritardata"
        ELSE
            lcdout " Istantanea"
        ENDIF
        tmp=tmp+1
    NEXT I
    GOSUB SIMBOLO_RIGA
MENU_RITC:
    GOSUB TASTIERA
    SELECT CASE TASTO
        CASE 12 'TASTO A
            RIGA_MENU=RIGA_MENU-1
            GOSUB SIMBOLO_RIGA
            if RIGA_MENU=3 or RIGA_MENU=6 or RIGA_MENU=9 or RIGA_MENU=12 or RIGA_MENU=15 or RIGA_MENU=18 or RIGA_MENU=21 THEN
                Goto MENU_RITARDO
            ENDIF
        CASE 13 'TASTO B
            RIGA_MENU=RIGA_MENU+1
            GOSUB SIMBOLO_RIGA
            if RIGA_MENU=4 or RIGA_MENU=7 or RIGA_MENU=10 or RIGA_MENU=13 or RIGA_MENU=16 or RIGA_MENU=19 or RIGA_MENU=22 THEN
                Goto MENU_RITARDO

```

della centralina, ossia quello installato nel PIC18F46K20. La precisazione è d'obbligo in quanto nella centrale d'allarme ci sono più microcontrollori che governano ognuno una specifica funzione e che rispondono tutti a quello principale, cui è demandata la gestione dell'insieme.

Le modifiche hardware (che nello schema elettrico qui accanto trovate evidenziate nelle cornici a bordo rosso) riguardano essenzialmente la linea di alimentazione positiva e l'interfacciamento del microcontrollore principale (U1) con il modulo RTC (U9); sulla linea che porta l'alimentazione principale al circuito di carica della batteria è stato rimosso il transistor regolatore ed è stato inserito un diodo Schottky, il quale, presentando una caduta di tensione diretta di appena $0,2 \div 0,3$ volt, consente di alimentare i morsetti + e - PWR della centralina con tensioni più basse di quella prevista per la versione originale della centralina FT1015, dato che tra il positivo +PWR e la batteria ci sono solo due diodi, che comportano un massimo di 1 volt di caduta. In queste condizioni, con un alimentatore da 15 V si ottengono i 14 V che corrispondono alla tensione di carica ottimale delle batterie al piombo. Inoltre il regolatore U6, che insieme a U10 forma lo stadio di alimentazione a tensioni stabilizzate che deve funzionare anche in assenza della tensione di rete, viene alimentato dal diodo D4, che prende tensione direttamente dal positivo di batteria.

Detto ciò, passiamo la parola al Signor Talamini, che ci illustrerà le sue modifiche.

.....ho realizzato mesi fa la

centrale antifurto FT1015 per installarla nella mia abitazione; con l'uso, mi si sono rese necessarie alcune funzionalità non previste in origine, per cui ho provveduto ad apportare qualche piccola modifica per meglio adattare la centrale alle mie esigenze. Ritenendo che le nuove funzioni implementate possano divenire utili anche ad altri lettori di Elettronica In, ho pensato di mandarvi il nuovo firmware (**Listato 1**) affinché lo rendiate disponibile liberamente al pubblico. Qui di seguito, vi elenco le modifiche apportate.

FIRMWARE DEL PIC18F46K20

- 1) Aggiunta di una nuova voce nel menu Utente: "Ritardo zona ON/OFF"; per ognuna delle 24 zone è possibile impostare se l'allarme deve intervenire in modo istantaneo o ritardato dalla rilevazione del sensore.
- 2) La retroilluminazione del display ora è automatica: si accende solo quando viene usata la tastiera per spegnersi automaticamente dopo circa 15 secondi di inutilizzo. Si consiglia di cambiare la resistenza R18 come indicato in "Modifiche Hardware". Questa modifica è molto utile in quanto riduce il consumo di elettricità dell'intera centralina (anche se di poco) privando il display dell'alimentazione della retroilluminazione quando non serve averla accesa.
- 3) Menu "Rileva alim. rete": ho modificato il comportamento nella funzione corrispondente. A centrale attivata, l'assenza di tensione di rete non provoca allarme, ma viene inviato solamente un SMS di avviso. Al ripristino della tensione

Listato 1

```

        ENDIF
        CASE 14 'TASTO C
            RIGA_MENU=5
            esci=1
            riavvia_menu=1
        CASE 15
            ZONA_TMP=RIGA_MENU
            RIGA_MENU=1
            GOSUB MENU_RIT_ONOFF
        END SELECT
        if esci=0 then
            if riavvia_menu=1 then
                riavvia_menu=0
                goto MENU_RITARDO
            endif
            GOTO MENU_RITC
        endif
        esci=0
        return

\***** MENU _ R I T _ O N O F F *****/
MENU_RIT_ONOFF:
    MENU=241
    lcdout $FE, 1 'cancella display
    lcdout $FE, $02, "***** ZONA"
    IF ZONA_TMP<10 THEN
        LCDOUT "0",DEC ZONA_TMP
    ELSE
        LCDOUT DEC ZONA_TMP
    ENDIF
    LCDOUT " *****"
    LCDOUT $FE, $C0, " Ritardata"
    lcdout $FE, $94, " Istantanea"
    gosub CTRL_ZONE_RITARDATE
    IF ZONA_RIT_ON=1 THEN
        riga_menu=1
    ELSE
        riga_menu=2
    ENDIF
    GOSUB SIMBOLO_RIGA
MENU_RIT_ONOFFC:
    GOSUB TASTIERA
    SELECT CASE TASTO
        CASE 12 'TASTO A
            RIGA_MENU=RIGA_MENU-1
            GOSUB SIMBOLO_RIGA
        CASE 13 'TASTO B
            RIGA_MENU=RIGA_MENU+1
            GOSUB SIMBOLO_RIGA
        CASE 14 'TASTO C
            RIGA_MENU=ZONA_TMP
            esci=1
            riavvia_menu=1
        CASE 15 'TASTO D
            SELECT CASE RIGA_MENU
                CASE 1
                    ZONA_RIT_ON=1
                CASE 2
                    ZONA_RIT_ON=0
            END SELECT
            GOSUB SET_ZONE_RITARDATE
            gosub menu_ok
            RIGA_MENU=ZONA_TMP
            esci=1
            riavvia_menu=1
        END SELECT
        if esci=0 then
            GOTO MENU_RIT_ONOFFC
        endif
        esci=0
    RETURN

\***** A V V I O _ A L L A R M E *****/
AVVIO_ALLARME:
.
.
.
.
    I=TMP
    if tipo_sens="P" then
        I=TEMPO_ALLARME+1
    else
        gosub CTRL_ZONE_RITARDATE
    
```

(Continua)

Listato 1

```

        if ZONA_RIT_ON=0 then
            I=TEMPO_ALLARME+1
        endif
    endif
.
.
.
.

***** CTRL_ZONE_RITARDATE *****
`Verifica la zona scelta se è ritardata su ON o su OFF e ne restituisce lo
stato
CTRL_ZONE_RITARDATE:
    select case ZONA_TMP
        CASE 1
            ZONA_RIT_ON=ZONAR01
        CASE 2
            ZONA_RIT_ON=ZONAR02
        CASE 3
            ZONA_RIT_ON=ZONAR03
        CASE 4
            ZONA_RIT_ON=ZONAR04
        CASE 5
            ZONA_RIT_ON=ZONAR05
        CASE 6
            ZONA_RIT_ON=ZONAR06
        CASE 7
            ZONA_RIT_ON=ZONAR07
        CASE 8
            ZONA_RIT_ON=ZONAR08
        CASE 9
            ZONA_RIT_ON=ZONAR09
        CASE 10
            ZONA_RIT_ON=ZONAR10
        CASE 11
            ZONA_RIT_ON=ZONAR11
        CASE 12
            ZONA_RIT_ON=ZONAR12
        CASE 13
            ZONA_RIT_ON=ZONAR13
        CASE 14
            ZONA_RIT_ON=ZONAR14
        CASE 15
            ZONA_RIT_ON=ZONAR15
        CASE 16
            ZONA_RIT_ON=ZONAR16
        CASE 17
            ZONA_RIT_ON=ZONAR17
        CASE 18
            ZONA_RIT_ON=ZONAR18
        CASE 19
            ZONA_RIT_ON=ZONAR19
        CASE 20
            ZONA_RIT_ON=ZONAR20
        CASE 21
            ZONA_RIT_ON=ZONAR21
        CASE 22
            ZONA_RIT_ON=ZONAR22
        CASE 23
            ZONA_RIT_ON=ZONAR23
        CASE 24
            ZONA_RIT_ON=ZONAR24
    end select
RETURN

***** SET_ZONE_RITARDATE *****
`Imposta la zona scelta se è ritardata su ON o su OFF
SET_ZONE_RITARDATE:
    select case ZONA_TMP
        CASE 1
            ZONAR01=ZONA_RIT_ON
        CASE 2
            ZONAR02=ZONA_RIT_ON
        CASE 3
            ZONAR03=ZONA_RIT_ON
        CASE 4
            ZONAR04=ZONA_RIT_ON
        CASE 5
            ZONAR05=ZONA_RIT_ON
        CASE 6
            ZONAR06=ZONA_RIT_ON
        CASE 7
            ZONAR07=ZONA_RIT_ON
        CASE 8

```

di rete, viene inviato un altro SMS di ripristino tensione. Questa variante consente di mantenere "silenzioso" il sistema di allarme, ovvero di evitare che in caso di blackout la centrale si metta a suonare insieme a quelle delle abitazioni di tutto il circondario, creando fastidio e falsi allarmi che alla fine fanno ignorare gli allarmi veri. Notate che se non viene montato il modulo GSM, la voce del menu "Rileva alim. rete" non è attivabile.

- 4) Riarmo centrale: ho notato che la centralina originale, dopo un allarme, una volta spenta la sirena dopo il tempo prestabilito, rimane inibita (per non provocare allarmi continui) fino alla disattivazione/riattivazione. Fino a quando non si interviene sul posto, la centrale non è più in grado di segnalare allarmi successivi al primo. Ho previsto, per questi casi, la possibilità manuale di riarmare la centrale da remoto tramite SMS, in modo da rendere possibile la segnalazione di nuovi allarmi. Sarà l'utente a decidere se lasciare la centrale inibita o riattivarla. Il comando SMS che ho aggiunto è: "RIA!" seguito dal consueto codice utente separato dal ";". La conferma dell'avvenuto riarmo viene fornita dalla centrale con risposta tramite SMS contenente la configurazione attuale.

MODIFICHE HARDWARE

Vediamo nel dettaglio le modifiche apportate all'hardware: la prima riguarda la resistenza R18, la quale è stata portata a 10 ohm dagli 82 iniziali; ho cambiato il valore, abbassandolo

```

        ZONAR08=ZONA_RIT_ON
CASE 9
        ZONAR09=ZONA_RIT_ON
CASE 10
        ZONAR10=ZONA_RIT_ON
CASE 11
        ZONAR11=ZONA_RIT_ON
CASE 12
        ZONAR12=ZONA_RIT_ON
CASE 13
        ZONAR13=ZONA_RIT_ON
CASE 14
        ZONAR14=ZONA_RIT_ON
CASE 15
        ZONAR15=ZONA_RIT_ON
CASE 16
        ZONAR16=ZONA_RIT_ON
CASE 17
        ZONAR17=ZONA_RIT_ON
CASE 18
        ZONAR18=ZONA_RIT_ON
CASE 19
        ZONAR19=ZONA_RIT_ON
CASE 20
        ZONAR20=ZONA_RIT_ON
CASE 21
        ZONAR21=ZONA_RIT_ON
CASE 22
        ZONAR22=ZONA_RIT_ON
CASE 23
        ZONAR23=ZONA_RIT_ON
CASE 24
        ZONAR24=ZONA_RIT_ON
end select
WRITE ZONA_TMP+49,ZONA_RIT_ON
RETURN

***** R E S E T *****
*****
RESET:
.
.
.
.
.
'Ritardo zone, tutte e 24 su Ritardata (0:Istantanea - 1:Ritardata)
  for i=1 to 24
    WRITE i+49,1
    pause 10
  next i
.
.
.
.

***** C T R L _ S M S *****
*****
CTRL_SMS:
.
.
.
  case "R"
    if BUFF[1]="I" and BUFF[2]="A" and BUFF[3]="!" and LUNGMESS>=11
then
      if allarme_eseguito=1 and SIRENA_SUONATA=1 THEN
        SMS_INVIATI=0
        ULTIMA_ZONA_ALLARME=0
        ZONA_ALLARME=0
        ALLARME_ESEGUITO=0
        ALLARME_IN_CORSO=0
        SIRENA_SUONATA=0
        CENTRALE_ATTIVAZIONE=1
        GOSUB MENU_0
        lcdout $FE, $02, " Riarmo in corso... "
        low RELE
        low sirena
      endif
      gosub SET_SMS_STATO
    endif
.
.
.

```

```

        ZONAR08=ZONA_RIT_ON
CASE 9
        ZONAR09=ZONA_RIT_ON
CASE 10
        ZONAR10=ZONA_RIT_ON
CASE 11
        ZONAR11=ZONA_RIT_ON
CASE 12
        ZONAR12=ZONA_RIT_ON
CASE 13
        ZONAR13=ZONA_RIT_ON
CASE 14
        ZONAR14=ZONA_RIT_ON
CASE 15
        ZONAR15=ZONA_RIT_ON
CASE 16
        ZONAR16=ZONA_RIT_ON
CASE 17
        ZONAR17=ZONA_RIT_ON
CASE 18
        ZONAR18=ZONA_RIT_ON
CASE 19
        ZONAR19=ZONA_RIT_ON
CASE 20
        ZONAR20=ZONA_RIT_ON
CASE 21
        ZONAR21=ZONA_RIT_ON
CASE 22
        ZONAR22=ZONA_RIT_ON
CASE 23
        ZONAR23=ZONA_RIT_ON
CASE 24
        ZONAR24=ZONA_RIT_ON
end select
WRITE ZONA_TMP+49,ZONA_RIT_ON
RETURN

***** R E S E T *****
*****
RESET:
.
.
.
.
.
'Ritardo zone, tutte e 24 su Ritardata (0:Istantanea - 1:Ritardata)
  for i=1 to 24
    WRITE i+49,1
    pause 10
  next i
.
.
.
.

