

Mensile di progettazione elettronica, attualità scientifica, novità tecnologiche.

Elettronica In

oltre l'elettronica

www.elettronica.in.it

€ 6,00

Anno XX - n. 191
Dicembre 2014
Gennaio 2015

Stampa in 3D
e realizza le tue
lampade natalizie RGB



Robot Spazzaneve

- Breakout board per sensori di temperatura umidità e pressione
- Smart scanner 3D con Kinect
- Scaldasiringhe per 3DRAG Chocolate
- SigFox, la tecnologia wireless long-range per Internet of Thing
- Impariamo a usare la board RandA
- Con Scratch tutti programmatori
- Al via i Corsi Futura Academy





TECNOLOGIE LONG-RANGE PER IoT: **SIGFOX**

di ARSENIO SPADONI

L'Internet of Things diventa a poco a poco realtà: alla scoperta di una tecnologia wireless long-range particolarmente indicata per realizzare semplici ed economiche reti wireless in grado di connettere tra loro miliardi di oggetti intelligenti. E che sta per arrivare anche in Italia.

A seconda delle previsioni, entro il 2020 il numero di dispositivi connessi in Rete sarà compreso tra 20 e 65 miliardi: la maggior parte saranno oggetti in grado di interagire con l'ambiente circostante che scambieranno dati tra di loro e con un'infrastruttura informatica al fine di migliorare i processi produttivi, la

produzione e distribuzione dell'energia e la logistica: fornire connettività e intelligenza agli oggetti consente di ottimizzare le risorse disponibili rendendo la nostra esistenza più piacevole e meno faticosa. È il nuovo paradigma di questo decennio, comunemente noto come Internet of Things, che dispiegherà i propri ef-

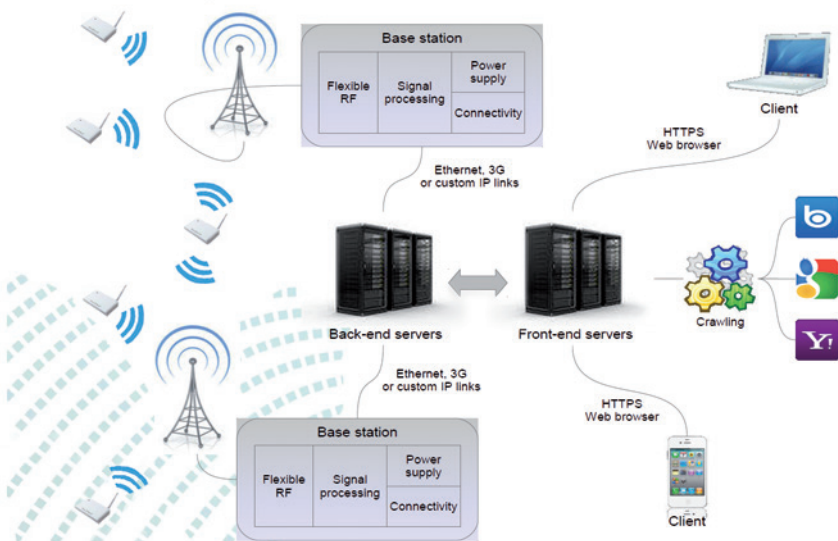


fetti nei prossimi 5-10 anni con implicazioni ancora tutte da scoprire, sicuramente con alcuni aspetti negativi ma con grandi opportunità per le persone e le aziende che le sapranno cogliere. Sicuramente, tanto per fare un esempio, il lettrista del contatore del gas sparirà (e questo rappresenta nell'immediato un fatto negativo) ma nel suo

complesso l'efficienza di sistema migliorerà creando nuove opportunità di lavoro. Quando ci si occupa di Internet of Things, il pensiero corre subito alle reti wireless (2G/3G e ora anche LTE/4G) che tutti noi siamo abituati ad utilizzare per le normali telefonate, per l'invio di SMS e, sempre di più, per la connessione



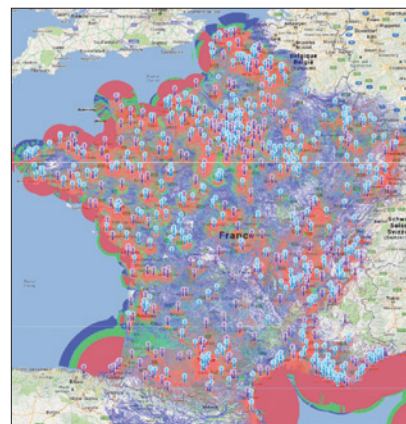
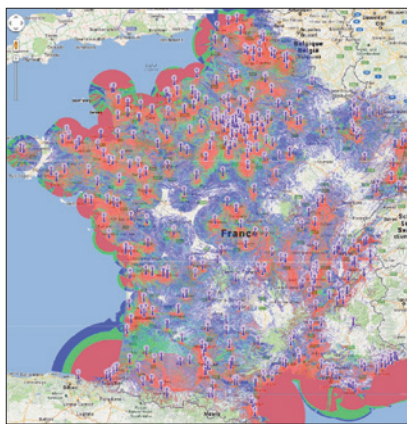
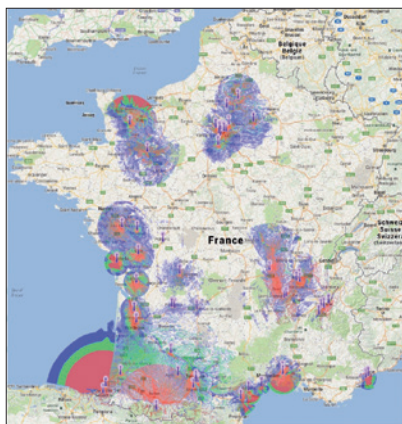
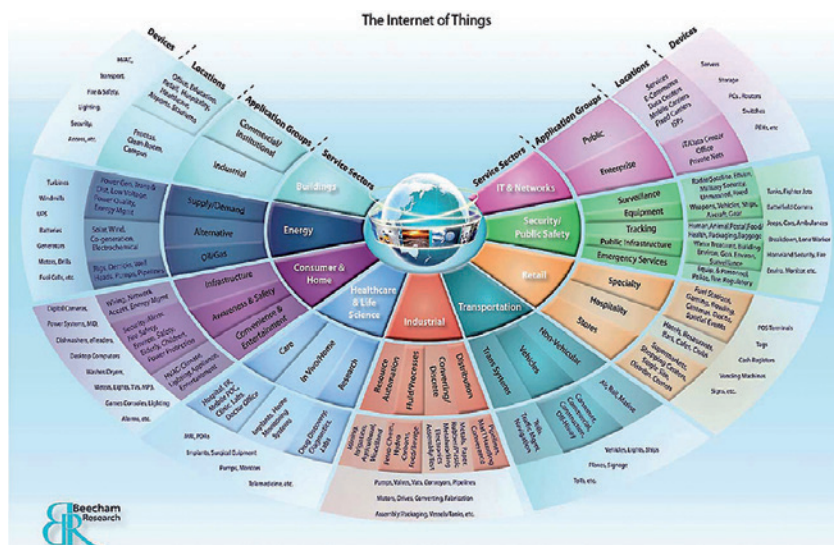
An example : SIGFOX network



I dati provenienti dalla rete SIGFOX vengono resi disponibili all'utente via Internet (mediante PC, smartphone o tablet) secondo il seguente schema.

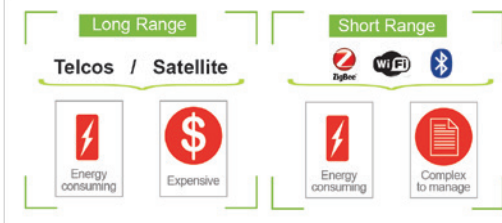
connettere deve consumare pochissimo in quanto alimentato a batteria, deve trasmettere poche informazioni, e magari deve avere un costo molto basso. Pensiamo, ad esempio, ad un rilevatore di fumo per impianti antincendio o ad un sensore di temperatura e umidità di una serra; il rilevatore di fumo deve inviare un paio di messaggi al giorno per dire che è "in vita" e, nel caso di incendio, un segnale d'allarme. Per oggetti di questo tipo, che rappresentano una grande fetta delle "Cose" da mettere in rete, è sicuramente più adatta una rete realizzata ad hoc, facilmente scalabile e altrettanto facilmente implementabile: e

ai servizi Internet, dalla posta elettronica alla navigazione, all'archiviazione di documenti. Il tutto tramite terminali intelligenti, dagli smartphone ai tablet, ai PC dotati di connessione wireless. Sicuramente la tecnologia cellulare è indispensabile e molto utile in molteplici applicazioni in ambito IoT e M2M, (pensiamo, ad esempio, al fleet-management); esistono, tuttavia, altrettante applicazioni in cui questa tecnologia non è appropriata, specialmente quando l'oggetto da



La copertura della Francia tramite la rete SIGFOX è stata realizzata in un solo anno: da sinistra a destra la copertura a inizio 2013, ad agosto e alla fine del 2013.

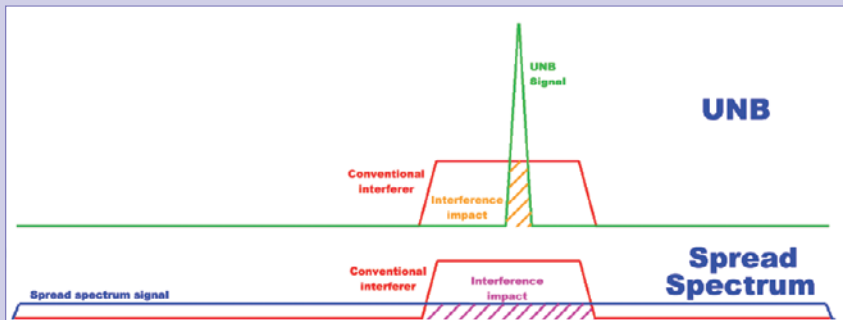
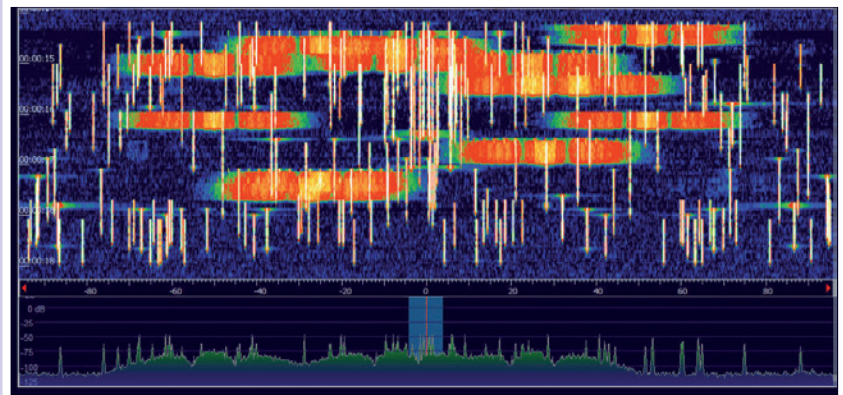
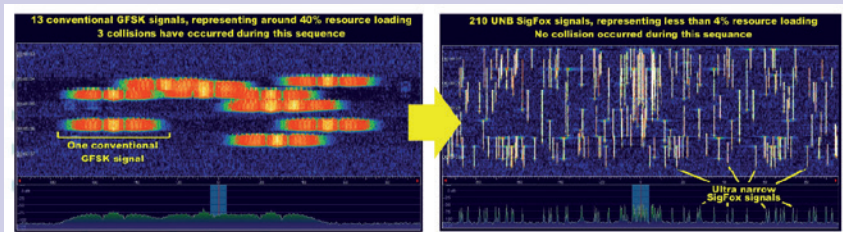
Le tecnologie **Ultra Narrow Band** e **Spread Spectrum**



sicuramente di gran lunga più economica di una rete GSM. Tuttavia, per trasferire nel mondo reale questa visione è necessario che la portata della connessione radio sia dell'ordine di 3-30 chilometri: solo così la rete dedicata potrà essere realizzata in pochissimo tempo e con risorse particolarmente contenute.

Già, ma come si fa a garantire portate di questo genere (long-range) con consumi ridotti e potenze di trasmissione altrettanto contenute, magari così basse da consentire l'utilizzo di quelle frequenze che non necessitano di alcuna concessione come, ad esempio, gli 868 MHz in Europa e i 915 MHz negli Stati Uniti?

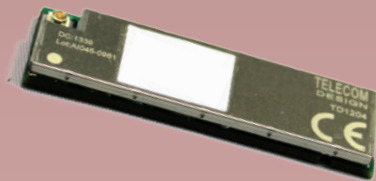
Le ricerche in questo settore hanno portato negli ultimi 3-4 anni a risultati sbalorditivi che consentono ad un singolo (e molto semplice) ricevitore di gestire più di un milione di trasmettitori entro un raggio d'azione che può andare da qualche chilometro (nelle aree densamente urbanizzate) a 15-30 chilometri nelle aree rurali. Pionieri in questo campo sono state le società Semtech (www.semtech.com) e SigFox (www.sigfox.com) che hanno messo a punto sistemi wireless low-cost in grado di raggiungere questi incredibili risultati. La francese SigFox (una start-up nata nel 2010) si è impegnata principalmente nella creazione di reti long-range che già coprono intere nazioni, lasciando a società partner la produzione dei dispositivi (chip e moduli) da utilizzare per realizzare i terminali; recentemente, ai produttori



Per aumentare la portata di un sistema radio si possono seguire due strade: aumentare la potenza del trasmettitore o migliorare la sensibilità del ricevitore. Nelle applicazioni IoT non è possibile aumentare la potenza del TX che, al contrario, deve essere la più bassa possibile per ridurre al minimo i consumi energetici. Da questo punto di vista potenze dell'ordine di 1-25 mW sulle frequenze sub-1 GHz sono accettabili. Con queste potenze e con una sensibilità standard del ricevitore dell'ordine di -110 dBm (garantita dalla tecnologia superrigenerativa o supereterodina) – la portata, in ogni caso, non supera i 30-300 metri. La sensibilità di un ricevitore è fortemente influenzata dal rumore intrinseco del dispositivo e da quello presente nell'etere. Per incrementare in maniera significativa la sensibilità (e quindi la portata del sistema), negli ultimi anni sono stati fatti notevoli passi in avanti grazie all'impiego di due particolari tecnologie: quella denominata spread spectrum (utilizzata nei dispositivi Lo-Ra di Semtech) e quella Ultra Narrow Band utilizzata da

SigFox. Nel caso dell'UNB, il principale problema da risolvere risiede nella stabilità di frequenza dell'apparato mentre lo svantaggio è la banda passante particolarmente bassa (nel caso della tecnologia SigFox i dati vengono trasmessi a 100 bit/secondo). Tuttavia, nel caso di applicazioni IoT, i messaggi sono molto brevi per cui, anche con questa velocità, possono essere inviati in un paio di secondi. L'altro grande vantaggio risiede nella possibilità di gestire da parte di un singolo ricevitore un numero incredibilmente alto di trasmettitori come dimostrano i grafici nei quali si vedono le emissioni di pochi TX convenzionali (con già alcune collisioni) paragonate con le emissioni di 210 TX UNB senza alcuna collisione. Da questo grafico si ricava che un ricevitore SigFox con tecnologia UNB integrata con Cognitive SDR (oltre a garantire una sensibilità di ben -126 dBm a cui corrisponde una portata di circa 1-10 km) è in grado di gestire fino a 3 milioni di dispositivi nel caso ognuno di questi invii 3 messaggi giornalieri occupando appena l'8% dello spettro del canale utilizzato.

Device SigFox

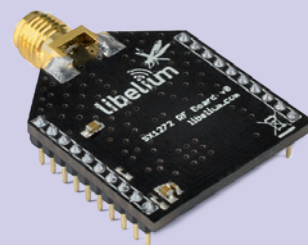


Fin dalla sua nascita, la rete ed il protocollo SigFox sono stati supportati dai dispositivi prodotti dalla società francese Telecom Design che attualmente ha a catalogo i moduli TD1204, TD1205 e TD1208, gateway e ricetrasmittitori operanti sulla frequenza europea ISM 868 MHz. A questi prodotti si sono aggiunti i dispositivi Telit (LE51-868S, un gateway SigFox a 868 MHz) e il ricetrasmittitore Silicon Labs Si446x in grado di operare tra 142 e 1050 MHz e quindi di funzionare sia sulla frequenza europea ISM 868 MHz che su quella americana di 915 MHz. A queste frequenze, questo compatto chip presenta una sensibilità di -127 dBm ed una potenza massima di +16/+20 dBm (50-100 mW). A fianco di SigFox è recentemente scesa in campo Atmel che proprio durante Electronica 2014 a Monaco ha annunciato e presentato il SoC ATA8520 che ha ottenuto di recente l'omologazione SigFox. Il chip è in grado di operare tra 315 e 915 MHz e può essere controllato tramite interfaccia SPI. Il dispositivo - che contiene lo stack SigFox e il relativo identificativo univoco a 32 bit, oltre al protocollo di sicurezza AES - è in grado di assicurare una portata compresa tra 15 e 30 km, grazie all'elevata sensibilità del ricevitore e alla potenza di uscita di +14,5 dBm.

di chip già attivi in questo settore, si è aggiunta Atmel che, in occasione di Electronica 2014 ha presentato il primo SoC certificato SigFox, un dispositivo low-cost dalle dimensioni particolarmente contenute che sicuramente darà un notevole impulso a questa tecnologia e alle reti in fase di realizzazione.

Per aumentare la portata di un sistema radio è possibile agire sulla potenza del trasmettitore, sulla sensibilità del ricevitore o su entrambi questi aspetti. Nel caso delle applicazioni per IoT la potenza emessa non può superare i 10-25 mW per rientrare nelle norme che regolano l'utilizzo delle frequenze ISM ma soprattutto per contenere i consumi. È anche vero che, per tornare all'esempio del rilevatore di fumo, il trasmettitore rimane in funzione per pochi secondi al giorno dopodiché va in modalità sleep assorbendo non più di qualche nanoampere, ma dobbiamo considerare che in molti casi la batteria deve garantire un'autonomia di 10-20 anni. Per questo motivo non è possibile andare oltre i valori di potenza citati. A questo proposito c'è da osservare anche che sulle frequenze ISM non è possibile trasmettere in continuazione ma con un "active time" limitato; questo è anche uno dei motivi per il quale SigFox ha scelto un massimo di 140 messaggi al giorno per

La tecnologia Lo-Ra (Long Range), sviluppata dalla società Semtech (www.semtech.com), offre sostanzialmente le stesse prestazioni della tecnologia SigFox essendo in grado di realizzare una connessione wireless su lunga distanza (15 km in ambiente rurale, 3 km in ambienti urbani ad alta densità) per l'invio di dati a bassa velocità. Tuttavia, al contrario di SigFox, che oltre ad aver messo a punto il protocollo e realizzato dei network che coprono già alcune nazioni, Semtech è un produttore di silicio e non un fornitore di servizi. Pertanto le reti IoT con tecnologia Semtech verranno realizzate da provider indipendenti; attualmente le società che più si sono date da fare da questo punto di vista sono Kerlink, IMST, Multitech, Cisco e Link-Lab. Per quanto riguarda, invece, la gestione dell'infrastruttura di rete, recentemente si è mossa IBM col suo centro ricerche di Zurigo che ha messo a punto il sistema LRSC (IBM Long-Range Signaling And Control) ed ha creato una serie di network-demo. L'infrastruttura IT creata da IBM è molto efficiente in termini di dispositivi supportati: è scalabile da pochi a un gran numero di dispositivi ed è ottimizzata per richiedere la minor potenza possibile consentita dall'applicazione con le comunicazioni protette da un motore crittografico. Per quanto riguarda i terminali, Semtech fornisce chip e IP a numerose aziende che



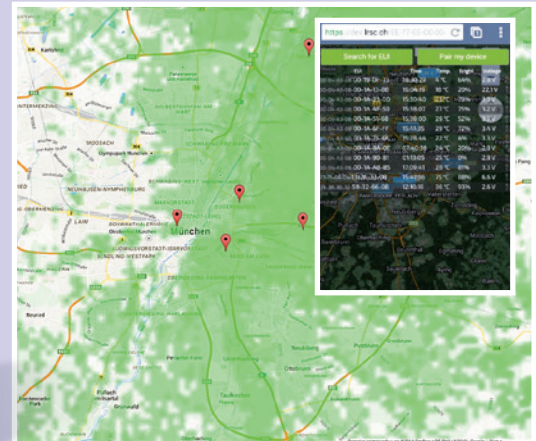
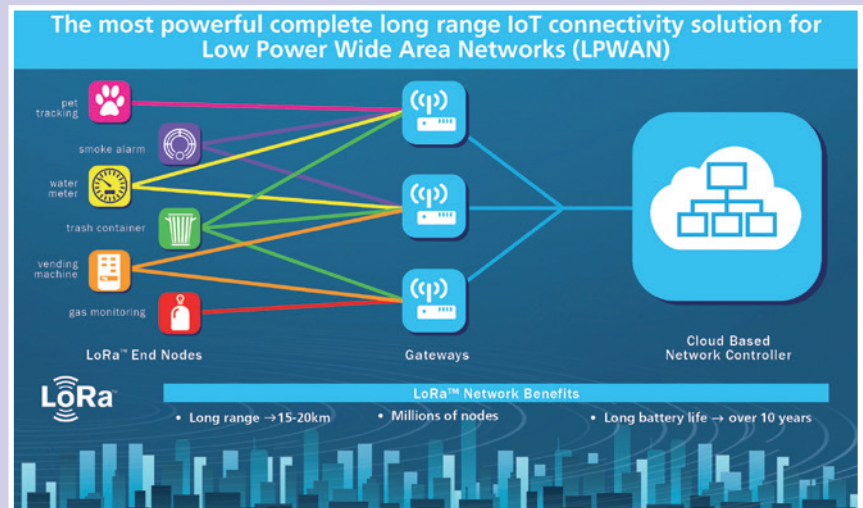
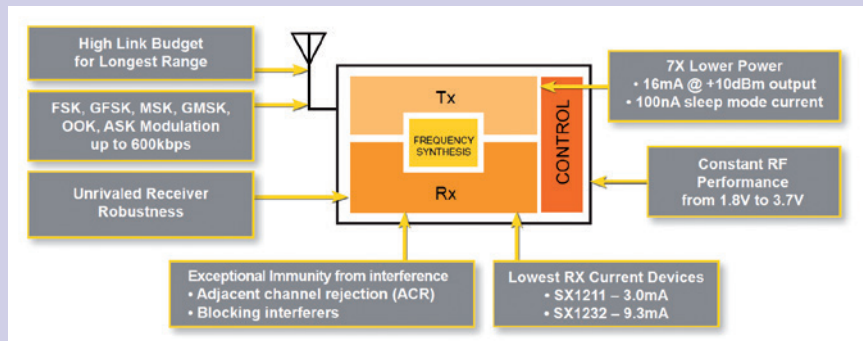
oggetto. Tornando al sistema radio, non resta dunque che agire sulla sensibilità del ricevitore e per fare ciò è possibile seguire strade differenti: SigFox ha scelto la tecnologia Ultra Narrow Band, Semtech quella dello Spread Spectrum. Entrambe le tecnologie hanno lo scopo di ridurre il rumore (prodotto localmente e/o

stanno sviluppando moduli e applicazioni. Durante la recente edizione di Electronica 2014 a Monaco sono state numerose le aziende che hanno presentato prodotti basati su questa tecnologia, alcune per realizzare dispositivi per connessioni punto-punto come l'italiana Aurel, altre per realizzare moduli da integrare in reti più o meno estese (Libelium, Microchip, ecc.). Sempre a Monaco, durante i quattro giorni della manifestazione, Microchip, IBM e Semtech hanno attivato una struttura di test per il nuovo modulo con tecnologia Lo-Ra che Microchip si appresta a lanciare sul mercato tra qualche mese.

Noi siamo stati tra i beta-tester del nuovo modulo Microchip alimentato a batteria: durante i giorni della manifestazione abbiamo potuto verificare come, nonostante la bassa potenza RF, l'alimentazione a batteria e il bassissimo consumo, il sistema fosse in grado di rilevare correttamente il segnale a parecchi chilometri di distanza in ambiente urbano ricevendo correttamente il dato di temperatura (il dato inviato era quello fornito dal sensore di temperatura della board).

Tra i chip prodotti da Semtech, il più utilizzato è il ricetrasmittitore SX1272 che è in grado di operare tra 862 e 1020 MHz (e quindi può essere utilizzato per entrambe le bande libere, quella europea e quella americana); il dispositivo utilizza la tecnologia spread spectrum con un algoritmo proprietario che è in grado di garantire una sensibilità (col più basso data-rate di 0,3 kHz) di ben -138 dBm mentre il trasmettitore può erogare al massimo +20 dBm (100 mW). L'espansione di spettro (che nel caso di questo chip può essere compresa tra 6 e 12 volte) si basa sulla relazione esistente tra la banda del segnale, il suo modulo e il suo contenuto energetico; lo scopo principale è quello di migliorare il rapporto segnale/rumore eliminando il maggior numero di interferenze e consentendo l'utilizzo contemporaneo della stessa gamma di frequenze a più utenti.

Utilizzando un algoritmo particolarmente valido (come quello Lo-Ra) è possibile raggiungere sensibilità che hanno dell'incredibile come, appunto, -138 dBm del chip SX1272.



captato dall'ambiente circostante) in modo da incrementare la sensibilità. Con queste tecnologie e con particolari algoritmi software, è stato possibile raggiungere sensibilità comprese tra -126 dBm e -138 dBm, performance legate anche alla bassissima velocità di trasmissione dei dati, compresa tra circa 100 e 300

bit al secondo. Per la maggior parte delle applicazioni questa bassa velocità non influisce sulle prestazioni della rete. In questo articolo ci occuperemo in maniera specifica della tecnologia e della rete SigFox per due motivi principali: la disponibilità di una copertura piuttosto ampia a livello europeo con i

primi gateway che stanno per essere installati anche in Italia e la disponibilità di chip low-cost (come quello di Atmel) con i quali realizzare le prime interessanti applicazioni.

One Network A billion dream
È questo lo slogan di SigFox, la giovane società francese che per

Il nuovo chip ATA8520



Derivato dalla famiglia di ricetrasmittitori sub-GHz di Atmel, il nuovo ATA8520 è espressamente studiato per funzionare con la rete SigFox in quanto ne implementa lo stack oltre all'ID univoco a 32 bit che identifica tutti i terminali collegati al network SigFox.

	Atmel ATA8520
Sigfox protocol handling	Device internal
TX output power (max)	14.5dBm
Power supply	1.9-3.6V
Current consumption - TX active	32.7mA
Current consumption - OFF mode	5nA
BOM	1xQ, 2xL, 7xC
Control I/F	SPI @ 500kHz
ESD protection (HBM)	4kV

prima ha creduto in una rete destinata specificamente agli oggetti. Una rete semplice, economica, scalabile e rapidamente implementabile. A pagina 24 vediamo la configurazione di questa rete con una serie di gateway radio in grado di colloquiare con gli oggetti (un milione e più per ciascun gateway) che si trovano nell'area di copertura radio. I dati ricevuti vengono inviati via GSM o rete fissa ad un sistema centrale di memorizzazione ed elaborazione e "distribuiti" ai Clienti mediante API di Internet in grado di automatizzare la gestione dei dispositivi e implementare l'integrazione dei dati.

L'infrastruttura è talmente semplice e poco costosa che, come si vede sempre a pagina 24, la copertura dell'intero territorio francese è stata realizzata in poco più di un anno. Utilizzando partner locali, SigFox ha completato la copertura anche in Olanda, Spagna, Regno Unito e parte della Russia e sta stringendo accordi con numerose altre società in grado di realizzare infrastrutture in tutta Europa ed anche in alcune zone degli Stati Uniti.

In ultima analisi, dunque, una rete dedicata destinata all'IoT deve presentare le seguenti caratteristiche:

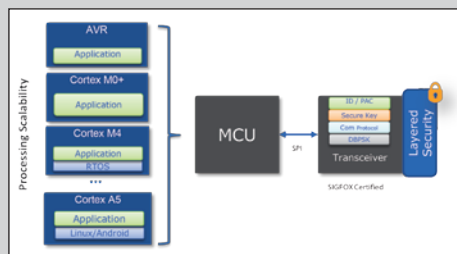
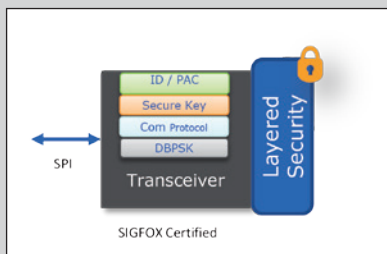
- Consumo di energia dei terminali molto basso
- Elevata portata della connessione radio per ridurre il numero di gateway
- Economica e facilmente integrabile e scalabile
- Sicura ed affidabile per evitare vulnerabilità

Tutti questi requisiti sono pienamente garantiti dai dispositivi e dall'infrastruttura SigFox. Per la copertura della Francia, ad esempio, sono bastati meno di mille gateway. La possibilità di avere a disposizione un'infrastruttura dedicata per dare connettività a qualsiasi oggetto fisico permette una moltitudine di opportunità, dalla ottimizzazione di processi esistenti fino alla creazione di business completamente nuovi. L'impiego di oggetti connessi non è certo una novità, ma la crescita del comparto sta accelerando rapidamente: si calcola che l'IoT genererà ricavi per 1200 miliardi entro il 2020 rispetto ai 200 miliardi di oggi. Gli oggetti collegati sono spesso molto semplici, isolati e funzionanti a batteria, con sensori che rilevano determinati eventi o informazio-

ni una decina o un centinaio di volte al giorno e li inviano ad un sistema informatico centralizzato. Le informazioni possono riguardare qualsiasi cosa, dal consumo di energia, alla temperatura, umidità, posizione, presenze, dati sanitari e tanto altro ancora. La sfida per i fornitori tradizionali di connettività di rete sta nella loro capacità di fornire soluzioni adeguate a questi prodotti che, spesso le tecnologie esistenti non possono fornire, sia per quanto riguarda i consumi energetici che per ciò che concerne i costi. Oggi giorno risulta evidente che le applicazioni legate all'Internet of Things hanno requisiti molto diversi rispetto ai cellulari ed agli smartphone; in quest'ultimo caso si procede verso la strada di una maggior larghezza di banda ed una maggior capacità di elaborazione a scapito della durata della batteria e di costi che risultano in ogni caso elevati, improponibili per infrastrutture con miliardi di oggetti connessi.

L'altra possibilità, quella della connettività satellitare, presenta gli stessi problemi con costi elevati e consumi energetici incompatibili con lo sviluppo di diffuse reti low-cost. Anche le soluzioni di connettività a corto raggio (Wi-Fi, ZigBee, ecc.) presentano un elevato consumo energetico e sono molto complesse da gestire.





Le risorse interne del dispositivo ATA8520 non sono accessibili all'utente in quanto ID/PAC, Secure Key, protocollo e modulazione DBPSK sono programmate in fabbrica. Il sistema è scalabile e può essere controllato mediante differenti processori in funzione anche delle prestazioni che si vogliono ottenere.

La connettività Wi-Fi, per esempio, richiede la configurazione di ogni oggetto mentre infrastrutture come ZigBee richiedono un elevato numero di concentratori, con conseguente complessità di installazione e manutenzione e persino maggior consumo energetico. In conclusione, dunque, si torna a quelle che sono le caratteristiche della rete SigFox e più in generale dei sistemi "low power long range" o LPWA (Low Power Wide Area) che risultano i più adatti ad un rapido sviluppo di infrastrutture IoT. Tecnicamente anche un sistema cellulare GSM/GPRS/LTE potrebbe essere definito long range, ma sicuramente non è low power. Senza considerare il fatto che al contrario dei sistemi GSM e satellitari, i dispo-

sistivi LPWA operano su frequenze radio libere e non soggette a concessione governativa. Il rapido dispiegamento di soluzioni destinate all'Internet of Things richiede dunque infrastrutture dedicate con elevata portata ma con consumi ridotti, dell'ordine (in trasmissione) di pochi milliwatt per garantire molti anni di funzionamento senza manutenzione di alcun tipo. Si calcola che un terminale SigFox con chip di ultima generazione alimentato con due batterie a stilo da 2.700 mA sia in grado di funzionare per circa 20 anni inviando 140 messaggi al giorno. Dal punto di vista tecnico la rete SigFox prevede l'invio di un massimo di 140 messaggi per oggetto connesso al giorno.

Ciascun messaggio è formato da un payload di 12 byte; nel caso di dati più lunghi, gli stessi vengono suddivisi in più messaggi in modo da rispettare i 12 byte previsti. Ciascun dispositivo è identificato da un ID di 32 bit e da PAC utilizzabile una sola volta per registrare l'ID SigFox. I nuovi dispositivi Atmel, ed in particolare il chip ATA8520, integra un TX a basso consumo sia in trasmissione (appena 32,7mA) che in modalità OFF (appena 5 nA); il chip può funzionare con una tensione compresa tra 1,9 e 3,2 V ed è in grado di erogare una potenza RF di +14,5 dBm. Inoltre il dispositivo integra lo stack SigFox, l'ID, il PAC ed è predisposto per un sistema crittografico AES in grado di garantire la massima sicurezza da questo punto di vista. La comunicazione avviene attraverso una porta SPI e il chip richiede circa 10 componenti esterni per funzionare. Nella classica configurazione questo dispositivo viene controllato da un micro AVR ad 8 bit (ATmega328P). Sono questi i componenti che utilizzeremo nei prossimi progetti IoT long-range che abbiamo in fase di progettazione. Stay Tuned!

Elettronica In

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

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

COLLABORATORI HARDWARE della nostra Rivista

Realizzare un progetto per Elettronica In è molto semplice!

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

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

Non perdere questa opportunità!
Contattaci oggi stesso!



ALBERO DI NATALE "CAMALEONTE"

di MARCO MAGAGNIN

Spesso, in redazione nascono discussioni "spontanee" sul modo di affrontare i progetti che vengono poi presentati sulla rivista; uno dei temi principali è quale approccio utilizzare in un progetto: orientarsi verso un circuito a componenti discreti, o usare uno o più integrati, oppure affidarsi a un microcontrollore e, nel caso, quale. Chiaramente le soluzioni possono essere più di una e sono spesso legate alla formazione e all'esperienza dei singoli progettisti. E questi sono anche i

motivi per i quali le discussioni diventano "accese". La situazione descritta si è puntualmente presentata durante la decisione su cosa presentare come "progetto di Natale": alla fine si è concordato di proporre una lampada a LED in grado di assumere il colore della superficie sulla quale viene appoggiata e da inserire in un albero di Natale cavo. A questo punto si è posta l'inevitabile serie di domande: quale lampada, come riconoscere i colori, come riprodurli?

Due differenti approcci per realizzare un gadget natalizio che prende il colore del piano cui si appoggia.

È l'occasione per vedere la differenza tra lo sviluppo di un progetto "amatoriale" e di un prodotto "commerciale".