***** C T R L _ S M S *****
*****
CTRL_SMS:
.
.
.
  case "R"
    if BUFF[1]="I" and BUFF[2]="A" and BUFF[3]="!" and LUNGMESS>=11
then
      if allarme_eseguito=1 and SIRENA_SUONATA=1 THEN
        SMS_INVIATI=0
        ULTIMA_ZONA_ALLARME=0
        ZONA_ALLARME=0
        ALLARME_ESEGUITO=0
        ALLARME_IN_CORSO=0
        SIRENA_SUONATA=0
        CENTRALE_ATTIVAZIONE=1
        GOSUB MENU_0
        lcdout $FE, $02, " Riarmo in corso... "
        low RELE
        low sirena
      endif
      gosub SET_SMS_STATO
    endif
.
.
.

```

36^a MOSTRA REGIONALE DELL'ELETTRONICA SCANDIANO

(REGGIO EMILIA)

14/15 FEBBRAIO
CENTRO FIERISTICO 2015



**MERCATINO DELLE PULCI
RADIOAMATORIALI**

**TELEFONIA - COMPONENTISTICA
COMPUTER - HI-FI CAR - RADIANTISMO
CB e OM - VIDEOREGISTRAZIONE**

ORARI

**Sabato 14 Febbraio 9 - 18,30
Domenica 15 Febbraio 9 - 18**



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

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

la Futura Elettronica -www.futurashop.it- e che viene codificato come 7300-LLCONBI. Conseguentemente, ho tolto il resistore di pull-up R22 presente, nel progetto originale, per mantenere a livello alto RD3 (evitando false commutazioni che avrebbero potuto causare errori nell'esecuzione del firmware) nel caso non venisse montato l'RTC (ricordo infatti che questo modulo si può anche non montare). Ho preferito inserire l'adattatore di livello in quanto la soluzione a resistore di pull-up non è comunque ottimale per il micro U1, dato che porterebbe i 5 volt su una linea a 3,3V. L'ultima modifica circuitale riguarda la sezione di carica della batteria tampone: alimentando la centrale a 15V non riuscivo ad avere almeno 13V ai capi della batteria. Ho quindi tolto T6 e cortocircuitato le relative piazzole di collettore ed emettitore, rimosso la resistenza R2 e sostituito D2 con un Schottky per avere una caduta inferiore (in casa avevo un 30WQ04FNPBF da 40V 3,5A). Infine, ho sostituito R1 con una resistenza da 10 ohm - 2W.



per il MATERIALE

Ricordiamo che questo progetto è stato presentato sui numeri 172 e 173 della rivista. La scatola di montaggio dell'antifurto è disponibile al prezzo di Euro 126,00 (cod. FT1015K). Il kit comprende tutti i componenti, i micro programmati, il PCB forato e serigrafato nonché tutte le minuterie. Gli unici elementi non inclusi (ma disponibili separatamente) sono: Modulo GSM completo di antenna (TDGGS-M_900, Euro 51,00), batteria ricaricabile cod. NP08-12 (Euro 21,00), alimentatore AL08-12N (Euro 8,00), contenitore G313 (Euro 11,00), telecomando TX3750-2CS (Euro 19,00). Tutti i prezzi si intendono IVA compresa.

Il materiale va richiesto a:

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

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



4 AHB
(Automatic High Beam Assistance)

5 TSR
(Traffic Sign Recognition)

3 PCW
(Forward Pedestrian Collision Warning)

AUTOVEICOLI SEMPRE PIU' SICURI GRAZIE ALL'ELETTRONICA

di KLAUS NEUENHUESKES*

2 FCW
(Forward Vehicle Collision Warning)

8 FCW
(Oncoming Vehicle Collision Warning)

6 SfM
(Detection of 3D Objects such as falling objects, curbstone and road damages by SfM [Structure from Motion])

I nuovi sistemi di assistenza avanzata alla guida (ADAS, Automatic Driver Assistance Systems) utilizzeranno ampiamente il riconoscimento delle immagini per aiutare gli automobilisti a identificare i rischi ed evitare incidenti; ciò aumenterà la richiesta di processori adeguati che possano garantire risultati in tempo reale e un'elevata efficienza energetica.

7 SLR
(traffic Signal Lamp Recognition)



1 LDW
(Lane Departure Warning)

Il miglioramento della sicurezza delle automobili si è rivelato il fattore più importante che ha contribuito alla riduzione del numero dei morti sulle strade in Europa negli ultimi decenni, come documentato da un rapporto elaborato congiuntamente da EuroRAP ed EuroNCAP e pubblicato nel 2013. Questo rapporto intitolato *Roads that Cars Can Read*, (Le strade che le auto possono leggere) comprende un invito a standardizzare elementi come segnali stradali e strisce bianche per un più agevole riconoscimento da parte dei sistemi di bordo. Si prevede che la diffusione di sistemi autonomi dotati di funzioni di riconoscimento visivo aiuterà a migliorare ulteriormente la sicurezza stradale negli anni a venire. In effetti, EuroNCAP (*European New Car Assessment Program*), il programma europeo di valutazione dei nuovi modelli di automobili, ha creato da qualche tempo dei protocolli di prova per diversi sistemi di sicurezza autonomi. I sistemi di segnalazione delle collisioni frontali (FCW, *Forward Collision Warning*) e di frenatura autonoma di emergenza (AEB, *Autonomous Emergency Braking*) sono stati inclusi nella valutazione di sicurezza EuroNCAP per la prima volta nel

2014. Il programma aggiungerà nel 2016 il riconoscimento dei segnali stradali (TSR, *Traffic-Sign Recognition*) e la frenatura autonoma di emergenza con riconoscimento diurno dei pedoni (PD, *Pedestrian-Detection*) e nel 2018 la frenatura autonoma di emergenza con il riconoscimento dei pedoni a tutte le ore. Altri importanti sistemi di assistenza alla guida che si stanno oggi diffondendo comprendono l'avviso di cambio corsia o l'assistenza al mantenimento della corsia (LDW/LKA, *Lane-Departure Warning / Lane-Keeping Assist*), l'assistenza automatica all'uso degli abbaglianti (AHB, *Automatic High-Beam Assistance*) in grado di rivelare i veicoli in avvicinamento frontale e i fari posteriori dei veicoli che ci precedono al fine di ridurre al minimo il bagliore, l'avviso di collisione frontale con pedoni (PCW, *Pedestrian Forward Collision Warning*), il riconoscimento dei segnali stradali (TSR, *Traffic Sign Recognition*) e il riconoscimento delle luci semaforiche (SLR, *Signal Lamp Recognition*). La Figura 1 mostra come alcuni di questi sistemi possono aiutare gli automobilisti a dare la giusta priorità ai rischi che si possono incontrare oggi sulla strada. Caratteristiche più avanzate comprendono

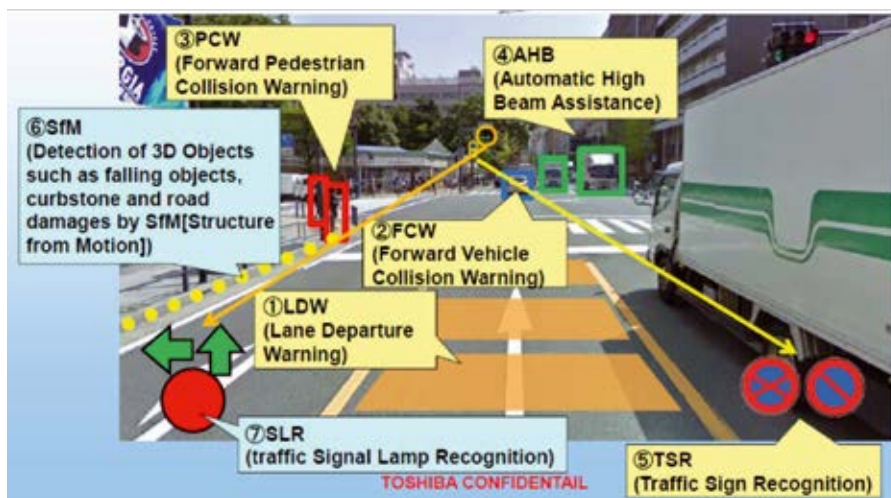


Fig. 1 - I sistemi basati sulla visione aiutano il guidatore a identificare e a prestare attenzioni ai rischi potenziali.



la rivelazione di oggetti tridimensionali, nonché il monitoraggio del conducente per rilevare segnali visivi di stanchezza e autenticarne l'identità.

È naturale che le auto di domani saranno in grado di "vedere" ampiamente al posto del conducente. E pur comprendendo che il conducente non deve affidarsi troppo a questo tipo di assistenza, la probabilità di ridurre il numero di incidenti gravi e mortali che continuano a verificarsi sulle strade europee è significativa.

LA SICUREZZA STRADALE OGGI: C'È ANCORA SPAZIO PER MIGLIORARE

Anche se la sicurezza sta migliorando, nel 2013 sulle strade europee sono morte in media 70 persone al giorno, in base ai dati resi

noti dalla Commissione Europea. Inoltre, il Commissario Europeo per la Mobilità e i Trasporti, Siim Kallas, ha precisato che per ogni incidente mortale ci sono 10 incidenti gravi, ad esempio con lesioni al cervello o alla spina dorsale. I sistemi di assistenza alla guida come VFCW e LDW/LKA sono principalmente studiati per prevenire le cause delle collisioni, che spesso derivano da disattenzioni dell'automobilista. Il sistema è in grado di rivelare il pericolo e allertare il guidatore. Sistemi autonomi come l'AEB rappresentano una migrazione verso un coinvolgimento più attivo dei sistemi di bordo, che in questi casi prendono il controllo dell'automobile per applicare l'azione correttiva. Sebbene molti dei primi sistemi di assistenza alla guida erano

studiati per proteggere i passeggeri, sviluppi più recenti tendono a ridurre i pericoli verso utenti della strada più indifesi come i pedoni. Questi sono particolarmente vulnerabili nelle aree urbane, oltre che sulle strade rurali prive di marciapiede. Il rilevamento automatico dei pedoni, tuttavia, presenta problemi di tipo tecnico. L'ampia variabilità delle forme e dimensioni umane, nonché la presenza di altri oggetti, come cassonetti, siepi, barriere e cartelli stradali, richiedono un potente riconoscimento delle immagini per distinguere un pedone nel campo visivo. Il riconoscimento dei pedoni nelle ore notturne o in condizioni di scarsa illuminazione è ancora più problematico.

Tra i vari utenti vulnerabili della strada, gli anziani meritano un discorso a parte. Gli anziani sono più vulnerabili, non solo come pedoni, ma anche come automobilisti. Fattori come il rallentamento delle reazioni mentali e fisiche agli eventi della strada e il deterioramento delle capacità visive possono aumentare la probabilità di essere coinvolti in un incidente. D'altro canto, in Europa c'è la necessità di continuare a muoversi in modo indipendente anche a tarda età, soprattutto perché l'età pensionabile continua ad aumentare. La frenatura automatica e gli altri sistemi di assistenza possono aiutare i conducenti più anziani a restare in sicurezza per più tempo.

REQUISITI DEI SISTEMI DI RICONOSCIMENTO DELLE IMMAGINI

Sistemi in grado di "vedere" i rischi si basano in modo significativo su un riconoscimento immagini ad alte prestazioni, e pertanto richiedono migliorie nelle tecnologie di elaborazione dei sistemi di riconoscimento immagini, migliorie che siano adatte alle applicazioni automobilistiche.

L'espressione "ad alte prestazioni" può voler dire tempi di risposta brevi e deterministici che permettano al sistema di notificare i rischi quasi istantaneamente, mentre i risultati devono essere precisi e affidabili. Se il sistema produce un eccessivo numero di "falsi positivi", il conducente inizierà a non fidarsi del sistema e a deselectare diverse opzioni, invalidando i vantaggi dell'assistenza automatica. In futuro, quando la guida con questi sistemi sarà accettata come norma, gli automobilisti che non li usassero potrebbero trovarsi in posizione di svantaggio. Storicamente, il riconoscimento delle immagini si è sempre basato estensivamente sul software, in particolare in applicazioni non automobilistiche. In un'applicazione ADAS automobilistica, tuttavia, la potenza dissipata da un processore di segnali digitali (DS, *Digital Signal*) ad alte prestazioni operante a frequenze dell'ordine di parecchi megahertz presenterebbe delle inaccettabili problematiche di tipo termico. I limiti di spazio interni al veicolo richiedono tipicamente che l'unità contenente l'elettronica di sistema e le telecamere venga montata dietro il parabrezza, dove può essere esposta alla diretta luce solare. È quindi importante garantire un comportamento in tempo reale del sistema senza dover ricorrere a dispositivi energivori interni alla scatola e che dissipano grandi quantitativi di calore. Gli effetti di un approccio software sull'economia di sistema e sull'efficienza generale del veicolo e di conseguenza sulla sua autonomia, sono anch'essi fattori che contribuiscono alla necessità di sviluppare una soluzione economicamente conveniente e a basso consumo.

L'HARDWARE LA FA DA PADRONE

L'esecuzione di funzioni di

elaborazione complesse o usate di frequente come trasformazioni, filtri, istogrammi e adattamenti piramidali con un hardware dedicato, piuttosto che con un software dedicato, può accelerare l'esecuzione degli algoritmi per favorire il funzionamento in tempo reale e massimizzare al contempo l'efficienza energetica. Ma un processore adeguato deve anche essere in grado di eseguire più applicazioni simultanee di assistenza alla guida. Ciò suggerisce la presenza di un processore a più motori con una serie di acceleratori hardware multicanale dedicati come piattaforma ideale per l'utilizzo in auto in grado di "vedere".

Poiché il numero di sistemi di assistenza alla guida che richiedono il riconoscimento delle immagini è aumentato, Toshiba ha sviluppato la sua famiglia di processori di riconoscimento immagini. I dispositivi si sono evoluti da una configurazione di base che combinava coppie di motori di elaborazione multimediale (MPE, *Media-Processing Engine*), un ingresso per singola telecamera e blocchi elementari di accelerazione hardware alla nuova famiglia di processori TMPV7608XBG di quarta generazione che comprendono fino ad otto unità MPE, otto ingressi per telecamere e acceleratori hardware multipli. Questi comprendono il calcolo e il filtraggio dei pixel, una trasformazione affine migliorata per la riduzione della distorsione e il ridimensionamento delle immagini, e acceleratori per la manipolazione e l'adattamento degli istogrammi. Alcuni acceleratori hanno prestazioni ancora più veloci rispetto ai loro predecessori, grazie all'utilizzo di tecniche più avanzate come l'accelerazione piramidale nella trasformazione affine.

Nell'hardware dedicato sono

anche implementate le funzioni Toshiba CoHOG (*Co-occurrence Histograms of Oriented Gradients*), CoHOG migliorato e SfM (*Structure from Motion*). Con questi miglioramenti, questo processore di quarta generazione offre prestazioni dieci volte superiori rispetto al precedente dispositivo quad-MPE.

TECNOLOGIE ORIGINALI PER IL RICONOSCIMENTO DELLE IMMAGINI

Toshiba ha creato la propria tecnologia CoHOG per migliorare il riconoscimento degli esseri umani in svariate situazioni, come ad esempio mentre camminano, vanno in bicicletta, sono seduti o sono in sedia a rotelle, e per permettere quando occorre il rilevamento degli animali, in modo da distinguere i corpi viventi da quelli inanimati, velocemente e affidabilmente. Un vantaggio essenziale è il fatto che la funzione CoHOG elimina l'elaborazione intensiva delle immagini richiesta per effettuare altre tecniche di riconoscimento come l'analisi delle shapelet.

CoHOG confronta i dati rilevati elaborati rispetto alle caratteristiche note delle forme e dei movimenti del corpo umano. La tecnica cerca coppie di gradienti comprese entro determinati limiti superiori e inferiori, corrispondenti a elementi come spalle, braccia,





Fig. 2 - Il CoHOG migliorato permette il riconoscimento dei pedoni 24 ore su 24.

gambe e fianchi, sapendo che questi elementi avranno anche un preciso orientamento reciproco, entro limiti fisiologici come lunghezze massime e minime degli arti, range di movimento e altezza totale. Toshiba ha anche sviluppato l'acceleratore CoHOG migliorato, che fornisce una precisione elevatissima nel riconoscimento dei pedoni nelle ore notturne analizzando i gradienti cromatici di immagini rilevate da più telecamere Full-HD (Fig. 2).

L'acceleratore SfM assiste il guidatore identificando con precisione ostacoli ignoti non inclusi nella libreria delle immagini, come guardrail, marciapiedi e piccoli oggetti presenti sulla carreggiata. Il rilevamento di ostacoli sconosciuti differisce dal classico riconoscimento dei modelli grazie all'uso della tecnologia di ricostruzione tridimensionale, utilizzando l'acceleratore SfM per analizzare rapidamente l'immagine fornita da una telecamera monoculare. Lo sviluppo del progetto è supportato da un kit di sviluppo software con driver e programmi applicativi campione, nonché da un ambiente di sviluppo integrato dedicato, un debugger per il processore multimediale e un simulatore. Sono anche inclusi un set di API per i driver software e gli acceleratori hardware insieme

a una libreria di elaborazione immagini.

Toshiba possiede diversi algoritmi come il riconoscimento dei pedoni, dei veicoli e dei contorni, mentre altri algoritmi sono a livello beta e in fase di preparazione per il lancio sul mercato. Il dispositivo TMPV7608XBG dà inoltre agli sviluppatori la possibilità di implementare i propri algoritmi qualora lo desiderino.

ELABORAZIONE ENERGETICAMENTE EFFICIENTE E SICUREZZA FUNZIONALE

Oltre a favorire un riconosci-

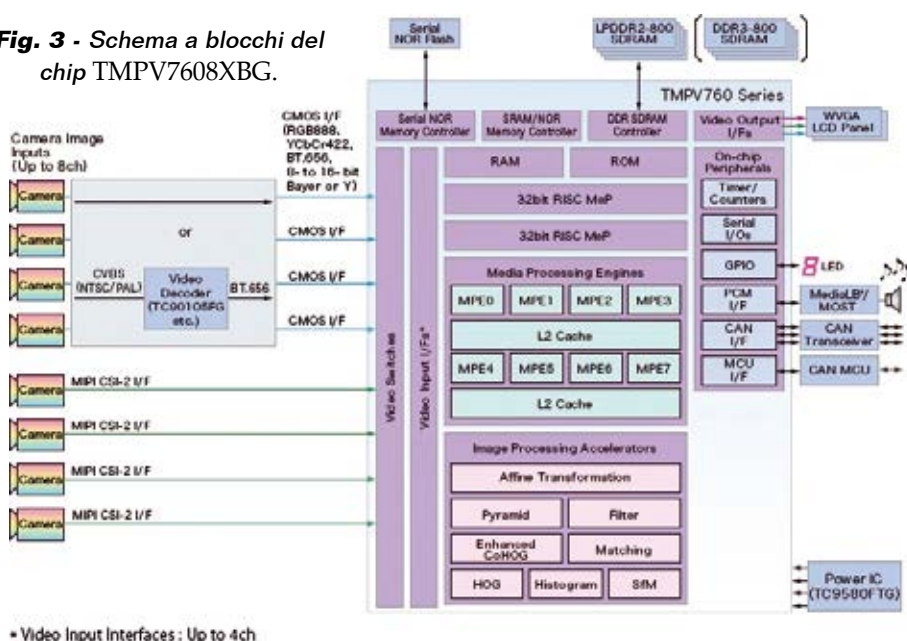
mento in tempo reale, i diversi acceleratori hardware integrati contribuiscono anche a ridurre il consumo generale di potenza del processore.

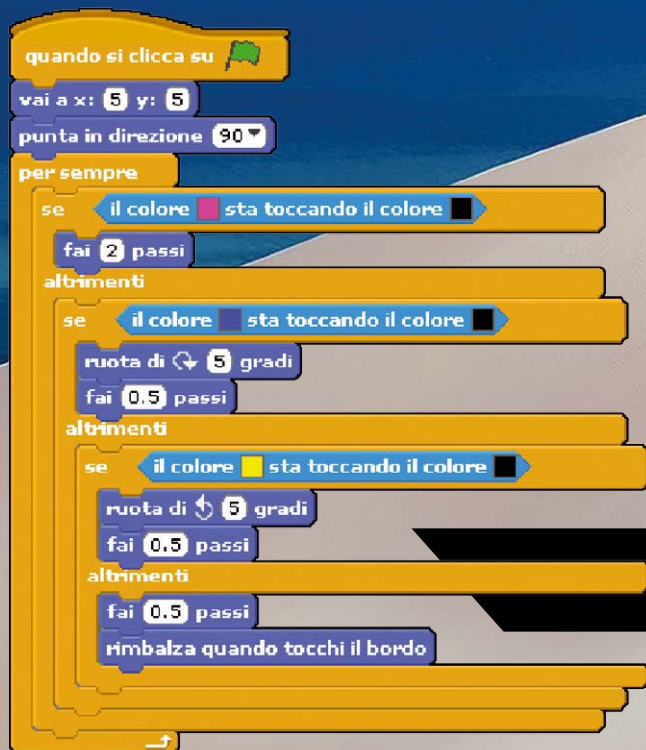
Il processore TMPV7608XBG per il riconoscimento delle immagini è stato sviluppato utilizzando il flusso di processo Toshiba certificato ISO26262. In quanto componente per la sicurezza di guida, il dispositivo è progettato per ottenere il necessario livello ASIL (*Automotive Safety Integrity Level*), garantendo la massima robustezza ai rischi sulla sicurezza.

Gli incidenti stradali si sono ridotti considerevolmente negli ultimi anni e i nuovi sistemi di assistenza alla guida promettono di realizzare ulteriori miglioramenti in termini di sicurezza. Molti dei sistemi oggi proposti dipendono in larga misura dal riconoscimento visivo dei rischi e trarranno vantaggio da una piattaforma di elaborazione economicamente conveniente ed energeticamente efficiente in grado di riconoscere immagini molto rapidamente.

**Klaus Neuenhueskes, Toshiba Electronics Europe*

Fig. 3 - Schema a blocchi del chip TMPV7608XBG.





SCRATCH: LA PROGRAMMAZIONE RESA FACILE

di MARCO MAGAGNIN

Dopo aver preso dimestichezza con l'ambiente di sviluppo che accompagna il linguaggio Scratch approfondiamo le principali strutture logiche che stanno alla base dell'attività di programmazione dei calcolatori elettronici.

Un paio di simpatici esempi renderanno tutto più semplice.

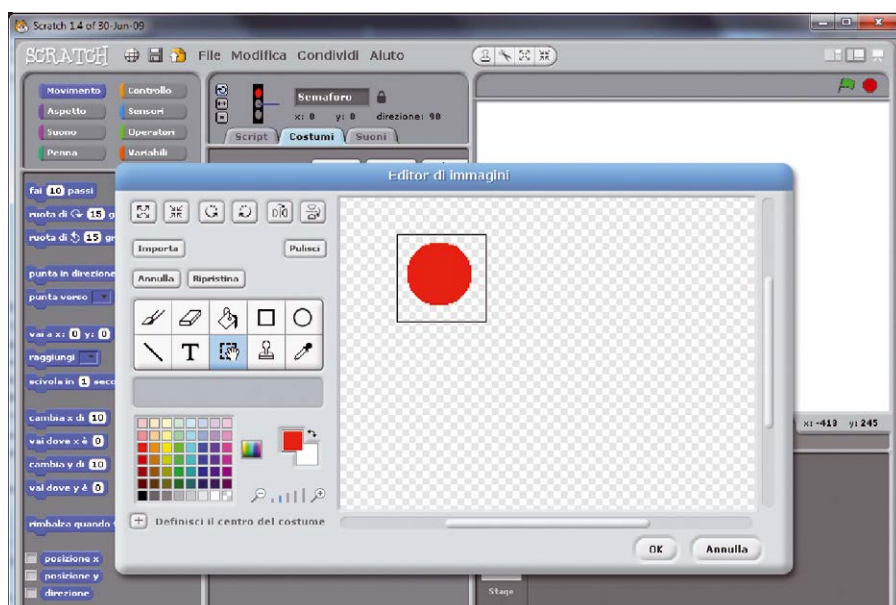
Seconda parte.

Riprendiamo il nostro cammino nell'apprendimento dell'arte della programmazione dei calcolatori elettronici, approfondendo le strutture di base della programmazione in generale ed in particolare di come sono state implementate nel linguaggio Scratch. Non ci stancheremo mai di ricordare che la progettazione e realizzazione di programmi e, da un punto di vista più ampio, di applicazioni, richiede sicuramente la padronanza delle tecniche e delle pratiche proprie della disciplina ma anche, se non soprattutto, la capacità di saper "progettare" gli elementi di base in una "architettura" ben strutturata, dal funzionamento affidabile, con prestazioni adeguate e, non ultima, bella da vedere e facile ed intuitiva da utilizzare. Questa capacità è ciò che di solito usiamo chiamare creatività, estro e perché no, genio. Non a caso il



Fig. 1

Fig. 2



corso di “programmazione” più prestigioso al mondo, tenuto dal Prof. Donald Knuth, all’università di Stanford si intitola “The art of computer programming”. I temi del suo corso sono raccolti in un’opera di quattro volumi monumentali dal titolo omonimo. Se riuscite a studiarli tutti un posto in Google non ve lo leva nessuno. Non meravigliatevi se sottolineo in continuazione la componente artistica associata alle attività di programmazione, così come avviene in molte altre discipline umane, come la musica, la pittura e la scultura, per citarne solo alcune. Prendiamo il solo esempio della pittura, supporti come tele, tavole e intonaci così come pennelli e colori sono gli “ingredienti” di base, che richiedono, per essere realizzati, conoscenze tecniche di chimica, muratura, tessitura, conoscenza dei materiali, ecc. Le “strutture” disciplinari principali sono costituite dalla conoscenza degli effetti di luce, delle ombre, della prospettiva, delle tecniche di miscelazione e deposizione dei colori, ecc. Tutte queste conoscenze, tuttavia, non danno grandi garanzie sulla possibilità di realizzare un bel quadro, e

nemmeno esiste un manuale che possa guidare un artista passo passo a realizzare un’opera d’arte, o comunque qualcosa di giudicabile come “bello”. Potremmo continuare con un numero infinito di esempi, ma torniamo ad occuparci degli ingredienti di base con i quali possiamo “comporre” i nostri programmi e le nostre applicazioni. L’essenza della programmazione, come abbiamo già anticipato nella prima puntata di questa serie di articoli, consiste nello spiegare a qualcosa che assomiglia ad un computer, cosa deve fare per ottenere ciò che abbiamo in mente. E questo richiede di aver studiato accuratamente il problema che dobbiamo risolvere, aver identificato una soluzione, verificata in tutti i suoi dettagli e condizioni operative e poi espressa in una forma di scrittura comprensibile al “computer” o qualsivoglia altra diavoleria che si incaricherà, come un fedele maggiordomo, di eseguire “scrupolosamente” le nostre disposizioni, fosse anche accendere il barbeque di domenica alle 11 dentro il garage di fianco all’automobile. Con mia figlia, quando era piccola, facevamo un gioco che chiamavamo “Ordine a

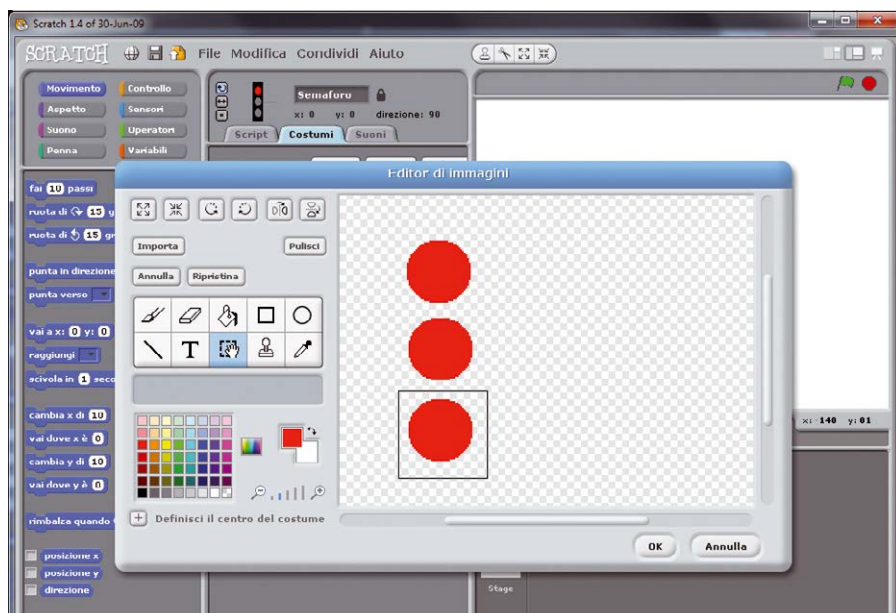
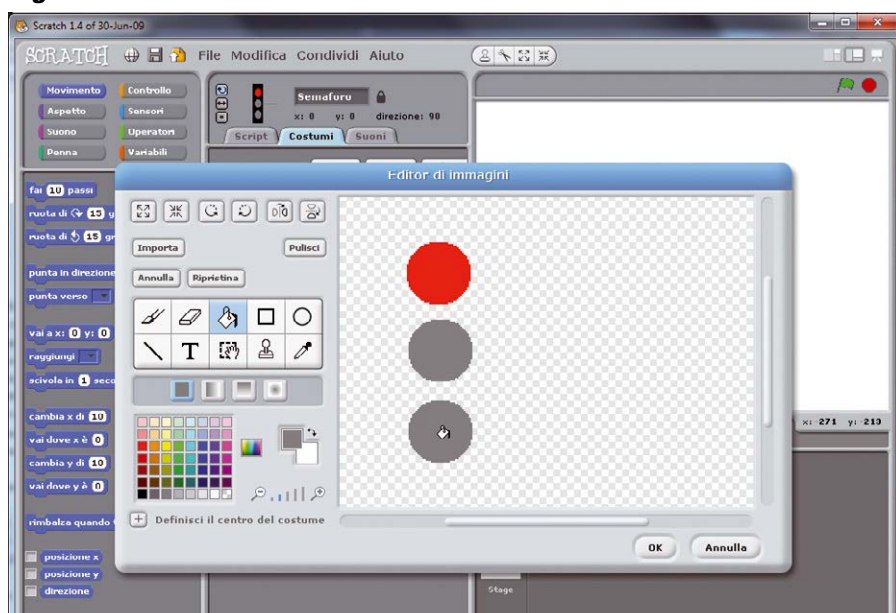
Robot”. Il robot ero io ed eseguivo “scrupolosamente” le istruzioni che mi venivano impartite, con risultati esilaranti, tipo mettere la pasta nell’acqua senza toglierla dal sacchetto. Semplificando al massimo, spiegare ad un computer cosa deve fare, significa fornire a quest’ultimo una serie di operazioni da eseguire in sequenza, nel caso ripeterle per un numero di volte indefinito o che può dipendere da particolari condizioni che si possono verificare di volta in volta, e dare disposizioni precise in modo che possa prendere “decisioni” che modificano il proprio comportamento, in base allo stato generale del sistema o di sue parti, che si concretizzano nelle cosiddette “variabili”. In una visione “moderna” della programmazione, possiamo istruire il nostro computer ad essere pronto a rilevare e rispondere a situazioni estemporanee, che vanno sotto il nome di “eventi”. Nei diversi linguaggi di programmazione intendiamo per programma un testo che contiene le tipologie di istruzioni che abbiamo appena citato, nella forma adatta ad essere poi “tradotta” in un ulteriore linguaggio comprensibile al no-

Fig. 3

stro computer, quello che viene genericamente definito come linguaggio macchina. Per poter essere tradotto senza ambiguità il testo deve essere redatto, come se si trattasse di un normale linguaggio umano, rispettando il vocabolario, la grammatica e la sintassi del linguaggio di programmazione adottato. Tipici linguaggi di programmazione comunemente adottati sono le varie "versioni" di C, il Python, il PHP, JAVA, HTML, Javascript e numerosissimi altri. Alcuni linguaggi adottano, in alternativa all'approccio testuale, un approccio grafico, dove le singole istruzioni sono caratterizzate da "simboli" grafici che ne mettono in evidenza le funzioni e, entro certi limiti, obbligano ad utilizzarle in modo corretto. Uno di questi è il linguaggio Scratch, che stiamo presentando in questi articoli, un altro esempio è il linguaggio di LabVIEW, oltre a molti altri. Un modo per rappresentare la "logica" di un programma in modo (quasi) indipendente dal linguaggio di sviluppo è il diagramma a blocchi, una serie di convenzioni grafiche che permettono di disegnare in modo schematico il flusso delle elaborazioni che devono essere eseguite dal codice che dobbiamo realizzare. In questa rivista avete avuto modo di vedere spesso diagrammi a blocchi a supporto della spiegazione della logica di funzionamento di programmi destinati a micro controllori come le schede della serie Arduino e gli integrati delle famiglie Microchip, ed a micro-computer come le diverse schede che ospitano il sistema operativo GNU/Linux.

STRUTTURE DI CONTROLLO FONDAMENTALI

Quelle che descriviamo di seguito sono le strutture di controllo

**Fig. 4**

fondamentali così come codificate nel teorema di Böhm-Jacopini, enunciato nel 1966 dagli informatici Corrado Böhm e Giuseppe Jacopini nel quale si afferma che qualunque algoritmo può essere implementato in fase di programmazione utilizzando tre sole strutture di controllo: la sequenza (o concatenazione), il ciclo (iterazione) e la selezione. Con questa impostazione ha avuto inizio l'era della programmazione strutturata e dei relativi

linguaggi di supporto. L'origine di questo approccio, storicamente, è stato sviluppato per risolvere le problematiche generate dall'utilizzo dei linguaggi di programmazione tradizionali, come COBOL, Fortran e vari dialetti Basic, che ammettono il "salto incondizionato". L'esagerazione nell'utilizzo di questo costruito ha portato alla realizzazione di programmi e applicazioni di complessità tale da essere praticamente impossibili

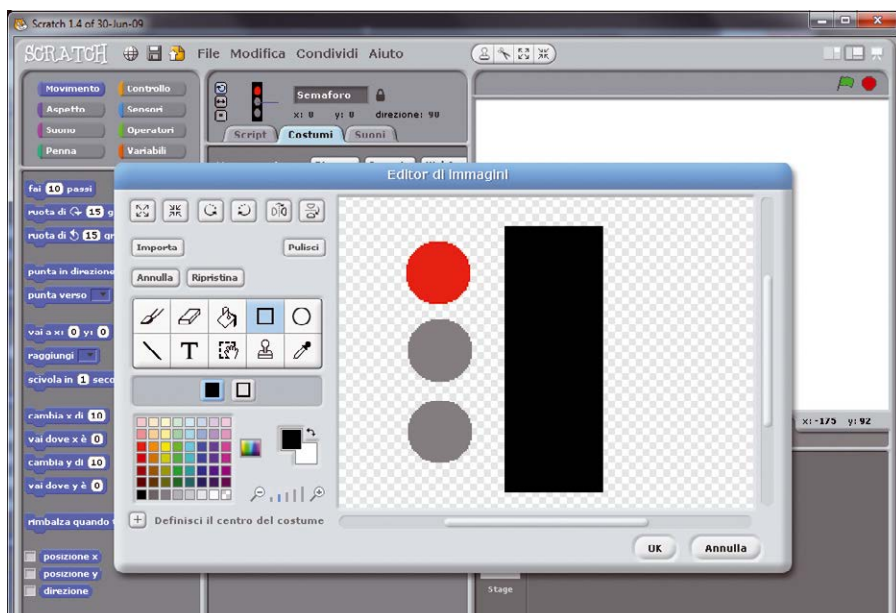


Fig. 5

asincroni rispetto al flusso principale del programma, necessità che ha dato origine al paradigma della programmazione orientata agli eventi, un paradigma comunque fortemente strutturato (event driven programming). Ci avventureremo nel seguito in queste ultime estensioni. Per ora approfondiamo le strutture di controllo proprie della programmazione strutturata: sequenza (o concatenazione), ciclo (iterazione) e selezione.

SEQUENZA (O CONCATENAZIONE)

Per sequenza si intende un blocco di istruzioni che vengono eseguite una dopo l'altra in un ordine prestabilito. Per fare un esempio, prendiamo le istruzioni necessarie ad eseguire la sequenza di funzionamento di un semaforo. Partiamo dalla configurazione iniziale, che vede il nostro semaforo con la luce verde accesa e le altre due spente. Introduciamo un'istruzione di attesa che tenga ferma la configurazione del semaforo per un certo tempo. Al termine dell'intervallo di tempo specificato passiamo a spegnere la luce verde ed accendere il giallo. Un altro periodo di attesa e poi passiamo a spegnere la luce gialla ed accendere la luce rossa. Aggiungiamo un ulteriore periodo di attesa, dopo il quale la nostra "sequenza" di istruzioni termina. Approfittiamo per documentare questa "logica" di tipo sequenziale rappresentandola graficamente con la tecnica del diagramma a blocchi, visibili in Fig. 1. Ora realizziamo praticamente questo blocco di istruzioni utilizzando il linguaggio Scratch. Approfittiamo di questo esempio per impratichirci con l'utilizzo delle funzionalità dell'ambiente di sviluppo Scratch, che abbia-

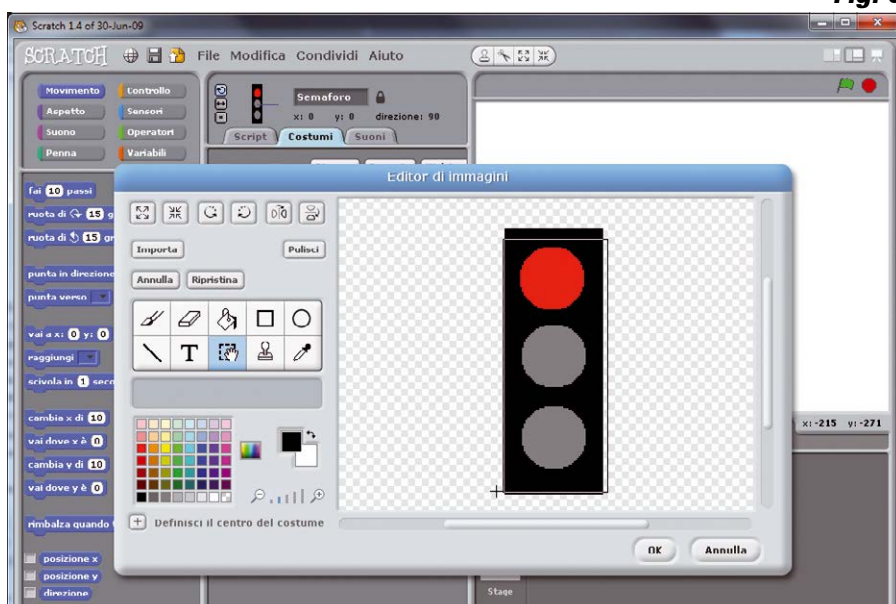


Fig. 6

da mantenere e da modificare per aggiungere nuove funzionalità. La loro complessità portò a definire questa categoria di programmi con la denominazione di "spaghetti code". La programmazione strutturata così come enunciata è sicuramente applicabile a tutti i tipi di programmi procedurali di tipo sincrono, cioè che non devono occuparsi di gestire eventi che possono accadere con modalità asincrona rispetto al ciclo logico stabilito. Un'estensione alla programmazione strutturata

è data dall'introduzione del concetto di procedura (detto anche modulo o blocco), del concetto di oggetto (programmazione orientata agli oggetti) e dalla esigenza di gestire comunque alcuni eventi asincroni rispetto al flusso principale del programma come la necessità di "intrappolare" e gestire eventuali errori e/o situazioni anomale che possono presentarsi nel corso dell'elaborazione (on conditions). Una ulteriore evoluzione è poi data dalla necessità di gestire eventi

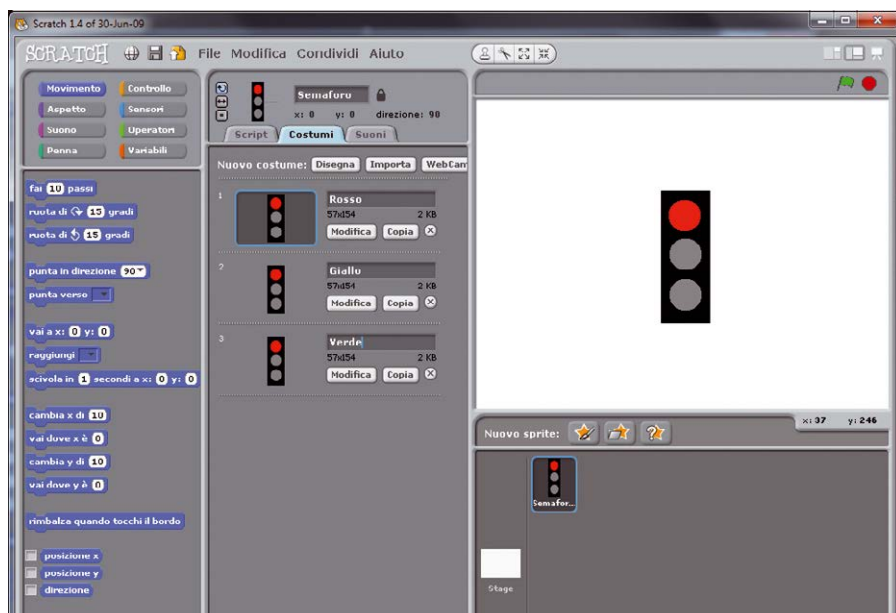
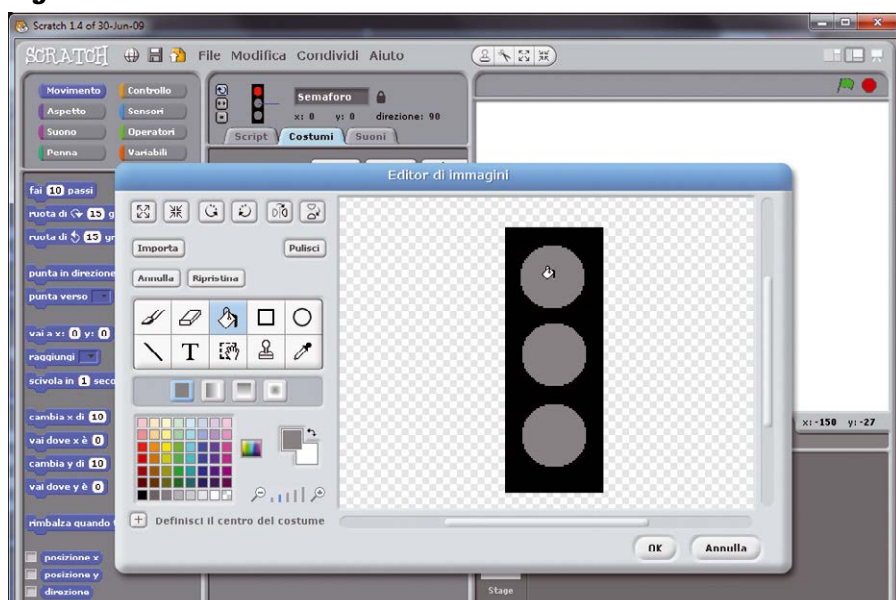
Fig. 7

mo presentato nella prima puntata di questa serie. Per prima cosa apriamo il nostro ambiente di sviluppo Scratch, cancelliamo lo sprite del gatto cliccandovi sopra col tasto destro del mouse e selezionando "Cancella". Nello spazio "Nuovo sprite" clicchiamo sulla funzione "Disegna nuovo sprite", la prima stellina a sinistra. Ci appare una finestra in stile "Paintbrush" con un'area di lavoro e gli strumenti necessari a comporre un disegno. Nella tavolozza dei colori selezionate il colore rosso. Poi selezionate lo strumento "ellissi" e con questo tracciate un cerchio come quello visibile in **Fig. 2**.

Ora selezionate lo strumento "selezione" ed "inquadrate" il cerchio appena realizzato, sempre come visibile in **Fig. 2**. Premete i tasti Control-C per duplicare la selezione, spostate il secondo cerchio al di sotto del primo ed eseguite l'operazione una seconda volta. Il risultato finale è visibile in **Fig. 3**, tre cerchi rossi allineati verticalmente.

Per realizzare la versione "rossa" del semaforo dobbiamo dare un colore neutro alle due luci inferiori. Nella tavolozza dei colori clicchiamo su un grigio, selezioniamo lo strumento "riempimento" (il secchiello che rovescia il colore) e clicchiamo al centro del secondo cerchio e, successivamente, del terzo. Vediamo il risultato in **Fig. 4**.

Per disegnare il "corpo" al semaforo, selezioniamo il colore nero dalla tavolozza, selezioniamo lo strumento "rettangolo", e tracciamo un rettangolo a destra dei cerchi in grado di contenerli tutti con un po' di margine. Se qualche operazione riesce male, niente paura. Si può tornare indietro con i tasti Control-Z oppure utilizzare lo strumento "gomma da cancellare". Notate che potete

**Fig. 8**

personalizzare gli strumenti di disegno. Ad esempio per lo strumento gomma, sotto il pannello degli strumenti, potete scegliere lo spessore dello strumento, più o meno fine. Per lo strumento riempimento potete scegliere la sfumatura, mentre per i poligoni potete scegliere se realizzarli pieni o con il solo contorno. Una volta ottenuto il risultato visibile in **Fig. 5**, selezionate ancora una volta lo strumento "selezione" e "circondare i tre cerchi, per poi trascinarli all'interno del rettan-

golo nero, come in **Fig. 6**. Per posizionare correttamente la selezione potete aiutarvi con i tasti freccia. Una volta soddisfatti del risultato confermate con il tasto "OK" e tornate nella finestra principale dell'IDE di Scratch. Se non è attivo il pannello "Costumi" lo attiviamo cliccando sulla linguetta "Costumi". A questo punto avete il disegno del semaforo come sprite ed il primo "costume" disponibile, quello con la versione "rossa" del semaforo. Per realizzare le versioni "gial-

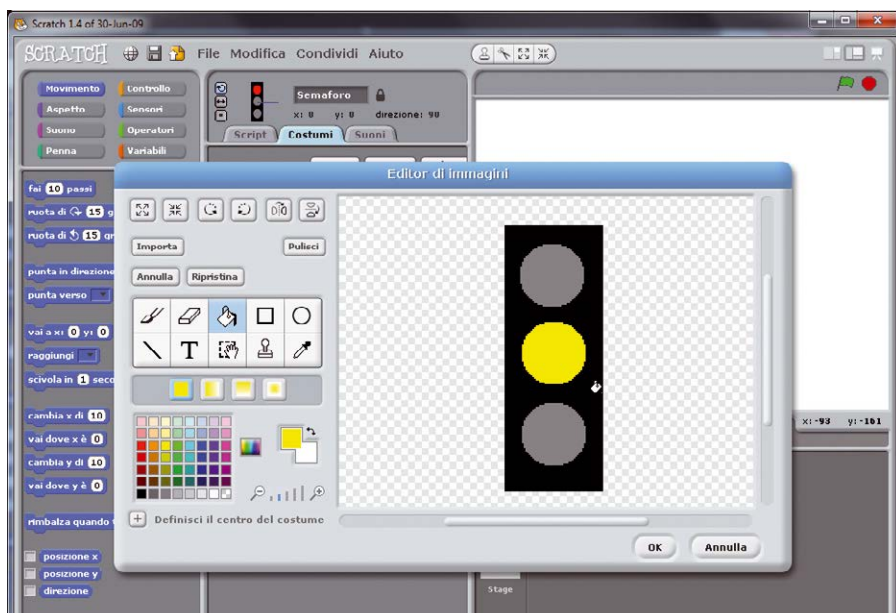


Fig. 9

Ora selezionate un bel giallo dalla tavolozza dei colori e cliccate sulla luce centrale, come in **Fig. 9**. Salvate con il pulsante "OK" e ripetete l'operazione per il costume "verde" fino ad ottenere il risultato visibile in **Fig. 10**.

Ora è tempo di animare il semaforo con le istruzioni che concretizzano il diagramma a blocchi di **Fig. 1** e che potete vedere nella **Fig. 11**. Il significato delle istruzioni si commenta da solo. Per realizzarlo cliccate sullo sprite "Semaforo" nell'area in basso a destra e poi cliccate sulla linguetta "Script". Nell'area degli script riproducete il blocco di **Fig. 11**. I componenti si trovano nelle "collezioni" selezionabili mediante i pulsanti in alto a destra. I colori che evidenziano i pulsanti corrispondono al colore dei blocchi contenuti. Quindi il cappello con la bandierina, che è un blocco di colore giallo lo trovate nella collezione "Controllo" contraddistinta dalla fascetta di colore giallo. Così per gli altri blocchi. Per "portare" i blocchi nell'area degli script basta trascinarli con il mouse e rilasciarli dove desiderato. Abbiamo già descritto nella prima puntata le modalità di editing degli script. Per comodità ne riportiamo un riassunto. Quando si trascina un'istruzione all'interno di uno script il posizionamento viene facilitato dall'apparire di una linea bianca che indica che in quel punto può essere "agganciata" l'istruzione che si sta trascinando. Blocchi di istruzioni possono essere staccati e spostati oppure duplicati con le apposite funzioni raggiungibili cliccandovi sopra con il tasto destro del mouse. Per eliminare un'istruzione dallo script bisogna prima "staccarla" dallo script trascinandola in una zona libera dell'area degli script. Poi cliccarvi sopra col

Fig. 10

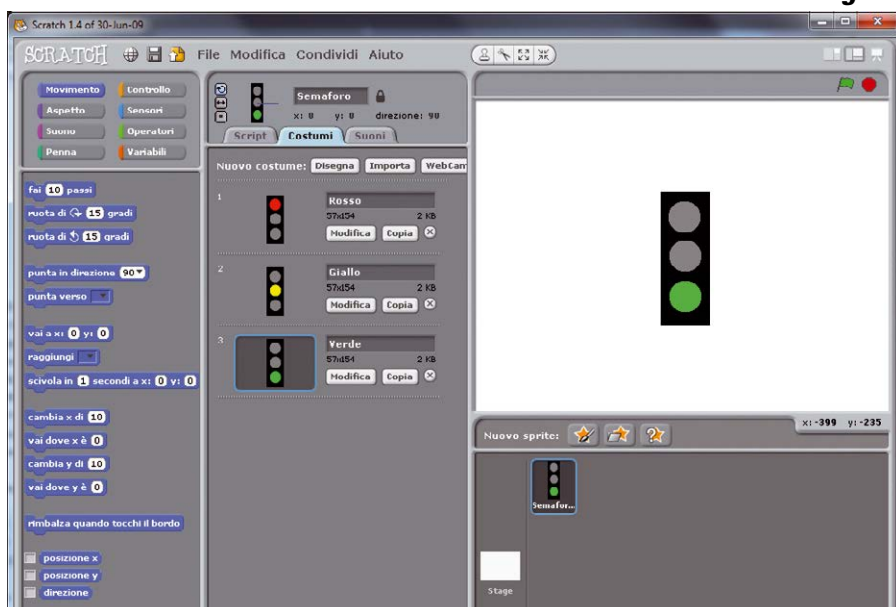


Fig. 11



la" e "verde" clicchiamo per due volte sul bottone "copia" a fianco del primo "costume". Approfit-

tate per dare dei nomi significativi allo sprite ed ai suoi costumi. Noi abbiamo scelto il nome "Semaforo" per lo sprite e "rosso", "giallo" e "verde" per i tre costumi come visibile in Fig. 7. Ora cliccate sul pulsante "Modifica" del costume "giallo". Nella finestra di modifica selezionate lo strumento "pipetta" cliccate sul colore grigio di una delle due "luci", selezionate lo strumento "secchiello" e cliccate sulla luce rossa, in modo da riempirla di grigio come in **Fig. 8**.

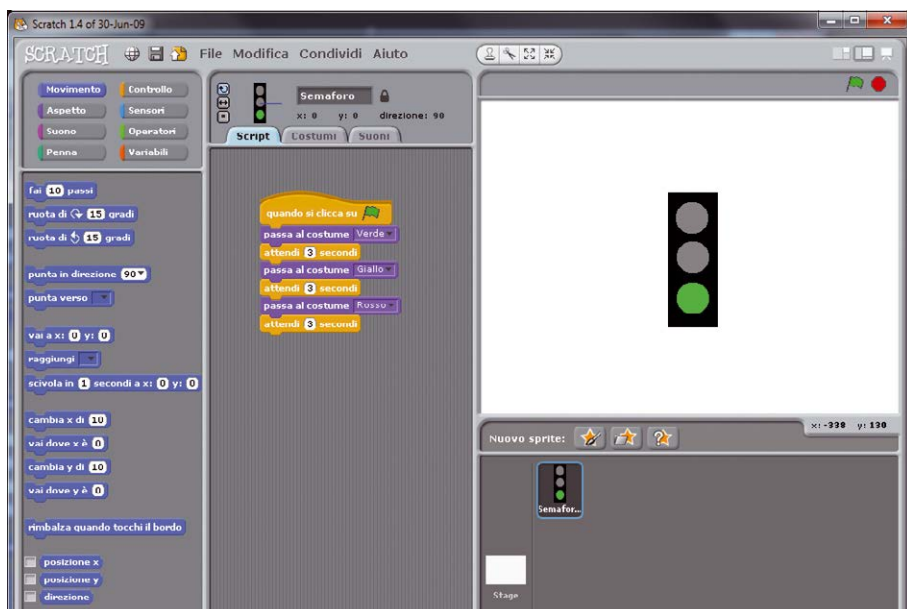


Fig. 12



Fig. 13



Fig. 14



Fig. 15



Fig. 16

tasto destro del mouse e selezionare "Cancella". Per selezionare il costume nelle istruzioni "passa al costume", basta selezionarli dalla lista che si apre cliccando sulla piccola freccetta nera rivolta verso il basso. Per variare il tempo nelle istruzioni "attendi <x> secondi", cliccate nell'area bianca dell'istruzione ed inserite il valore desiderato. Alla fine il risultato deve apparire come in Fig. 12.