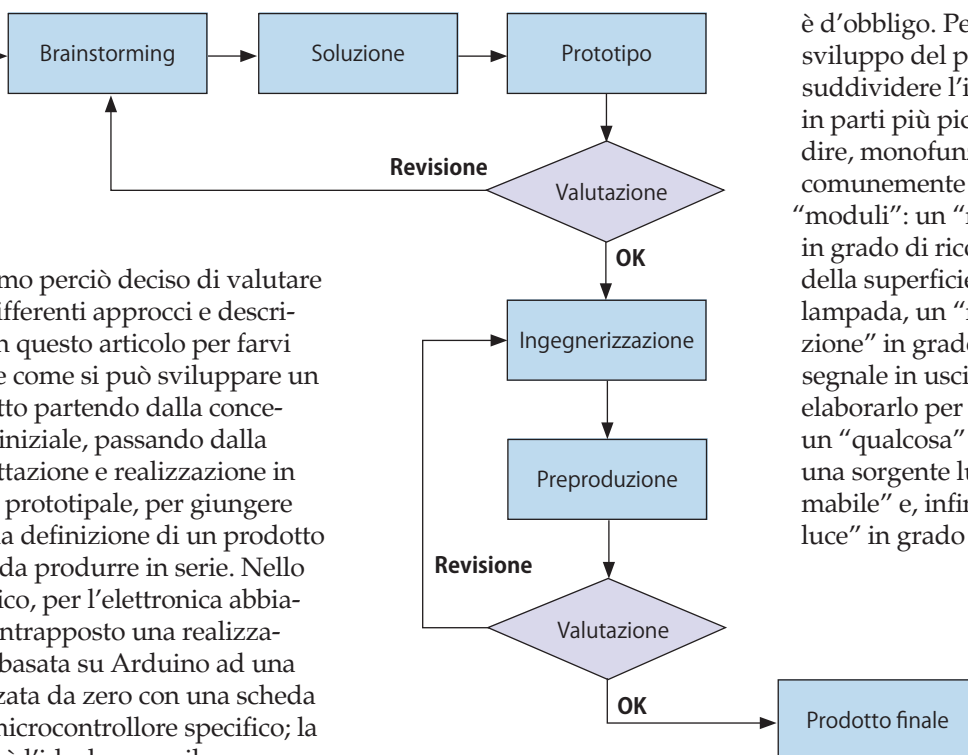
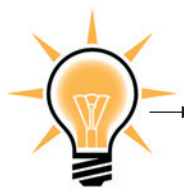


Fig. 1

Abbiamo perciò deciso di valutare due differenti approcci e descriverli in questo articolo per farvi vedere come si può sviluppare un progetto partendo dalla concezione iniziale, passando dalla progettazione e realizzazione in forma prototipale, per giungere poi alla definizione di un prodotto finale da produrre in serie. Nello specifico, per l'elettronica abbiamo contrapposto una realizzazione basata su Arduino ad una realizzata da zero con una scheda e un microcontrollore specifico; la prima è l'ideale per sviluppare un prototipo, ma per costo e ingombri è inadatta a una produzione di serie, dove invece scegliere l'hardware fornisce la soluzione migliore.

L'APPROCCIO AL PROGETTO

Lo schema di massima del processo di progettazione e ingegnerizzazione di un prodotto è mostrato in Fig. 1: si parte dall'ipotesi iniziale, nel nostro caso la volontà di realizzare la nostra lampada di Natale in grado di emettere luce dello stesso colore della superficie sulla quale viene appoggiata. Il primo passo che porta da una semplice idea ad un'ipotesi di progetto è la definizione delle funzionalità, la scelta dei componenti e, in questo caso, anche un'ipotesi di design.

Il processo prevede una prima fase dedicata alla scelta dei componenti, per quanto riguarda la realizzazione elettronica, e alla definizione dell'aspetto estetico, per quel che concerne il corpo esterno della lampada. Sulla base

delle prime scelte si realizza un primo prototipo da valutare e migliorare fino a giungere ad una versione adatta ad essere replicata in produzione.

La decisione (o, meglio, l'obbligo, detto anche "vincolo di progetto") è stata di realizzare la lampada con il processo di stampa 3D, in modo da ottenere un oggetto molto simile per aspetto e consistenza a quello che sarebbe risultato il prodotto definitivo.

La forma iniziale sarà quella di un albero a tre dimensioni che ricordi quello di Natale.

Ora pensiamo alla funzionalità della lampada, che consiste nel rilevare il colore della superficie sottostante la base e la sua riproduzione come tonalità di luce. Già questo introduce un ulteriore vincolo: il corpo della lampada deve essere di materiale traslucido e di un colore che non interferisca con la tonalità di luce emessa, in modo da non falsare i colori. In pratica il bianco

è d'obbligo. Per semplificare lo sviluppo del progetto ci conviene suddividere l'intero problema in parti più piccole e, per così dire, monofunzione, quelli che comunemente vengono chiamati "moduli": un "modulo sensore" in grado di riconoscere il colore della superficie di appoggio della lampada, un "modulo di elaborazione" in grado di comprendere il segnale in uscita dal "sensore" ed elaborarlo per produrre in uscita un "qualcosa" in grado di pilotare una sorgente luminosa "programmabile" e, infine, un "modulo luce" in grado di riprodurre una

tonalità luminosa a fronte di un input predefinito.

Definite le funzionalità dei singoli "moduli" o "stadi" come si diceva una volta, per ciascuno è necessario decidere quale tecnologia adottare e la conseguente "ingegnerizzazione". Per il modulo di riconoscimento del colore si presentano diverse alternative: possiamo utilizzare una telecamera e poi elaborare le immagini acquisite per estrapolare il colore predominante, ma tale soluzione richiede un hardware con una potenza di elaborazione abbastanza significativa, qualcosa come ciò che abbiamo presentato nell'articolo "Computer Vision con Raspberry Pi ed il modulo Camera Pi" nel fascicolo numero 186. Ma sarebbe come utilizzare un cannone per andare a caccia di zanzare.

Potremmo allora ipotizzare la costruzione di un sensore utilizzando componenti discreti, sensori di luce, e diodi multicolori

da interfacciare ad una logica di elaborazione e controllo: una soluzione già più adatta, ma ancora troppo complicata da realizzare e mettere a punto. Una terza scelta è di adottare un criterio che sta avendo sempre più seguito: andare alla ricerca, sul mercato, di moduli preconfezionati che svolgano le funzioni di cui abbiamo bisogno. Lo stesso criterio che adottiamo quando realizziamo i nostri progetti in GNU/Linux: ricerca della maggior quantità di "mattoni" disponibili nel "mondo" e la realizzazione delle sole integrazioni tra le diverse parti. Questa ricerca sarà maggiormente fruttuosa quanto più siamo stati accurati nella scomposizione del problema in moduli.

Cerca che ti cerca, abbiamo trovato il TCS230, un modulo con sensore di riconoscimento dei colori prodotto da LC Technology. Per riprodurre la tonalità di luce rilevata con il sensore, la scelta è molto semplice: un LED RGB andrà benissimo in fase di prototipo e ci apre la strada a diverse scelte in fase di prodotto finale, in quanto il pilotaggio RGB si applica ad una vasta gamma di prodotti di produzione di luce, reperibili sul mercato in una varietà di forme, dimensioni, potenze e prezzi. Ne consegue la definizione della funzione principale del "modulo di elaborazione": applicare ai piedini che pilotano ciascun colore un segnale PWM proporzionale alla intensità della corrispondente componente di colore rilevata dal sensore.

Ci resta da scegliere cosa usare come "modulo di elaborazione": per un oggetto monofunzione, con molta probabilità la scelta giusta è utilizzare un microcontrollore con un'adeguata dotazione di pin di ingresso/uscita sia analogici che digitali. Le prestazioni non sono molto importanti, in quanto

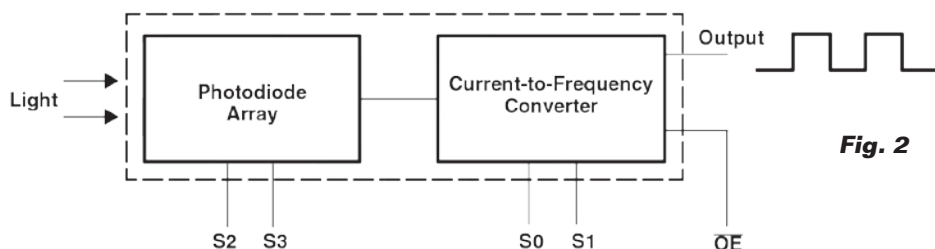


Fig. 2

una lampada è un oggetto statico: una volta appoggiata su una superficie se ne sta lì fino a quando qualcuno non la sposta, giusto per vedere se cambia colore. Per intenderci, la nostra applicazione non richiede una soluzione real time basata su FPGA con tempi di campionamento dell'ordine dei nanosecondi: un microcontrollore economico è più che sufficiente per le esigenze di questo progetto. Per un primo prototipo la scelta è facile: basta allungare la mano sulla scrivania e utilizzare l'onnipotente Arduino Uno. La sua vocazione è proprio quella di rendere semplice e veloce la realizzazione di prototipi ... funzionanti. L'IDE e il linguaggio di sviluppo sono stati messi a punto con il preciso obiettivo di essere accessibili a tutti. Abbiamo tutto per realizzare il nostro prototipo. Per il prodotto finale dovremo sostituire Arduino Uno con una soluzione più adatta ad una produzione in serie, così come probabilmente per il LED RGB, ma ce ne preoccupiamo a tempo debito. Arduino Uno è già stato oggetto di numerosi libri e articoli sulla rivista, quindi rimandiamo a questi per chi già non ne conosce a sufficienza l'utilizzo. Ci concentriamo invece sulle caratteristiche

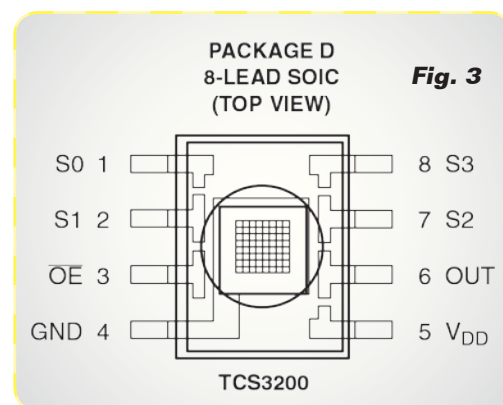


Fig. 3

del modulo TCS230 e sull'utilizzo dei LED RGB.

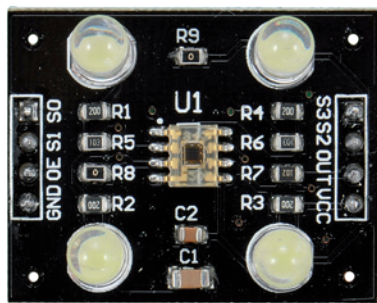
IL MODULO TCS230

Il modulo TCS230 è prodotto da LC Technology (<http://www.lctech-inc.com>), una società con sede in Cina, nella regione dello Shenzhen. Questo modulo non è altro che una breakout board con a bordo il sensore TCS230, un convertitore programmabile che permette di trasformare la luce in ingresso in una frequenza proporzionale all'intensità della luce percepita. Il circuito integrato in tecnologia CMOS comprende una matrice di 8x8 fotodiodi ed un convertitore corrente-frequenza secondo lo schema a blocchi visibile in Fig. 2. Nel sensore sono presenti tre filtri di colore R, G e

Tabella 1

TERMINAL NAME	NO.	I/O	DESCRIPTION
GND	4		Power supply ground. All voltages are referenced to GND.
OE	3	I	Enable for f_o (active low).
OUT	6	O	Output frequency (f_o).
S0, S1	1, 2	I	Output frequency scaling selection inputs.
S2, S3	7, 8	I	Photodiode type selection inputs.
VDD	5		Supply voltage

Fig. 4

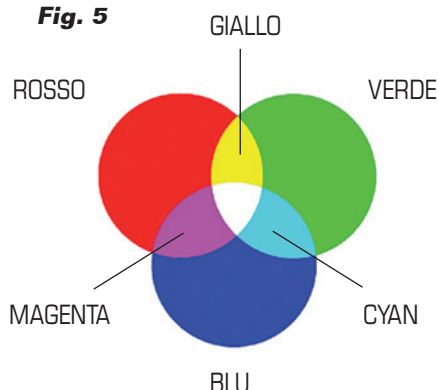


B (guarda caso) e la misura delle componenti di colore consiste in una prima misura dell'intensità della luce composta (senza filtri) e nel misurare successivamente le intensità relative a ciascun colore applicando i filtri uno alla volta. In **Fig. 3** è visibile la pin-out dell'integrato, mentre nella **Tabella 1** sono descritti i significati dei singoli pin. I pin del modulo TCS230 trovano corrispondenza uno a uno con i pin dell'integrato, come visibile nella **Fig. 4**. I diodi presenti sul modulo sono alimentati in parallelo e servono ad "illuminare" il campo di "visione" del sensore. Il modulo può essere alimentato con tensioni da 2,7V a 5,5V ed è quindi compatibile con i livelli dei microcontrollori alimentati sia a 3,3V che a 5V. Il

Tabella 2

S2	S3	PHOTODIODE TYPE
L	L	Red
L	H	Blue
H	L	Clear (no filter)
H	H	Green

Fig. 5



convertitore TCS230 legge i fotodiodi che compongono la matrice. In base alla configurazione dei pin di controllo S2 ed S3 (vedi **Tabella 2**) la matrice di diodi filtra la luce rossa, verde, blu o bianca ed emette in uscita sul pin OUT un segnale ad onda quadra con duty-cycle del 50% e frequenza direttamente proporzionale all'intensità della luce percepita (irradianza). Il funzionamento del sensore sfrutta il fatto che le differenti sfumature di colore possono essere ricondotte ad una combinazione dei tre colori fondamentali, rosso, verde e blu, in proporzioni appropriate (**Fig. 5**). Quando viene impostato il filtro per un determinato colore predominante, ad esempio il blu, l'integrato TCS230 misura la sola intensità del colore esaminato, escludendo gli altri; in questo modo è possibile determinare la componente blu del colore in esame. Stesso procedimento per determinare l'intensità delle componenti rossa e verde. L'ampiezza dello spettro di frequenza emessa può essere scalata dei fattori espressi in **Tabella 3** configurando opportunamente i livelli dei pin S0 ed S1, in modo da potersi adattare alle capacità di elaborazione del microcontrollore utilizzato. Gli ingressi e le uscite possono essere interfacciate ad un microcontrollore per la configurazione e la lettura dei valori di frequenza, cosa che noi faremo con il nostro Arduino Uno.

IL LED RGB

Per riprodurre la tonalità di colore

Tabella 3

S0	S1	OUTPUT FREQUENCY SCALING (f_o)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

ricepite dal sensore, utilizziamo un LED RGB, almeno nel nostro prototipo, ovvero un dispositivo optoelettronico composto da tre LED che emettono rispettivamente luce Rossa, Blu e Verde (**Fig. 6**). Esistono due tipologie di questi componenti: ad anodo comune e a catodo comune. Nel primo, tutti i LED hanno l'anodo, ovvero il terminale positivo in comune, mentre i tre LED vengono comandati singolarmente pilotando il catodo di ciascuno. Nei componenti a catodo comune, i tre LED condividono il terminale negativo, mentre il pilotaggio di ciascun singolo LED avviene utilizzando l'anodo, ovvero il terminale positivo. Una configurazione, questa, più adatta ad essere utilizzata con Arduino Uno. Infatti per pilotare i LED utilizzeremo tre uscite analogiche (DAC) di Arduino Uno, in grado di convertire le grandezze digitali tra 0 e 255 in valori di tensione proporzionali tra 0 e +5V. Ciascuna sfumatura di colore, nella rappresentazione RGB ad otto bit di precisione, è rappresentata da una terna di valori nell'intervallo 0-255 che rispecchia l'intensità di ciascuna componente di colore. Per esempio, il rosso intenso è rappresentato dalla configurazione (255, 0, 0). Sul web, all'indirizzo www.rapidtables.com/web/color/RGB_Color.htm potete trovare una tabella interattiva nella quale, inserendo i valori delle tre tonalità di colore base, vedrete a cosa corrispondono in RGB. Vale la pena sottolineare una particolarità relativa ai toni di grigio: chi si è occupato di elaborazione di immagini, sa che i toni di grigio sono espressi da terne di valori RGB uguali, per esempio (150, 150, 150). Non essendo possibile riprodurre, con il nostro LED, le tonalità di grigio, faremo in modo che in queste condizioni il LED

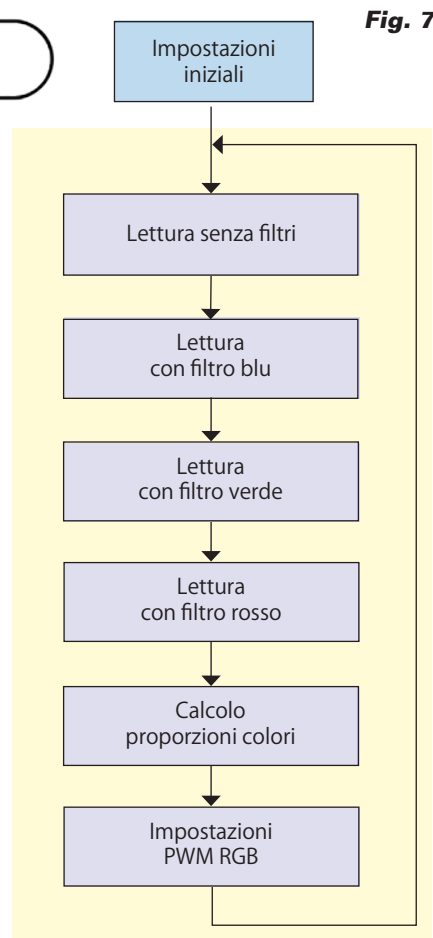
emetta luce bianca. Il LED RGB scelto per il nostro prototipo è siglato 540R2GBC-CC.

STRATEGIA DI ELABORAZIONE

La strategia di elaborazione per il microcontrollore del progetto è visibile nel diagramma a blocchi di **Fig. 7**: dopo le definizioni iniziali dell'ambiente di elaborazione, all'accensione del microcontrollore viene innescato il ciclo di elaborazione principale, che esegue all'infinito le operazioni descritte qui di seguito.

Come prima operazione, tentiamo di verificare se la lampada è effettivamente appoggiata ad una superficie, oppure la stiamo tenendo sollevata, eventualmente per spostarla da un luogo ad un altro; in questo caso sospendiamo la funzione di "mimetismo" saltando tutto il processo di elaborazione successivo. Se verificiamo che la lampada è appoggiata ad una superficie, dobbiamo operare sul sensore in modo da riconoscere la tonalità di colore della superficie di appoggio. Per ottenere questo risultato, eseguiamo una

serie di letture della frequenza di uscita, la quale, come abbiamo già visto, è proporzionale all'intensità della componente luminosa, modificando di volta in volta la configurazione dei filtri. La prima lettura viene eseguita in assenza di filtri, in modo da ottenere la luminosità totale riflessa dalla superficie. Eseguiamo le successive letture inserendo ogni volta un filtro diverso, rispettivamente il filtro per il colore rosso, il verde e il blu. Dopo ogni lettura siamo in grado di determinare quanto la componente del colore selezionato è presente nella tonalità complessiva, in termini percentuali. Al termine delle letture prendiamo il valore maggiore tra quelli letti e gli assegniamo il livello 255 per poi rapportare gli altri due in modo proporzionale, con alcuni aggiustamenti che abbiamo trovato sperimentalmente sulla base del comportamento del sensore, come descritto nel datasheet. La terna di valori ottenuti è utilizzata per configurare le tre uscite PWM

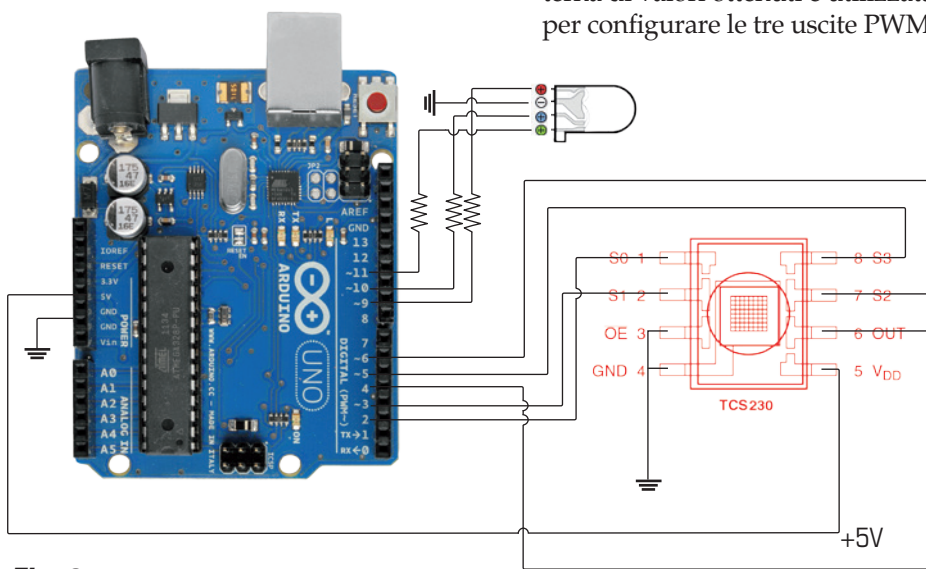
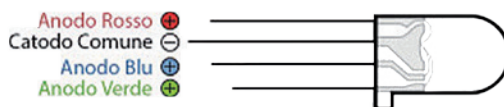


che pilotano i terminali positivi del LED RGB.

SCHEMA ELETTRICO DEL PROTOTIPO

Nello schema in **Fig. 8** possiamo vedere i collegamenti tra il sensore TCS230, la scheda Arduino Uno e il LED RGB.

L'assemblaggio, in puro stile "prototipo", è visibile in **Fig. 9**. I terminali S0 ed S1, che servono a configurare il range di frequenza della tensione emessa dall'uscita del sensore TCS230, sono collegati ai pin 2 e 3 di Arduino. Le uscite S2 ed S3 che permettono di definire se e/o quale filtro utilizzare nella lettura delle intensità luminose, sono collegati rispettivamente alle uscite digitali 6 e 5. Il terminale di uscita del sensore TCS230 è collegato all'ingresso 4



```

/*
Sketch lampada di Natale
*/
int S0 = 2;    // pin controllo frequenza di output
int S1 = 3;    // pin controllo frequenza di output
int S2 = 6;    // pin controllo filtri colore
int S3 = 5;    // pin controllo filtri colore
int taosOutPin = 4; // pin uscita frequenza
int LED = 13;  // pinD
int R = 9;     // PWM colore rosso
int G = 10;    // PWM colore verde
int B = 11;    // PWM colore Blu
double Coeff = 0;
int Ro = 0;    // p inE
int Go = 0;    // pinE
int Bo = 0;    // pinE

void setup() {
  TCS230setup();
  Serial.begin(115200);
  Serial.print("\n\n\nready\n\n\n");
  pinMode(R,OUTPUT); //S2 pinE
  pinMode(G,OUTPUT); //S2 pinE
  pinMode(B,OUTPUT); //S2 pinE
  delay(100);
}

void loop() {
  Serial.print(detectColor(taosOutPin)); // impostazione seriale per debug
  Serial.print("\n\n\n");
  delay(1000);
}

int detectColor(int taosOutPin){
  // controllo presenza superficie
  double isPresentTolerance = 5;
  double isPresent = colorRead(taosOutPin,0,0)/colorRead(taosOutPin,0,1);
  Serial.print("isPresent:");
  Serial.println(isPresent,2);
  Serial.print("valore isPresentTolerance:");
  Serial.println(isPresentTolerance,2);
  if(isPresent < isPresentTolerance){
    Serial.println("nessuna superficie rilevata");
    return 0;
  }

  double red,blue,green;
  // lettura intensita della luce (senza filtri)
  double white = colorRead(taosOutPin,0,1);
  // lettura intensita componente rossa
  red = white/colorRead(taosOutPin,1,1);
  // lettura intensita componente blu
  blue = white/colorRead(taosOutPin,2,1);
  // lettura intensita componente verde
  green = white/colorRead(taosOutPin,3,1);
  Serial.print("red");
  Serial.println(red);
  Serial.print("blue");
  Serial.println(blue);
  Serial.print("green");
  Serial.println(green);

  if(red > blue && red > green){
    Coeff = 255 / red;
    Ro = 255;
    Go = green * Coeff;
    Bo = blue * Coeff;
    LED_RGB(Ro, Go, Bo);
    return 1;
  }
  if(blue > green && blue > red){
    Coeff = 255 / blue;
    Bo = 255;
    Go = green * Coeff;
    Ro = red * Coeff;
    LED_RGB(Ro, Go, Bo);
    return 2;
  }
  if(green > blue && green > red){
    Coeff = 255 / green;
    Go = 255;
    Ro = red * Coeff;
    Bo = blue * Coeff;
    LED_RGB(Ro, Go, Bo);
    return 3;
  }
}

double colorRead(int taosOutPin, int color, boolean LEDstate){
  //make sure that the pin is set to input
  pinMode(taosOutPin, INPUT);

```

di Arduino. Infine, i tre terminali del LED RGB sono collegati alle uscite PWM di Arduino 9 (R), 10 (G) e 11 (B). Le uscite PWM sono quelle identificate con un “-” sulla serigrafia della scheda. Abbiamo usato le tre uscite citate, in quanto sono caratterizzate dallo stesso intervallo di frequenza del segnale PWM; se avessimo scelto altri pin, avremmo potuto riscontrare un funzionamento anomalo in quanto i pin 5 e 6 hanno frequenza di uscita di 976 Hz mentre gli altri pin (3, 9, 10 e 11) hanno frequenza di uscita di 488 Hz. Per evitare sorprese e anomalie di funzionamento difficilmente individuabili, quando si affronta un progetto è sempre opportuno leggere i data-sheet e la documentazione dei componenti coinvolti. Nel nostro caso, abbiamo collegato i terminali del LED RGB a tre uscite PWM caratterizzate dalla stessa frequenza. Le tre resistenze da 470 ohm limitano la corrente in ingresso ai terminali di uscita a circa 20 mA.

SKETCH PER ARDUINO UNO

Lo sketch nel Listato 1 permette di gestire la logica della nostra lampada, ovvero il processo di riconoscimento del colore della base di appoggio e la sua riproduzione come tonalità luminosa del LED RGB.

Lavorando a livello di prototipo con uno strumento semplice come

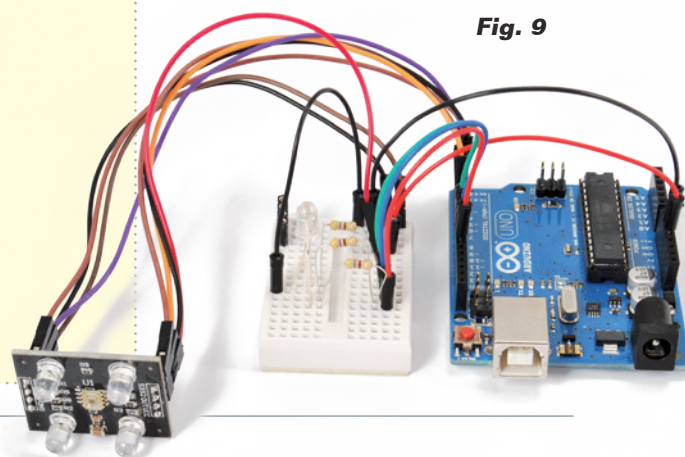


Fig. 9

Arduino, possiamo introdurre nel codice una serie di istruzioni utili ai fini del debug che poi elimineremo in fase di realizzazione del prodotto finale. Nel nostro caso utilizzeremo il Monitor Seriale per evidenziare i valori letti ed i risultati dei calcoli eseguiti. La costante *isPresentTolerance* permette di impostare un valore di soglia per verificare se la lampada è appoggiata o meno su una superficie. Abbiamo usato l'accorgimento di eseguire una lettura con i LED di illuminazione accesi ed una con i LED spenti; per accendere e spegnere i LED, si configurano opportunamente i pin S0 ed S1. Nel caso di lampada appoggiata su una superficie, avremo due valori molto diversi, mentre nel caso di lampada "sospesa" la luce ambiente raggiungerà comunque il sensore, fornendo due letture con valori più vicini.

Se la lampada è appoggiata ad una superficie, si misura l'intensità totale della luce riflessa e le intensità relative delle singole componenti di colore, applicando a ciascuna lettura i filtri opportuni mediante la configurazione dei pin S2 ed S3 del sensore TCS230. L'intensità relativa di ciascuna componente viene calcolata in rapporto all'intensità del colore complessivo, letto senza la presenza di alcun filtro. Per leggere la frequenza emessa in corrispondenza dell'applicazione di ciascun filtro viene utilizzata l'istruzione "pulseIn", che legge la lunghezza di un impulso su uno specifico pin. L'istruzione può essere configurata per leggere la durata dell'impulso nel livello LOW o nel livello HIGH: nel nostro caso, LOW. In questa evenienza l'istruzione attende che il livello del pin assuma il valore LOW, poi inizia la misurazione del tempo fino a quando il livello torna HIGH, interrompendo la misurazione. Il

```

taosMode(1);
int sensorDelay = 3;
//imposta i pin per selezionare i filtri
if(color == 0){// bianco (no filtri)
    digitalWrite(S3, LOW); //S3
    digitalWrite(S2, HIGH); //S2
    // Serial.print(" w");
}else if(color == 1){// attiva filtro rosso
    digitalWrite(S3, LOW); //S3
    digitalWrite(S2, LOW); //S2
    // Serial.print(" r");
}else if(color == 2){// attiva filtro blu
    digitalWrite(S3, HIGH); //S3
    digitalWrite(S2, LOW); //S2
    // Serial.print(" b");
}else if(color == 3){// attiva filtro verde
    digitalWrite(S3, HIGH); //S3
    digitalWrite(S2, HIGH); //S2
    // Serial.print(" g");
}
double readPulse;
if(LEDstate == 0){
    taosMode(0);
    // digitalWrite(LED, LOW);
}
if(LEDstate == 1){
    taosMode(1);
    // digitalWrite(LED, HIGH);
}
delay(sensorDelay);
readPulse = pulseIn(taosOutPin, LOW, 80000);
// gestione timeout della lettura impulso
if(readPulse < .1){
    readPulse = 80000;
}
// spegne i LED del sensore
taosMode(0);
return readPulse;
}
// gestione modalita emissione frequenza. TaosMode(0);
// zero sensore in quiete

void taosMode(int mode){
    if(mode == 0){
        //power OFF
        digitalWrite(LED, LOW);
        digitalWrite(S0, LOW); //S0
        digitalWrite(S1, LOW); //S1
        // Serial.println("mOFFm");
    }else if(mode == 1){
        // frequenza 1:1
        digitalWrite(S0, HIGH); //S0
        digitalWrite(S1, HIGH); //S1
        // Serial.println("m1:1m");
    }else if(mode == 2){
        // frequenza 1:5
        digitalWrite(S0, HIGH); //S0
        digitalWrite(S1, LOW); //S1
        //Serial.println("m1:5m");
    }else if(mode == 3){
        // frequenza 1:50
        digitalWrite(S0, LOW); //S0
        digitalWrite(S1, HIGH); //S1
        //Serial.println("m1:50m");
    }
    return;
}

void LED_RGB(int Rx, int Gx, int Bx){
    analogWrite(R, Rx);
    analogWrite(G, Gx);
    analogWrite(B, Bx);
    return;
}

void TCS230setup(){
    // inizializza pin gestione frequenza
    pinMode(LED,OUTPUT);
    // pin di selezione dei filtri colore
    pinMode(S2,OUTPUT); //S2
    pinMode(S3,OUTPUT); //S3
    // pin di uscita conversione luce-frequenza
    pinMode(taosOutPin, INPUT);
    // pin selezione modo frequenza di uscita
    pinMode(S0,OUTPUT); //S0 pinB
    pinMode(S1,OUTPUT); //S1 pinA
    return;
}

```

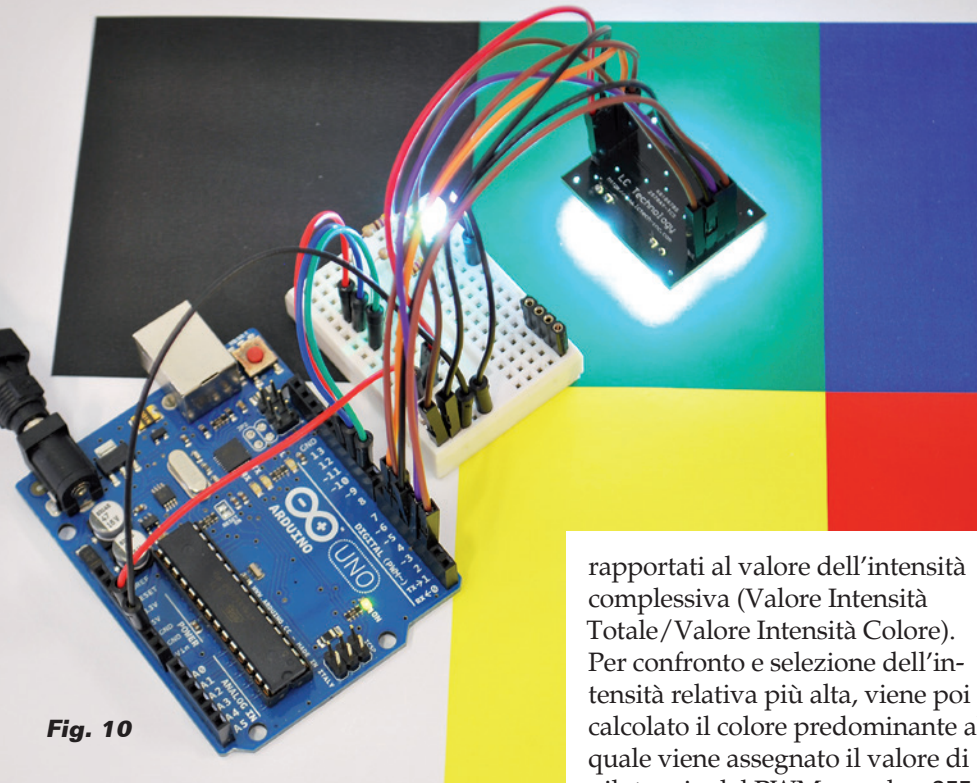


Fig. 10

risultato della lettura è in microsecondi. Il formato dell'istruzione è:

pulseIn(pin, value, timeout)

dove *pin* è il riferimento del pin per il quale si vuole leggere l'impulso (*int*), *value* il livello dell'impulso da leggere **HIGH** o **LOW**. (*int*) e *timeout* (opzionale) corrisponde al numero di microsecondi entro i quali deve essere rilevato un impulso. Il valore predefinito è di un secondo (*unsigned long*). Se entro il timeout non viene rilevato alcun impulso, viene prodotto come risultato il valore 0.

L'istruzione "pulsein" funziona per impulsi della durata tra 10 microsecondi e tre minuti, anche se per impulsi particolarmente lunghi possono verificarsi errori nelle letture. La logica di riconoscimento del colore richiede una prima lettura in assenza di filtri, condizione che si ottiene impostando i pin S2 ed S3 rispettivamente a livello alto e basso, come specificato in **Tabella 3**. Viene letto il valore presente sul pin OUT e, in sequenza, vengono impostati i filtri corrispondenti ai tre colori base e letti i valori delle relative intensità,

rapportati al valore dell'intensità complessiva (Valore Intensità Totale/Valore Intensità Colore). Per confronto e selezione dell'intensità relativa più alta, viene poi calcolato il colore predominante al quale viene assegnato il valore di pilotaggio del PWM uguale a 255. I valori di pilotaggio dei PWM per gli altri colori vengono calcolati in proporzione tra il valore letto per il colore in esame ed il valore del colore predominante rapportato a 255.