Cliccate sulla bandierina verde per mettere in funzione il semaforo.

CICLO (O ITERAZIONE)

Carino il nostro semaforo? Peccato che funzioni una sola volta, per poi rimanere congelato nella configurazione finale, un rosso perenne che crea una coda di veicoli tendente a infinito. Per

Fig. 19

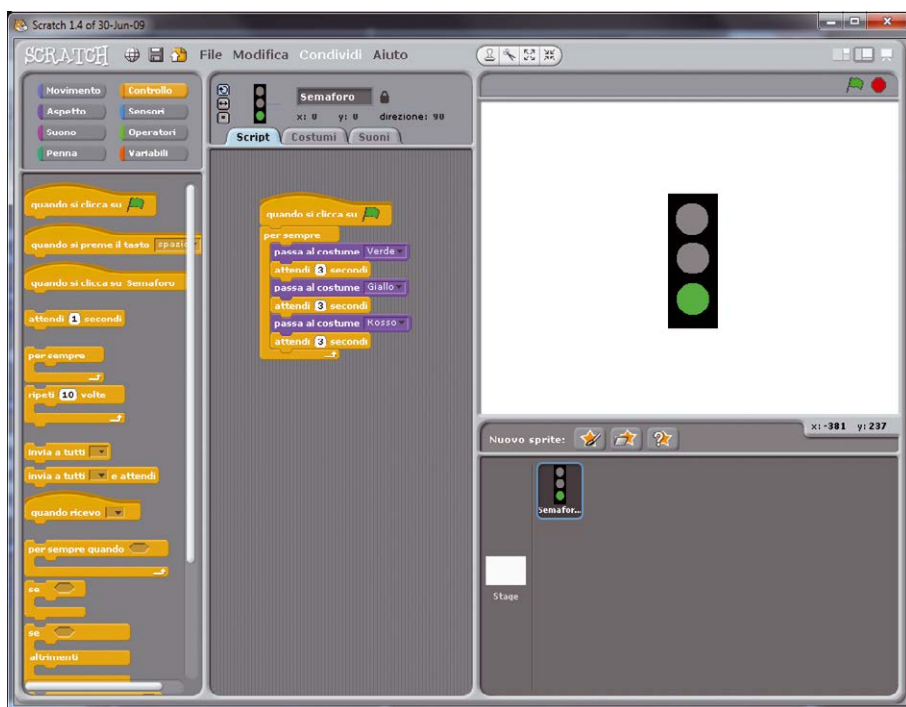


Fig. 17



Fig. 18



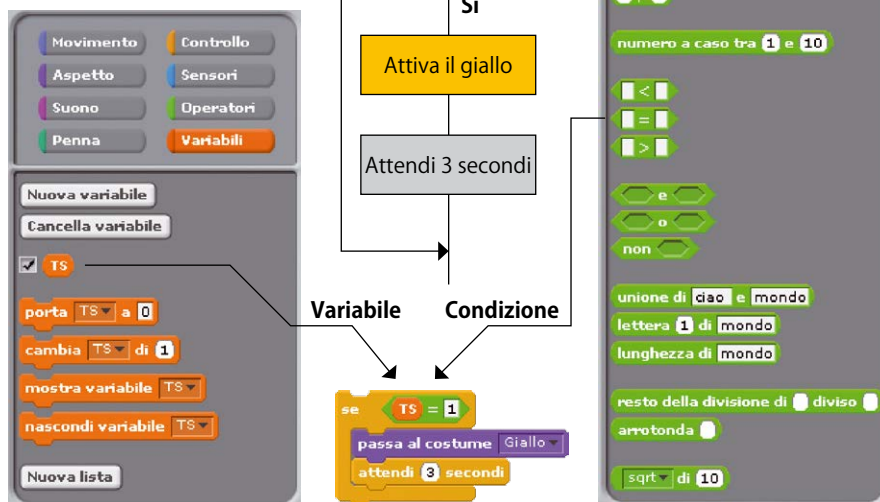
Fig. 20



Fig. 21

comportarsi da vero semaforo, dopo un certo periodo di luce rossa deve tornare il verde e così di seguito. Il blocco di istruzioni che realizza il comportamento del semaforo, una volta terminato, deve essere rieseguito dall'inizio, in un ciclo continuo senza fine. Per realizzare questo comportamento ci viene in aiuto la struttura di controllo "ciclo". In realtà esistono diverse "versioni" della struttura ciclo, a seconda delle "regole" in base alle quali vogliamo che sia controllata la ripetizione del blocco di istruzioni. A seconda dei linguaggi la struttura di controllo ciclo è implementata dalle istruzioni "while", "loop", "repeat", "for" e "do ... until". La grande distinzione sta tra i cicli incondizionati come "loop" o, in Scratch, "forever" o "per sempre" in italiano (Fig. 13), ed i cicli condizionati. Questi ultimi a loro volta possono essere "pilotati" per eseguire un blocco di istruzioni per un certo numero di volte (in Scratch, per esempio, "repeat 5" o "ripeti 5 volte" (Fig. 14) oppure, in base al verificarsi o meno di condizioni particolari, come "per sempre quando <condizione>" (Fig. 15) o "ripeti fino a quando <condizione>" (Fig. 16). Siccome il blocco di istruzioni che anima il nostro semaforo deve essere ripetuto incondizionatamente, adottiamo il blocco "per sempre". Il diagramma a blocchi della logica di funziona-

Fig. 22



mento del semaforo viene modificato come quello visibile in Fig. 17, dove il flusso di elaborazione, una volta terminato, torna all'inizio della sequenza. Per tradurlo nel linguaggio Scratch, visibile in Fig. 18, "stacciamo" la sequenza di istruzioni dal "cappello" iniziale. Nel pannello "Controllo" identifichiamo il blocco a forma di "C" "per sempre" e lo trasciniamo nell'area

degli script agganciandolo sotto al cappello con la bandierina verde. Poi trasciniamo il blocco di istruzioni, che "governano" il semaforo, all'interno della "bocca" del blocco "per sempre". Fatto? Verificate di trovarvi nella condizione equivalente a quella visibile in Fig. 19. Cliccate sulla bandierina et voilà, ora il semaforo funziona in modo corretto. Divertitevi a modificare il suo

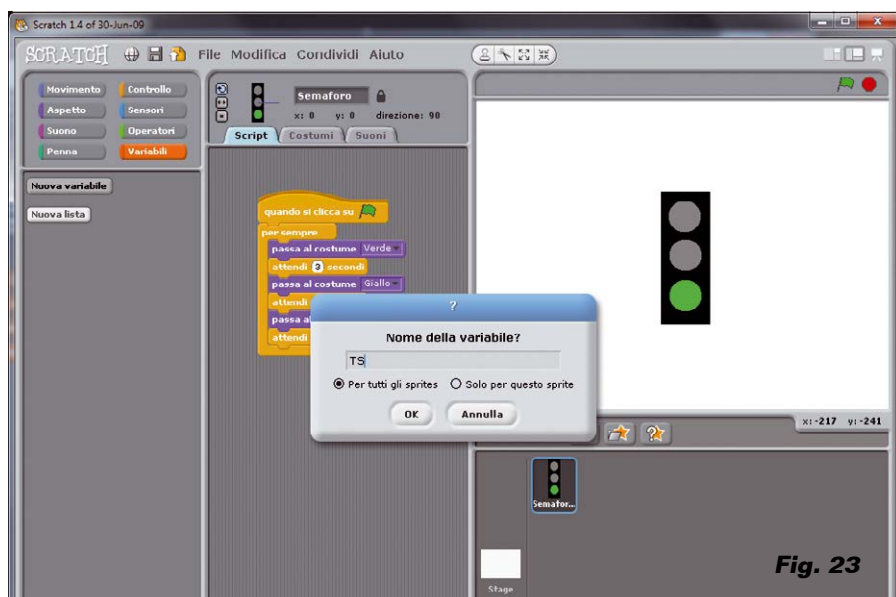


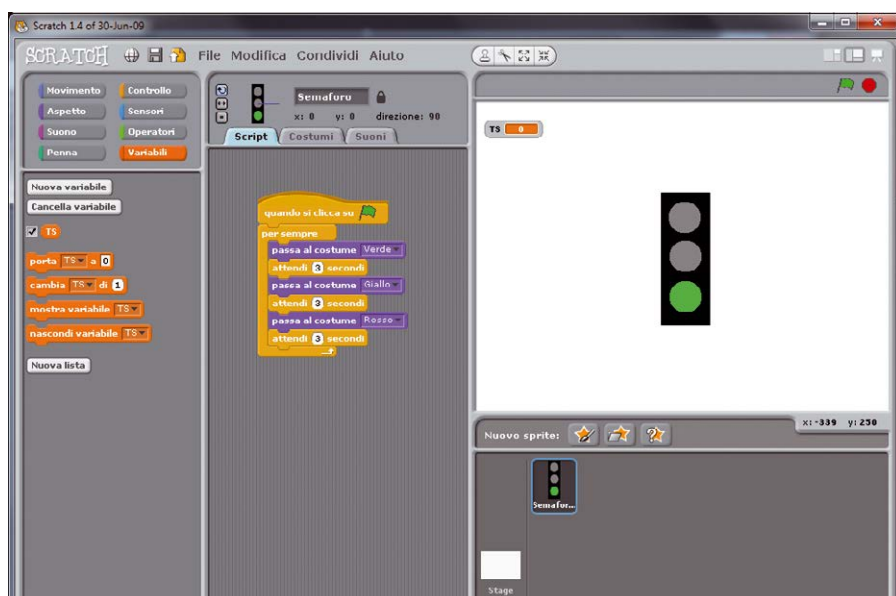
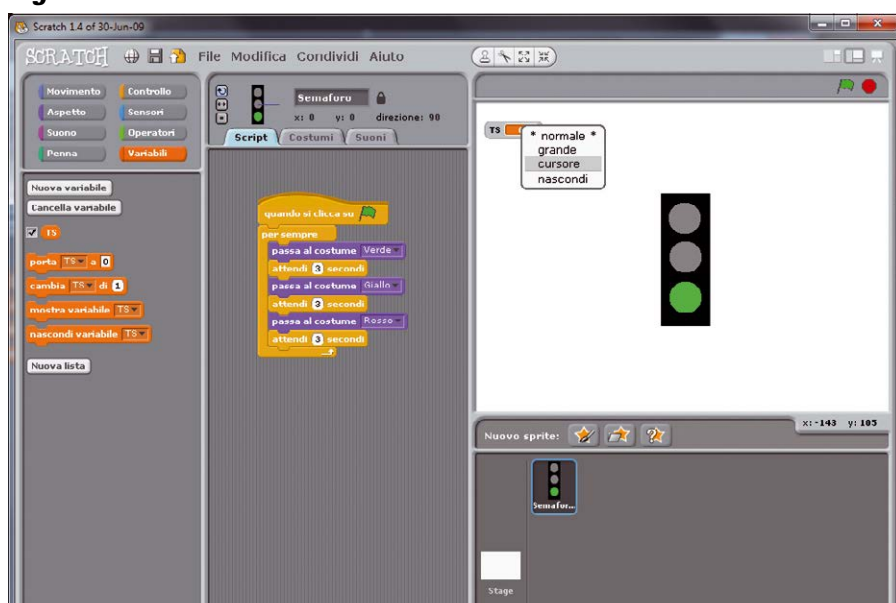
Fig. 23

Fig. 24

comportamento, per esempio ripassando dal giallo tra rosso e verde, come avviene in alcuni nazioni, o modificando le temporizzazioni. Ora è il momento di cominciare a complicarci la vita, ovvero impariamo a prendere delle decisioni.

SELEZIONE

Bella la vita del semaforo anche se decisamente noiosa e, per dirla tutta, non particolarmente intelligente. Il suo unico scopo è di cambiare colore a intervalli fissi. Poco importa se sul lato che diventa verde non c'è nessuno e dalla parte rossa interrompe un flusso di veicoli chilometrico. O, peggio ancora, non prolunga il giallo finché da un lato non si libera l'incrocio, dando via libera nell'altra direzione, generando un ingorgo inestricabile. Nella maggior parte delle attività umane, o che gli umani delegano alle macchine, i problemi non possono essere risolti con la semplicità degli esempi precedenti. A seconda delle condizioni che si presentano, occorre prendere delle decisioni. Ciò è vero sia per gli individui umani che per le macchine. Per questo ci viene in aiuto la struttura di controllo della "selezione", ovvero la capacità di intercettare e valutare diverse tipologie di condizioni che si possono verificare, ed in base a queste, condizionare l'esecuzione di alcuni blocchi di istruzioni in alternativa ad altri. Per esempio potremmo voler fare in modo che il semaforo sia predisposto per funzionare sia con il passaggio del giallo tra il rosso ed il verde che con il passaggio diretto dal rosso al verde. Tutto questo utilizzando un flag o "variabile" di configurazione, impostabile dall'esterno, senza bisogno di dover modificare il codice del programma per

**Fig. 25**

cambiarne il modo di funzionamento. Per prima cosa dobbiamo creare la variabile "TS" (Tipo Semaforo). Poi modifichiamo il comportamento del programma introducendo la struttura di controllo selezione nella gestione del giallo durante la transizione dal rosso al verde. A questo proposito specifichiamo la possibilità di scegliere tra due differenti "sfumature" del blocco di controllo selezione: la selezione ad una via e la selezione a due vie. La prima possibilità permette

di controllare l'esecuzione di un blocco di istruzioni al verificarsi della condizione che pilota la struttura di controllo, nel caso la condizione non sia verificata il blocco viene saltato e l'elaborazione prosegue con l'istruzione successiva alla chiusura del blocco, dopo la parte inferiore della "Bocca" (Fig. 20). La seconda possibilità permette di eseguire due blocchi di istruzioni in alternativa a seconda del verificarsi o meno della condizione che pilota la struttura di controllo (Fig. 21).

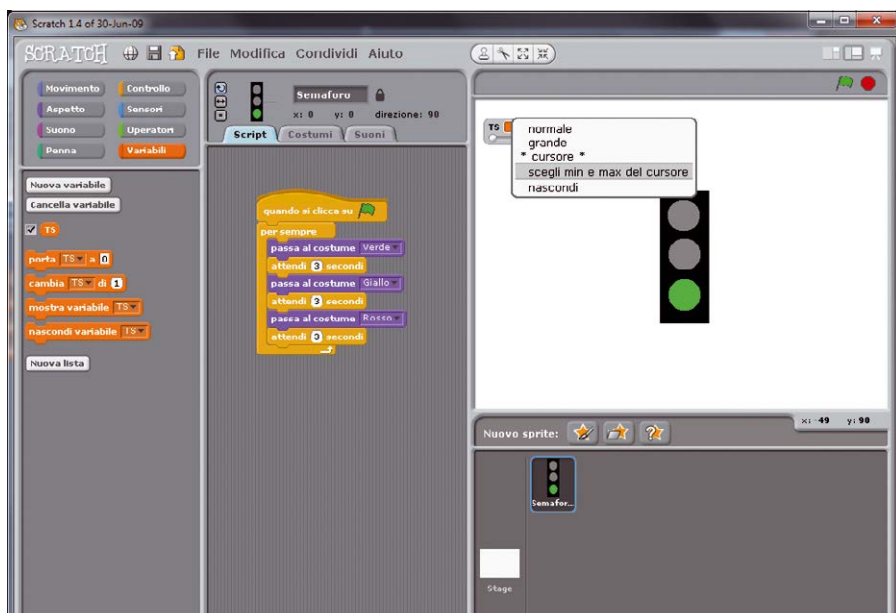


Fig. 26

riabile, corredata dei blocchi di istruzioni per poterla gestire. Il primo blocchetto in alto con il segno di spunta è il riferimento della variabile stessa, da trascinare negli spazi di forma corrispondente nelle istruzioni. Il segno di spunta permette di visualizzare la variabile sul "palcoscenico" (Stage).

Lasciate il segno di spunta e cliccate col tasto destro del mouse sul segnaposto della variabile sul palcoscenico.

Selezionate l'opzione "cursore" come in Fig. 24.

Questa operazione fa apparire un cursore al di sotto della variabile, che può essere utilizzato per modificarne il valore, anche con il programma in esecuzione. Proprio quello che ci serve, però i valori possibili devono essere limitati a "0" e "1". Quindi cliccate di nuovo col tasto destro del mouse sul segnaposto della variabile e selezionate la voce di menu "scegli min e max del cursore" come in Fig. 25.

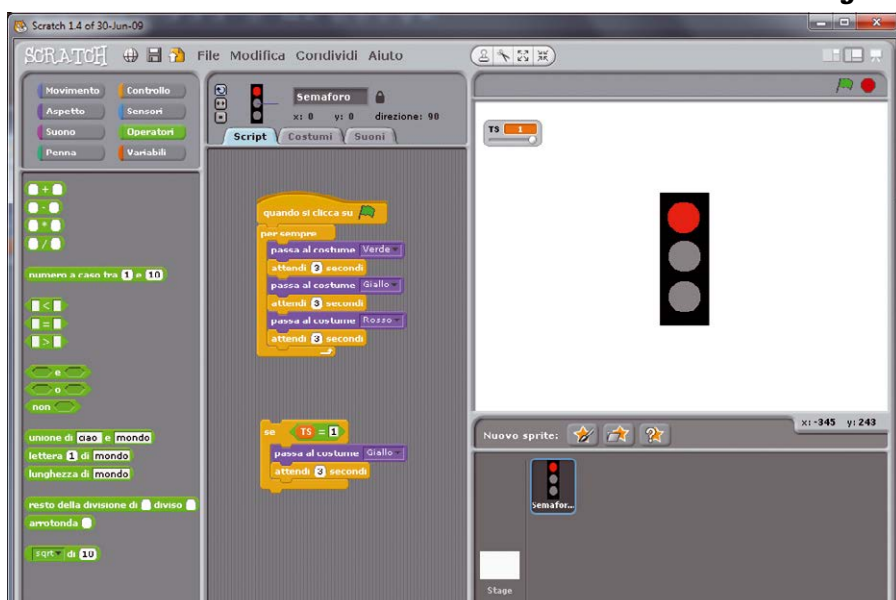
Nella finestra di dialogo indicate "0" come valore minimo e "1" come valore massimo come in Fig. 26.

Create il blocco "se" trascinando e collocando i vari "pezzi" come indicato nello schema di Fig. 22 ed inseritelo all'interno del blocco principale come visibile in Fig. 27.

Potete mettere in funzione il semaforo cliccando sulla bandierina verde.

A seconda della posizione del cursore della variabile "TS", e quindi del valore impostato nella variabile stessa, vedrete il semaforo comportarsi in modo diverso nella transizione dal colore rosso al verde, con o senza passaggio per il giallo. Se volete conservare il vostro lavoro salvatelo selezionando "Salva con nome" dal menu "File".

Fig. 27



Per il nostro semaforo è sufficiente adottare la struttura ad una via, come visibile in Fig. 22. Nel caso la variabile TS tenga il valore "1", facciamo apparire il giallo nella transizione tra rosso e verde, altrimenti saltiamo il blocco di istruzioni ed eseguiamo in sequenza le istruzioni successive. Per realizzare il blocco di selezione visibile in Fig. 22 dovete per prima cosa creare la variabile TS. Per creare la variabile TS assicuratevi di aver selezionato

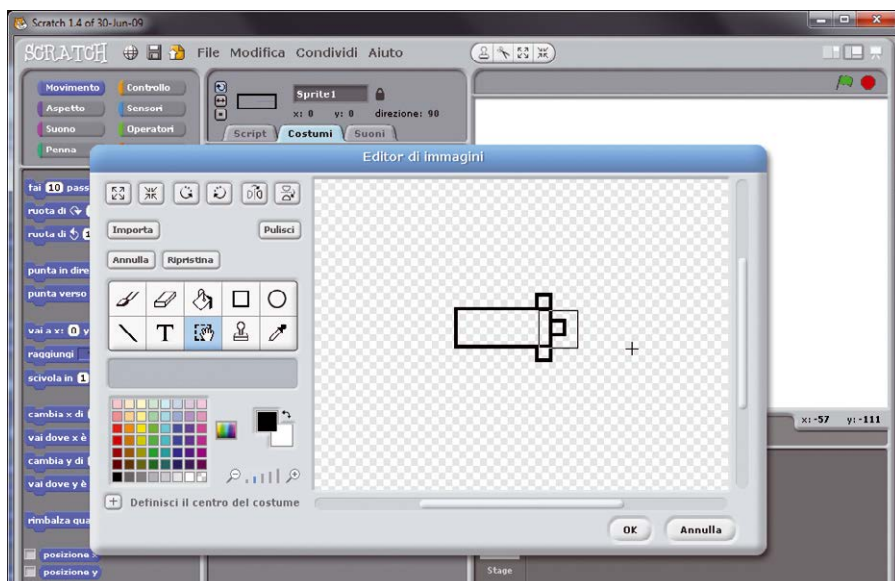
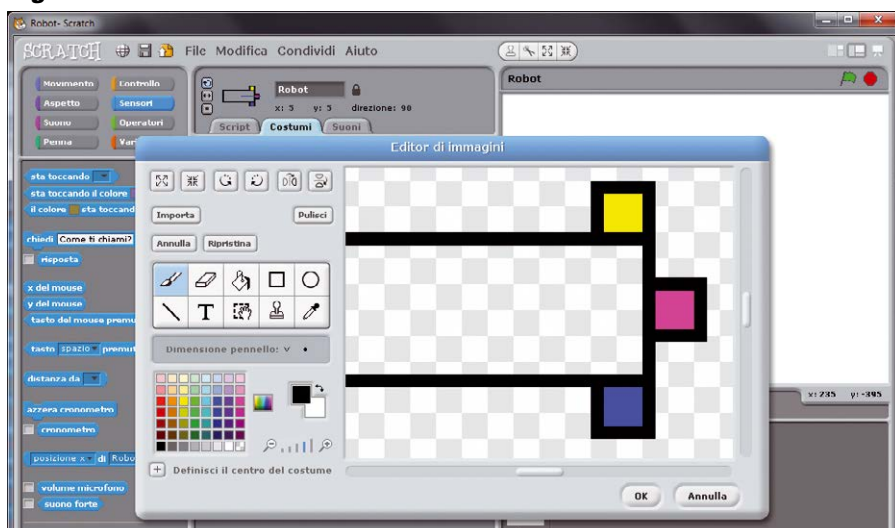
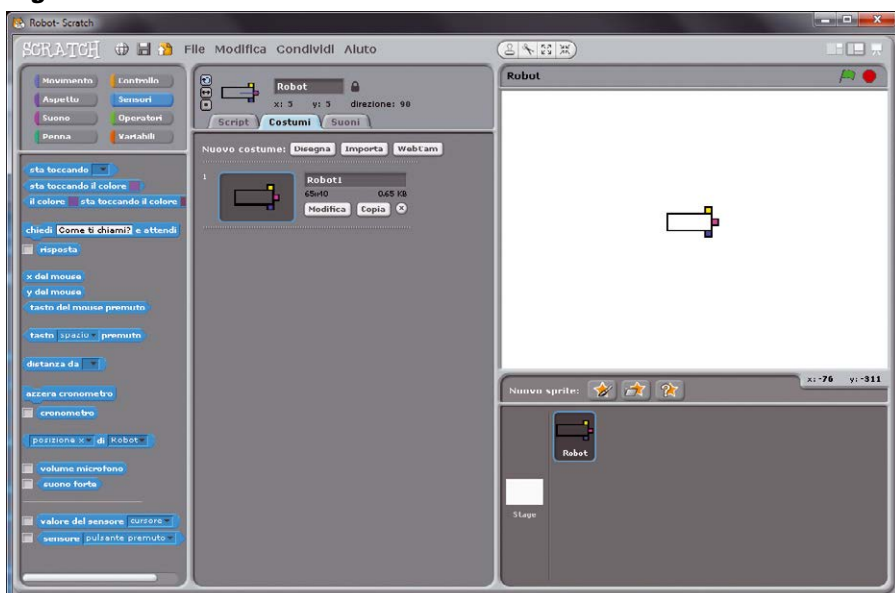
lo sprite "Semaforo", per essere certi cliccate sullo sprite stesso nell'area in basso a destra. Poi cliccate sul pulsante "Variabili" nel menu delle collezioni di blocchi in alto a sinistra. Nel pannello che si apre cliccate sul pulsante "Nuova variabile". Si aprirà una finestra di dialogo nella quale inserire il nome della variabile, nel nostro caso "TS", come visibile in Fig. 23. Lasciate invariato il resto e confermate cliccando sul tasto "OK". Vedrete apparire la nuova va-

Fig. 28

FACCIAMOCI UN ROBOT

Una volta familiarizzato con le “strutture” di base della programmazione possiamo cimentarci con qualcosa di decisamente più ambizioso della gestione di un semaforo. Per esempio un simulacro di robot che deve seguire un percorso realizzato con una linea nera su un “pavimento” bianco. Con il linguaggio Scratch? Sì, con il linguaggio Scratch. Normalmente i robot che “inseguono” una linea sono dotati frontalmente di un fila di sensori sensibili all’infrarosso, almeno tre, che misurano il grado di riflessione del terreno sottostante, alto nel caso il colore sottostante sia bianco e basso nel caso sia nero. Nella simulazione del robot in Scratch, al posto dei sensori tradizionali, utilizziamo un metodo basato sui colori, in modo da poterci avvalere della condizione di selezione, disponibile nella collezione Scratch, “se il colore <X> sta toccando il colore <Y>”. A questo punto possiamo usare dei sensori a “colori” che possiamo utilizzare con una logica simile a quella utilizzata con i sensori “veri”. Prima di approfondire la “logica” di controllo realizziamo il nostro robot ed il “terreno” con la linea nera da seguire. Dopo aver fatto questo la logica di controllo apparirà più chiara. Aprite un nuovo progetto, cancellate lo sprite gatto e cliccate sulla stellina “Disegna nuovo sprite”. Disegnate una struttura come quella visibile in **Fig. 28**.

Il rettangolo di dimensioni maggiori rappresenta il corpo del robot mentre i tre quadrati più piccoli costituiscono i sensori. Per disegnare il corpo del robot selezionate lo strumento “pennello”, primo in alto a sinistra, selezionate un tratto “fine” (per esempio la seconda selezione

**Fig. 29****Fig. 30**

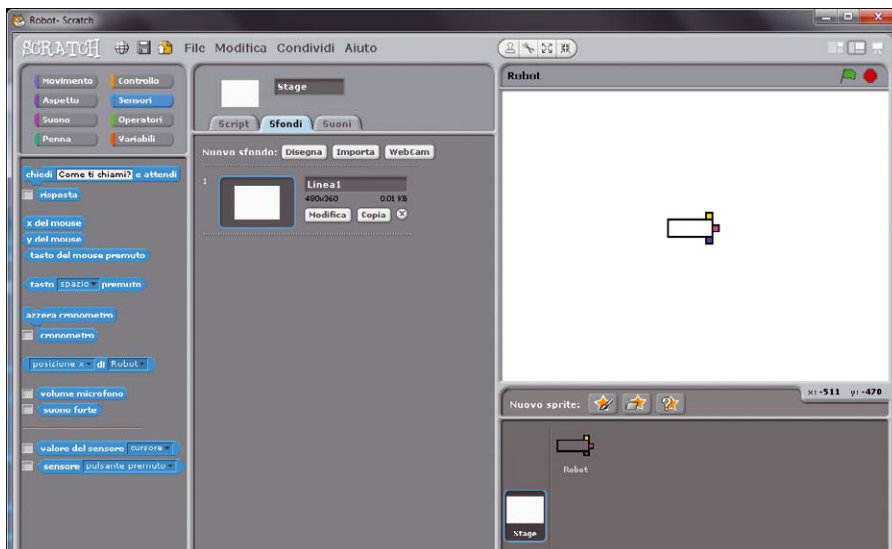


Fig. 31

disponibile), e tracciate un rettangolo dalle proporzioni come quello in figura. Poi disegnate un quadrato, in un qualunque punto del palcoscenico. Selezionate lo strumento "selezione". Selezionate il quadrato, cliccate col tasto destro del mouse e selezionate "Copia", trascinete la copia nella posizione di uno dei sensori. Risselezionate, copiate di nuovo l'originale e trascinatela nella posizione del secondo sensore. Infine selezionate il quadrato originale e trascinatelo nella posizione del terzo sensore. Ora dovete "personalizzare" ciascun sensore con un colore diverso. Allargate il disegno con lo strumento "zoom" ovvero le tacchette in basso a destra della tavolozza. Cliccando su una o l'altra tacchetta si allarga o restringe il disegno. Ora colorate i tre sensori con colori diversi, noi abbiamo scelto il giallo, il blu ed il viola. Selezionate il colore giallo cliccandovi sopra nella tavolozza. Poi selezionate lo strumento "secchiello" e riempite il sensore di sinistra con il colore giallo. Vi consigliamo di seguire lo schema di colori che abbiamo adottato in modo da avere riscontro con le istruzioni e la logica complessiva dello script. Ripetete l'operazione per i colori blu e viola fino ad ottenere il risultato finale visibile in Fig. 29. Confermate cliccando sul pulsante "OK" e date un nome allo sprite ed al costume appena realizzati, per esempio "Robot" e "Robot1", come visibile in Fig. 30. Ora dobbiamo preparare il "percorso" che il nostro robot dovrà seguire. Cliccate sull'icona "Stage" nell'area in basso a destra e nel pannello date un nome allo "sfondo" e cliccate sul pulsante "Modifica" (Fig. 31). Nella finestra di disegno selezionate lo strumento "pennello",

Fig. 32

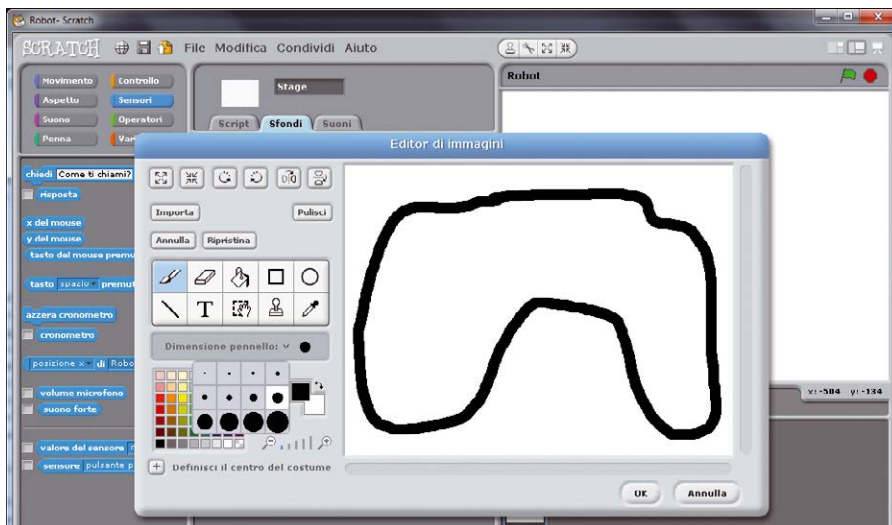


Fig. 33

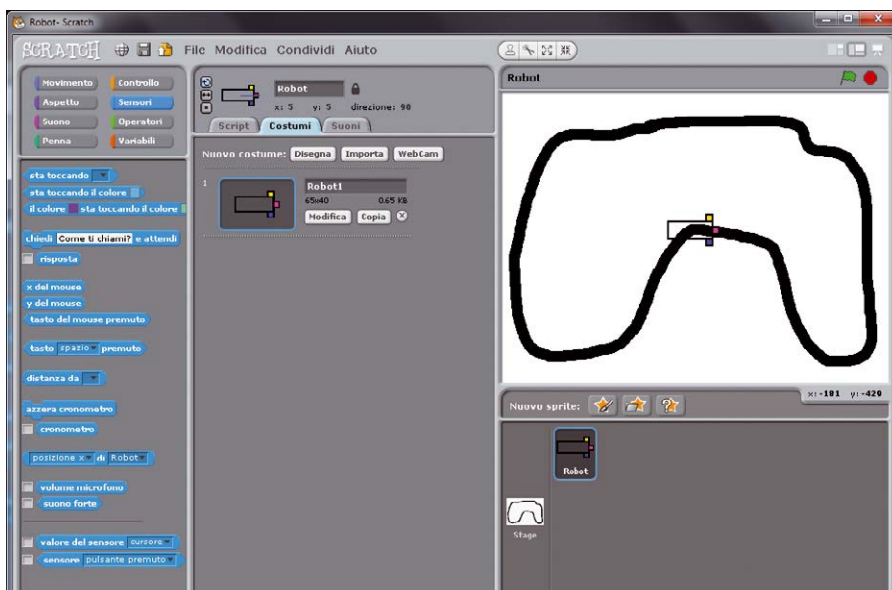
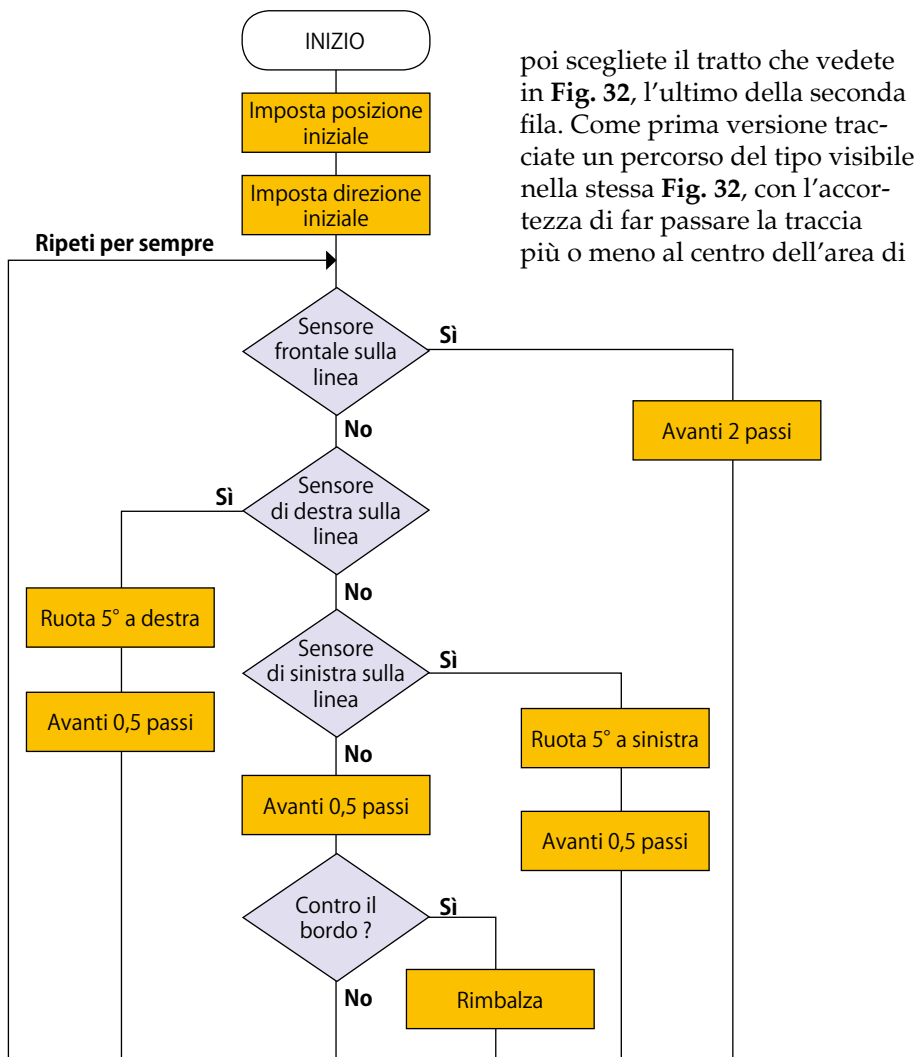


Fig. 34



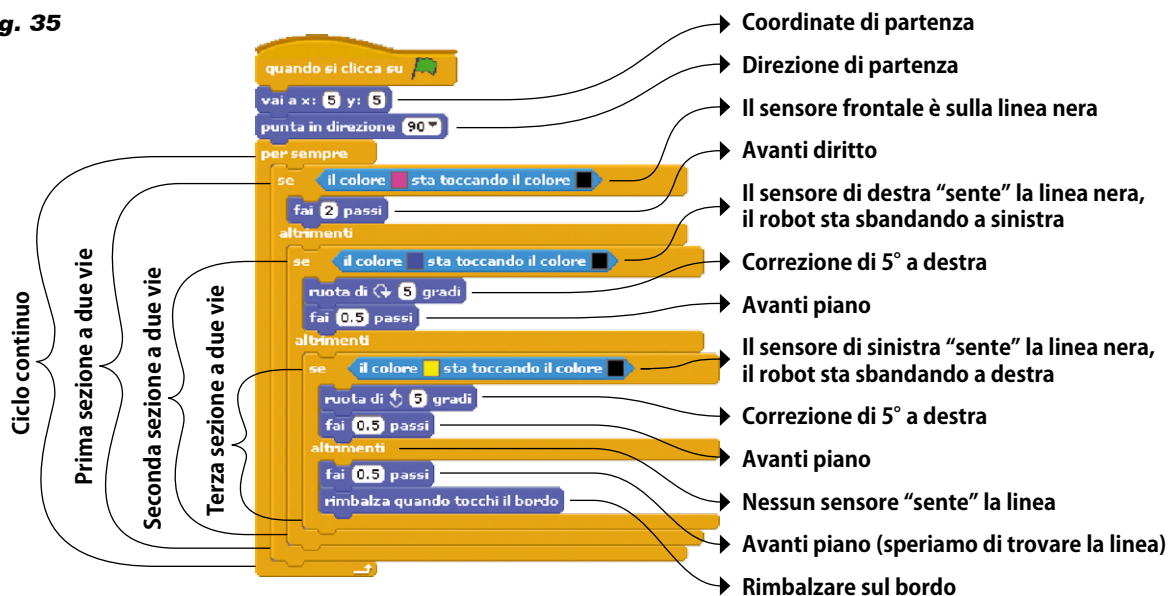
poi scegliete il tratto che vedete in Fig. 32, l'ultimo della seconda fila. Come prima versione tracciate un percorso del tipo visibile nella stessa Fig. 32, con l'accuratezza di far passare la traccia più o meno al centro dell'area di

disegno. Confermate con "OK". Il risultato finale è visibile in Fig. 33.

Ora dobbiamo istruire a dovere il nostro Robot per fargli seguire la linea nera. È a questo punto che entra in gioco la capacità di ragionare in modo logico e creativo per trovare una soluzione al problema. Le variabili in gioco sono molte, la tipologia del percorso con curve più o meno accentuate, il posizionamento dei sensori, la forma del robot, eccetera. Una prima semplice strategia, adatta al robot ed al tipo di percorso che avete disegnato può essere la seguente:

- Se il sensore frontale, quello viola sta "toccando" la linea nera, il robot prosegue dritto;
- Se il sensore di "destra", di colore blu, sta toccando la linea nera, significa che il robot sta "sbandando" a sinistra, oppure che la linea sta incurvandosi verso destra. In questo caso il robot deve "girare" verso destra di un certo angolo e, prudenzialmente, avanzare più lentamente;
- Se il sensore di "sinistra", di

Fig. 35



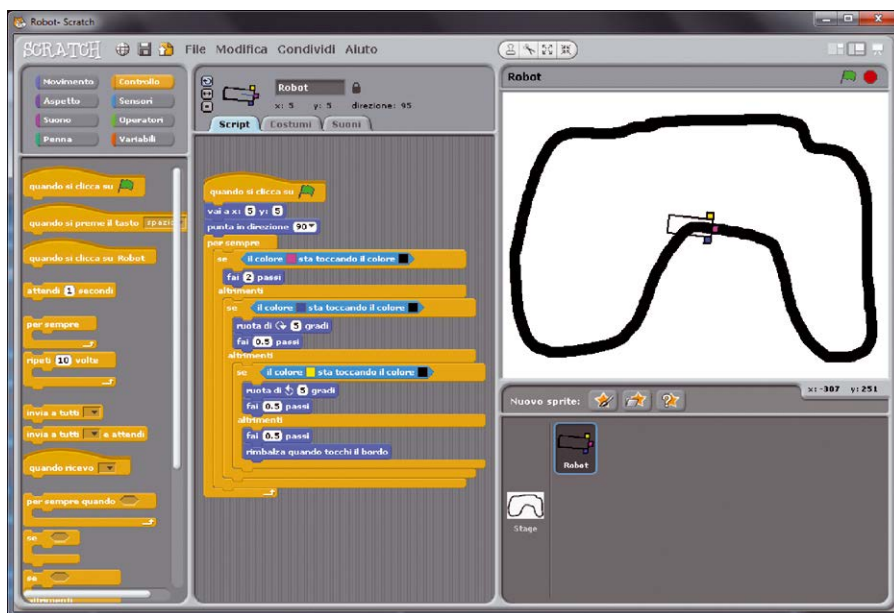


Fig. 36

caso viene eseguito il blocco di istruzioni contenuto nella prima "bocca" della selezione a due vie. Se la condizione non è soddisfatta viene eseguito il blocco di istruzioni contenuto nel ramo "else" della selezione, ovvero la seconda "bocca" del blocco di selezione. Una volta terminato il lavoro dovrete trovarvi nella situazione visibile in Fig. 36. Salvate il progetto con un nome, se non l'avete ancora fatto.

Ora potete provare il tutto cliccando sulla bandierina verde e mettendo in moto il robot. Se per caso funziona male e non segue la linea, il problema potrebbe essere la linea di spessore troppo grande, che impegna più di un sensore alla volta, oppure la distanza tra i sensori troppo piccola. Rifate il disegno del percorso con una linea meno spessa oppure disegnate un robot leggermente più largo. Carino? A questo punto potete sperimentare moltissime opzioni alternative. Modificate il percorso rendendolo più difficile. Spostate i sensori o cambiate la forma del robot. Modificate i parametri di sterzata e di velocità per tentare il record del percorso. Introdurrete incroci a "T" e a forma di "+" come avviene per le prove della RoboCup. Ovviamente dovrete modificare la strategia, e magari la composizione del robot con un numero maggiore di sensori in posizioni diverse. Nelle nuove strategie potete sfruttare le condizioni composte "AND" e "OR", aumentando il numero dei blocchi di selezione utilizzati. Altra possibilità è avvalersi della condizione "distanza da <X>" inserendo nel palcoscenico degli sprite "ostacoli" e provando ad evitarli, oppure a passare tra due sprite "barriera". Fatevi aiutare da amici e bambini. Il divertimento è assicurato. ■

colore giallo, sta toccando la linea nera significa che il robot sta sbandando verso destra oppure che la linea nera sta piegando a destra. Il robot deve girare verso sinistra di un certo angolo ed avanzare più lentamente;

- Infine, se nessuno dei sensori tocca la linea nera, significa che il robot si è perso, oppure la linea è interrotta (provate, cancellando un pezzo di linea). Il robot rallenta la velocità e prosegue dritto, sperando di incontrare la linea nera di nuovo, prima o poi.
- Nel caso vada a sbattere contro un bordo, deve rimbalzare, rientrando in campo e proseguendo come prima.

Abbiamo tradotto questa strategia nel diagramma a blocchi di Fig. 34, ed in base a questo abbiamo realizzato lo script Scratch visibile in Fig. 35.

Lo script prevede un ciclo "per sempre" e tre blocchi di selezione a due vie "annidati" in modo da coprire tutti i casi previsti nella strategia. Potete ovviamente adottare tutte le soluzioni alternative che esprimano lo stesso comportamento "logico", per

esempio utilizzando una sequenza di blocchi di selezione ad una via. Diverse soluzioni possono differire per il comportamento e per le prestazioni, provate più che potete per fare esperienza. Componete lo script di Fig. 35 attivando lo sprite "Robot", selezionando il pannello "Script" e poi "scovando" i blocchi nelle diverse collezioni e trascinandoli sull'area dello script, collegandoli assieme nel modo corretto. Un'unica nota relativamente alla condizione di selezione "se il colore <X> sta toccando il colore <Y>". Per impostare correttamente i colori, cliccate sul primo quadratino della condizione, poi andate a selezionare il colore cliccando sul sensore desiderato. Controllate bene il risultato, bisogna operare con una certa precisione e "centrare" con precisione il colore del sensore. Nel caso ripetete l'operazione. Il colore del secondo quadratino è più facile da impostare, in quanto la linea nera è decisamente più semplice da "centrare". Il blocco di selezione ritorna il valore vero, se si verifica la condizione nel "blocco" verde, ovvero il sensore del colore impostato sta toccando la linea nera. In questo

TUTTO PRONTO PER LA FINALE DELLA ROBOCUP JR 2015

Oltre 150 team provenienti da tutta Italia si contenderanno il titolo di campioni italiani per l'anno 2015 nelle varie categorie della gara di robotica "RoboCup Junior" che si svolgerà la seconda settimana di aprile presso il centro fieristico Malpensafiere in provincia di Varese.

a cura della REDAZIONE

La settima Finale Nazionale della RoboCup Jr Italia – la più importante manifestazione italiana di Robotica Educativa – si svolgerà quest'anno nel periodo 8-11 aprile 2015 presso il complesso fieristico Malpensafiere di Busto Arsizio (VA). La manifestazione, organizzata dall'ISIS Facchinetti di Castellanza (VA), in collaborazione con la Rete di Scuole per la RoboCup Jr Italia, vedrà la presenza di oltre 150 team di studenti provenienti da tutte le regioni d'Italia, che si sfideranno in differenti categorie, mettendo in campo robot autonomi da loro



stessi realizzati e programmati.

Sono sempre più numerose, infatti, le Scuole di ogni ordine e grado che adottano la Robotica Educativa per promuovere una maggiore motivazione all'apprendimento e allo sviluppo di nuove competenze.

La Finale Nazionale è la naturale conclusione di un percorso formativo che consente agli studenti di presentare il frutto del



Finanziarsi col crowdfunding

proprio lavoro, confrontandosi con team provenienti da altre regioni d'Italia nel tentativo di conquistare il diritto di partecipare alla finale mondiale che quest'anno si terrà a luglio a Hefei in Cina.

La manifestazione vedrà coinvolti circa 1.500 tra studenti ed insegnanti impegnati nelle gare e negli eventi collaterali, nonché migliaia di altri studenti della zona nel ruolo di spettatori. Oltre alle gare, durante i giorni della manifestazione sono previsti convegni, presentazioni e workshop, a cura della Rete di Scuole per la RoboCup Jr Italia, di Università, Associazioni e Aziende del settore.

RoboCup è un'iniziativa scientifica internazionale con l'obiettivo di migliorare lo stato dell'arte dei robot intelligenti; nata nel 1997, la missione originale era di mettere in campo una squadra di robot in grado di vincere, entro il 2050, la finale di Coppa del Mondo di Calcio. Da questa iniziativa è nata RoboCup Junior, una manifestazione destinata ai giovani fino ai 19 anni con l'intento di promuovere la Robotica Educativa come metodo trasversale, multidisciplinare e fortemente laboratoriale per guidare i ragazzi ad un uso formativo e corretto delle tecnologie.

L'apertura della Rete alle scuole del Primo ciclo dell'istruzione (scuole medie ed elementari) è avvenuta nell'a.s. 2010/11 a seguito di progetti di continuità e orientamento che hanno trovato nella Robotica Educativa un potente strumento didattico. Le gare di robotica si articolano in quattro principali tipologie:

- Rescue A
- Rescue B
- Soccer

The screenshot shows the 'School Raising' website interface for 'RoboCup Jr Italia'. It features a grid of crowdfunding projects. Each project includes a photo of the robot, a title, a description, a goal amount, and a progress bar. The projects shown are:

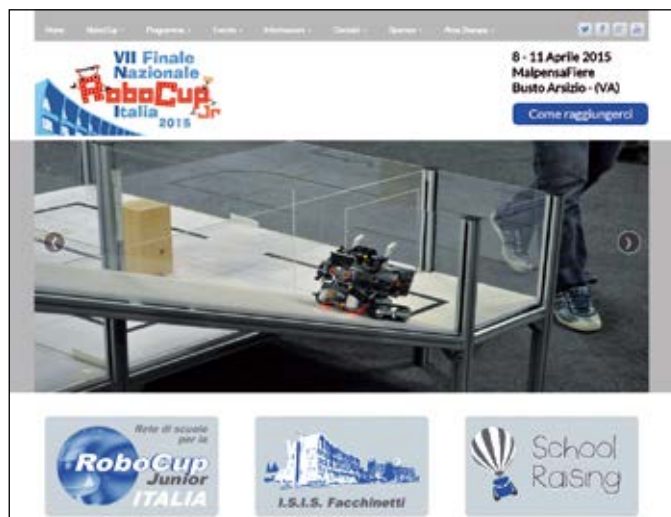
- Camisiamo Robotando**: Goal 516 €, 43% progress.
- ROBOCUP JUNIOR 2015 RESCUE B: Murphy 2.0**: Goal 345 €, 38% progress.
- RCI RESCUE B**: Goal 5 €, 1% progress.
- TUTTI INSIEME ROBOTICAMENTE!!**: Goal 516 €, 43% progress.
- HelpBot**: Goal 345 €, 38% progress.
- Il CFP di San Giovanni Bianco a RoboCup Italia 2015**: Goal 5 €, 1% progress.

Uno degli ostacoli che frenano la diffusione di iniziative quali la RoboCup Junior - col coinvolgimento di un numero sempre maggiore di scuole - è la mancanza di fondi pubblici per questo genere di attività (e non solo per queste); lo stesso problema riguarda gli studenti e le loro famiglie che debbono farsi carico delle spese di partecipazione alle gare e dei materiali necessari per realizzare i prototipi di robot. Quest'anno, dopo il test del 2014, è completamente operativa la piattaforma di crowdfunding School Raising che gli studenti possono utilizzare per sollecitare finanziamenti per i propri progetti robotici, ma non solo. In questo caso viene applicata la formula "Keep it all", ovvero alla fine della campagna il budget raccolto viene comunque versato al team, anche se non è stato raggiunto l'obiettivo della raccolta come avviene solitamente nelle più famose piattaforme di crowdfunding (Kickstarter, Indigogo, ecc.).

Sono già numerosi i team che quest'anno stanno utilizzando questa innovativa forma di finanziamento; ricordiamo che le squadre sono ancora in tempo per proporre i propri progetti o le proprie iniziative (magari per farsi finanziare, in caso di vittoria, il viaggio in Cina per partecipare alla finale mondiale...).

- Dance/Theatre

a loro volta suddivise in Under 19 (U19, studenti di età inferiore ai 19 anni) e Under 14 (U14, studenti fino ai 14 anni di età). Sono previste inoltre delle gare di co-spazio e delle gare non competitive, tipicamente per i più piccini. Nella categoria Rescue A i robot sono impegnati ad individuare in modo rapido e preciso le vittime all'interno di uno scenario che simula le rovine di un disastro. All'interno di un'arena i robot si dovranno confrontare con difficoltà via via crescenti, dal seguire una linea su una superficie piana (percorso guidato), al muoversi attraverso gli ostacoli di un terreno irregolare (percorso libero), al percorrere una rampa ed afferrare e porre in salvo la vittima. Il robot deve eseguire la missione



in maniera completamente autonoma, seguire il percorso, muoversi tra le varie stanze, individuare la vittima (rappresentata da una lattina), afferrarla e porla in salvo su

Volandia, una visita obbligatoria

Per molte scuole la Finale Nazionale della RoboCup Jr Italia è un'occasione per fare coincidere la gara di robotica con la tradizionale gita scolastica primaverile: mentre il team gareggia, gli altri studenti visitano musei, luoghi d'arte o località di interesse dal punto di vista paesaggistico. Quest'anno la sede della manifestazione è

ubicata ad appena 20 minuti di treno da Milano, e altrettanto tempo è sufficiente per raggiungere i bellissimi laghi lombardi (Maggiore e Como). Per gli appassionati di tecnologia, ad appena 10 minuti d'auto, segnaliamo uno dei musei più belli realizzati nel nostro paese, il Parco e Museo del Volo di Volandia, adiacente all'aeroporto della Malpensa. Un Parco e Museo dedicato al "sogno

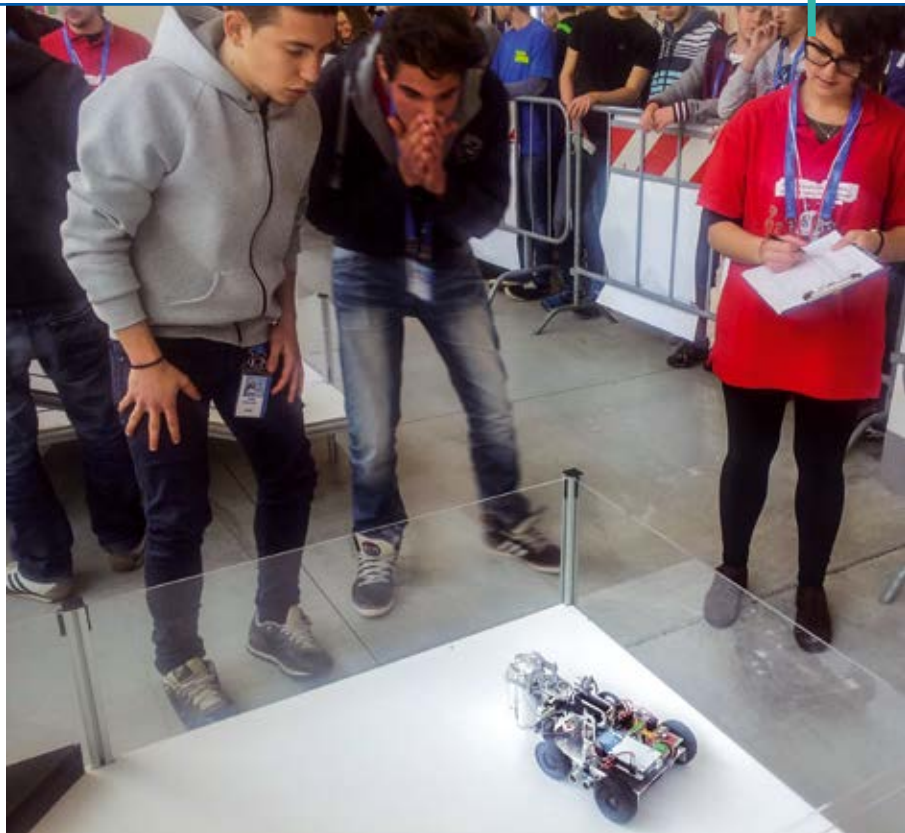
del volo", un tuffo nella storia dell'aeronautica e dello spazio per una gita indimenticabile.

Sono 11 le aree del museo: dai voli pionieristici in mongolfiera fino al Convertiplano, perfetta fusione tra volo orizzontale e verticale.

Tante emozionanti ambientazioni, sale video, un planetario didattico ed un padiglione interamente dedicato allo spazio. Ampie aree giochi anche al coperto e per i più grandi una sezione con simulatori di volo. L'intento del Museo è quello di far conoscere la storia dell'a-



una apposita pedana senza alcun intervento da parte del capitano della squadra. Possono essere utilizzati robot commerciali adattati e programmati (tipicamente Lego Mindstorm) o robot autocostruiti. La prima soluzione è sicuramente la più semplice dal momento che l'hardware necessita di pochissimi interventi; molto più complessa è la seconda soluzione in quanto – oltre a dover scegliere la piattaforma elettronica – i team si trovano ad affrontare problemi di meccanica e trazione non banali. Per questo motivo, in alcune edizioni passate della RoboCup Jr sono stati previsti dei premi speciali per i robot autocostruiti. Nella categoria Rescue B i robot sono impegnati nell'esplorazione di un labirinto con l'individuazione al suo interno delle vit-



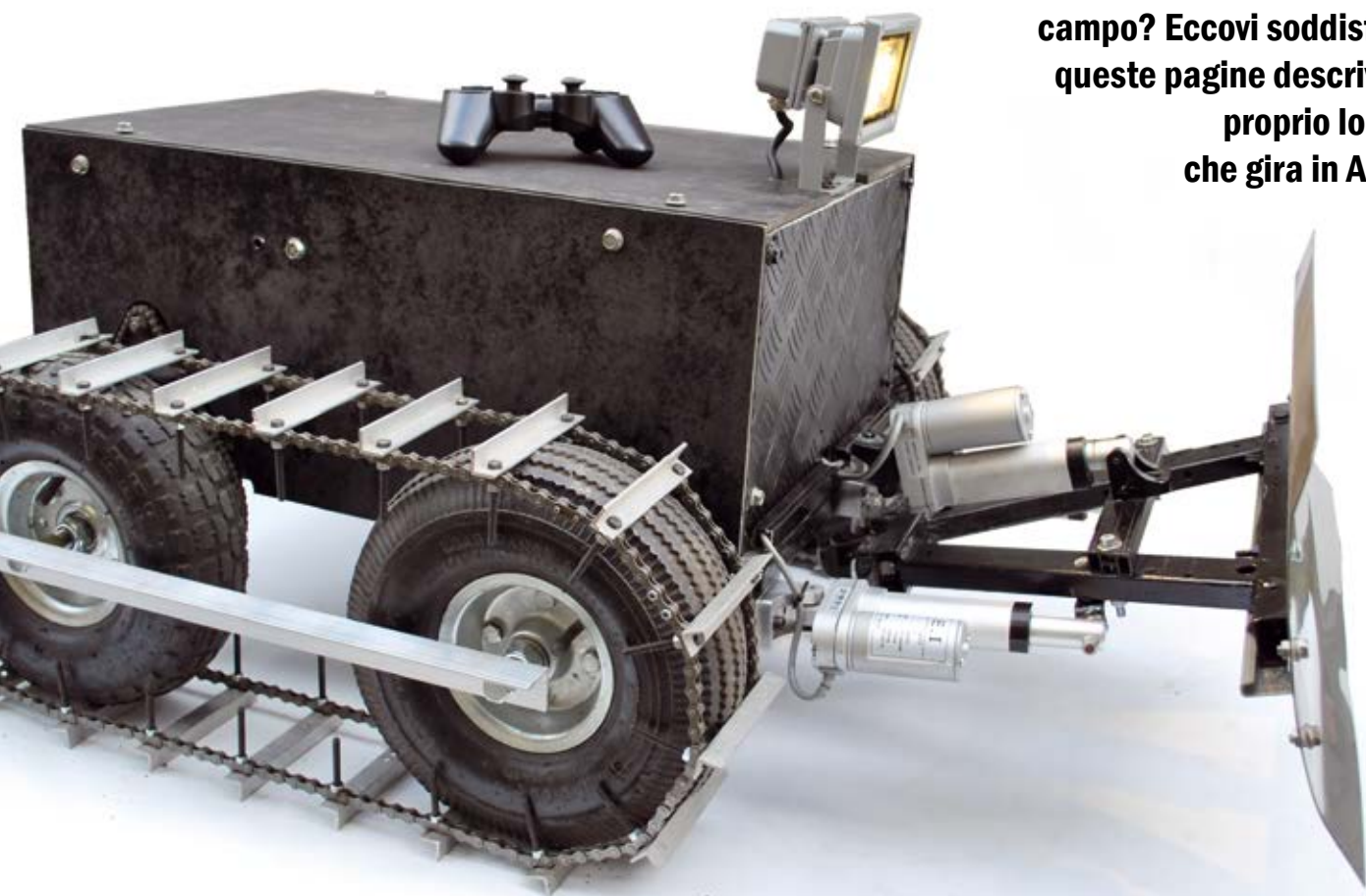
viazione mondiale, con particolare attenzione alle aziende che hanno fatto la storia del volo in Italia, come la Caproni, l'Agusta, l'Aermacchi, la SIAI-Marchetti, tutte con sede nella Provincia di Varese. Il Museo mette a disposizione anche dei simulatori di volo, una biblioteca che raccoglie numerosi volumi sul tema della conquista dell'aria e un ricco archivio storico. Recentemente è stato aggiunto uno spazio dedicato ai velivoli autonomi gestito dall'associazione GULLP che comprende una parte espositiva e un vero e proprio "LAB" dedicato allo studio, alla progettazione e alla realizzazione di velivoli UAV/Droni e in generale di progetti legati al volo e alle nuove tecnologie.

time che sono rappresentate da sorgenti di calore. Una volta individuata la vittima, il robot deve lasciare nelle vicinanze un segnale per le squadre di soccorso. Anche in questo caso tutte le vittime debbono essere individuate nel minor tempo possibile e il robot, anche in questo caso, deve effettuare la missione nel minor tempo possibile. Nella categoria Soccer (a sua volta suddivisa in due sottocategorie) i due robot in gara si muovono autonomamente in un campo di calcio in miniatura inseguendo una speciale pallina che emette raggi IR, nel tentativo di mettere a segno quanti più goal possibili. Infine, nella Dance, uno o più robot vestiti in costume si muovono e danzano seguendo una coreografia, in armonia con la musica di sottofondo. Come per l'edizione 2014, anche il calendario della manifestazione lombarda si articola in 4 giorni: dall'apertura ufficiale del mercoledì pomeriggio alla cerimonia di premiazione del sabato a mezzogiorno. Quest'anno tutte le gare e le attività connesse si svolgeranno nei padiglioni del Centro Fieristico Malpensafiere, a cavallo tra le province di Milano e Varese e poco distante dall'aeroporto della Malpensa. □

ROBOT SPAZZANEVE: IL FIRMWARE

di VITTORIO LOSCHIAVO

Avete costruito il cingolato e state aspettando il firmware per caricarlo e fare le prove sul campo? Eccovi soddisfatti: in queste pagine descriveremo proprio lo sketch che gira in Arduino.



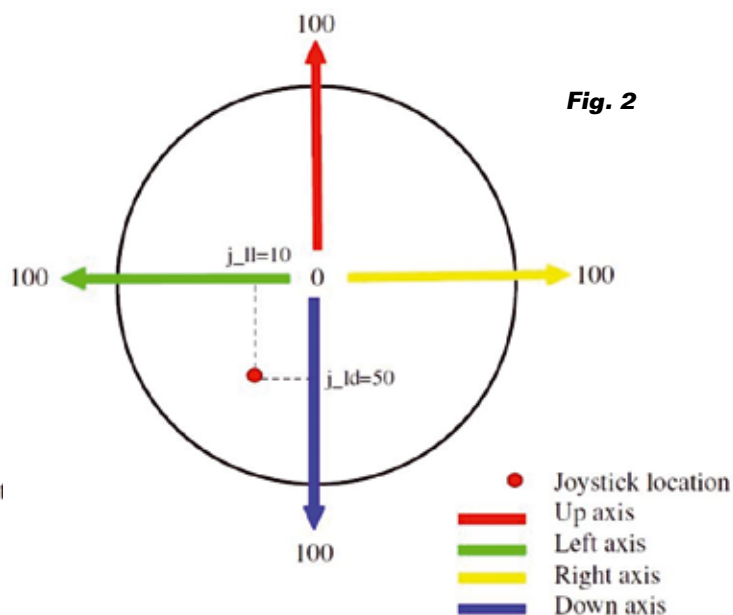
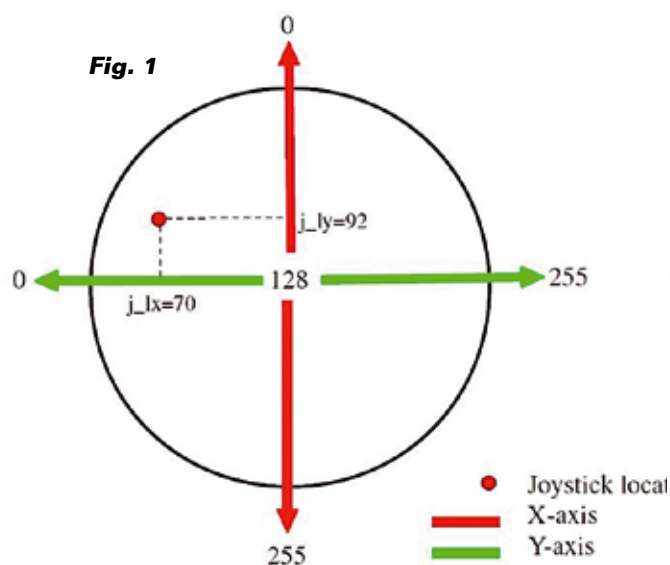
Ve lo abbiamo presentato nel fascicolo di novembre/dicembre descrivendolo per quello che è: la versione da Maker degli spalaneve motorizzati che si trovano nei più forniti centri del Fai da te e che all'estero già spopolano. Di fatto il nostro robot è un particolare cingolato, versione miniatura dello spalaneve tradizionale, solo

che può essere radiocomandato con facilità utilizzando, per renderlo ancor più familiare, un controller per Sony PS2. Nella prima puntata abbiamo descritto l'hardware del progetto, ossia la meccanica più l'elettronica di governo, spiegando sommariamente che è centrata intorno ad Arduino Uno; ora descriviamo il firmware da caricare in

Arduino per realizzare tutte le funzioni richieste, ovvero per azionare i motori di trazione, comandare la pala e accendere/spegnere il proiettore frontale in base ai comandi impartiti via radio.

UN BREVE RIEPILOGO

La parte elettrica ed elettronica dello spazzaneve è basata su Ar-



Valore decimale	Elemento controller	Descrizione
0	Select button	Lo shield PS2 restituisce lo stato del pulsante corrispondente quando lo specifico valore decimale viene ricevuto: 0 corrisponde a pulsante premuto; 1 corrisponde a pulsante a riposo. Esempio Se Arduino invia il valore 10 (decimale), lo shield-PS2 verifica lo stato del pulsante L1 del controller PS2. Poi lo shield PS2 risponderà con: 0 se il pulsante L1 è premuto; 1 se il pulsante L1 si trova a riposo.
1	left joystick centre button	
2	right joystick centre button	
3	START button	
4	up button	
5	right button	
6	down button	
7	left button	
8	L2 button	
9	R2 button	
10	L1 button	
11	R1 button	
12	triangle button	
13	circle button	
14	cross button	
15	square button	
16	left joystick x-axis	Lo shield PS2 restituisce lo stato corrispondente al valore decimale dell'asse selezionato con il joystick, ovvero della relativa combinazione. Esempio Se Arduino invia il valore 20 (decimale), lo shield PS2 leggerà lo stato del joystick e ritornerà il valore corrispondente.
17	left joystick y-axis	
18	right joystick x-axis	
19	right joystick y-axis	
20	left joystick up value	
21	left joystick down value	
22	left joystick left value	
23	left joystick right value	
24	right joystick up value	
25	right joystick down value	
26	right joystick left value	
27	right joystick right value	
28	Ritorna lo stato di connessione del controller PS2 allo shield PS2	Il PS2-Shield legge lo stato del controller PS2 e ritorna 1 se il controller viene rilevato via radio o è collegato al connettore PS2 dello shield.

Tabella 1

1° byte inviato [decimale]	2° byte inviato [decimale]	Controller	Descrizione
29	motor1 value	leggera vibrazione a sinistra	motor1 = 1 (motore on) motor1 = 0 (motore off)
30	motor2 value	intensa vibrazione a destra	motor2 = 0 ÷ 255 (velocità regolabile)

Tabella 2

duino Uno Rev. 3, che ospita tre shield e si interfaccia con queste schede:

- il driver per motori DRI0018;
- il circuito a relè RELAY1CH;
- il DC/DC converter elevatore STEPUP30VADJ.

Gli shield montati su Arduino sono il motor shield (cod. VMA03) per controllare i tre motori lineari che gestiscono il movimento della pala, e il PS2SHIELD, che permette di interfacciarsi con la console della PlayStation 2 (PS2); quest'ultimo richiede che vi sia montato, inserendolo nell'apposito connettore, l'RX-PS2, che è un ricevitore radio a 2,4 GHz specifico per ricevere i comandi dalla predetta console. Tutte le schede e gli elementi utilizzati nel circuito si possono acquistare da Futura Elettronica, www.futurashop.it con i codici indicati nel disegno. Lo schema di cablaggio della macchina è stato pubblicato nel fascicolo n° 191, cui rimandiamo. La fonte di alimentazione è costituita da una serie di due accumulatori a piombo gel da 7,2 Ah: nel punto intermedio dei due preleviamo 12 V, mentre tra il negativo e il positivo della serie prendiamo i 24 volt.

Con i 12 volt facciamo funzionare

Arduino, il quale alimenterà la logica degli shield attraverso i propri pin-strip; il LED (dotato internamente della resistenza di limitazione) indicherà quando Arduino è sotto tensione.

Con i 12 volt si alimenta anche la sezione di potenza del Motor Shield VMA03; infatti, visto l'assorbimento dei tre attuatori lineari, è sconsigliabile prendere i 12V da Arduino: bisogna optare per l'alimentazione esterna, da fornire all'apposita morsettiera PWR.

Ciascun attuatore lineare è composto da un motoriduttore a 12 Vcc che utilizza una vite senza fine per spostare un albero avanti e indietro lungo la sua lunghezza (escursione massima di 5 cm). L'attuatore ha un coefficiente di carico dinamico di 50 kg e una velocità massima di 1,3 cm/s. E in grado di sostenere fino a circa 250 kg, quando non si muove, e la coppia a vite assicura il mantenimento della posizione dell'albero anche in assenza di alimentazione.

Due interruttori di limitazione garantiscono l'arresto del motore quando arriva alla massima estensione e contrazione, mentre i diodi permettono di invertire la direzione dopo aver raggiunto il punto limite. L'attuatore è in metallo ed è sigillato per proteggerlo dalla polvere e dall'acqua, condizione indispensabile per operare in mezzo alla neve.

Con la linea di alimentazione a 24 volt funziona la sezione di potenza dello spazzaneve, vale a dire la trazione e la sezione del proiettore (opzionale): la prima si basa sul driver per motori di potenza DRI0018, da collegare con quattro fili ad altrettante linee digitali di Arduino. Le morsettiere di uscita del modulo permettono di collegare i due motori a 24 volt impiegati per la trazione.

Oltre al controller dei motori, i 24 volt vanno ad alimentare la sezione

Listato 1