Come ultima operazione, i valori PWM calcolati vengono utilizzati per pilotare i rispettivi pin collegati ai terminali del LED RGB. Lo sketch del **Listato 1** può essere scaricato dal sito www.elettronica-in.it. Per eseguirlo, aprite l'IDE di Arduino e copiate lo sketch nell'editor dell'IDE. Collegate Arduino Uno al PC con l'apposito cavo USB, impostate la porta COM e la scheda come indicato in precedenza, compilate ed eseguite l'upload dello sketch nel microcontrollore.

Aprirete il Monitor Seriale: posizionate il sensore su oggetti o su fogli di carta di colore diversi e divertitevi a verificarne il funzionamento (**Fig. 10**).

Ora andiamo a vedere se la nostra stampante 3D ha terminato di realizzare l'alberello che rappresenta l'involucro della lampada. Assembliamo il tutto e andiamo in giro a raccogliere i commenti sul nostro prototipo: nel mondo professionale, questa attività si chiama test di mercato.

INGEGNERIZZAZIONE DEL PRODOTTO

Una volta che il prototipo è stato "approvato", il che, come è capitato anche a noi, può richiedere diverse modifiche delle specifiche e rivisitazioni di soluzioni soprattutto estetiche, dobbiamo iniziare a pensare a produrre una versione "commerciale". Non possiamo certo mantenere lo schema a fili volanti ed Arduino Uno è decisamente troppo ingombrante e costoso per un prodotto di questo tipo. Dunque dobbiamo operare delle scelte con l'obiettivo di:

- sostituire Arduino con un microcontrollore più economico e di ridotte dimensioni; la scheda Arduino Uno è sovrabbondante per un utilizzo in produzione e richiederebbe comunque la realizzazione di uno shield collegato con dei pin-strip, con ingombri e costi che posizionerebbero il nostro prodotto fuori mercato ancor prima di produrlo;
- realizzare un circuito stampato che si adatti alla disposizione dei componenti, ovvero il sensore verso il basso e il o i LED RGB verso il centro della lampada, e che si adatti ad essere alloggiato all'interno della lampada stessa;
- proteggere il codice in modo che non sia facilmente replicabile da chi volesse copiare il progetto per un uso commerciale;
- realizzare il tutto in modo che sia facilmente gestibile a chi dovrà occuparsi di assemblaggi, montaggio, gestione a magazzino e spedizione.

Per la lampada, anche qui dipende dal numero di pezzi da ottenere: con la stampante 3D possono essere realizzati prototipi e piccole serie, mentre per la produzione di massa è probabilmente meglio rivolgersi a stampatori con impianti a iniezione; in questo caso la pro-

duzione deve prevedere volumi veramente grandi, per ammortizzare il costo di allestimento degli stampi e dell'attrezzaggio delle macchine di produzione.

Quanto al microcontrollore, avremmo potuto scegliere un micro Atmel compatibile con il codice scritto per il prototipo, per poi compilarlo e caricarlo sul micro stesso con una tecnica che non richieda il bootloader, come ampiamente descritto nel libro "Arduino e le Tecniche di programmazione dei microcontrollori Atmel" edito da Futura Group. Invece abbiamo preferito una soluzione "scomoda" ma che viene adottata molto spesso in questo tipo di realizzazioni: siamo "andati a cadere" su un microcontrollore ultraeconomico della Microchip, e precisamente il PIC16F1827. Definiamo la scelta "scomoda" perché dobbiamo riscrivere o meglio "portare" il codice scritto per Arduino UNO in una forma utilizzabile per il micro PIC16F1827.

I motivi per cui abbiamo scelto un prodotto Microchip e in special modo questo micro, sono i seguenti:

- ha un costo contenuto;
- è disponibile sia in contenitore PDIP che SMD;
- è dotato di clock interno, il che ci evita il ricorso a quarzi e componenti aggiuntivi, con il conseguente ulteriore contenimento dei costi;
- presenta consumi ridotti;
- dispone di 15 pin di I/O, ciascuno in grado di pilotare un LED;
- ha un numero di pin PWM (4) sufficienti per i nostri scopi;
- presenta compatibilità pin to pin con micro della stessa famiglia con più funzionalità.

Come ulteriore "ingegnerizzazione" di prodotto, abbiamo sostituito il LED RGB con uno più

potente e con una gestione migliore della luce prodotta, anche se questo ha comportato la necessità di pilotare ciascun "colore" con un circuito di potenza basato su un transistor MOSFET.

Il risultato finale è visibile nello schema elettrico qui sotto (notate che il LED RGB è composto dai diodi LD1r, LD1g, LD1b). Possiamo vedere, con riferimento anche al **Listato 2**, le assegnazioni dei pin del microcontrollore Microchip PIC16F1827. Nello schema elettrico è indicato sia il nome "funzionale" del pin che il suo numero di riferimento. Come si vede nel **Listato 2**, i pin RB1, RB2, RB4 ed RB5 sono pin digitali di output, che pilotano gli ingressi del sensore TCS230. Su RB7 viene letta la durata dell'impulso risultato della misurazione dell'intensità di luce. Come pin di uscita

PWM abbiamo utilizzato CCP2, CCP3 e CCP4, ai quali fanno capo i terminali del LED di potenza B, G ed R. L'alimentazione per il microcontrollore e per il resto del circuito è fornita dallo stadio di alimentazione costruito attorno all'integrato U2, un regolatore di tensione lineare di tipo 7805, con un diodo di protezione dall'inversione di corrente e le celle di livellamento e stabilizzazione della tensione realizzate con i condensatori C1, C2, C3 e C4. Dato l'alto consumo del circuito, dovuto principalmente al LED di potenza, raccomandiamo di utilizzare un alimentatore esterno in CC stabilizzato che eroghi una tensione non superiore a 7,5 V. Questo, per evitare un eccessivo surriscaldamento dell'integrato 7805. Per quanto riguarda la breakout board TCS230, alimentazione a

[schema **ELETTRICO**]

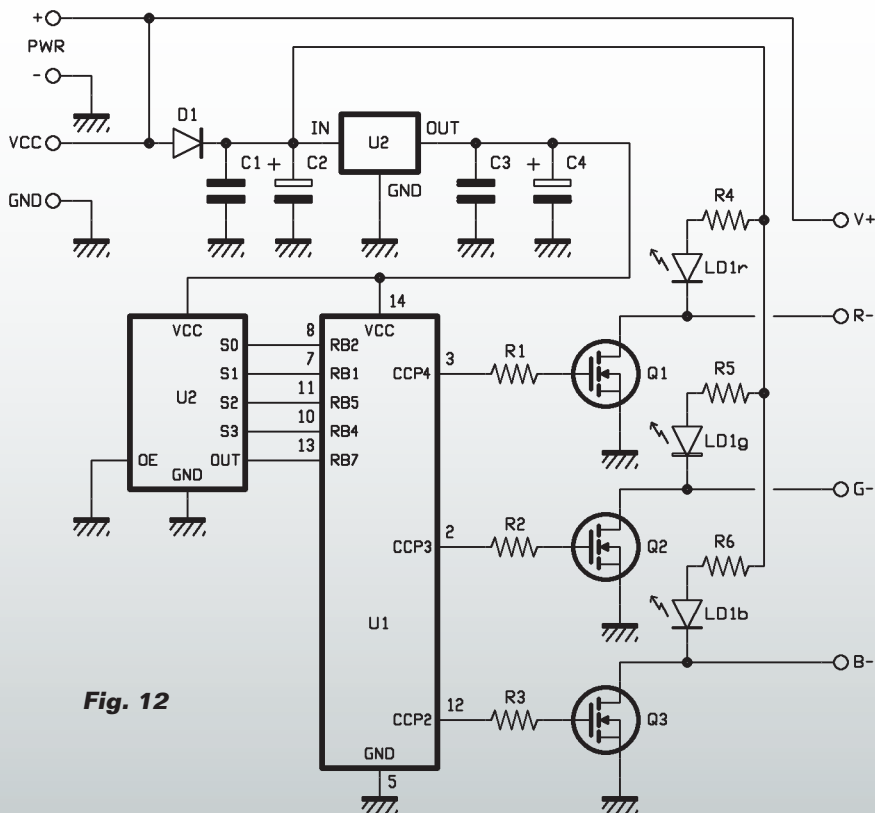


Fig. 12

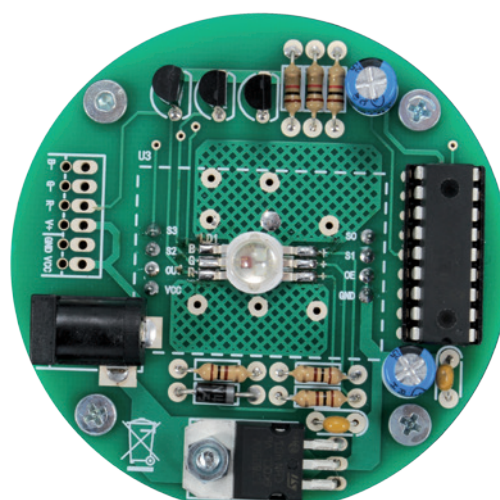
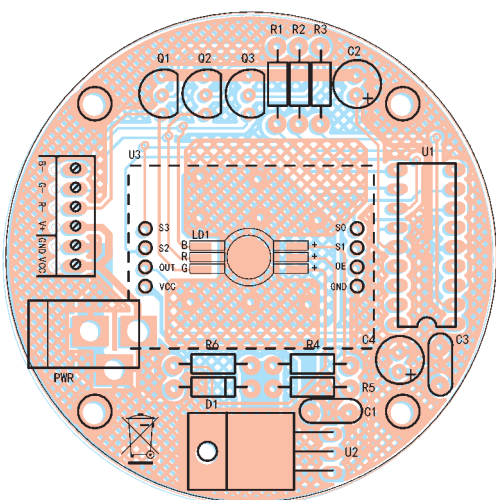
parte, l'unica nota da aggiungere, come nel caso del prototipo, è il collegamento a massa del terminale OE, in modo da mantenere sempre in funzione il sensore. Per pilotare correttamente il LED di potenza, per ciascun terminale "colore" del LED è inserito un circuito di potenza costituito da un MOSFET BS170, con relativa resistenza di limitazione. Le resistenze R4, R5 ed R6 limitano la corrente assorbita da ciascun terminale del LED di potenza. Se date un'occhiata al circuito stampato potete notare l'adattamento della realizzazione

per l'utilizzo con la lampada. Abbiamo previsto di montare il sensore al di sotto dello stampato, rivolto verso la superficie della quale si vuole rilevare il colore, mentre gli altri componenti, sono montati superiormente, in particolare il LED di potenza in posizione "centrale", in modo da diffondere la luce nel modo migliore. La disposizione dei componenti è visibile nel "Piano di montaggio". Abbiamo montato sull'alloggiamento della breakout board TCS230 due strip di connettori femmina, così come abbiamo

adottato uno zoccolo per l'integrato PIC16F1827. Chiaramente, in caso di montaggio di produzione questi ultimi componenti verrebbero saldati direttamente sullo stampato. Ma tant'è, alla fine ha prevalso lo spirito del "kit". Quattro "colonnine", dalla parte della breakout board, costituiscono una solida base sulla quale poggiare il tutto. Portiamo l'attenzione sul software di gestione della lampada, da installare sul microcontrollore PIC16F1827.

Per il "porting" del software abbiamo scelto il linguaggio di svilup-

[piano di **MONTAGGIO**]



Elenco Componenti:

R1: 1 kohm	D1: 1N4007
R2: 1 kohm	Q1: BS170
R3: 1 kohm	Q2: BS170
R4: 100 ohm	Q3: BS170
R5: 100 ohm	LD1: LEDRGB1W
R6: 100 ohm	
C1: 100 nF multistrato	Varie:
C2: 100 μ F 25 VL elettrolitico	- Plug alimentazione
C3: 100 nF multistrato	- Zoccolo 9+9
C4: 100 μ F 25 VL elettrolitico	- Strip femmina 4 poli (2 pz.)
U1: PIC16F1827-I/P (MF1170)	- Vite 3MA 6mm (4 pz.)
U2: 7805	- Torretta F/F 23mm (4 PZ.)
U3: Brackout Board TCS230	- Circuito stampato

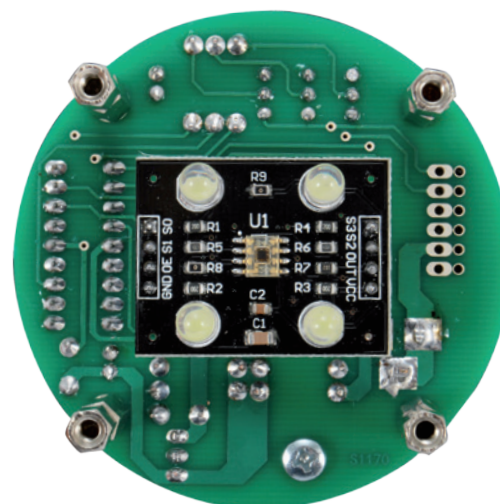




Fig. 11

po Pic Basic Pro, una versione di BASIC appositamente destinato allo sviluppo sui microcontrollori Microchip. Per la programmazione dobbiamo utilizzare un programmatore Pic, nel nostro caso il modello Pickit 3, disponibile presso Futura Elettronica (www.futurashop.it).

Nel **Listato 2** vediamo il risultato del "porting", che rispetta la logica di elaborazione vista nel diagramma a blocchi di **Fig. 7**. I commenti nel listato e la sua "somiglianza" con lo sketch di Arduino non dovrebbero lasciare dubbi di interpretazione. Per rendere "modulare" l'utilizzo della lampada, ovvero per offrire la possibilità di sostituire l'albero con contenitori di altre forme, abbiamo separato la base della lampada dal "lume". Come base abbiamo stampato un cilindro di 70 mm di diametro e 40 mm di altezza, in modo da contenere tutto

Listato 2

```
*****
;* Name : LampadaCamaleonte.pbp *
;* PIC : 16F1827 *
;* Notes : Firmware Lampada Camaleonte *
;* Riproduce i colori della superficie di appoggio *
*****

INCLUDE "modedefs.bas"
'@ DEVICE INTRC_OSC_NOCLKOUT
'@ DEVICE MCLR_OFF
DEFINE OSC 16
'DEFINE PULSIN_MAX 1350
'DEFINE PULSIN_MAX 1750

ADCON0 = %00000000
OPTION_REG.7=0
'CMCON0=%00000111
ANSELA=%00000000
ANSELB=%00000000
'OSCCON=%01101010
OSCCON=%01111010
APFCON0.3=0 'CCP2 su pin Rb6
'MDSCR=%00000011
'WPU=%00010000

***** Definizione porte del PIC *****

SYMBOL TX = PORTB.3 'Pin 9 - Dati Seriali in uscita

SYMBOL S0 = PORTB.2 ' Pin 8
SYMBOL S1 = PORTB.1 ' Pin 7
SYMBOL S2 = PORTB.5 ' Pin 11
SYMBOL S3 = PORTB.4 ' Pin 10
SYMBOL OUTC = PORTB.7 ' Pin 13

Coeff VAR WORD

Ro VAR word
Go VAR word
Bo VAR word

white VAR WORD
white_off VAR WORD
red VAR WORD
blu VAR WORD
green VAR WORD

Tolerance VAR WORD
mode VAR BYTE

DEFINE CCP3_REG PORTA 'Hpwm 1 pin port
DEFINE CCP3_BIT 3 'Hpwm 1 pin bit
DEFINE CCP4_REG PORTA 'Hpwm 1 pin port
DEFINE CCP4_BIT 4 'Hpwm 1 pin bit
DEFINE CCP2_REG PORTB 'Hpwm 1 pin port
DEFINE CCP2_BIT 6 'Hpwm 1 pin bit

INPUT OUTC
OUTPUT S0
OUTPUT S1
OUTPUT S2
OUTPUT S3

***** Dichiarazione variabili *****
CONTA VAR WORD 'Contiene la lunghezza dell'impulso
ERRORE VAR BYTE '0:Nessun errore nel segnale - 1:Segnale non valido

Ro = 0
Go = 0
Bo = 0

CONTA=0
ERRORE=0

***** Inizializzazione programma *****
INIZIO:
CLEAR
SEROUT tx,T9600,[13,10,"SYSTEM STARTUP v1.0",13,10]

***** M A I N *****
'Lettura lunghezza impulso "0" e successivamente lettura treno di bit ricevuti
MAIN:
mode = 3
GOSUB TAOS_MODE
CONTINUA:
' Legge il bianco
```

(Continua)

```

LOW S3
HIGH S2
GOSUB LEGGI_COLORE
SEROUT tx,T9600,[13,10,"W:",#CONTA," (\",#S2,"-",#S3,")",13,10]
white = CONTA * 100
SEROUT tx,T9600,[13,10,"Wx:",#white," (\",#S2,"-",#S3,")",13,10]

    \ Legge il rosso

LOW S3
LOW S2
GOSUB LEGGI_COLORE
SEROUT tx,T9600,[13,10,"R:",#CONTA," (\",#S2,"-",#S3,")",13,10]
red = white / CONTA
SEROUT tx,T9600,[13,10,"Rx:",#red," (\",#S2,"-",#S3,")",13,10]

    \ Legge il verde

HIGH S3
HIGH S2
GOSUB LEGGI_COLORE
SEROUT tx,T9600,[13,10,"G:",#CONTA," (\",#S2,"-",#S3,")",13,10]
green = white / CONTA
SEROUT tx,T9600,[13,10,"Gx:",#green," (\",#S2,"-",#S3,")",13,10]

    \ Legge il blu

HIGH S3
LOW S2
GOSUB LEGGI_COLORE
SEROUT tx,T9600,[13,10,"B:",#CONTA," (\",#S2,"-",#S3,")",13,10]
blu = white / CONTA
SEROUT tx,T9600,[13,10,"Bx:",#blu," (\",#S2,"-",#S3,")",13,10]

IF red >= blu AND red >= green THEN
    Coeff = 25500 / red
    Ro = 255
    Go = green * Coeff / 100
    Bo = blu * Coeff / 100
    Bo=Bo * 3 / 4
    Go=Go * 3 / 4
ENDIF

IF blu >= red AND blu >= green THEN
    Coeff = 25500 / blu
    Bo = 255
    Go = green * Coeff / 100
    Ro = red * Coeff / 100
    Ro=Ro * 3 / 4
    Go=Go * 3 / 4
ENDIF

IF green >= blu AND green >= red THEN
    Coeff = 25500 / green
    Go = 255
    Ro = red * Coeff / 100
    Bo = blu * Coeff / 100
    Bo=Bo * 3 / 4
    Ro=Ro * 3 / 4
ENDIF

HPWM 2,b0,100
HPWM 3,g0,100
HPWM 4,r0,100

SEROUT tx,T9600,[13,10,"RGB:",#Ro,"-",#Go,"-",#Bo,13,10]

    pause 500
GOTO CONTINUA

TAOS MODE:
    SELECT CASE mode
        CASE 0
            LOW S0
            LOW S1
        CASE 1
            HIGH S0
            HIGH S1
        CASE 2
            HIGH S0
            LOW S1
        CASE 3
            LOW S0
            HIGH S1
    END SELECT
    pause 50
RETURN

LEGGI_COLORE:
    pause 50
    PULSIN OUTC,0,CONTA
RETURN

```

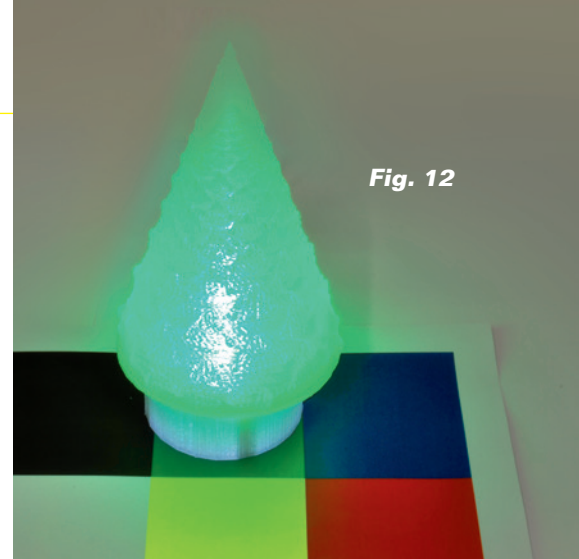
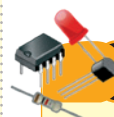


Fig. 12

il circuito della lampada e fornire supporto alla parte superiore (vedere la Fig. 11). Abbiamo quindi praticato un foro laterale da 10 mm di diametro in corrispondenza del plug di alimentazione per fare passare il jack dell'alimentatore. Finalmente (lo vedete nella Fig. 12) abbiamo raggiunto il risultato finale del nostro processo: la lampada di "Natale" che si adatta al colore dell'oggetto dove viene appoggiata. Provate ad appoggiarla sulle confezioni dei regali o sulla tovaglia della tavola di Natale e vedrete che prenderà il loro colore. Ora non ci resta che salutarvi, augurarvi buon divertimento e ...Buone Feste!



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità ad eccezione del Sensore Colore TCS230 (cod. 2846-SENSORECOLORE - Euro 8,00), del LED RGB da 1W (cod. 2846-LEDRGB1W - Euro 3,00) e del microcontrollore, non programmato, (cod. PIC16F1827-I/P - Euro 4,00) che possono essere acquistati presso Futura Elettronica. Stampato e Firmware del microcontrollore sono scaricabili dal sito della rivista.

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



ROBOT SPAZZANEVE

di VITTORIO LOSCHIAVO e GIUSEPPE OLIVIERI

Robot cingolato basato su Arduino Uno, comandato a distanza grazie al controller wireless della PlayStation 2. Prima puntata.

Anche quest'anno, probabilmente arriverà la neve. Molti lo sperano: gli studenti, cui fornirebbe una scusa per rimanere a casa da scuola, gli appassionati degli sport invernali (che la aspettano in montagna), i bambini, per giocare a palle di neve o realizzare il proprio pupazzo, i nostalgici, che la aspettano per la notte di Natale, ma c'è anche chi la vuole per la sola soddisfazione di godersi un panorama imbiancato caratterizzato da quei suoni ovattati che conferiscono alla neve qualcosa di unico. Indubbiamente la neve è suggestiva e gradevole, o meglio, lo sarebbe se non limitasse i nostri movimenti in auto e se non dovessimo

spalarla per uscire di casa o dal nostro garage, per rendere agibile il marciapiede, il vialetto che porta al condominio, l'ingresso del negozio, le aree di parcheggio o altro.

Dopo molti anni che si vedono in nazioni come Canada, Stati Uniti d'America e non solo, anche in Italia sono arrivate presso in centri del fai da te le cosiddette "Frese per la neve": se ne trovano ormai di vari modelli e dimensioni, alimentate dalla corrente elettrica di casa o a benzina, da spingere a mano o da guidare come un trattorino. A seconda del modello, queste macchine sono in grado di rimuovere, sparandola a lato, una piccola o grande quantità di neve.

Noi che non siamo insensibili alle soluzioni per semplificare la vita, abbiamo preso in considerazione l'idea di realizzare e proporre qualcosa che ci evitasse il freddo e la fatica di utilizzare le solite pale a mano, tuttavia abbiamo optato per non replicare ciò che si trova in commercio, ma qualcosa di divertente e tecnologico. Abbiamo quin-



Fig. 1

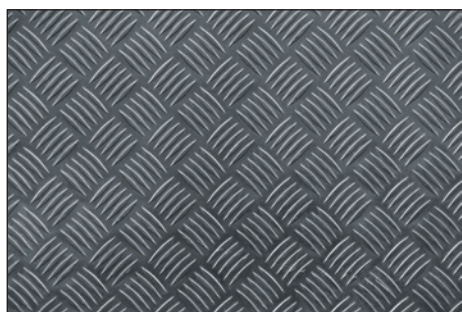


Fig. 2

di deciso di mettere in cantiere un Robot Spazzaneve controllato a distanza, nel quale fanno bella mostra di loro una robusta meccanica e una raffinata elettronica tutta implementata con schede facilmente reperibili in commercio.

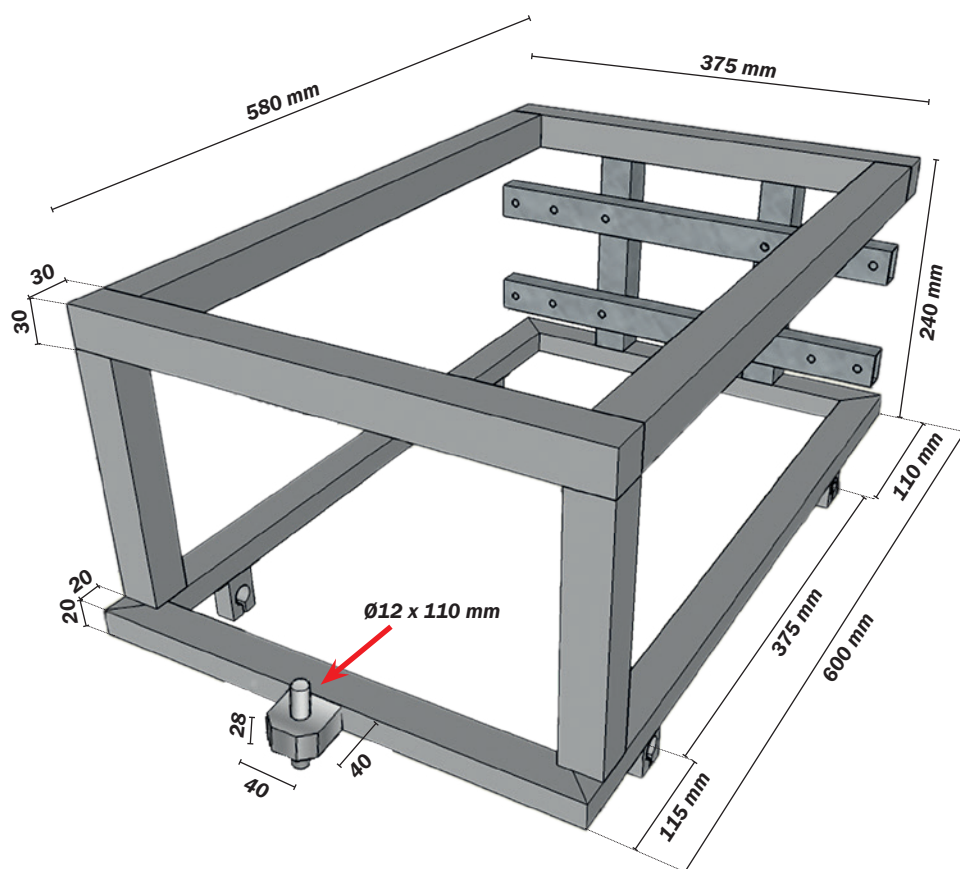
Il risultato è la versione miniatura di uno spazzaneve vero, con tanto di cingoli, pala e proiettore montato sulla sommità per illuminare la strada al buio; il tutto si comanda mediante un telecomando da Play-Station 2. Vediamo dunque come è costruito il nostro cingolato, partendo dalla meccanica e descrivendo poi l'elettronica ed il firmware che le occorre.

La meccanica

Dal lato meccanico, il robot spazzaneve consta di un telaio che sostiene una pala -motorizzata per essere sollevata e orientata lateralmente- e che supporta il sistema di trazione a motori elettrici, ruote e cingoli.

Il telaio

Inizialmente si era pensato di realizzare l'intera struttura con tubolari in alluminio tenuti insieme da angolari, ma date le elevate sollecitazioni meccaniche cui sarebbe stato sottoposto e la robustezza



Misure del telaio portante dello spazzaneve; il perno indicato dalla freccia ha, nella parte bassa, un diametro di 14 mm per un'altezza di 5 mm, che serve ad andare in battuta nel telaio.



Fig. 3

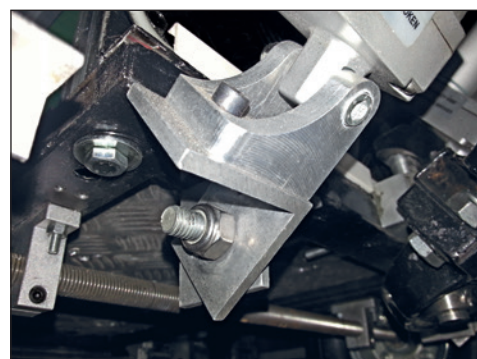


Fig. 4

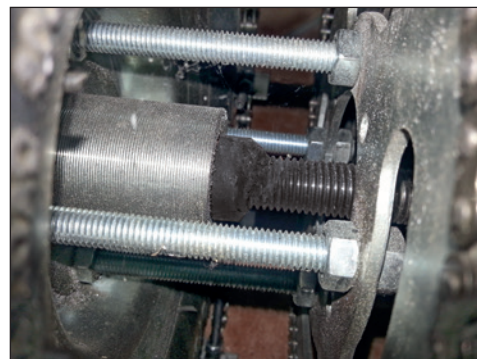


Fig. 5

richiesta, abbiamo deciso di utilizzare, per la base, quattro tubolari di ferro a sezione rettangolare saldati insieme, ottenendo così un rettangolo da 600x375 mm (Fig. 1). Come base d'appoggio per i vari componenti meccanici, abbiamo utilizzato una piastra di alluminio 600x375 spessa 3 mm (Fig. 2) fissata sopra la base del telaio in ferro. Due profilati di alluminio 30x30 mm lunghi 190 mm montati verticalmente, più due orizzontali 30x30x580 mm (lato lungo) e altrettanti da 30x30x375 mm (lato corto), sono state utilizzate come scheletro cui applicare la copertura esterna del robot, quest'ultima realizzata utilizzando 5 pannelli in alluminio da 3 mm di spessore; per fissare i montanti alla base, bisogna praticare nella loro sezione un foro 8MA, poi filettato (Fig. 3). Due staffe a "L" (Fig. 4) in alluminio sono fissate, sempre sulla parte frontale del telaio, e servono da supporto per i due attuatori lineari che permettono alla pala di ruotare a destra e sinistra.

Il sistema di trazione

Il movimento dello spazzaneve è affidato a quattro ruote per carrelli, dotate di pneumatici e mozzo provvisto di cuscinetto a sfera con foro del diametro di 16 mm; ogni ruota ha un diametro esterno di 26 cm e pneumatico largo circa 8,5 cm. Facilmente reperibili in molti centri



Fig. 6

del fai da te, queste ruote sono in grado di sopportare (singolarmente) un peso di 150 kg, per un totale di 600 kg, quindi più di quello che ci serve per il nostro robot. Per gli assi delle ruote sono state utilizzate delle barre filettate da 16 mm di diametro, lunghe 63 cm, fatte passare attraverso dei supporti in alluminio e qui bloccate, facendo in modo che da ogni lato del telaio escano in misura uguale. I supporti sono fissati alla base inferiore del telaio del robot.

Per bloccare le ruote alla giusta distanza (lato verso il supporto) si può optare per due soluzioni; dado e controdado, oppure, come nel nostro caso, saldare un dado direttamente sulla barra filettata (Fig. 5). Per la prima soluzione, una volta fatto ciò bisogna inserire un dado 16 MA nella barra filettata, avvitarlo fino a farlo arrivare a circa 7 cm dal supporto della barra (Fig. 6), verificare più volte la misura e quindi saldarlo. Lo stesso vale per l'altro lato e l'altra barra.

Per montare la corona su ogni ruota, bisogna togliere i quattro bulloni originali e ricavare altri quattro fori, di pari diametro, che combacino con quelli della rispettiva ruota; per posizionare i fori con precisione, conviene creare una dima in cartoncino ricavando l'interasse dei fori delle ruote. Forate le corone, bisogna inserire i quattro da 8MA lunghi 100 mm e avvitare un dado autobloccante in modo da fissare il bullone al cerchio della ruota (Fig. 7); poi si inserisce un altro dado autobloccante (o dado e controdado) ad una distanza di circa 72 mm dal dado avvitato nel lato interno del cerchio. Ora si può applicare la corona, centrandone i fori rispetto ai quattro retrostanti e avvitare con quattro dadi ben stretti. Fatto ciò, si monta la ruota: se tutto è corretto avrete i dadi della corona ad una distanza massima di 10 mm dal telaio. Su

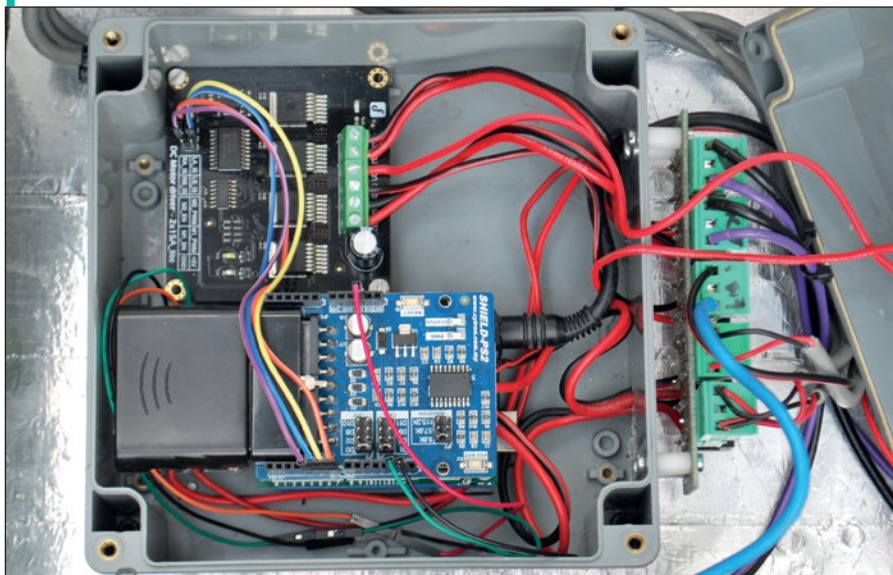
ogni asse si devono inserire i dadi autobloccanti 16 MA. Esternamente, le due ruote sono distanziate e tenute in tensione tramite una barra angolare di alluminio da 30x30 mm, spessa 2 mm (Fig. 8) e vanno fissate tramite un bullone al centro della barra filettata. Quando abbiamo abbassato la pala per spalare ci siamo accorti che era necessario aumentare l'aderenza utilizzando dei cingoli (o delle ruote adatte, tipo quelle dei trattorini), ma abbiamo deciso di trovare una soluzione alternativa. Prendendo spunto dalle varie soluzioni trovate in Internet, e adattandole alle nostre esigenze, abbiamo realizzato dei cingoli "casalinghi" utilizzando quattro confezioni di catene per bicicletta da 1/2"x1/8" con 114 maglie ciascuna (due per lato, per un totale di 120 maglie, corrispondenti a una lunghezza di circa 166 cm), 40 angolari in alluminio (mm) 12x12 lunghi 120 con spessore 2 e 80 bulloni 5MA lunghi 50 mm e relativi dadi autobloccanti (Fig. 9).