```
if (ps2.getval(p_circle)==0) {
  Serial.println("CERCHIO");
  if (stato==0)
  {
    stato=stato+1;
    delay(100);
    Serial.println(stato);
    digitalWrite(RELAY, HIGH);
    ps2.vibrate(2, 255);    // attiva la vibrazione sul controller

    delay(500);            // pausa 500 ms
    ps2.vibrate(2,0);      // disattiva vibrazione
    delay(500);
    ps2.vibrate(2,255);
    delay(500);
    ps2.vibrate(2,0);
  }
  else
  {
    stato=0;
    delay(100);
    Serial.println(stato);
    digitalWrite(RELAY, LOW);
    ps2.vibrate(2, 255);    // attiva la vibrazione sul controller
    delay(500);            // pausa 500 ms
    ps2.vibrate(2,0);      // disattiva vibrazione
  }
}
```

del proiettore a LED, che potete montare o meno e la cui attività è governata in base alla luminosità rilevata nell'ambiente dall'apposito fotoreistore previsto nell'elettronica del nostro spazzaneve.

Il proiettore è progettato per funzionare a 220 Vac, tuttavia siccome questa tensione non è disponibile a bordo del robot, per evitare di usare un inverter abbiamo modificato il proiettore aprendolo, rimuovendo l'alimentatore AC/DC e portando i due fili del LED di potenza all'uscita di un converter DC/DC (cod. STEPUP30VADJ); quest'ultimo è uno switching a tensione d'uscita regolabile, che va registrato in modo da fornire al corpo a LED del proiettore una corrente tale da farlo lavorare a circa 10 watt.

Della ricezione dei dati provenienti dal telecomando si occupa lo shield PS2SHIELD, per il quale il costruttore rende disponibile la specifica libreria che abbiamo integrato nello sketch per Arduino. Lo shield effettua la ricezione e la decodifica dei comandi; la parte radio a 2,4 GHz

viene fornita separata (RX_PS2) e va inserita nell'apposito connettore dello shield.

IL FIRMWARE

Bene, dopo aver riassunto la composizione e le caratteristiche del robot spazzaneve, possiamo passare a ciò che più vi interessa: il firmware, ovvero lo sketch da caricare in Arduino Uno affinché governi lo spazzaneve per fargli eseguire i comandi inviati tramite il controller della PlayStation 2.

Il firmware di gestione del robot è relativamente semplice; per comprenderne la struttura è opportuno, prima, spendere qualche parola sugli shield che deve controllare e, in particolare, su quello PS2. Infatti del Motor Shield VMA03 abbiamo già parlato diffusamente in altri articoli perché l'abbiamo impiegato in vari progetti, quindi ci limitiamo a dire che della sua gestione si occupa l'apposita libreria che abbiamo già previsto di includere nello sketch e che richiede un segnale PWM ed un livello logico di enable.

Listato 2