Fig. 7

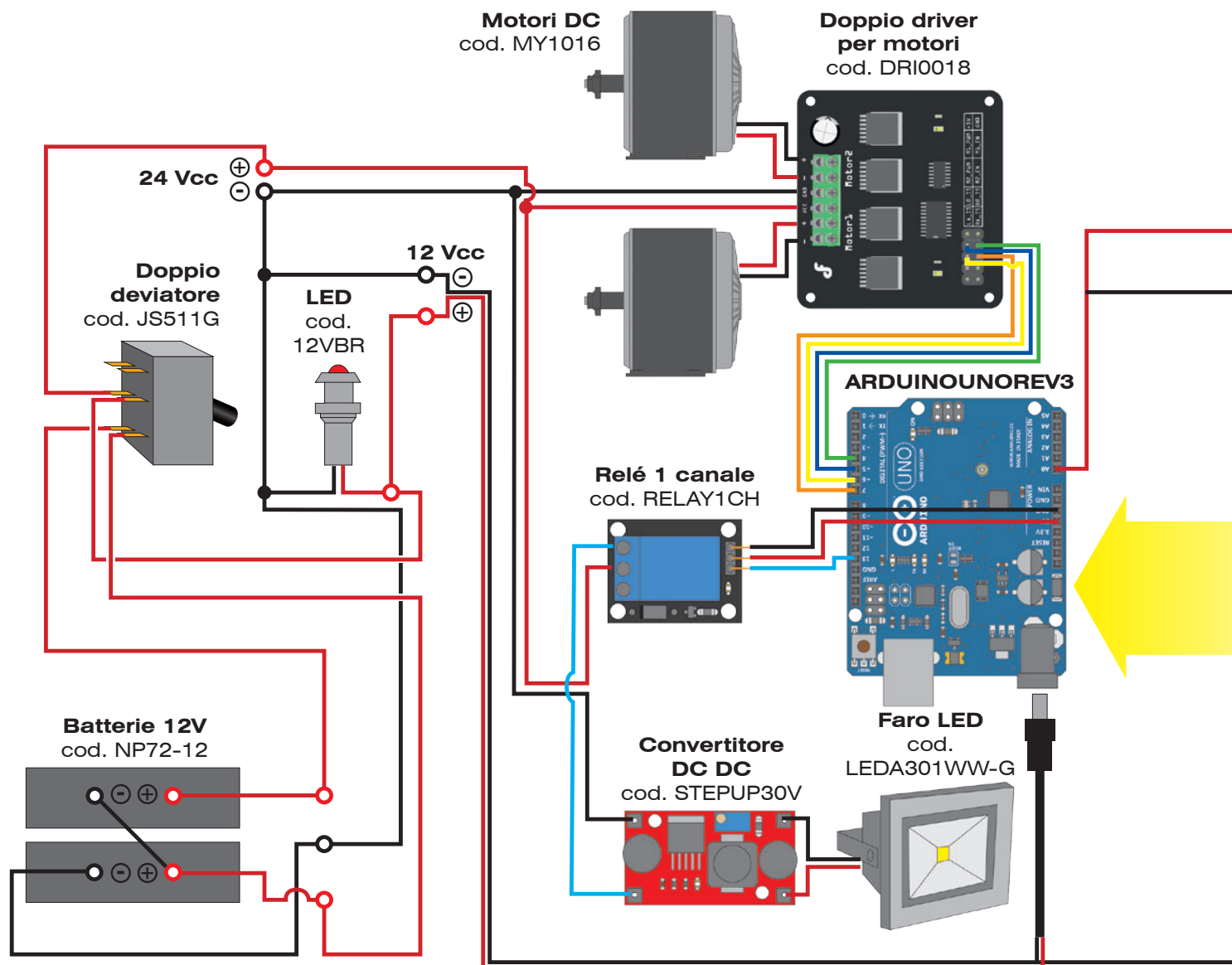


Fig. 8



Il cablaggio all'interno del contenitore in plastica.

Gli angolari in alluminio, distanziati dal centro di un bullone al centro dell'altro di 8 cm, oltre che a tenere insieme e parallele le catene di ciascuna coppia costituente un cingolo (avvolte intorno alla relativa coppia di ruote) permettono di avere la giusta aderenza al manto nevoso. Su ogni angolare vanno praticati due fori, nei quali bisogna inserire due bulloni 5MA x 50 mm, necessari a fissarli alla catena ed evitando che le catene scivolino da un lato o dall'altro. Per abbinare le ruote al pignone



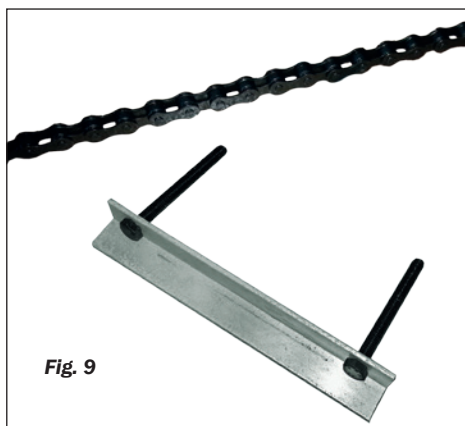
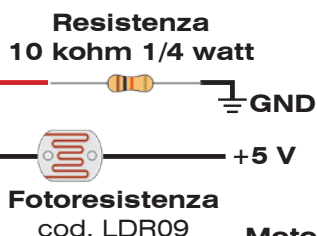
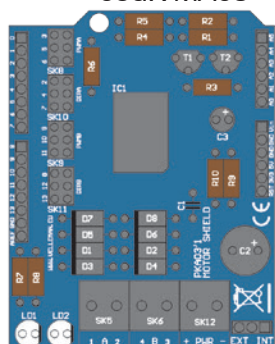


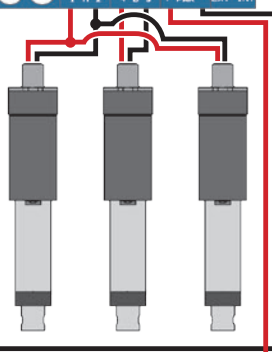
Fig. 9



Motor shield
cod.VMA03

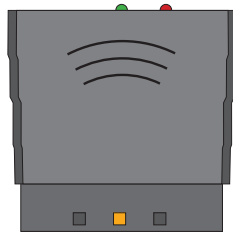


Attuatori lineari
cod. LACT2

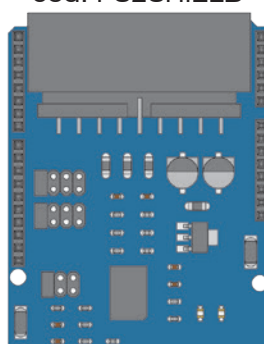


del motore MY1016, è stato necessario procurarsi due catene a passo corto (tipo quelle dei monopattini elettrici o delle minimoto), montare sulle due ruote motrici due corone di diametro 155 mm a 76 denti e distanziarle di circa 35 mm dalla ruota (lato interno del pneumatico) tramite 4 bulloni 8MA x 100 mm. Per fissare i motori abbiamo utilizzato dei supporti, ricavati da due barre rettangolari di ferro (mm) 30x20 lunghe 375 e forate. Partendo dall'esterno, il primo foro si trova a circa 14 mm, mentre il secondo a 42 mm dal primo; questa distanza corrisponde ai

Ricevitore PS2
RX_PS2



Shield PS2
cod. PS2SHIELD



fori del motore (parte più stretta della staffa del motore). Sulla barra sono stati praticati due fori, a 110 mm dall'esterno e da entrambi i lati, necessari per il fissaggio alle due barre verticali da 190 mm. La stessa foratura è stata fatta per la seconda barra. Le due barre verticali da 19 mm, una volta saldate



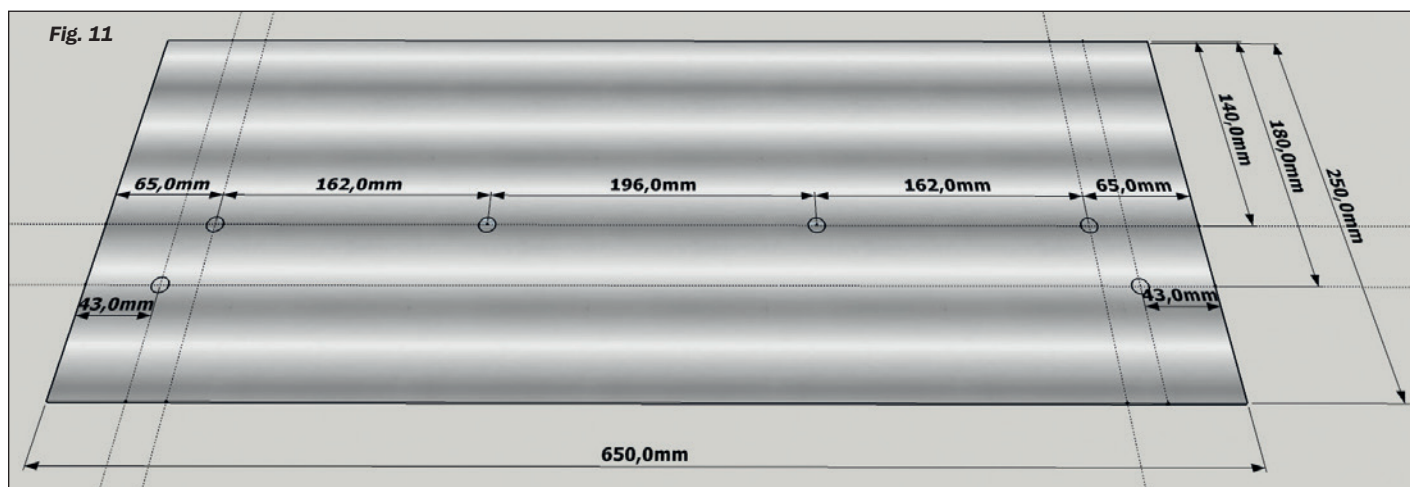
Fig. 10

alla base del telaio, sono state forate a una distanza (partendo dalla base del telaio) di circa 44 mm per il primo foro (a forma di asola) e 139 mm per il secondo foro (sempre a forma di asola), come mostrato in Fig. 10. A questo punto non rimane che fissare i motori, orientando il pignone verso l'esterno e verificando che sia in linea con la corona della ruota.

La pala

La parte meccanica, un po' più complessa, è quella del supporto della pala e del suo movimento. Per realizzare la pala abbiamo utilizzato una lastra di acciaio rettangolare da 650x250 mm, spessa 2 mm (Fig. 11), che poi abbiamo leggermente curvato (sopra e sotto, lato lungo) per agevolare la spalatura della neve. Per la struttura meccanica che supporta la pala e permette di muoverla (Fig. 12), abbiamo utilizzato uno scatolare di ferro da 30x20 mm lungo 265 mm, tagliato all'estremità con la richiesta angolazione, uno scatolare di ferro 40x20 mm lungo 13 cm, un profilato rettangolare da 30x20 mm lungo 46 mm, una traversa ad incastro lunga 600 mm (utilizzata per i ripiani/scaffali e reperibile presso i centri del fai da te), un supporto con cuscinetto da 16 mm e due scatolari da 40x30 mm lungo 26,5 cm tagliato all'estremità la richiesta angolazione. Abbiamo quindi realizzato un perno in alluminio lungo 11 cm (ϕ 12 mm tranne che nei primi 5 mm, dove ha ϕ 14 mm) da inserire nel cuscinetto da 12 mm, nella

Fig. 11

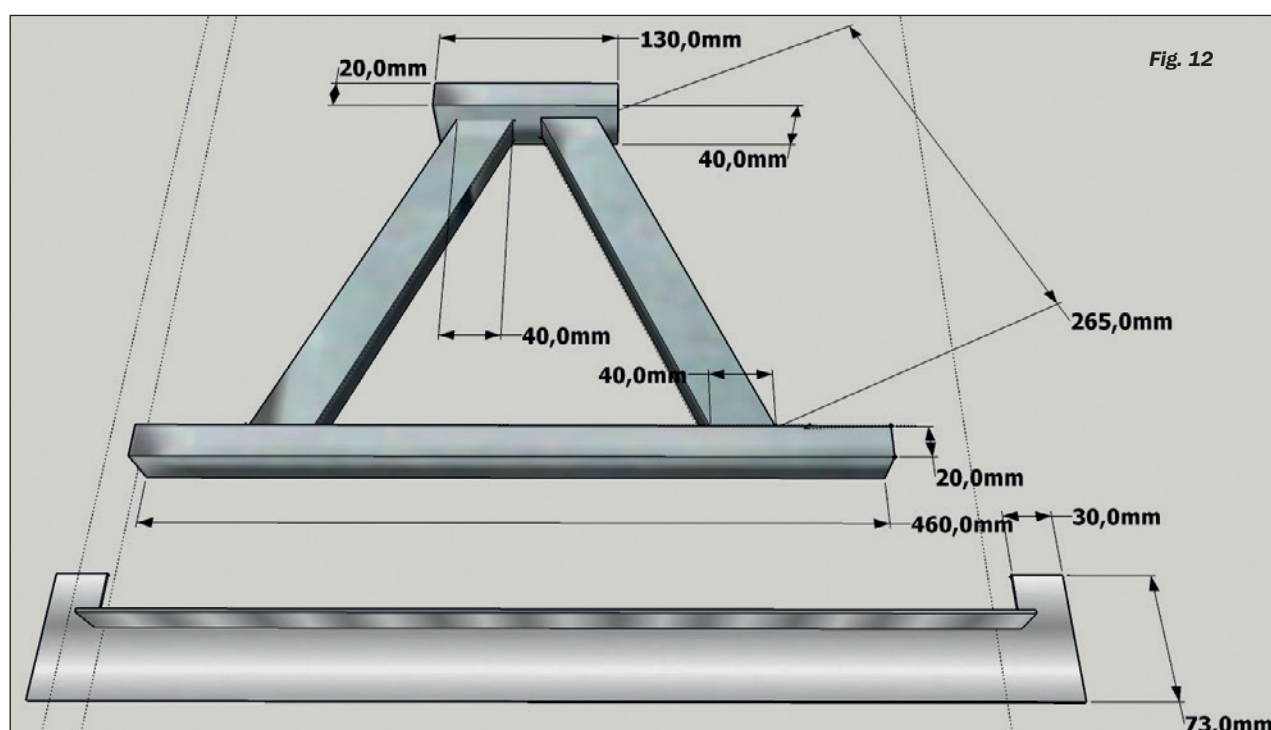


parte superiore del quale è stato praticato un foro da 5 mm, poi filettato, per fissare, mediante un bullone 5MA, la staffa per l'attuatore lineare che permette di alzare e abbassare la benna. La seconda staffa dell'attuatore è stata fissata al centro della barra con lunghezza 265 mm. Per ruotare a destra e sinistra la pala, abbiamo utilizzato per ogni lato due staffe a L in alluminio da 50x50 mm, larghe 40 e spesse 5, forate e avvitate inserendo tra le due una rondella,

come si vede in Fig.13. Una coppia di L è stata fissata, tramite due bulloni, nella parte inferiore (lato sinistro) del telaio e l'altra a quello destro. A una staffa della coppia di L è stata fissata la relativa staffa per l'attuatore lineare. La seconda staffa dell'attuatore destro e sinistro è stata fissata sulla struttura a triangolo della pala (Fig. 14). Prima di forare e fissare i tre attuatori lineari sul triangolo della pala, bisogna prendere i due attuatori lineari che servono per ruotare a

destra e sinistra la pala, applicare 12 Vcc a uno dei due in modo da avere l'estensione massima, mentre l'altro deve risultare completamente contratto. Ora vanno collegati i fili di alimentazione dei due attuatori in modo da avere il positivo del 1° attuttore collegato con il negativo del 2° e il negativo del 1° sul positivo del 2°. Bisogna quindi alimentare la coppia di fili degli attuatori e togliere tensione appena hanno raggiunto la stessa estensione. Adesso si possono

Fig. 12



Listato 1 - Lettura da seriale pulsanti del controller PS2, tramite PS2SHIELD

```
#include <Shield_PS2.h>

//declare class objects
PS2 ps2=PS2();          //PS2 class object: ps2

#define LEDPIN 13

void setup()
{
  Serial.begin(9600);
  ps2.init(9600, 2, 3); //initialize the main board to use desired (baudrate, rx, tx)
                        //for Arduino Mega use RX:10, TX: 11 for software serial
                        //for Arduino Leonardo use pin 8, 9, 10, 11 as RX and TX for software serial

  pinMode(LEDPIN, OUTPUT);
  digitalWrite(LEDPIN,LOW);
}

void loop()
{
  if( ps2.getval(p_up)==0) {
    Serial.println("UP");
  }
  if (ps2.getval(p_down)==0) {
    Serial.println("DOWN");
  }
  if (ps2.getval(p_right)==0){
    Serial.println("RIGHT");
  }
  if (ps2.getval(p_left)==0) {
    Serial.println("LEFT");
  }
  if (ps2.getval(p_triangle)==0) {
    Serial.println("TRIANGOLO");
  }
  if (ps2.getval(p_cross)==0) {
    Serial.println("X");
  }
  if (ps2.getval(p_circle)==0) {
    Serial.println("CERCHIO");
  }
  if (ps2.getval(p_square)==0) {
    Serial.println("QUADRATO");
  }
  if (ps2.getval(p_L1)==0) {
    Serial.println("L1");
  }
  if (ps2.getval(p_L2)==0) {
    Serial.println("L2");
  }
  if (ps2.getval(p_r1)==0) {
    Serial.println("R1");
  }
  if (ps2.getval(p_r2)==0) {
    Serial.println("R2");
  }
  if (ps2.getval(p_start)==0) {
    Serial.println("START");
  }
  if (ps2.getval(p_select)==0) {
    Serial.println("SELECT");
  }
}
```