```
#####  
# Constants (LITERAL1)  
#####  
p_select pulsante "select"  
p_start pulsante "start"  
p_up pulsante "up"  
p_right pulsante "right"  
p_down pulsante "down"  
p_left pulsante "left"  
p_triangle pulsante "triangolo"  
p_circle pulsante "cerchio"  
p_cross pulsante "X"  
p_square pulsante "quadrato"  
p_joy_lu Joystick sinistro "up"  
p_joy_ld Joystick sinistro "down"  
p_joy_ll Joystick sinistro "left"  
p_joy_lr Joystick sinistro "right"  
p_joy_ru Joystick destro "up"  
p_joy_rd Joystick destro "down"  
p_joy_rl Joystick destro "left"  
p_joy_rr Joystick destro "right"
```

Spieghiamo un po' le caratteristiche salienti dello Shield PS2 e le principali funzioni della libreria ad esso dedicata.

Lo shield permette di utilizzare il controller wireless (o quello via cavo) della Playstation 2, consentendo di comandare a distanza robot o altri dispositivi; per come è stato progettato, può essere utilizzato con Arduino Uno Rev. 3,

Listato 3

```
if (ps2.getval(p_triangle)==0) {  
  Serial.println("TRIANGOLO");  
  if (velo<250)  
{  
    velo=velo+25;  
  }  
  else  
{  
    ps2.vibrate(2, 255); //non fa niente  
    delay(500);  
    ps2.vibrate(2,0);  
  }  
  analogWrite (M1, velo);  
  analogWrite (M2, velo);  
  Serial.println(velo);  
}
```

Duemilanove, Mega e Leonardo. È dotato di pulsante di reset per ripetere quello di Arduino, di jumper per impostare vari Baud Rate (9.600, 57.600, 115.200 bps), nonché di LED di stato e di jumper per selezionare i pin utilizzati per il TX e l'RX dei dati, alternativi a quelli predefiniti dell'UART.

A tale riguardo diciamo che sulla scheda Arduino Uno i pin D0 e D1 corrispondono alla seriale hardware e che il loro significato è, rispettivamente, TX e RX. È comunque possibile riallocare le linee TX e RX via software, assegnandole ad altri pin (TX: pin 1, 3, 9, 11 – RX: 0, 2, 8, 10) caricando l'apposita libreria New Software Serial, che permette di emulare la porta seriale su coppie di I/O digitali purché non già utilizzati da altre risorse hardware (ovvero altri shield).

Notate che se utilizzate i pin della seriale hardware (D0, D1), bisogna ricordarsi di staccare il connettore PS2 dallo Shield prima di iniziare a programmare Arduino Uno.

Lo shield PS2 è in grado, tramite l'apposita libreria, di riconoscere e gestire tutti i pulsanti e i joystick analogici del controller della PS2. Ogni joystick analogico (destro e sinistro) ha due assi (X e Y) e due formati di output: analogico 1 e 2. Nel formato di output analogico 1 ci sono 2 variabili per ogni joystick, assi X e Y. In base a come spostiamo la leva del joystick, si ottiene in uscita un valore da 0 a 255 (guardate la **Fig. 1**), corrispondente alla posizione della levetta.

Asse Y:

- Posizione centrale (neutro), il valore è 128.
- Spingendo in up, il valore cambia da 128 a 0
- Spingendo in down, il valore cambia da 128 a 255.

Asse X:

- Posizione centrale (neutro), il

valore è 128.

- Spingendo verso sinistra, il valore cambia da 128 a 0.
- Spingendo verso destra, il valore cambia da 128 a 255.

Nel formato di output analogico 2 ci sono quattro variabili per ogni joystick (up, down, left e right). Quando l'utente muove il joystick in una direzione, il valore delle 4 variabili cambia da 0 a 100. I joystick di destra e di sinistra avranno quindi assegnate ciascuno quattro variabili indipendenti.

Possiamo perciò decidere tra i seguenti due formati di output:

Le variabili del formato 1 sono:

- j_lx - Left joystick, X axis
- j_ly - Left joystick, Y axis
- j_rx - Right joystick, X axis
- j_ry - Right joystick, Y axis

Quelle del formato 2 sono:

- j_lu - Left joystick, up axis
- j_ld - Left joystick, down axis
- j_ll - Left joystick, left axis
- j_lr - Left joystick, right axis
- j_ru - Right joystick, up axis
- j_rd - Right joystick, down axis
- j_rl - Right joystick, left axis
- j_rr - Right joystick, right axis

Per quanto riguarda la gestione dei pulsanti del controller della PS2, basta semplicemente verificare lo stato del pulsante che ci interessa e andare a leggere il valore restituito, che può essere:

- 0 se il pulsante è premuto;
- 1 se il pulsante non è premuto.

Ad esempio, se vogliamo sapere se è stato premuto il pulsante "UP", dovremo scrivere la seguente istruzione:

```
ps2.getval(p_up)==0
```

È anche possibile sostituire "p_up" con il valore decimale corrispondente, che in questo caso è "4".

Guarda il Robot Spazzaneve in azione su YouTube!



Vuoi vedere in azione il nostro Robot Spazzaneve? Nulla di più semplice: collegati alla pagina <http://youtu.be/L4X1DqNTaRw>. Potrai anche vedere i video di moltissimi altri nostri progetti proposti in passato.



Per il resto delle corrispondenze fate riferimento al manuale fornito con lo shield (si può anche scaricare dal web all'indirizzo <http://cytron.com.my/p-shield-ps2>). Lo shield PS2 è in anche in grado di gestire i due piccoli motori interni al controller della PS2 che determinano la vibrazione (sinistra e destra): i relativi comandi richiedono due byte di dati, il primo dei quali indica quale motore deve vibrare, mentre il byte successivo (secondo byte) indica lo stato del motore o la velocità (come mostrato nella **Tabella 2**).

Per utilizzare lo shield che gestisce i comandi provenienti dal controller della PS2 abbiamo utilizzato la libreria `Shield_PS2.h` realizzata dal produttore, la Malese Cytron. All'inizio del nostro sketch includiamo, con l'istruzione `#include <Shield_PS2.h>`, la libreria necessaria alla corretta gestione dello shield per il controller della PS2. Andiamo avanti e vediamo il controllo del proiettore a LED, che viene operato tramite il pin analogico A0 di Arduino Uno Rev3, cui è col-

legata la fotoresistenza che rileva la luminosità ambientale; quando la luce si abbassa oltre la soglia prevista, lo sketch attiva il proiettore, o meglio, fornisce lo stato logico alto alla scheda `RELAY1CH`, la quale che contiene un relé -di cui è reso disponibile l'intero scambio- attivato da livelli logici. Nel progetto, di questa scheda utilizziamo lo scambio tra C ed N.A. e con esso commutiamo l'alimentazione diretta al DC/DC converter che alimenta il proiettore, perciò quando Arduino Uno fornisce lo stato logico 1 (tramite il proprio pin digitale 13) il relé scatta e chiude i predetti contatti, facendo accendere il proiettore a LED.

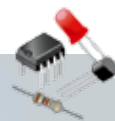
Proseguiamo con la descrizione e andiamo alla gestione dei tre attuatori lineari che permettono di alzare/abbassare e ruotare a destra/sinistra la pala: visto che richiedono una certa corrente, non potendo pilotarli direttamente con le linee di Arduino ci siamo affidati a un'interfaccia driver di potenza, che nello specifico è il Motor Shield della Futura Elettronica. Dato che questa scheda richiede di essere pilotata con segnali PWM, il firmware che abbiamo progettato invia ai pin digitali 2 e 11 segnali di PWM e ai pin 3 e 12 i segnali di enable diretti al Motor Shield.

Quanto ai motori di trazione MY016, anche in questo caso Arduino Uno non può pilotarli direttamente ma li gestisce attraverso due pin digitali che forniscono i segnali di controllo al driver di potenza corrispondente; i pin utilizzati allo scopo sono 5 e 6, anch'essi impostati per fornire dei segnali PWM con i quali i driver forniscono impulsi di larghezza variabile in modo da modulare la potenza fornita ai motori. Nello sketch, il controllo PWM dei motori corrisponde alle variabili `"E1"` ed `"E2"`, corrispondenti rispettivamente ai pin

digitali 5 e 6. Quanto ai pin digitali 4 e 7, corrispondono alle variabili, rispettivamente, `"M1"` e `"M2"`. Riassunto, per la gestione da parte di Arduino Uno abbiamo creato le seguenti variabili: `"FOTORES"` per la fotoresistenza, `"RELAY"` per l'attivazione del relé che accende il proiettore, `"sensorValue"` dove viene memorizzato il valore della fotoresistenza; le variabili che gestiscono il movimento alza/abbassa della pala sono `"ALZAPALA"` e `"ONALZAPALA"`, mentre quelle che gestiscono il movimento destra/sinistra della pala sono `"RUOTAPALA"` e `"ONRUOTAPALA"`. Le variabili dei motori sono, come già detto,

Presso Futura Elettronica è possibile reperire gran parte degli elementi necessari per questa realizzazione. La scheda Arduino (cod. ARDUINOUNOREV3) costa 24,50 Euro, il faro a LED 10 w (cod. LEDA301WW-G) costa 22,00 Euro, la scheda relé (cod. RELAY1CH) costa 7,00 Euro, la motor shield (cod. VMA03) costa 25,00 Euro, il motore (cod. MY1016) costa 57,00 Euro, attuatore lineare (cod. LACT2) costa 153,00 Euro, la staffa (cod. SUPPLACT) costa 22,00 Euro, il driver (cod. DRI0018) costa 58,00 Euro, il convertitore (cod. STEPUP30V) costa 6,50 Euro, lo shield PS2 (cod. PS2SHIELD) costa 36,00 Euro, la batteria (cod. NP72-12) costa 29,00 Euro. Tutti i prezzi si intendono IVA compresa.

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



per il MATERIALE

Area Rebus

Rovigo Fiere

Provincia di Rovigo

Fiera

dell'informatica elettronica e radiantismo

Rovigo

Cen.Ser. - Viale Porta Adige 45

28 febbraio e
1 marzo 2015

apertura dalle ore 9,00 alle ore 19,00

iniziative collaterali

MODEL EXPO
mostra di modellismo statico e dinamico

VINTAGE VILLAGE
mercato hobbysti radio cine foto ottica

TRASHWARE
riuso materiale informatico per il sociale

per riduzioni e info visita il sito
www.arearebus.com

"E1", "E2" e "M1", "M2". La variabile **"velo"** è quella dove viene salvato il valore della velocità dei motori, mentre la **"motori"** è quella dove viene salvato lo stato dei motori e permette di fermare i motori una volta che i pulsanti di direzione o il joystick di direzione vengono rilasciati. La variabile **"stato"** riguarda la gestione del proiettore a LED: se vale 0, l'accensione è subordinata alla condizione di illuminazione della fotoresistenza, mentre se vale 1 si comanda il proiettore con il relativo pulsante del controller PS2.

Visto che per sapere se è attiva una delle due funzioni non abbiamo alcun feedback, sul controller viene attivata la vibrazione; per l'esattezza, il controller PS2 vibra due volte e per un breve periodo per il pulsante, mentre una volta per la fotoresistenza (**Listato 1**):

Per i pin TX e RX abbiamo utilizzato rispettivamente il pin 9 e il pin 8, mentre come Baud Rate abbiamo selezionato 9.600. L'istruzione **"ps2.getval(p_up)==0"** permette di conoscere quale pulsante è stato attivato sul controller PS2: nel nostro caso abbiamo inserito **"(p_up)=0"** che permette di sapere se è stato premuto il pulsante **"Up"** del controller; se volete utilizzare un altro pulsante potete sostituire quello attuale con il pulsante che vi interessa (nel **Listato 2** trovate l'elenco). È anche possibile sostituirla con **"(p_joy_lu)==100"** per sapere se il joystick di sinistra è in posizione **"UP"** oppure sostituendola con **"(p_joy_ld)==100"** potete sapere se il joystick è in posizione **"Down"**, **"(p_joy_lr)==100"** se è **"Right"** e **"(p_joy_11)==100"** se è **"Left"**. Infine, potete utilizzare insieme le due istruzioni con **"if(ps2.getval(p_up)==0 || ps2.getval(p_joy_lu)==100)"** che permette di sapere se il pulsante **"UP"** è stato premuto o il joystick di sinistra è in posizione **"UP"**.

Il nostro sketch comincia con verificare il valore della variabile **"motori"**, impostata con un valore uguale a **"0"**. Se viene premuto il pulsante **"Up"** o il joystick sinistro viene spostato nella posizione **"Up"** [istruzione **if(ps2.getval(p_up)==0 || ps2.getval(p_joy_lu)==100)**], corrispondente nel nostro caso al movimento in avanti, verranno attivati i motori con il valore di velocità PWM **"velo"**, che può essere incrementato da 0 a 250 con il pulsante **"triangolo"** e diminuito con **"X"**. Quando viene raggiunto il valore di velocità massima, corrispondente a 250, sul controller viene attivata per un breve periodo la vibrazione, come mostrato nel **Listato 3**. □

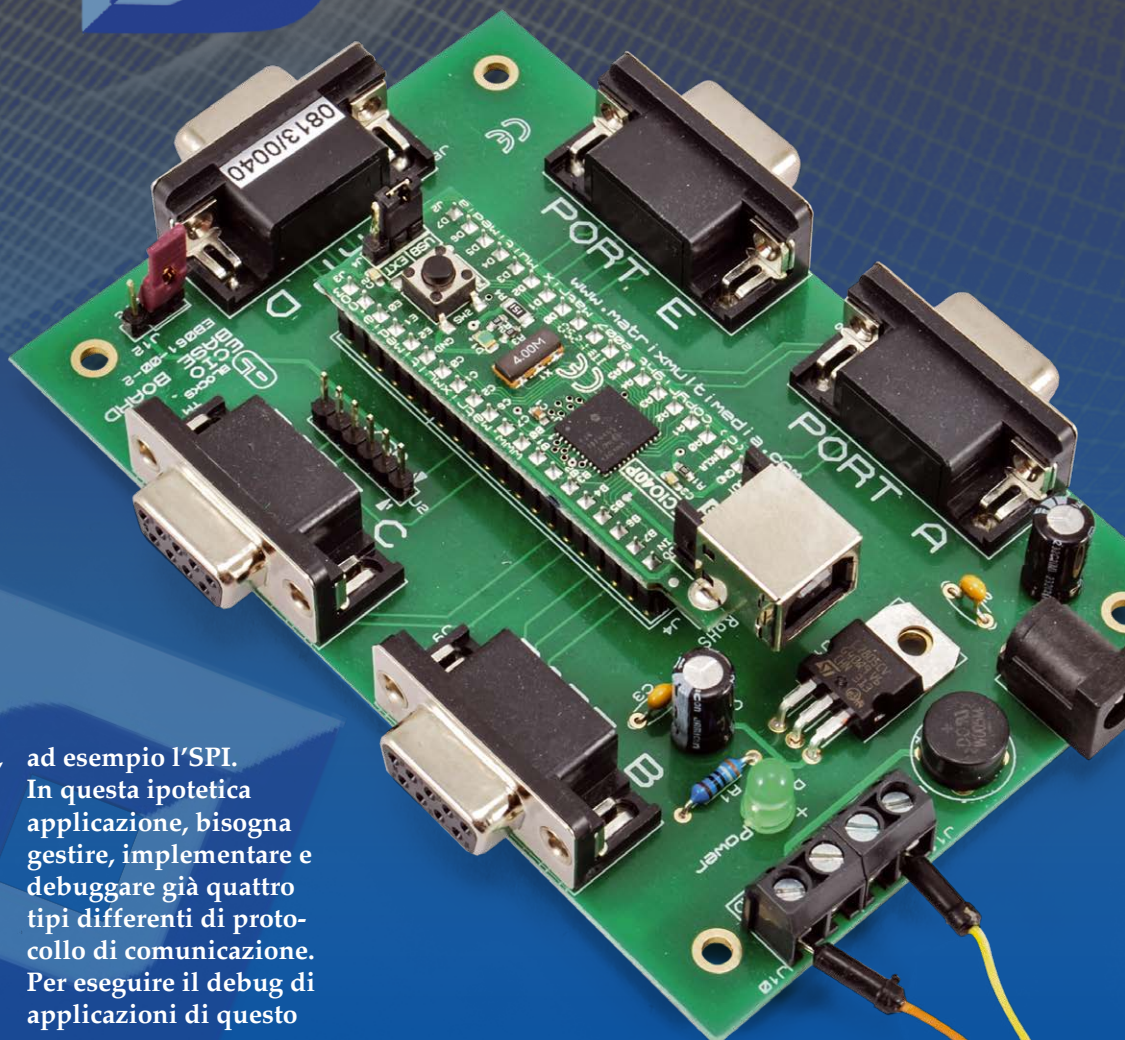
EMI: ECIO MULTI INTERFACE

di FRANCESCO FICILI

Una delle problematiche più comuni nello sviluppo di sistemi embedded di una certa complessità, è sicuramente gestire un insieme di interfacce di comunicazione di basso livello diverse tra loro, e farlo spesso con una singola unità a microcontrollore. Pensiamo ad esempio al caso di una scheda a microcontrollore per una determinata applicazione: è molto facile che questa debba interfacciarsi con una EEPROM per l'immagazzinamento di dati di configurazione o di altro tipo, usando ad esempio un'interfaccia I²C-Bus; nella stessa scheda potremmo trovare un link RS232, che a livello logico TTL è rappresentato da un'interfaccia UART verso un trasceiver, ad esempio di tipo MAX232. Ancora, potremmo trovare un display LCD (interfaccia HD44780) e vari tipi di sensori che potrebbero interfacciarsi al controllore principale attraverso canali analogici o linee seriali come

ad esempio l'SPI. In questa ipotetica applicazione, bisogna gestire, implementare e debuggare già quattro tipi differenti di protocollo di comunicazione. Per eseguire il debug di applicazioni di questo

Scheda di sviluppo, capace di interfacciarsi con i più comuni protocolli di comunicazione utilizzati in ambito embedded, governata da una ECIO 40P prodotta dalla Matrix Multimedia e programmata in Flowcode.



Standard	Tx Type	# Signal Wires	Data Rate & Distance	Hardware \$	Scalability	Application Example
UART	Asynchronous	2	20kbps @ 15m	Medium (transceiver)	Low (point-to-point)	Diagnostic display
LIN	Asynchronous	2	20kbps @ 40m	Medium (transceiver)	High (identifier)	Washing machine subsystem network
SPI	Synchronous	4+	25Mbps @ 0.1m	Low	Medium (chip selects)	High speed chip to chip link
I2C	Synchronous	2	1Mbps @ 0.5m	Low (resistors)	High (identifier)	System sensor network

Fig. 1 - Confronto tra protocolli seriali di basso livello.

tipo potremmo dotarci di un costoso analizzatore di stati logici, magari dotato di software di interfaccia per la decodifica di protocolli specifici, oppure valutare l'opportunità di realizzare uno strumento più snello e semplice che ci permetta di eseguire tale tipo di operazioni.

In Fig. 1 è riportata una cross reference tra alcuni dei più comuni standard seriali di basso livello.

DESCRIZIONE DEL PROGETTO

Ciò che ci proponiamo di realizzare in questo articolo, non è uno strumento complesso come un analizzatore di stati logici, ma qualcosa di più semplice: una scheda multi interfaccia che ci permetta di interfacciarci con periferiche che implementano alcuni dei più comuni standard di comunicazione di basso livello attraverso un bus utilizzatissimo negli attuali Personal Computer: l'USB.

La nostra applicazione sarà dotata di un software di gestione che permetta di impostare la scheda per interfacciare lo standard desiderato e, attraverso un semplice set di comandi di alto livello, di eseguire semplici operazioni come la scrittura e la lettura di byte o stringhe e la modifica di alcuni parametri di controllo. I protocolli di comunicazione che dovranno essere implementati sono i seguenti:

- UART – Universal Asynchronous Receiver Transmitter;
- PC Master – Inter Integrated Circuit (solo master);
- SPI Master – Serial Peripheral Interface (solo master);
- LCD HD44780 – Interfaccia LCD Hitachi.

Doteremo inoltre il nostro sistema di un bootloader USB, in modo da garantirne l'espandibilità con protocolli o funzioni aggiuntive anche

dopo il rilascio della prima versione.

LA SCHEDA ECIO

La piattaforma hardware che intendiamo utilizzare per questo progetto è la scheda ECIO, prodotta dalla Matrix Multimedia. La ECIO è una famiglia di dispositivi basati su microcontrollore e pensati per il rapid prototyping a basso costo di applicazioni general purpose. Le ECIO sono nativamente dotate di interfaccia USB e di un bootloader per l'aggiornamento del Firmware e possono essere programmate utilizzando l'apposito SW di programmazione liberamente

Fig. 2 - Board ECIO.

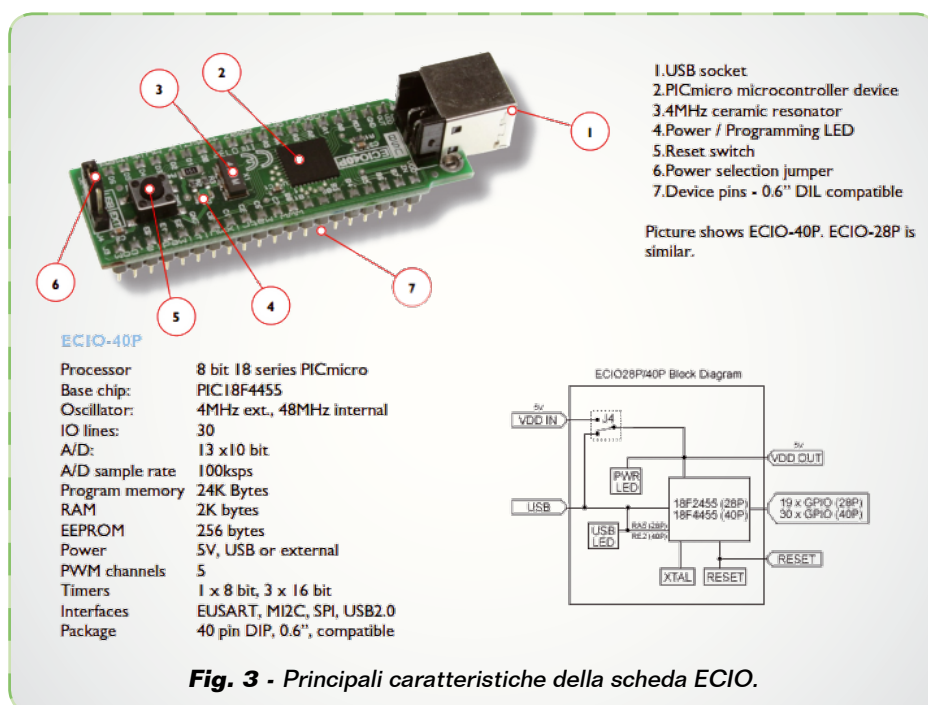
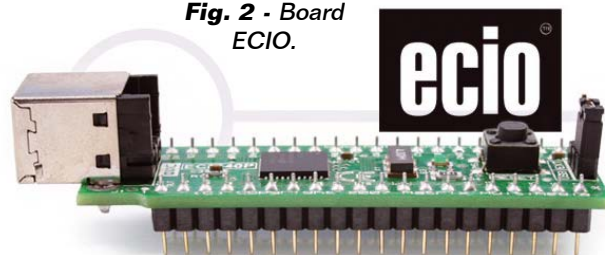


Fig. 3 - Principali caratteristiche della scheda ECIO.

scaricabile dal sito della Matrix. Le schede hanno un fattore di forma adatto alla prototipazione, in quanto la piedinatura è costituita da una doppia fila di pin a passo 2,54 mm (600 mils), quindi adatto ad essere inserito su una breadboard o su una scheda millefori, oppure, addirittura, su un socket DIL40-600. In Fig. 2 è rappresentato un esempio di scheda ECIO.

Come detto, esistono diversi modelli di ECIO, differenziati in base al microcontrollore sul quale sono basati; per la nostra applicazione scegliamo di utilizzare un ECIO 40P, che costituisce un buon compromesso tra potenza di calcolo, quantità di memoria Flash e periferiche e, naturalmente, costo. In Fig. 3 è rappresentato un estratto del data-sheet che riporta le principali caratteristiche della ECIO 40P; come si può vedere, la scheda è basata su un microcontrollore PIC18F4455, clockato tramite un quarzo esterno da 4 MHz. Il micro dispone di 24 kB di memoria programma, di 2 kB di RAM e anche di 256 byte di EEPROM interna, utilizzabile come memoria non volatile per l'immagazzinamento, ad esempio, di parametri di funzionamento. La scheda dispone di 30 linee di I/O, delle quali fino a 13 possono essere utilizzate dal convertitore A/D interno (a 10-bit e 100 ksp/s) e 5 come canali PWM. Dal punto di vista delle periferiche, la scheda dispone di interfaccia hardware UART, I²C ed SPI, oltre, naturalmente, all'USB 2.0. Completano la dotazione, quattro timer interni. In Fig. 4 è rappresentata la pin-out della scheda.

PROTOCOLLO DI COMUNICAZIONE

Partiamo quindi da un piattaforma hardware che dispone già di quanto ci occorre, dal punto di

vista delle risorse, per realizzare la nostra applicazione, ed inoltre soddisfa nativamente il requisito sull'aggiornamento del firmware tramite bootloader USB.

A questo punto bisogna impostare un protocollo che ci permetta di configurare e gestire le varie interfacce che intendiamo realizzare tramite la ECIO. Decidiamo di servirci di un protocollo codificato ASCII con un set di comandi meno esteso possibile, in modo da rendere più agevole l'utilizzo della scheda. In Tabella 1 è riportato il formato del protocollo di comunicazione della nostra scheda.

Si noti che tutti i comandi sono composti da due parti distinte:

- 1) un *command type* più un *campo lunghezza*;
- 2) un *campo payload*.

Il comando completo viene quindi inviato in due transazioni distinte: prima si manda il command type (nella Tabella 1 è chiamato Tipologia di comando)

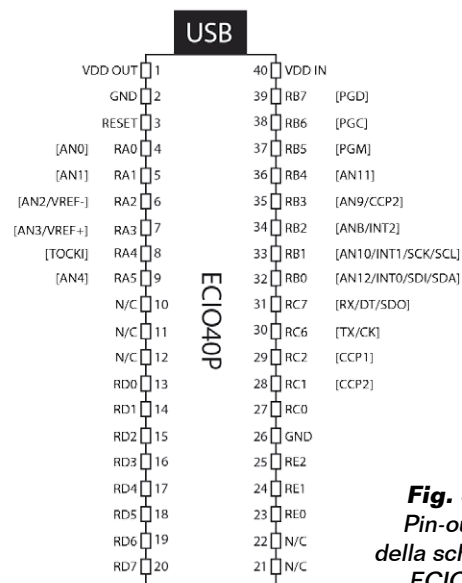


Fig. 4
Pin-out
della scheda
ECIO.

e la lunghezza del payload, e, non appena la scheda conferma la ricezione di questa parte del comando, si procede con l'invio del campo payload. Si noti che il campo lunghezza dati... è espresso in byte.

Di seguito è riportato il set di comandi completo della scheda, diviso in base alla categoria. Notiamo che abbiamo cinque

Tabella 1 - Formato del protocollo.

	Cmd Type	Sep.	Lenght	Sep.	Sep.	Payload	Sep.
Numero di Byte	3	1	2	1	1	0-20	1
Significato	Tipologia Comando	'<'	Lunghezza campo dati (byte)	'>'	'<'	Payload del campo dati	'>'

Tabella 2

Comando base		
Comand Type	Lenght	Payload
BMS	01	- 0 → Board Control Mode - 1 → I ² C Interface - 2 → UART Interface - 3 → SPI Interface - 4 → LCD Interface

Tabella 3

Comandi I ² C		
Comand Type	Lenght	Payload
IWB	01	Byte da scrivere sull'interfaccia I ² C
IRB	00	NA

Tabella 4

Comandi UART		
Comand Type	Lenght	Payload
USB	01	0 - 1200 1 - 2400 2 - 4800 3 - 9600 4 - 19200 5 - 38400 6 - 57600 7 - 115200
USS	01-20	Dati da inviare sulla porta seriale
URS	07	Numero di byte da leggere + timeout in ms

Tabella 5

Comandi SPI		
Comand Type	Lenght	Payload
SWS	01-20	Dati da inviare sull'interfaccia SPI
SRS	07	Numero di byte da leggere + timeout in ms

categorie distinte di comandi:

- 1) comandi base;
- 2) comandi I²C;
- 3) comandi UART;
- 4) comandi SPI;
- 5) comandi LCD.

Analizziamo di seguito le categorie.

COMANDI BASE

In questa categoria rientra effettivamente il solo comando BMS – Board Mode Set, che imposta la modalità operativa della scheda. La sintassi di BMS è descritta nella **Tabella 2**.

COMANDI I²C

Il primo comando di questa categoria è IWB – I²C Write Byte: scrive il byte contenuto all'interno del campo payload sull'interfaccia I²C-Bus.

Il secondo ed ultimo comando è IRB – I²C Read Byte; esso legge un byte dall'interfaccia I²C e lo restituisce con il messaggio successivo. Entrambi i comandi sono descritti, con la loro sintassi, nella **Tabella 3**.

COMANDI UART

I comandi dedicati alla gestione dell'UART sono tre: il primo è

USB – Uart Set Baud rate ed imposta il baud-rate dell'interfaccia UART; il secondo è USS – Uart Send String ed invia la stringa del campo dati sulla porta seriale. Il terzo ed ultimo è URS – Uart Read String e legge e restituisce con un messaggio di ritorno il numero di dati ricevuti sulla porta seriale indicato sul campo payload.

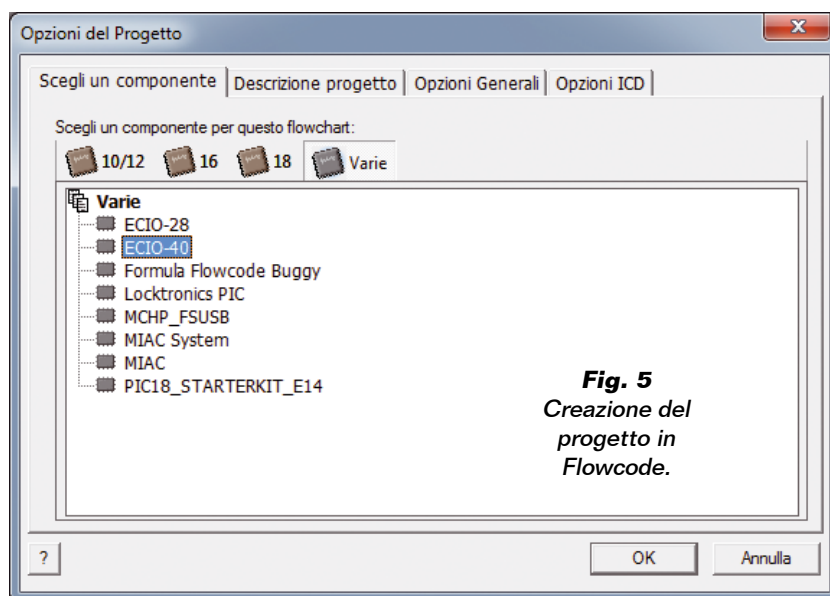
La sintassi per impartire questi tre comandi è descritta nella **Tabella 4**.

COMANDI SPI

In questa categoria ci sono due comandi, il primo dei quali è SWB – SPI Write Byte: questo scrive il byte contenuto all'interno del campo payload sull'interfaccia SPI. Il secondo comando è SRB – SPI Read Byte e legge un byte dall'interfaccia SPI e lo restituisce con il messaggio successivo. La sintassi dei comandi riguardanti la gestione dell'SPI è riportata nella **Tabella 5**.

COMANDI LCD

Concludiamo la panoramica del set di comandi analizzando i tre destinati alla gestione degli LCD; il primo è LWS – LCD Write String e scrive la stringa contenuta all'interno del campo payload sull'interfaccia LCD. Il secondo è LCS – LCD Clear Screen e cancella lo schermo del display LCD collegato all'interfaccia. Il terzo ed ultimo è LCP – LCD Cursor X Position e muove il cursore del display LCD alla posizione indicata nel capo payload (asse x,y). La sintassi di questi tre comandi è descritta nella **Tabella 6**. Chiaramente dovremo preoccuparci, durante lo sviluppo della



nostra applicazione, di garantire la massima scalabilità del protocollo, in maniera da poter aggiungere nuove funzionalità in futuro.

FIRMWARE

Passiamo adesso alla descrizione del firmware, che è stato sviluppato utilizzando Flowcode 5. Per prima cosa creiamo un progetto Flowcode per la ECIO 40P, selezionando l'opzione "Nuovo Progetto" e scegliendo la voce ECIO 40, sotto il tab "Varie", come illustrato in Fig. 5. Flowcode caricherà automaticamente le impostazioni adatte alla ECIO, quindi non è necessario impostarle dal tab "Opzioni Generali".

STRUTTURA DEL FIRMWARE

Come già visto in altri progetti, la base del nostro firmware è un semplice scheduler non pre-emptive, con gestione della periodicità dei task, basato su un interrupt di una risorsa timer (in particolare, si tratta del timer 2). Un oggetto del genere si può creare in flowcode usando il blocco interrupt e impostando un interrupt sul timer 2 avente un periodo di 1 ms (Fig. 6), associato ad una macro che chiamiamo MainScheduler; partendo da questa base tempi e utilizzando un contatore ed un blocco decisionale per ogni task, saremo in grado di generare le periodicità che ci servono.

In Fig. 7 è riportata la finestra delle proprietà dell'intervallo: la ECIO ha un clock interno di 48 MHz (ottenuti tramite un quarzo esterno ed un PLL interno a 96 MHz), quindi impostiamo un valore di prescaler pari a 1:16, un postscaler pari a 1:10 ed un rollover value di 75, in modo da ottenere un frequenza dell'intervallo pari a 1 kHz.

Tabella 6

Comandi LCD		
Comand Type	Lenght	Payload
LWS	01-20	Stringa di caratteri da visualizzare su LCD
LCS	00	NA
LCP	03	Posizione X (2byte) e Y (1byte) del cursore

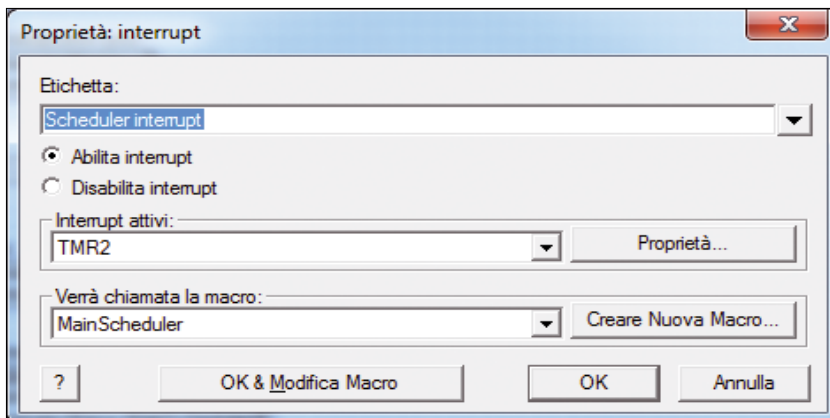


Fig. 6 - Impostazioni del blocco interrupt.

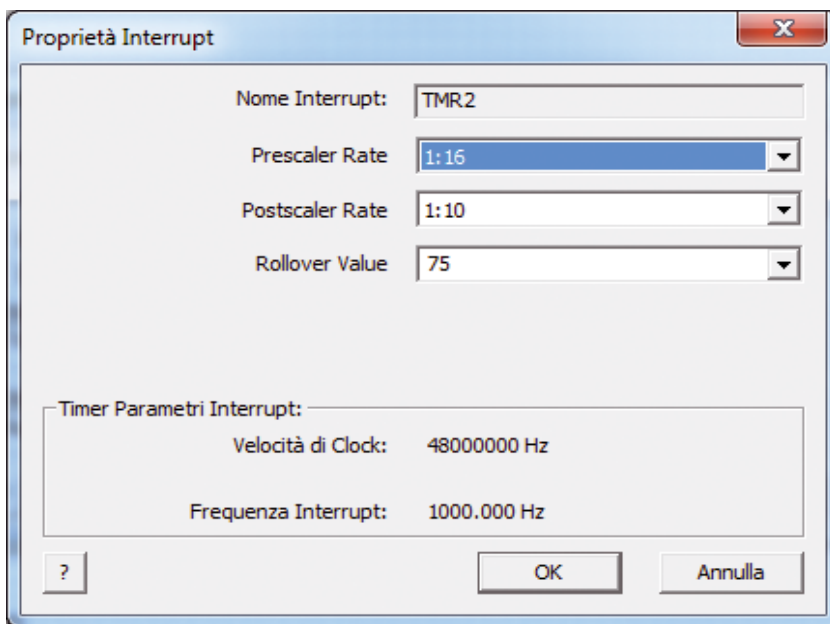


Fig. 7 - Proprietà dell'intervallo del timer 2.

Tabella 7 - Task e la loro periodicità

Task	Periodo [ms]	Descrizione
AppTask	50	Task di controllo applicazione
UsbRxTask	10	Task ricezione dati da USB
UsbTxTask	10	Task trasmissione dati verso USB
LcdTask	10	Task gestione interfaccia LCD
UartTask	10	Task gestione interfaccia UART
SpiTask	10	Task gestione interfaccia SPI
I2cTask	10	Task gestione interfaccia I2C
LedTask	50	Task gestione LED di stato

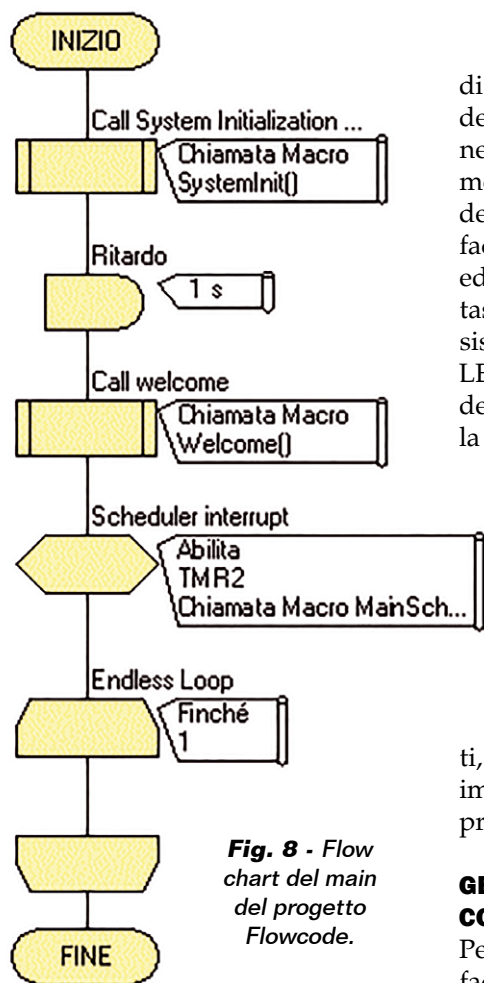


Fig. 8 - Flow chart del main del progetto Flowcode.

Nella Fig. 8, invece, è riportato il main del progetto. Come è possibile vedere, dopo una fase di inizializzazione (dove vengono inizializzate tutte le periferiche del sistema, ossia l'interfaccia USB, la UART, l'SPI, l'PC e l'interfaccia LCD), viene lanciata la macro Welcome, che ha il solo scopo di scrivere sull'interfaccia LCD e sull'interfaccia UART il nome del prodotto e dell'autore. Dopo, viene lanciato il blocco interrupt dello scheduler e il main si conclude con un ciclo infinito vuoto, in quanto tutte le operazioni da questo punto in poi sono gestite interamente dallo scheduler, che lancia ad uno ad uno i vari task in esecuzione. In particolare, lo scheduler controlla l'esecuzione di un totale

di otto task: uno di controllo dell'applicativo (per impostazione della modalità di funzionamento e sviluppi futuri), due task dedicati alla gestione dell'interfaccia USB (uno per la ricezione ed uno per la trasmissione), un task per la gestione del LED di sistema (la ECIO ha un singolo LED connesso al pin numero 2 della porta E), e quattro task per la gestione delle interfacce UART, SPI, PC ed LCD 44780. Nella Tabella 7 sono riportati i nomi di tutti gli 8 task, corredati della descrizione e della relativa periodicità. Passiamo adesso ad un'analisi di dettaglio dei task più importanti, in modo da comprenderne le implementazioni specifiche e le problematiche incontrate.

GESTIONE DEL COLLEGAMENTO USB

Per l'implementazione dell'interfaccia USB, abbiamo diverse possibili opzioni, perché Flowcode implementa tre diversi componenti USB, che corrispondono ad altrettante classi USB:

- Componente USB Serial (corrispondente alla classe USB CDC-Communication Device Class);
- Componente USB HID (corrispondente alla classe USB HID-Human Interface Device);
- Componente USB Slave (corrispondente alla classe USB custom).

Scegliamo di utilizzare la prima classe, che risulta la più adatta alla nostra applicazione: in pratica il nostro dispositivo sarà visto lato PC come una seriale virtuale (COM), semplificando notevolmente la gestione lato software. Per inserire il componente USB serial all'interno del nostro

progetto, basterà semplicemente trascinare l'icona del componente all'interno del pannello, come illustrato in Fig. 9.

A questo punto configuriamo il componente e generiamo il driver necessario all'interfacciamento verso il PC. Eseguire questa operazione è molto semplice, basta cliccare con il tasto destro sull'icona del componente e selezionare la voce "Proprietà Extra". Comparirà la finestra di dialogo di Fig. 10.

All'interno di questa finestra inseriamo il nome dell'interfaccia, il VID (Vendor ID), il PID (Product ID), il nome del produttore, la versione e, se vogliamo, un ritardo di enumerazione (in questo caso è stato impostato un ritardo pari ad un secondo). Una volta completato l'inserimento dei dati, generiamo il driver cliccando sul pulsante "Generate Driver File". Si aprirà una finestra di dialogo che permetterà di salvare un file .inf, il quale costituisce il driver che dovrà poi essere utilizzato per l'interfacciamento della periferica al PC. Ora il nostro componente USB è inserito all'interno del nostro progetto e disponiamo anche del driver per l'interfacciamento. Per completare la descrizione dell'interfaccia USB ci rimane da esaminare la struttura dei due task UsbRxTask ed UsbTxTask.

Iniziamo dal task di ricezione: il componente USB di Flowcode ci mette a disposizione una API che si chiama USBReadString, la quale permette di leggere dall'interfaccia USB una stringa di lunghezza prefissata, entro un certo timeout. Una volta definito il nostro protocollo, abbiamo stabilito che un comando completo viene inviato tramite due transazioni distinte: la prima contenente il comando in sé e la lunghezza del campo dati (all'in-

terno di due opportuni delimitatori) e la seconda contenete il payload effettivo (della lunghezza indicata dal campo dati della prima transazione). Strutturiamo quindi il nostro flow chart come una piccola macchina a stati, il cui stato iniziale si occupa della ricezione di una stringa di lunghezza fissa pari a 7 byte (3 byte di comando, 2 byte lunghezza dati, e 2 byte per i delimitatori del campo lunghezza dati), poi, una volta ricevuto correttamente questo primo segmento, si passa ad un secondo stato che si occupa di ricevere il payload. Inoltre il secondo stato ha anche una condizione di uscita a tempo (timeout), in maniera che il sistema non possa rimanere bloccato nell'attesa di un payload nel caso in cui quest'ultimo vada accidentalmente perso.

Il flow chart che implementa quanto descritto sopra è riportato in Fig. 11: come vedete, all'inizio del diagramma viene controllato lo stato della variabile globale

UsbRxTask_CompleteTransfer. Se si determina che non c'è un trasferimento in corso viene lanciato il segmento destro del flow chart, che ha lo scopo di ricevere la prima transazione da 7 byte (per ricevere i dati viene utilizzata la macro ReadString del componente USB Serial). Se sono stati ricevuti dei dati e se il formato è corretto, condizione che viene controllata verificando la corretta posizione dei delimitatori, tramite un blocco decisionale con la seguente condizione:

`(.UsbCommand[3] = '<') AND (.UsbCommand[6] = '>')`

Viene inviato un messaggio di conferma ('OK'), si esegue il parsing del buffer di ricezione, ricavando la stringa di comando (UsbRxTask_UsbCmd) ed un valore numerico contenente la lunghezza in byte del payload (UsbRxTask_UsbDataLen), e si procede alla ricezione del secon-

do frame. Nel caso il formato non sia corretto, viene invece inviato un messaggio di errore. Naturalmente si cambia lo stato del flag UsbRxTask_CompleteTransfer, in modo da indicare che una transazione dati è in corso e deve venire ricevuto il secondo frame (`UsbRxTask_CompleteTransfer = .GET_DATA`).

A questo punto, alla successiva iterazione va in esecuzione il segmento sinistro del flow chart; si utilizza nuovamente la macro ReadString per ricevere il frame relativo al payload (la lunghezza del frame è nota dalla transazione precedente, ed è stata caricata all'interno della variabile globale UsbRxTask_UsbDataLen). Se dei dati sono stati ricevuti viene controllato il formato (presenza dei delimitatori '<' e '>' rispettivamente all'inizio ed alla fine della stringa dati) e si procede al parsing. Se invece il formato non è corretto viene inviato un messaggio di errore. Il ciclo di attesa per la ricezione del se-

Fig. 9
Inserimento
del compo-
nente USB
Serial

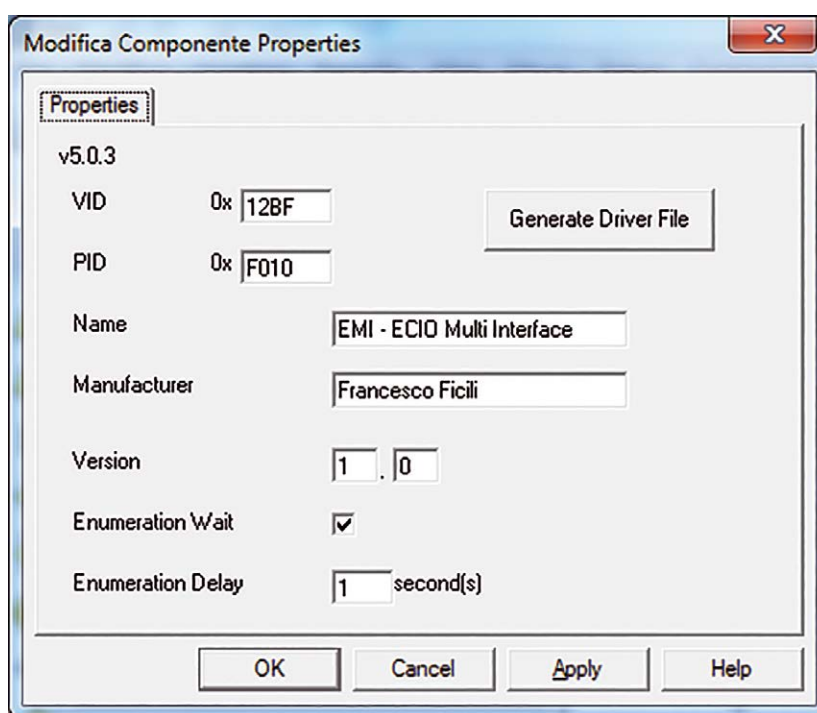


Fig. 10
Proprietà
Extra del
componente
USB Serial.

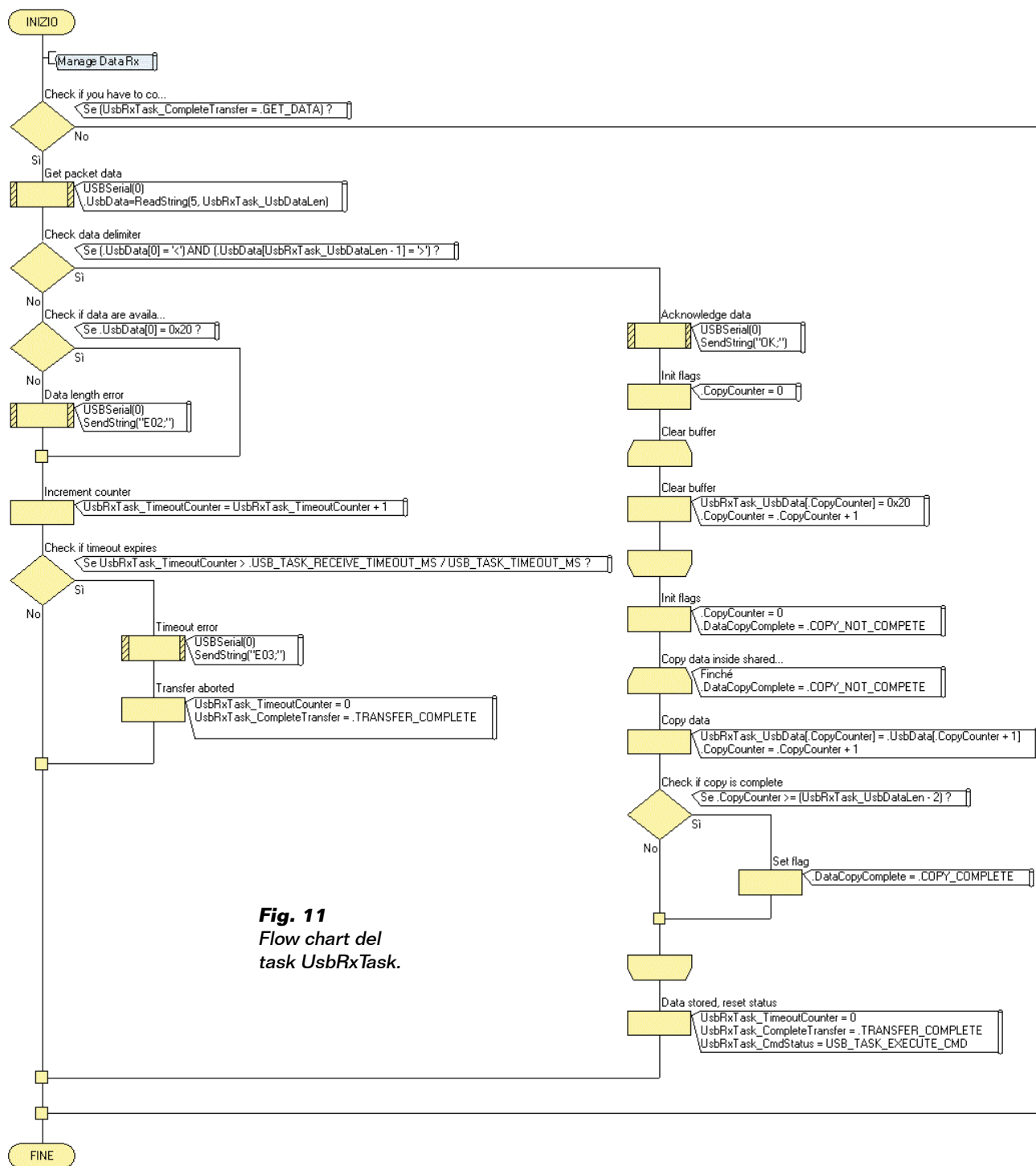


Fig. 11
Flow chart del
task UsbRxTask.

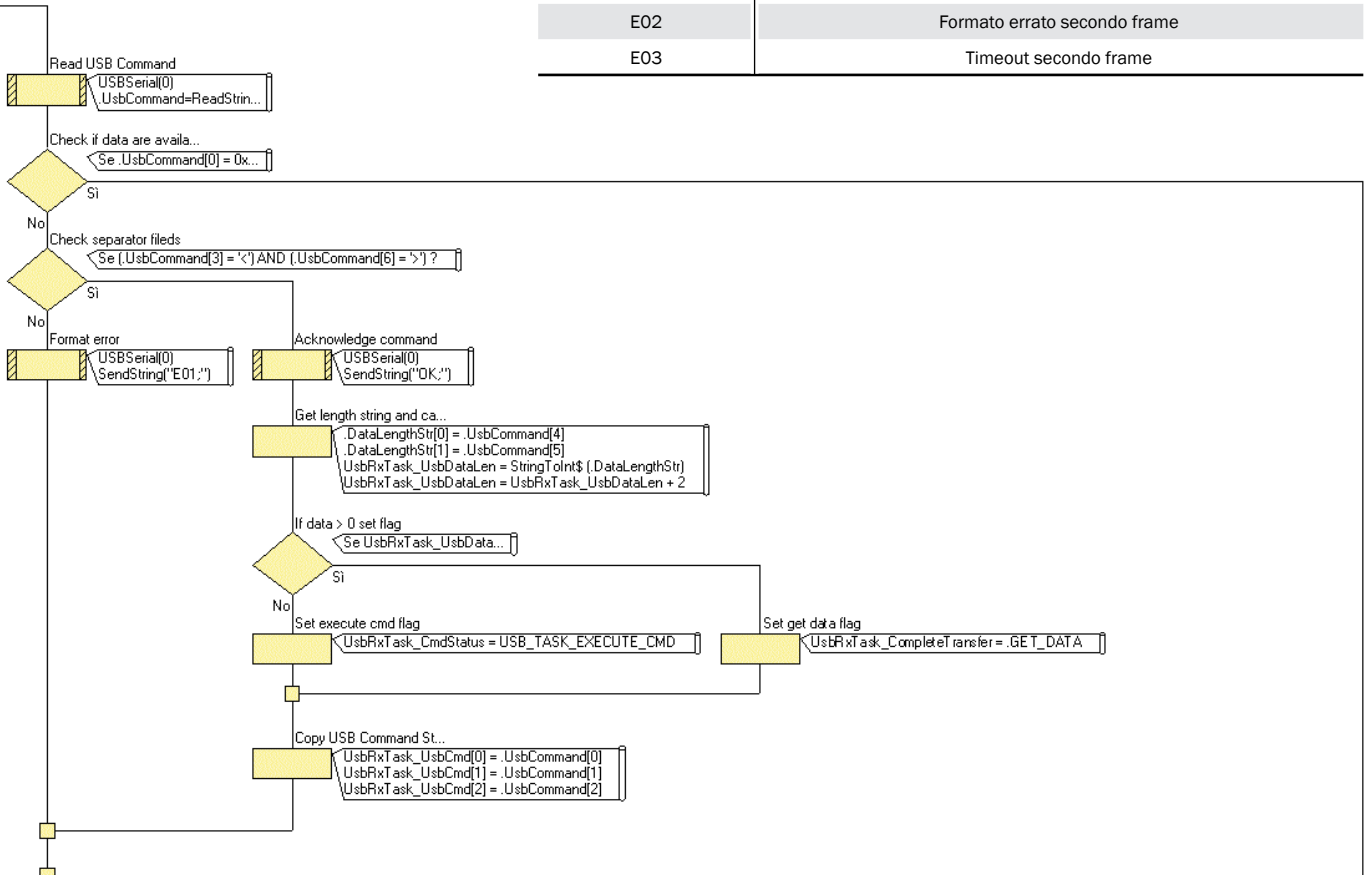
condo frame non dura in eterno, infatti durante questa fase viene periodicamente incrementato il contatore UsbRxTask_TimeoutCounter, e se il valore supera una soglia prefissata (nel nostro caso 10s), si abortisce l'operazione, si invia un messaggio di errore e si ritorna alla ricezione del primo

frame. Nella **Tabella 8** sono riportati i messaggi di errore utilizzati ed il loro significato. Passiamo adesso all'analisi del task UsbTx, il cui flow chart è riportato in **Fig. 12**; come vedremo, questo task è decisamente più semplice del precedente. In pratica, viene utilizzato un flag,

UsbTxTask_ReplyStatus (che può assumere i due valori USB_TASK_SEND_REPLY o USB_TASK_NOT_SEND_REPLY), per determinare se debba essere trasmesso o meno un messaggio di risposta. Nel caso in cui si debba effettuare una trasmissione, il flag viene resettato e si utilizza

Tabella 8 - Codici di errore

Messaggio	Significato
E01	Formato errato primo frame
E02	Formato errato secondo frame
E03	Timeout secondo frame



una struttura condizionale per verificare quale interfaccia sia attiva ed inviare in trasmissione il buffer corretto.

Task APP

Come abbiamo dedotto dal precedente esempio, può essere at-

tiva solo un'interfaccia per volta; questo, sia per evitare situazioni di conflitto, sia perché alcune interfacce possono potenzialmente condividere alcune risorse hardware. C'è quindi l'esigenza di avere un task che gestisca l'applicazione ed un meccanismo che garantisca che non venga lanciata in esecuzione più di un'interfaccia alla volta.

Per applicare questo tipo di

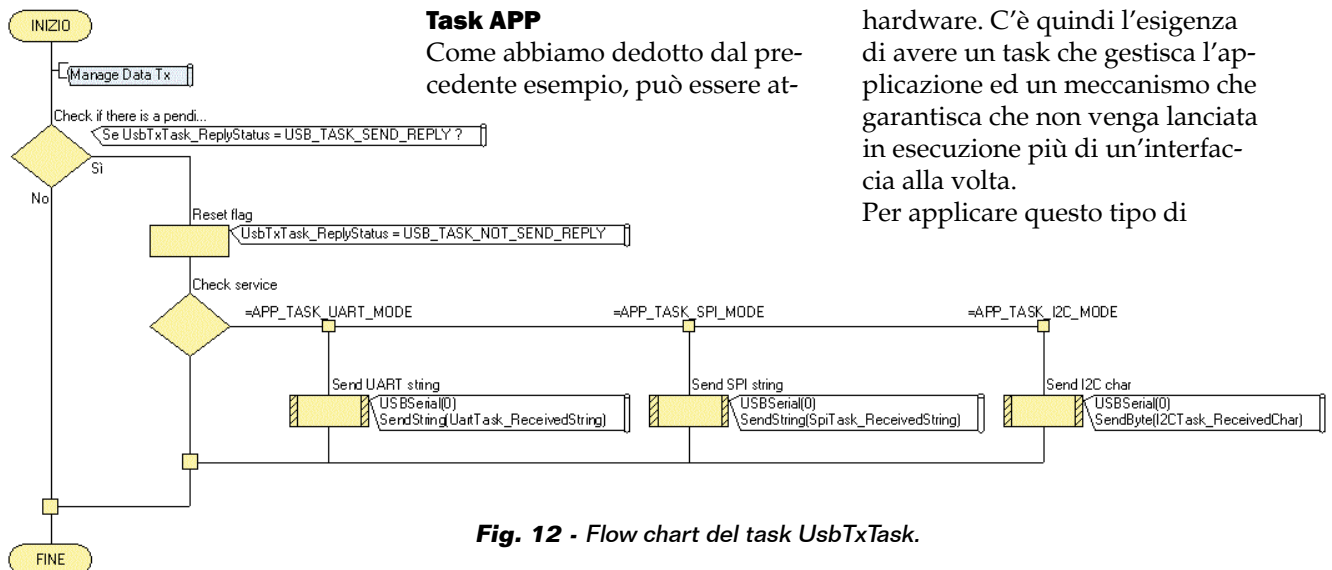
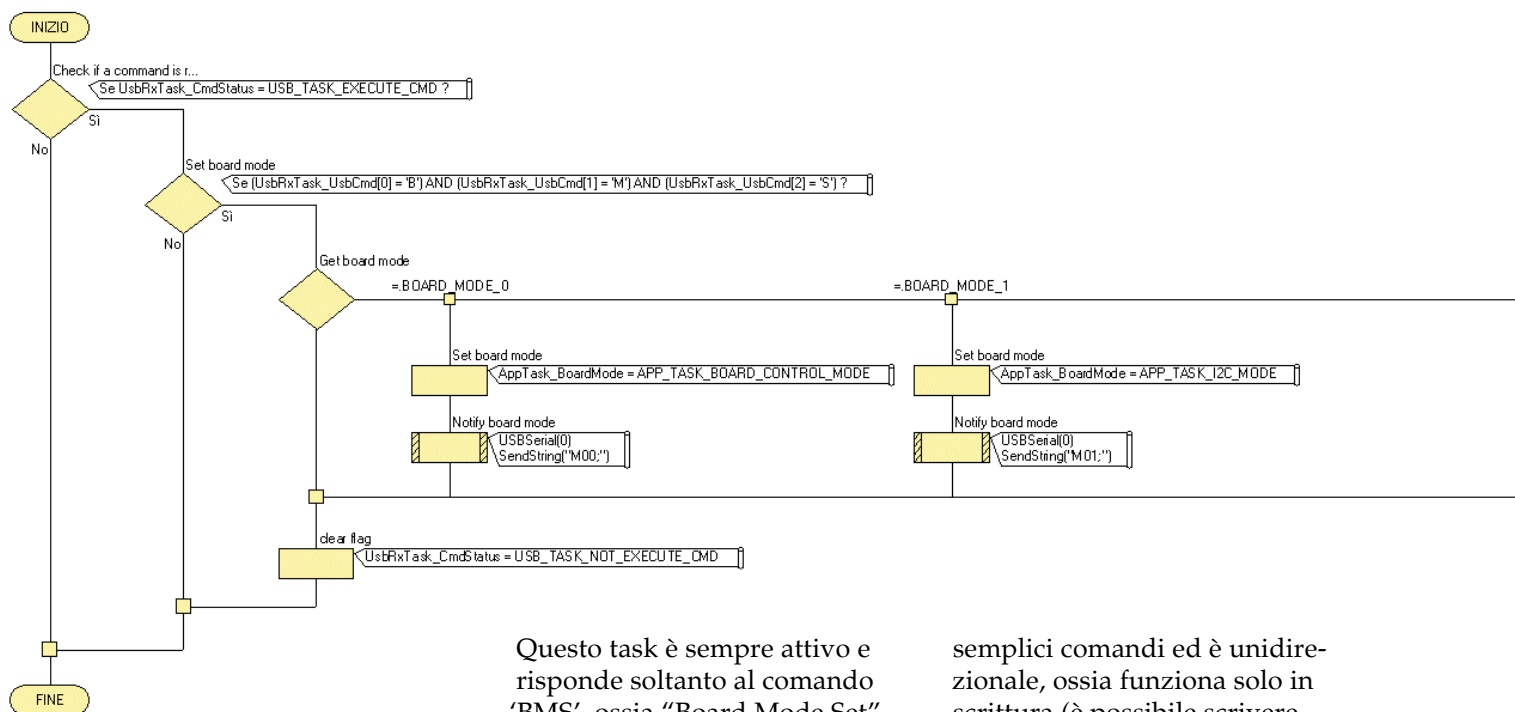


Fig. 12 - Flow chart del task UsbTxTask.



gestione abbiamo utilizzato un task applicativo, il cui compito è quello di gestire lo stato di una variabile globale che indica quale interfaccia è attiva al momento. Ogni task che sia dedicato alla gestione di un'interfaccia ha il compito di controllare lo stato di questa variabile per determinare se essa debba andare in esecuzione, oppure no. Il flow chart che implementa il task applicativo (AppTask) è riportato in **Fig. 13**.

Questo task è sempre attivo e risponde soltanto al comando 'BMS', ossia "Board Mode Set", il quale prende come parametro un numero che indica quale modalità l'utente vuole impostare (vedere **Tabella 1** e **Tabella 2**). Passiamo adesso ad un'analisi più dettagliata di alcune interfacce e prendiamo in analisi, a titolo di esempio, l'interfaccia LCD.

Task LCD

Quella verso l'LCD è una delle interfacce più semplici da gestire, in quanto dispone di pochi

semplici comandi ed è unidirezionale, ossia funziona solo in scrittura (è possibile scrivere su un pannello LCD collegato all'interfaccia LCD della EMI attraverso un software host su PC). In realtà l'interfaccia con l'LCD può essere bidirezionale, ma in questo caso la assumiamo unidirezionale. Le funzioni che l'interfaccia LCD deve implementare sono:

- scrittura di una stringa su display;
- spostamento del cursore;
- cancellazione dello schermo.

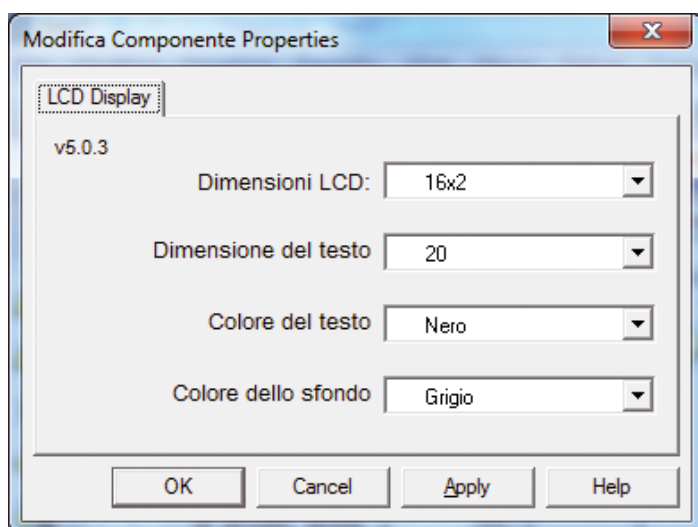


Fig. 14 - Proprietà extra componente LCD.

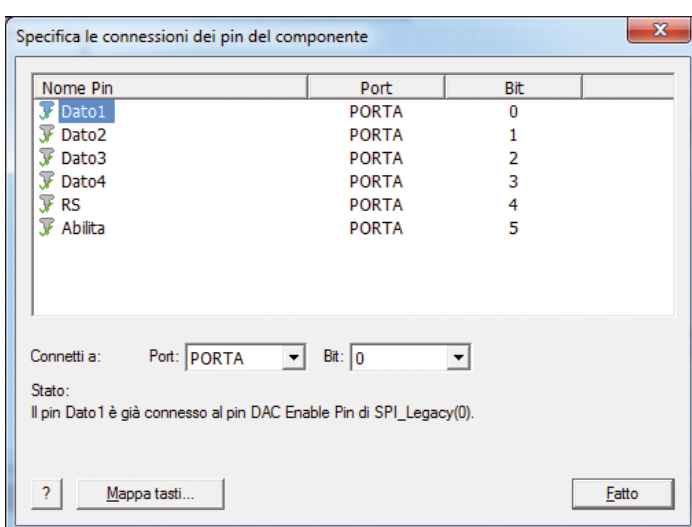
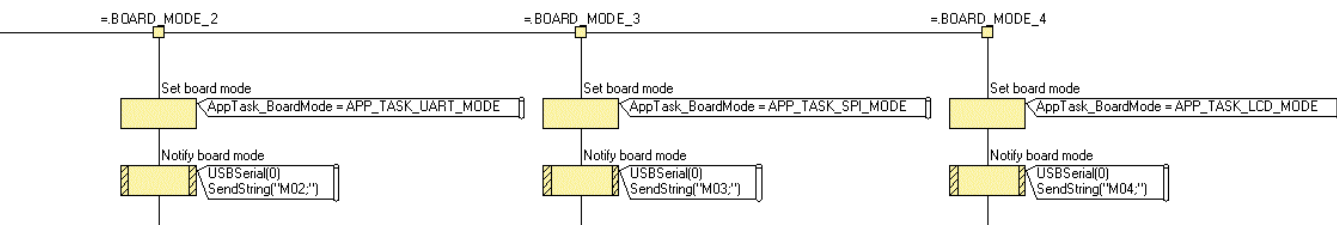


Fig. 15 - Connessioni componente LCD.

Fig. 13 - Flow chart del task AppTask.



Per prima cosa importiamo il componente LCD all'interno del nostro progetto, prelevandolo dalla barra dei componenti sotto la voce "Uscite\ LCD". Il componente va anche opportunamente inizializzato nella macro SystemInit. Inoltre vanno settate le proprietà extra del componente, che in questo caso si riducono semplicemente all'impostazione del tipo di display LCD che si desidera interfacciare (Fig. 14) e le connessioni.

Per questo progetto si è deciso che l'interfaccia LCD si troverà sulla porta A (Fig. 15), ma non sarebbe assolutamente un problema, in futuro, utilizzare un'altra porta, dato che non ci sia appoggiata su hardware specifico ma su semplici GPIO.

Il flow chart che implementa il task LCD è riportato in Fig. 16. Come è possibile vedere, prima di eseguire qualsiasi operazione vengono effettuati i seguenti due controlli in cascata:

`AppTask_BoardMode = APP_TASK_LCD_MODE`
e
`UsbRxTask_CmdStatus = USB_TASK_EXECUTE_CMD`

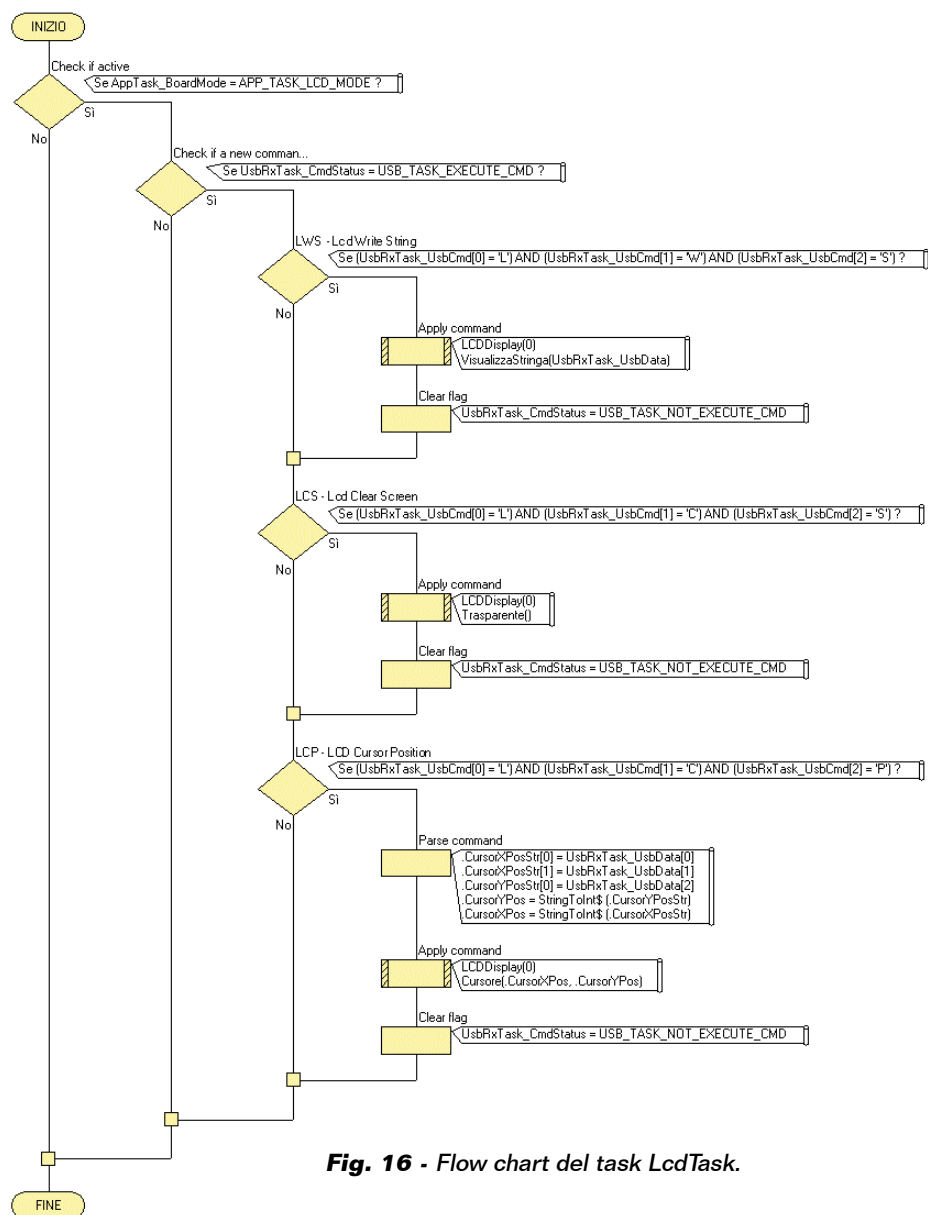


Fig. 16 - Flow chart del task LcdTask.

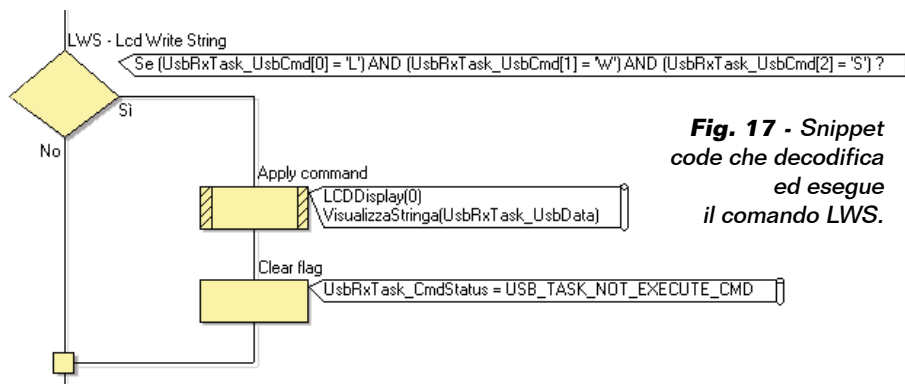


Fig. 17 - Snippet code che decodifica ed esegue il comando LWS.

Il primo controllo verifica che la modalità impostata sia la modalità LCD (mode 04) e, successivamente, viene verificato se è stato ricevuto un comando con il formato corretto sulla porta USB. Se questi due controlli danno esito positivo, si passa all'esecuzione del task, che consiste nell'effettuare la decodifica del comando ricevuto ed eseguire l'azione specifica richiesta. Ad esempio, il comando che permette di scrivere una stringa su un display LCD usando l'apposita interfaccia, è il comando "LWS" (Lcd Write String). Il segmento di codice grafico che esegue questa operazione è riportato in Fig. 17: come si può vedere, una struttura di selezione verifica il comando, tramite il seguente controllo:

$(UsbRxTask_UsbCmd[0] = 'L') \text{ AND } (UsbRxTask_UsbCmd[1] = 'W') \text{ AND } (UsbRxTask_UsbCmd[2] = 'S')$

Dopodichè viene attuata l'operazione richiesta tramite la macro VisualizzaStringa del componente LCD. I parametri della macro sono riportati in Fig. 18.

Come si vede dalla finestra delle proprietà della macro, la stringa da visualizzare passata come parametro è proprio quella ricevuta come payload del comando LWS, opportu-

namente decodificata e caricata sulla variabile condivisa UsbRxTask_UsbData, dal task UsbRxTask.

L'ultima operazione eseguita è il reset del flag UsbRxTask_CmdStatus, in modo da indicare che il task LCD ha decodi-

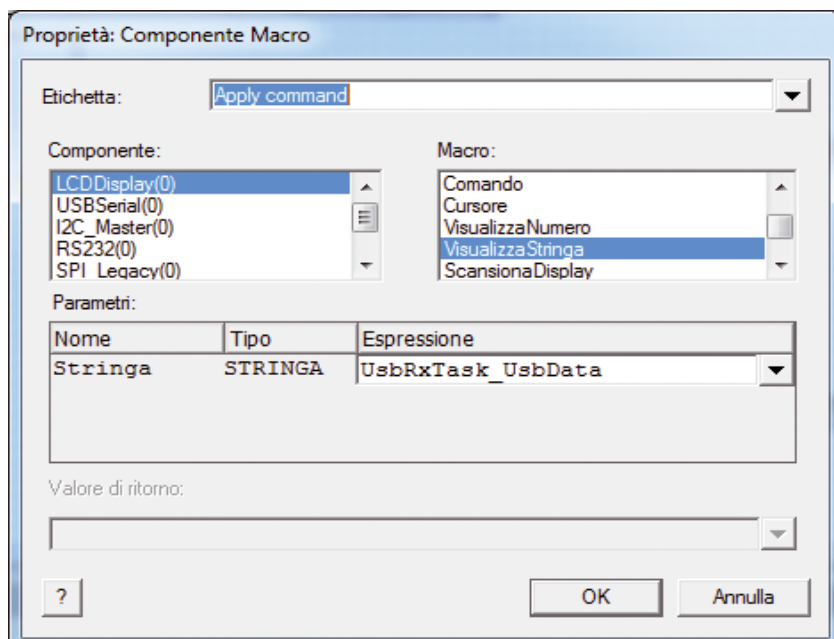


Fig. 18 - Parametri della macro componente VisualizzaStringa del componente LCD.

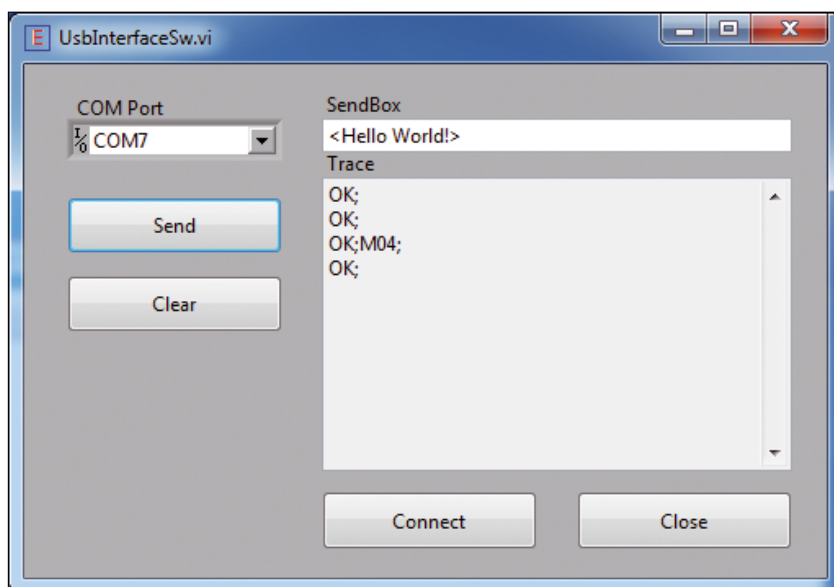
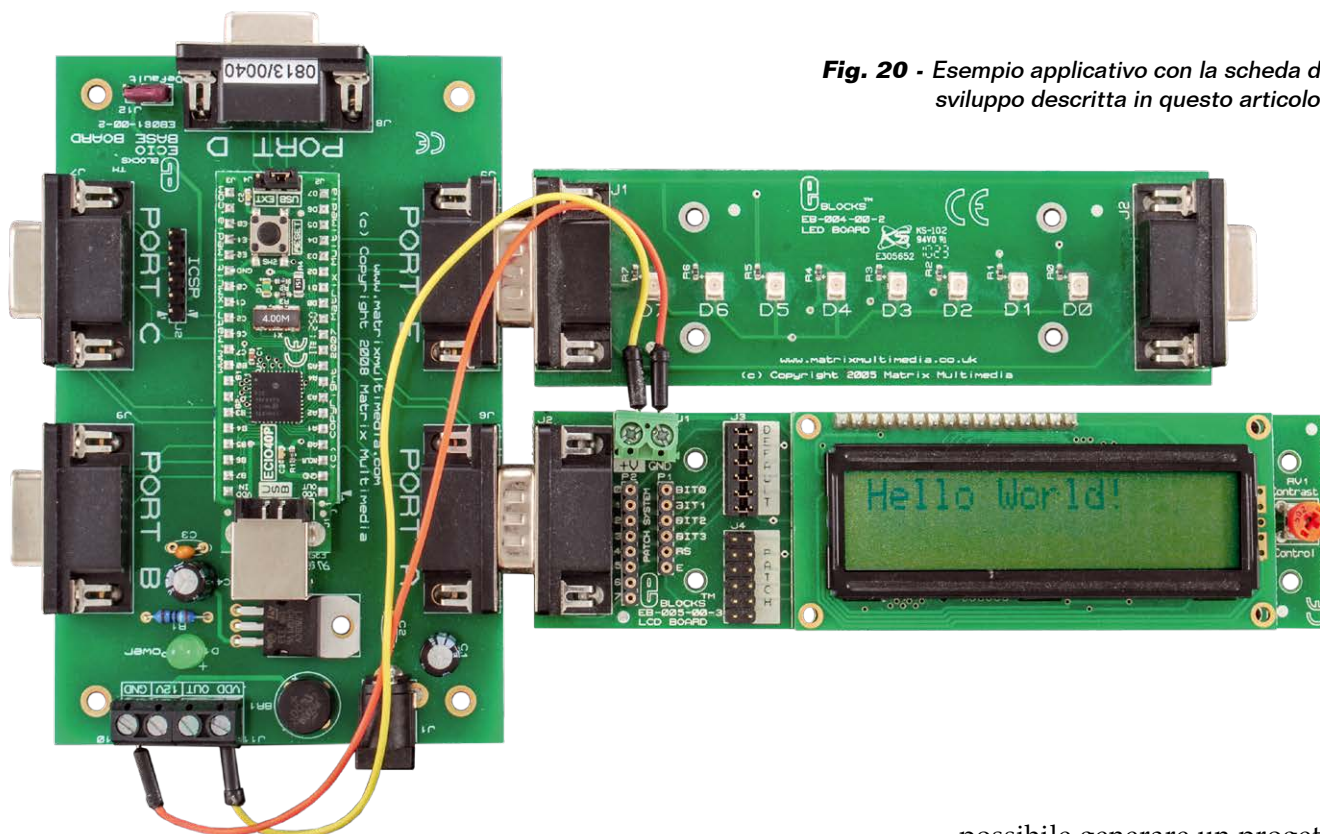


Fig. 19 - Interfaccia LabVIEW.

Fig. 20 - Esempio applicativo con la scheda di sviluppo descritta in questo articolo.



ficato ed eseguito il comando, ed è pronto per una eventuale nuova operazione. Similmente vengono eseguite anche le altre due funzioni, come la cancellazione dello schermo oppure lo spostamento del cursore sul display. La gestione delle altre interfacce di cui dispone il sistema è del tutto simile, con una maggiore complicazione nel caso di task che oltre che inviare dati su un certo canale, li debbano anche ricevere (come nel caso, ad esempio, dell'UART). In questo caso l'implementazione deve prendersi cura di ricevere eventuali dati dalla periferica e ritomarli indietro all'host che li ha richiesti (ad esempio un programma di interfaccia che gira sui PC). Lasciamo ai lettori più interessati alla materia, l'analisi dell'implementazione di queste interfacce ed, eventualmente, anche l'aggiunta di nuove funzionalità.

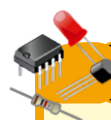
TEST DELL'INTERFACCIA

A scopo di test è stata sviluppata un'interfaccia in LabVIEW (l'ambiente di progettazione grafica sviluppato da National Instruments) che permette di dialogare con la scheda. Nell'esempio mostrato nella Fig. 19 la scheda viene impostata in modalità LCD (BMS<01> ..<4>) e poi viene scritta la stringa "Hello World!", tramite la sequenza di comandi LWS<12> .. <Hello World!>. Il risultato di tale operazione è visibile nella Fig. 20.

CONCLUSIONI

Come abbiamo visto, realizzare un sistema multi interfaccia capace di parlare cinque lingue differenti (USB, UART, I²C, SPI, LCD) con Flowcode è relativamente semplice e con alcuni accorgimenti (uso di uno scheduler come infrastruttura base di servizio) è anche

possibile generare un progetto fortemente scalabile: se un domani volessimo aggiungere la possibilità di controllare dei canali PWM o dei canali analogici, ci basterebbe aggiornare il nostro protocollo di comunicazione ed inserire i task che implementano le funzionalità aggiuntive. In sostanza, l'ambiente si presta ad essere utilizzato anche per progetti di una certa complessità e permette di ridurre notevolmente i tempi di sviluppo rispetto ad un ambiente testuale, chiaramente a prezzo di una maggiore astrazione di alcuni aspetti dello sviluppo. ■



per il MATERIALE

Il software FLOWCODE e le schede ECIO si possono acquistare sul sito www.matrixsl.com

CORSO

MPLAB X

3

di FRANCESCO FICILI
e VINCENZO GERMANO

Proseguiamo lo studio di MPLab X, il nuovo ambiente di sviluppo integrato di Microchip Technology, che soppianta il vecchio MPLab IDE. Approfondiamo la conoscenza di alcune periferiche di base dei microcontrollori PIC32, i primi dispositivi a 32-bit prodotti da Microchip. Terza puntata.

Nella puntata precedente abbiamo approfondito la nostra conoscenza dei microcontrollori PIC32 della Microchip e di alcune loro periferiche di base: i Timer e gli Interrupt. Ne abbiamo analizzato e compreso le principali caratteristiche e abbiamo imparato a utilizzarli, tramite facili esempi, grazie al software MPLab X. In questa terza puntata focalizziamo la nostra attenzione su altre periferiche di base dei PIC32, che verranno applicate e utilizzate su una demoboard pensata, progettata e realizzata appositamente per questo corso.

LE PERIFERICHE INPUT CAPTURE E OUTPUT COMPARE

Oltre alla gestione dei Timer, come visto nella

scorsa puntata, il microcontrollore PIC32 fornisce due periferiche di base che migliorano il funzionamento dei Timer e sono in grado di semplificare svariate applicazioni, trovando largo impiego nella registrazione di eventi, nella misurazione di periodi di tempo, nei conteggi, nella generazione di impulsi e di forme d'onda periodiche; tali periferiche sono la Input Capture e la Output Compare.

Ma vediamo maggiormente nel dettaglio per comprenderne meglio il funzionamento. Utilizzando la periferica Input Capture (IC) i registri funzionano come un cronometro, infatti quando si verifica un particolare evento, il tempo dell'evento viene registrato e il clock continua a funzionare normalmente (un cronometro,

funzionamento desiderata. Quando viene utilizzato in modalità impulso continuo (OCM=101), in particolare, il registro OCxR viene utilizzato per determinare l'istante (rispetto al valore del temporizzatore associato) in cui il pin d'uscita viene impostato, mentre il registro OCxRS determina quando il pin di uscita viene resettato.

LA PERIFERICA PWM

Come abbiamo visto, la periferica Output Compare può essere utilizzata come generazione di impulso singolo, generazione di impulsi continua e modulazione di larghezza di impulso (PWM). Possiamo affermare che la configurazione PWM è molto utile in svariate applicazioni ed è forse quella di gran lunga più utilizzata tra le tre.

La modalità adottata per la generazione di un segnale PWM è una combinazione tra la normale esecuzione di un Timer e la modalità Output Compare, grazie alla quale è possibile erogare un segnale PWM utilizzando il reset di un Timer a un valore specifico. Questo aspetto sarà maggiormente comprensibile nell'esempio di utilizzo del PWM.

Il modo in cui un segnale PWM funziona è piuttosto semplice: un impulso viene prodotto a intervalli regolari (T), tipicamente forniti da un Timer e da un suo registro di periodo; la larghezza di impulso (Ton), però, non è fissa ma è programmabile e può variare tra 0% e 100% del periodo del Timer. Il rapporto tra la larghezza dell'impulso (Ton) e il periodo del segnale (T) viene chiamato duty-cycle, come riportato in Fig. 3. Quindi un'uscita PWM è fondamentalmente un'onda quadra con un periodo e un ciclo di lavoro specifico.

I due valori limite del duty-cycle sono 0% e 100%: il primo corrisponderebbe a un segnale nullo, mentre il secondo corrisponde a un segnale di uscita costante.

Per quanto riguarda la risoluzione del PWM, tipicamente viene indicata come il logaritmo in base 2: per esempio, se ci sono 256 possibili ampiezze di impulso, diciamo che abbiamo un segnale PWM con una risoluzione di 8 bit.

ESEMPIO DI UTILIZZO DEL PWM

Passiamo a un esempio pratico di generazione del segnale PWM mediante i moduli OC del microcontrollore PIC32MX, dove genere-

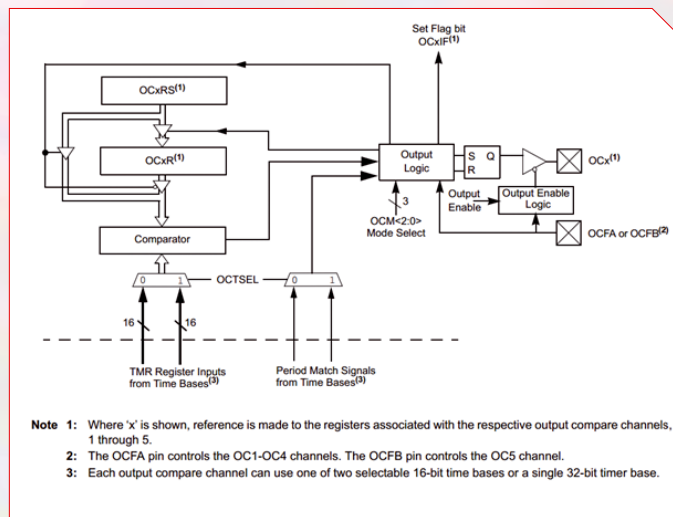


Fig. 2 - Schema a blocchi di un Output Compare.

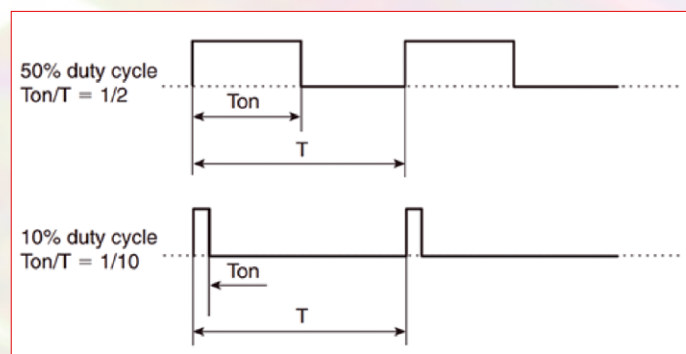


Fig. 3 - Segnale PWM di riferimento.

remo un flusso continuo di impulsi con duty-cycle desiderato.

Tutto quello che dobbiamo fare per inizializzare il modulo OC e generare un segnale PWM è impostare i tre bit del registro OCM all'interno del registro di controllo OCxCON; a riguardo, fate riferimento alla **Tabella 1**. Per ottenere un segnale PWM di base, conviene configurarlo come 0x110. È prevista una seconda modalità PWM (0x111) ma nel nostro caso non traiamo alcuna utilità nell'abilitare i "Fault" (guasto) sul pin, comunemente richiesto da diverse applicazioni come possono essere i meccanismi di protezione (ad esempio controllo motore o conversione di potenza).

Successivamente abbiamo bisogno di selezionare il Timer su cui basare il periodo di PWM, scegliendo tra Timer2 e Timer3. Nel caso in cui vogliamo essere in grado di produrre un periodo PWM a 40 kHz e supponendo un clock periferico di 16 MHz, impostando un prescaler a un rapporto 1:1 otteniamo un

Tabella 1 - Registro di controllo per la periferica Output Compare.

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
	ON ⁽¹⁾	—	SIDL	—	—	—	—	—
7:0	U-0	U-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	OC32	OCFLT ⁽²⁾	OCTSEL	OCM<2:0>		

periodo di 400 cicli. Da questo valore deriva anche la risoluzione del duty-cycle relativo al PWM per il modulo di Output Compare: infatti, dal momento che avremo 400 possibili valori del ciclo di lavoro, siamo in grado di avere una risoluzione compresa tra 8 e 9 bit, più di 256 (2^8) passi ma meno di 512 (2^9).

Dopo che il timer scelto viene configurato e poco prima di scrivere nel registro OCxCON, bisogna impostare, per la prima volta, il valore del duty-cycle nel registro OCxR e nel registro OCxRS. In modalità PWM, i due registri lavoreranno in configurazione master/slave e una volta che il modulo PWM verrà avviato, saremo in grado di cambiare il ciclo di lavoro scrivendo solo nel registro OCxRS. Il registro OCxR sarà aggiornato, copiando un nuovo valore dal OCxRS, solo e precisamente all'inizio di ogni nuovo periodo, allo scopo di evitare difetti (glitches) e avere un intero periodo (T) di tempo per preparare il prossimo valore del duty-cycle. Tutto questo sarà maggiormente chiaro nell'esempio *Dimmer digitale (LED con PWM da potenziometro)* che vedremo più avanti.

LE PERIFERICHE ANALOGICHE DEI PIC32

Guardandoci intorno comprendiamo subito che viviamo in un mondo analogico, infatti le varie grandezze come temperatura, umidità, pressione ma anche tensioni e correnti sono analogiche. Se vogliamo che le nostre applicazioni elettroniche debbano interagire con il mondo esterno, abbiamo la necessità di interpretare le informazioni analogiche e convertirle nel mondo digitale in modo tale che un microcontrollore possa essere in grado di elaborarle e produrre eventualmente ancora un'uscita analogica. Il modulo convertitore analogico/digitale (ADC, acronimo di Ana-

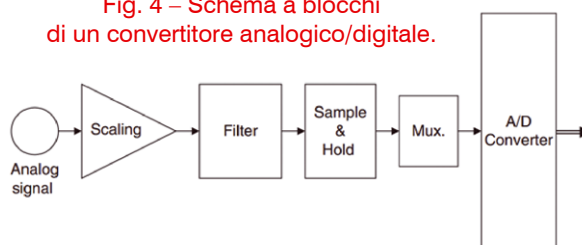
log to Digital Converter) è una delle interfacce chiave con il mondo "reale" e la famiglia PIC32MX è stata progettata per affrontare al meglio la natura analogica di questo mondo. Un convertitore veloce analogico/digitale, in grado di 500.000 conversioni al secondo, è disponibile su tutti i modelli con un multiplexer di ingresso che consente di monitorare un numero di ingressi analogici in modo rapido e con alta risoluzione. Passiamo ad analizzare le caratteristiche più salienti di questa periferica di base dei microcontrollori e nello specifico del PIC32.

La periferica ADC

Il convertitore analogico/digitale è una periferica che converte una tensione di ingresso analogica in un numero digitale in modo tale che possa essere elaborato da un microcontrollore o qualsiasi altro sistema digitale. Possiamo classificare le tensioni gestite dagli ADC come: unipolari e bipolari. Nel primo caso, vengono accettate tensioni di ingresso con una sola polarità, mentre nel secondo caso il range delle tensioni gestite è sia positivo che negativo. In linea generale, in Fig. 4 sono riportate le fasi tipiche coinvolte nella lettura e la conversione di un segnale analogico in forma digitale, un processo noto anche come "conditioning".

I segnali ricevuti dai sensori, di solito devono

Fig. 4 – Schema a blocchi di un convertitore analogico/digitale.



essere elaborati prima di essere processati da un convertitore ADC. Questo trattamento di solito inizia con lo "scalare" (Scaling) il segnale al valore di tensione corretto. Le componenti del segnale indesiderate vengono rimosse filtrando il segnale mediante filtri classici (ad esempio, un filtro passa-basso). Prima che il segnale entri in un convertitore ADC, esso viene fatto passare attraverso un dispositivo di campionamento e tenuta (Sample&Hold – S&H), utile per leggere segnali veloci (in tempo reale) il cui valore può cambiare tra gli istanti di campionamento; l'S&H è sostanzialmente uno switch bipolare solid state all'uscita del quale c'è un condensatore, che si carica al valore istantaneo del segnale quando lo switch è chiuso, mantenendo una tensione costante durante il processo di conversione.

Continuando lungo la catena di acquisizione mostrata in Fig. 4, molte applicazioni potrebbero necessitare di vari ADC e per sopperire a tale necessità normalmente viene utilizzato un multiplexer analogico (MUX) in ingresso al convertitore, che seleziona solo un segnale alla volta e lo porta in ingresso all'ADC. L'uscita del convertitore è digitale e tipicamente "parallela" digitale.

Possiamo riassumere il processo di conversione mediante un microcontrollore:

1. si applica il segnale da elaborare all'ingresso dell'ADC;
2. si avvia la conversione;
3. si attende fino a quando la conversione è completa;
4. si legge il dato digitale convertito.

La conversione viene avviata attivando il convertitore, che, a seconda della sua velocità, condiziona la velocità del processo di conversione stesso. Al termine della conversione, il convertitore genera un interrupt per indicare che la conversione è completa e i dati di uscita convertiti possono essere letti dal dispositivo digitale collegato all'ADC.

Facendo riferimento alla Fig. 5, in cui è riportato lo schema a blocchi dell'ADC a 10-bit di un PIC32, notiamo che ha fino a 16 pin di ingresso analogici, chiamati con AN0-AN15, oltre a due pin di ingresso analogici per l'applicazione di riferimenti di tensione esterni, che possono venire condivisi con altri pin di

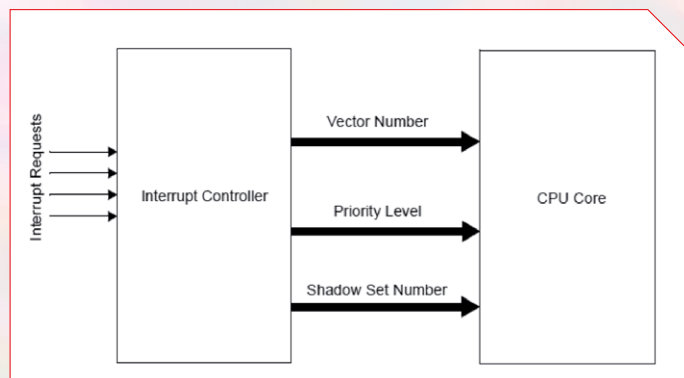


Fig. 5 – Schema a blocchi di un modulo ADC.

ingresso analogici e possono essere comuni ad altri riferimenti di moduli analogici. Quanto analizzato precedentemente in linea generale, lo riscontriamo anche in quest'architettura; infatti gli ingressi analogici sono collegati tramite due multiplexer (MUX) e possono essere commutati tra due set di ingressi. In più, si può notare la presenza del blocco di campionamento e tenuta (S&H). In una tale architettura è presente un registro di controllo che specifica quali canali di ingresso analogici saranno inclusi nella sequenza di scansione. Infine, dal diagramma a blocchi si evince che l'ADC è collegato ad un buffer (16-word) e ogni risultato a 10 bit viene convertito in uno degli otto formati di uscita a 32 bit quando viene letto dal buffer.

Riassumendo brevemente le caratteristiche di questo modulo interno al PIC32, abbiamo:

- conversione con registri ad approssimazioni successive (SAR - Successive Approximation Register);
- fino a 1 Msps di velocità di conversione e fino a 16 pin di ingresso analogici;
- pin d'ingresso per un riferimento di tensione esterna;
- sorgente di trigger selezionabile per la conversione;
- modalità di riempimento del buffer selezionabile;
- funzionamento durante la modalità di sospensione (Sleep) e inattività (Idle).

Per maggiori dettagli sulle caratteristiche/funzionalità e sul loro utilizzo, fate riferimento al data-sheet reperibile sul sito ufficiale della Microchip Technology.

DIMMER DIGITALE (LED CON PWM DA POTENZIOMETRO)

Passiamo adesso a illustrare un semplice esempio di come pilotare un LED mediante

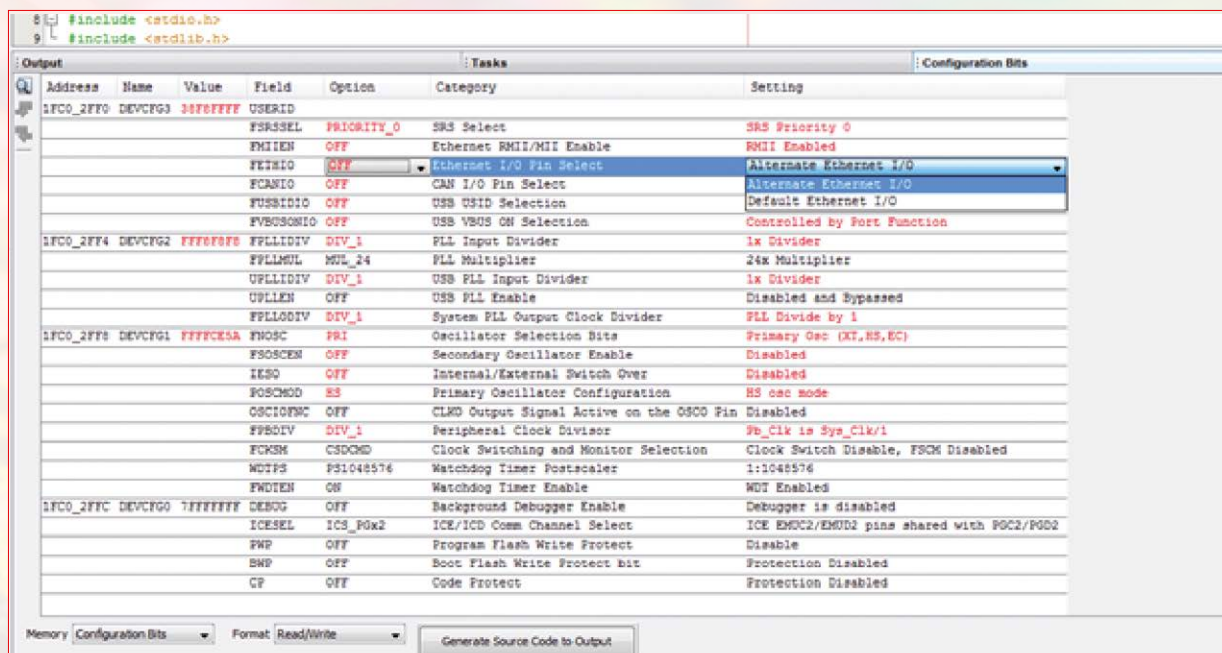


Fig. 6 – Esempio del tab relativo ai Configuration Bit.

l'utilizzo di un potenziometro, continuando a integrare le informazioni presentate nel corso delle puntate precedenti con le nozioni affrontate in queste pagine.

Una differenza sostanziale rispetto agli esempi proposti nelle puntate precedenti è nell'impostazione di questo progetto pratico e dei prossimi a venire: da ora in poi verrà utilizzata come piattaforma hardware associata a questo corso la scheda "DemoboardPIC32", pensata, progettata e realizzata appositamente (fate riferimento al riquadro DemoboardPIC32). Quindi come primo passo si abbandonerà il simulatore e si imporrà il progetto in MPLab X per riconoscere e utilizzare un hardware fisico: la DemoboardPIC32. Tutto questo comporta un ulteriore passaggio nella creazione del progetto in MPLab X, che riguarda la schermata di selezione nota come "Select Tool", all'interno della quale bisognerà selezionare il proprio "Hardware Tool" (ad esempio ICD3, PICkit2, PICkit3 e via dicendo) per la programmazione/debug della scheda, al posto del simulatore (Simulator). La restante parte di creazione del progetto è invariata. Perciò creiamo un quarto progetto da MPLab X tenendo presente quanto appena detto e nominiamolo Ex4; all'interno del progetto creiamo un file C di tipo main e apriamolo all'interno dell'editor.

Prima di passare alla stesura del codice,

un'attenzione particolare va all'impostazione del microcontrollore mediante i "Bit di configurazione" o "Configuration Bits", un insieme di bit speciali che possono essere modificati solo in fase di programmazione. Si tenga presente che dipendono dalle caratteristiche del microcontrollore stesso come vedremo a breve. Tali bit vengono "letti" nella fase di reset per attivare o disattivare le caratteristiche hardware del microcontrollore, che includono, tra le altre funzionalità, la sorgente di clock, timer watchdog, protezione della memoria e molto altro ancora. I Configuration Bits non sono codice eseguibile ma li potete immaginare essenzialmente come "fusibili" situati nello spazio di memoria del programma. Per poterli configurare in MPLab X, bisogna selezionare nel "main menu" Window→PIC Memory Views→Configuration Bits, verrà aperto un nuovo tab nell'area *Output* sotto l'editor; tale tab permetterà di configurare rapidamente questi bit fondamentali per la corretta esecuzione del programma.

Con riferimento alla Fig. 6, si evince che essendo il PIC32 un micro di gamma alta, i suoi bit di configurazione sono molti e in alcuni casi di non rapida intuizione, per questo si consiglia sempre di far riferimento al data-sheet del microcontrollore selezionato, per poter effettuare le impostazioni desiderate. In aggiunta, si può notare come MPLab X ten-

ga traccia delle modifiche di tali bit, colorando di rosso le impostazioni cambiate rispetto a quelle predefinite. Infine, grazie al tasto "Generate Source Code to Output" è possibile copiare e incollare il codice che viene automaticamente generato (secondo le impostazioni scelte) direttamente nel file d'interesse, il

quale può essere il main o un file distinto per una più corretta e pulita gestione del codice. A questo punto non ci resta che passare alla scrittura del codice relativo al nostro esempio di pilotaggio LED mediante potenziometro. Avendo già visto negli esempi delle precedenti puntate come si configurano i Timer e

DEMOBOARD PIC32

La "Demoboard PIC32" è stata pensata e realizzata in modo da poter essere utilizzata come starter kit e quindi come strumento di supporto hardware per questo corso; prevede tutte le periferiche base, più alcune periferiche avanzate, che verranno analizzate in dettaglio nel corso delle varie puntate. Come si vede nello schema a blocchi in questo riquadro, il cuore di tutta la scheda è il microcontrollore Microchip Technology PIC32MX795F512L, dotato di 512k di Flash e 128k di RAM (più 12k di auxiliary FLASH).

La board dispone di connettività Ethernet e USB, in modo da poter fungere da piattaforma dimostrativa per questa tipologia di applicazioni.

La connessione USB è sia di tipo device (per poter supportare classi quali la HID, la CDC e la MSB, oltre a poter essere utilizzata come bus per il Bootloading), sia di tipo Host.

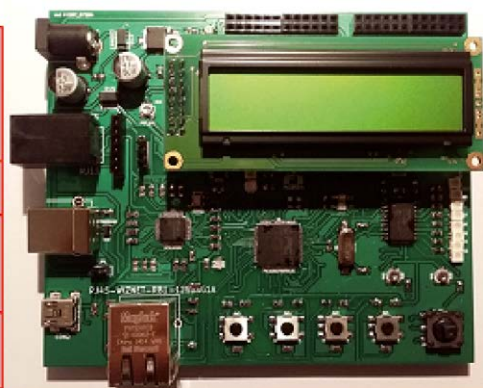
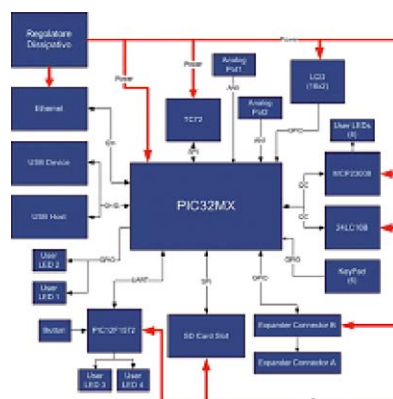
Le tensioni stabilizzate di alimentazione della scheda sono 5V e 3,3V.

Dallo schema a blocchi qui accanto, si nota facilmente che la MCU PIC32MX ha le seguenti connessioni principali:

- termometro a stato solido TC72 e slot per memoria SD mediante protocollo SPI (Serial Peripheral Interface);
- Port Expander MCP23008 (collegato a sua volta a 8 LED) e a una memoria EEPROM i2c 24LC16B mediante una piccola rete I²C, grazie alla quale è possibile sviluppare e verificare applicazioni I²C;
- microcontrollore PIC12F1572, attraverso la por-

ta UART, che a sua volta è collegato a un pulsante e due LED (gestibili anche mediante l'utilizzo di PWM);

- due potenziometri collegati ad altrettanti ingressi analogici;
- due user LED generici (User1 e User2), collegati direttamente a due pin del microcontrollore principale, utilizzabili anche mediante appositi PWM;
- display da 16x2 segmenti per permettere la visualizzazione di alcune informazioni d'interesse.



Schema a Blocchi e scheda assemblata della DemoboardPIC32.

La scheda prevede due connessioni per la programmazione dei microcontrollori, un header a sei poli per la programmazione mediante PicKit e un connettore RJ11 per la programmazione mediante ICD. In aggiunta, mediante uno switch sulla scheda è possibile selezionare quale dei due microcontrollori programmare.

Completa la dotazione della demoboard, un header di espansione a quaranta contatti su due file (20x2).

Listato 1 – Configurazione della periferica OC2

```
OC2CON = 0x0000;           // Turn off the OC2 when performing the setup
OC2R = 0x0064;             // Initialize primary Compare register
OC2RS = 0x0064;            // Initialize secondary Compare register
OC2CON = 0x0006;           // Configure for PWM mode without Fault pin enabled
OC2CONSET = 0x8000;        // Enable OC2
```

gli Interrupt, tralasciamo questo passaggio in quanto le loro impostazioni non cambiano. Il primo passo da compiere consiste nel verificare gli ingressi e le uscite d'interesse sul microcontrollore della DemoboardPIC32 dove sono collegati; nel nostro caso, volendo pilotare con un segnale PWM il LED (identificato sulla scheda come D1), dobbiamo abilitare e configurare il modulo Output Compare corrispondente. Guardando sul datasheet del microcontrollore su quale pin è collegato D1, si nota che la periferica OC corrispondente è la OC2, quindi passiamo a configurarla come segue nel **Listato 1**.

È buona norma spegnere sempre le periferiche prima di configurarle, cosa che attuiamo con la prima istruzione. Successivamente, come visto nella trattazione del modulo dobbiamo inizializzare i registri OC2R e OC2RS per la comparazione dei valori e la generazione del PWM.

Con la penultima istruzione configuriamo la periferica a lavorare come generatore di PWM mediante il registro OC2CON e infine la riattiviamo per poterla utilizzare.

L'ultimo passo da compiere per poter avere in uscita il PWM desiderato è legato all'impostazione del registro OC2RS. Volendo ottenere un segnale variabile e pilotabile mediante potenziometro, non dobbiamo far altro che acquisire il valore del potenziometro

mediante ADC e collegarlo a tele registro.

Ancora una volta bisogna verificare mediante datasheet del microcontrollore su quale pin è collegato il potenziometro d'interesse e configurarlo come ADC. Nel nostro caso vogliamo utilizzare il potenziometro che sulla DemoboardPIC32 è indicato come RV1, perciò configuriamo la periferica come riportato nell'esempio associato a questo corso e leggiamo il valore acquisito come riportato nel **Listato 2**.

Una volta memorizzato il valore del potenziometro nella variabile "pot", non ci resta che associarla al registro OC2RS per completare questo quarto progetto pratico e avere un LED pilotato (mediante PWM variabile) con un potenziometro.

CONCLUSIONI

In questa terza puntata abbiamo analizzato e compreso la teoria che c'è dietro ad alcune periferiche di base dei microcontrollori PIC32 come i convertitori analogico-digitale, Input Capture e Output Compare, per poi passare a utilizzarli nella pratica mediante MPLab X e la DemoboardPIC32.

Infine, con tutti questi strumenti abbiamo realizzato il "Dimming di un LED". Nella prossima puntata continueremo il nostro viaggio analizzando altre periferiche dei PIC32, come SPI, I²C e USART. ■

Listato 2 – Lettura e acquisizione valore del potenziometro

```
while ( !mAD1GetIntFlag() )
{
    // Wait for the first conversion to complete so there will be valid data in ADC _
    // result registers
}
// Read the result of pot conversion from the idle buffer
pot = ReadADC10(offset);
mAD1ClearIntFlag();
}
```