Fig. 13

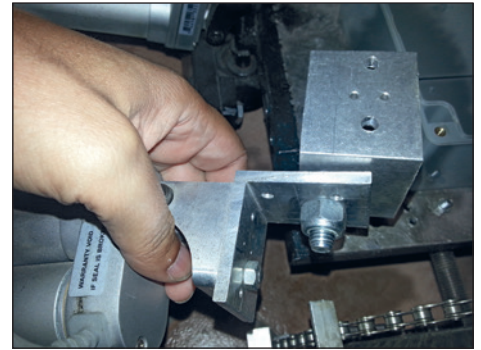
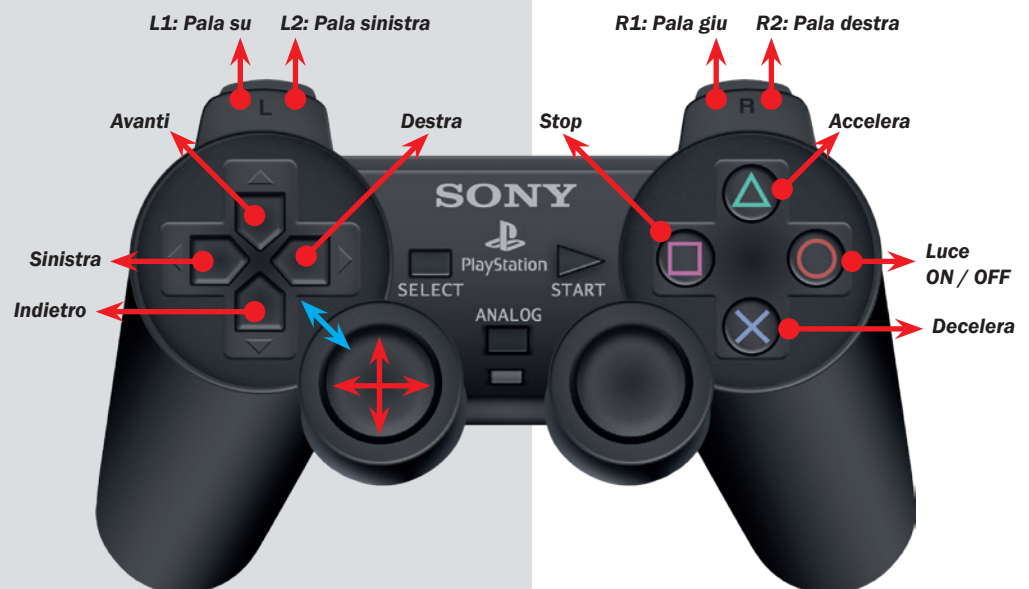


Fig. 14



fissare le staffe (cod. SUPPLACT) specifiche dell'attuatore lineare ai due attuatori laterali (sulla parte posteriore). La staffa dell'attuatore deve essere fissata alla coppia di "L" in alluminio (quella con interposta la rondella), precedentemente fissate sulla parte frontale bassa del telaio (Fig. 15). Ora si può posizionare la parte del braccio dell'attuatore vicino al triangolo della pala, fissare una piccola "L"



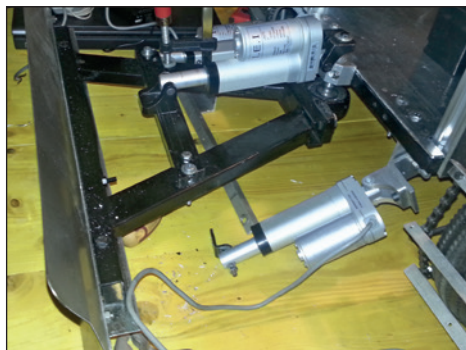


Fig. 15

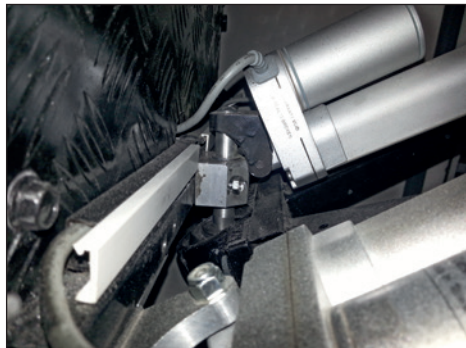


Fig. 16

(servirà come ancoraggio) e prendere bene le misure prima di forare definitivamente. Non rimane che posizionare anche l'attuatore centrale, fissando la staffa sul perno centrale del cuscinetto e fissando poi la staffa (dopo aver inserito la staffa specifica anche sulla parte del pistone) sulla zona centrale del triangolo della pala (Fig. 16). La traversa a incastro va fissata sulla barra orizzontale del triangolo della pala (Fig. 17), forata e fissata. La

pala in acciaio, opportunamente piegata, va posizionata sulla traversa in modo da segnare i punti da forare mostrati in Fig. 11.

L'elettronica

La parte elettrica ed elettronica dello spazzaneve è basata su Arduino Uno, che ospita tre shield e si interfaccia con queste schede:

- il driver per motori DRI0018;
- il circuito a relé RELAY1CH;
- il DC/DC converter elevatore STEPUP30V.

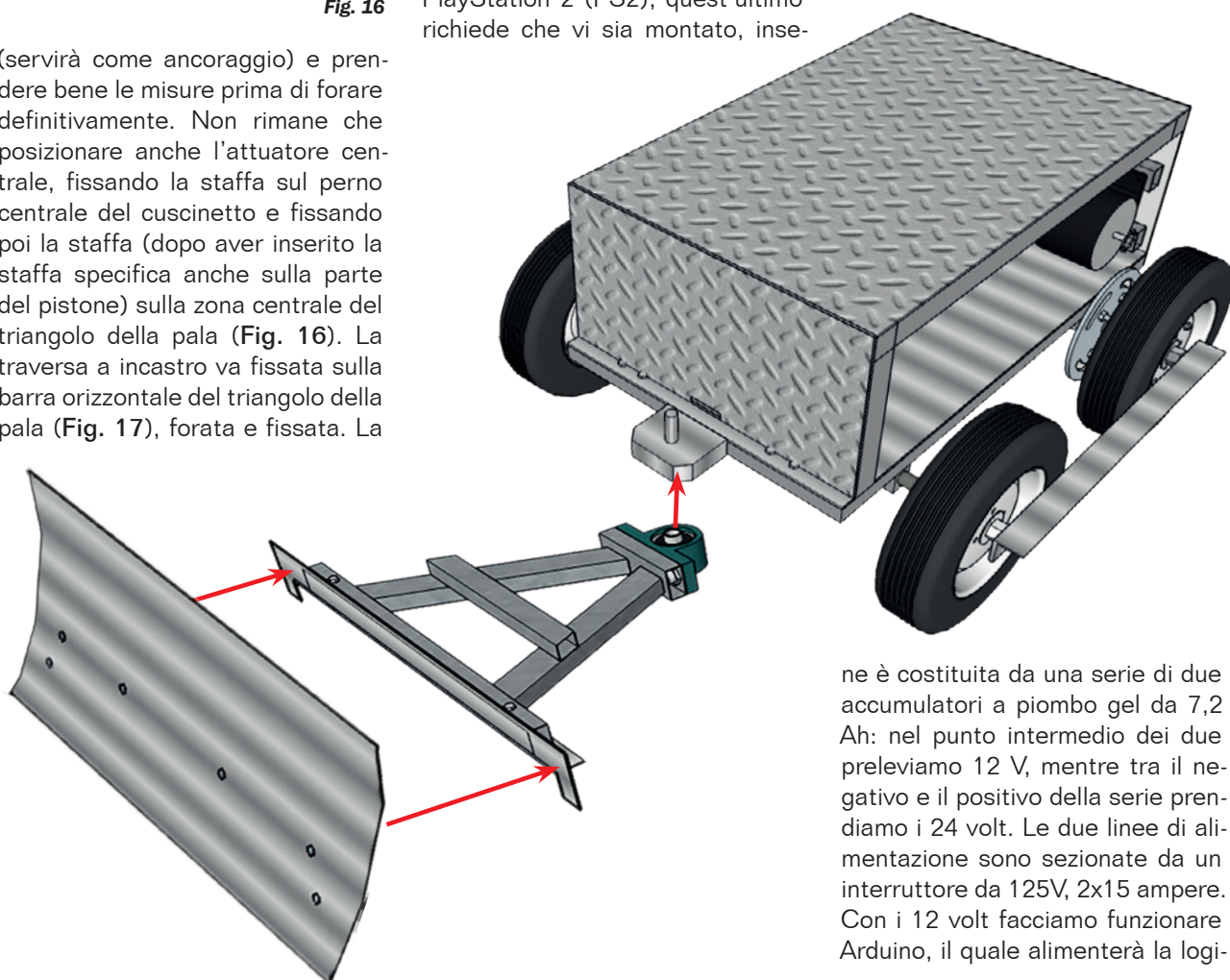
Gli shield montati su Arduino sono il motor shield (cod. VMA03) per controllare i tre motori lineari che gestiscono il movimento della pala, e il PS2SHIELD, che permette di interfacciarsi con la console della PlayStation 2 (PS2); quest'ultimo richiede che vi sia montato, inse-



Fig. 17

rendolo 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 quello che trovate in queste pagine; la fonte di alimentazio-



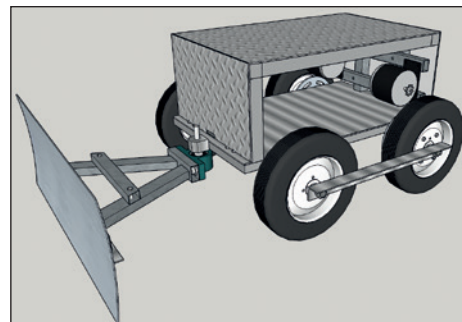
ne è costituita da una serie di due accumulatori a piombo gel da 7,2 Ah: nel punto intermedio dei due preleviamo 12 V, mentre tra il negativo e il positivo della serie prendiamo i 24 volt. Le due linee di alimentazione sono sezionate da un interruttore da 125V, 2x15 ampere. Con i 12 volt facciamo funzionare Arduino, il quale alimenterà la logi-

ca 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 (cod. VMA03); infatti, visto l'assorbimento dei tre attuatori lineari (cod. LACT2), è sconsigliabile prendere i 12 V da Arduino: bisogna optare per l'alimentazione esterna, da fornire all'apposita morsettiera PWR. Per prendere l'alimentazione esterna, bisogna spostare il jumper dello shield vicino alla morsettiera, tra il centrale e l'EXT. La gestione dello shield richiede un'apposita libreria per Arduino, fornita a corredo. 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. È in grado

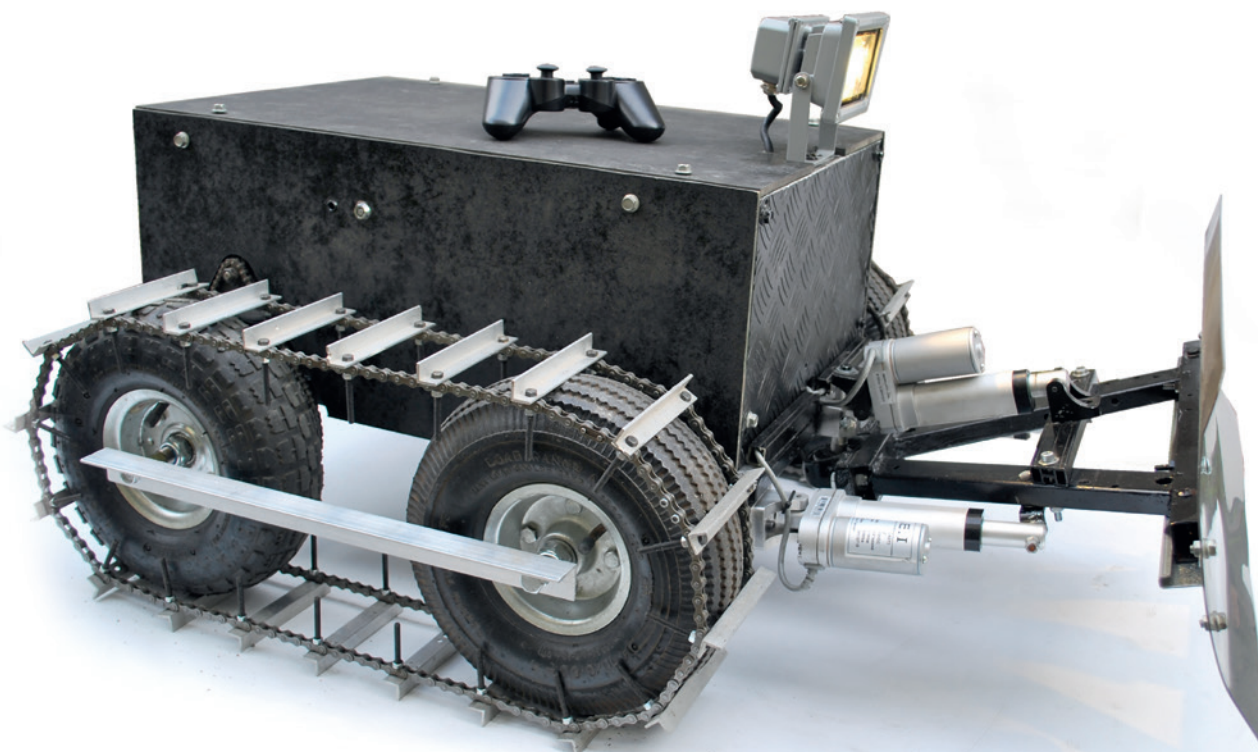
di sostenere fino a circa 250 kg, quando non si muove, e la coppia a vite assicura il mantenimento della posizione dell'albero anche in assenza di alimentazione. Due interruttori di limitazione garantiscono l'arresto del motore quando arriva alla massima estensione e contrazione, mentre i diodi permettono 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 (IP63 nominale).

Passiamo adesso alla linea di alimentazione a 24 volt, con cui viene fatta funzionare la parte 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 (cod. DRI0018), da collegare con quattro fili ad altrettante linee digitali di Arduino.

Il DRI0018 è un doppio driver di potenza a ponte in grado di gestire due motori brushed in continua con una tensione di alimentazione



max. di 35 Vcc e una corrente di 15 A (ciascuno). Interfacciandolo ad Arduino e utilizzando solo 4 I/O digitali (2 PWM), è possibile scegliere il verso e la velocità di rotazione dei motori. Il circuito dispone di quattro pin di controllo, quattro LED che indicano la direzione di rotazione dei motori, due pin per l'alimentazione della scheda (5 Vdc) e due dissipatori di calore in alluminio situati nella parte posteriore del circuito stampato, necessari a smaltire il calore sviluppato nel funzionamento alla massima potenza. Per la gestione del driver da parte di Arduino, il costruttore fornisce a corredo un'apposita libreria. Le morsettiere di uscita



I/O digitale Arduino pin DRI0018

4	DIR1
5	PWM1
6	PWM2
7	DIR2

Tabella 1

permettono di collegare i due motori, da connettere secondo l'indicazione dello schema (non scambiate la polarità, altrimenti i cingoli gireranno al contrario!) utilizzando cavi di diametro pari almeno a 1,5 mmq.

Quanto agli ingressi di comando, va rispettata la connessione tra Arduino e il connettore a passo 2,54 – 2 file del controller indicata nella **Tabella 1**; la corrispondenza vale a patto di usare la libreria fornita dal costruttore e il nostro firmware: se li modificate, dovete rivedere di conseguenza le connessioni. Ricordate che i DIR sono i livelli logici che impongono la direzione di rotazione dei motori e i PWM i segnali PWM da fornire ai driver di potenza.

Per identificare i contatti del connettore del controller è sufficiente riferirsi alla serigrafia visibile accanto ad esso sul lato componenti dello stampato.

Oltre al controller dei motori, i 24 volt vanno ad alimentare la sezione del proiettore a LED, che potete montare o meno; se la montate, rispettate il cablaggio indicato e ricordate di far passare il positivo dei 24 V dalla scheda a relé (cod. RELAY1CH). Quest'ultima ospita un relé e il rispettivo transistor di comando, controllato da Arduino mediante il Digital I/O 13; Arduino fornisce anche l'alimentazione alla scheda, mediante il 5V e la massa (GND).

La scheda a relé permette di accendere e spegnere il proiettore, intervenendo sull'alimentazione del circuito corrispondente.

Il proiettore è progettato per funzionare a 220 Vac, tuttavia siccome questa tensione non è disponi-

bile 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 del converter DC/DC (cod. STEPUP30V); 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. Di solito ciò si ottiene a circa 30 volt. Comunque ricordate che la potenza è data dal prodotto $V \times I$, quindi corrente per tensione (si ottiene in watt se V è espressa in volt e I in ampere), quindi a 30 V, la corrente assorbita dev'essere poco sotto i 340 mA. Sebbene il gruppo di LED sia internamente protetto da sovracorrente, considerato che lo STEPUP30V ha una limitazione della corrente di uscita a 2A, può essere utile sostituirlo con un DC/DC elevatore dotato di regolazione della corrente di uscita; un valido esempio è reperibile sul sito www.futurashop.it ed è codificato come STEPUP30VADJ.

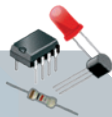
Della ricezione dei dati provenienti dal telecomando si occupa lo shield PS2SHIELD, per il quale il costruttore rende disponibile la specifica libreria; nel **Listato 1** potete vedere un esempio di sketch per implementare la ricezione sulla seriale di Arduino dei comandi trasmessi dai pulsanti della console PS2. Lo shield effettua la ricezione e la decodifica dei comandi; la parte radio è staccata e corrisponde all'RX_PS2, che va inserito nell'apposito connettore.

L'ultima cosa da notare è la fotoresistenza, impiegata per far rilevare ad Arduino l'illuminazione dell'ambiente e comandare la scheda RELAY1CH in modo che il proiettore a LED venga acceso; il componente è alimentato a partire di tensione da un resistore da 10 kohm (1/4 di watt) e viene letto

dall'input analogico A0 di Arduino. I 5 volt vengono prelevati dal 5V di Arduino.

Il funzionamento dell'insieme è semplice: in condizioni di buona illuminazione, la resistenza del fotoresistore è bassa e la tensione letta dall'ADC del microcontrollore di Arduino è bassa e non fa scattare il comando della scheda relé; al buio la resistenza incrementa molto e l'ADC legge una tensione prossima ai 5 volt, che causa l'accensione del proiettore.

Per l'hardware è tutto: nella prossima puntata spiegheremo lo sketch per Arduino, che comunque si può già scaricare dal nostro sito www.elettronica.in. □



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.

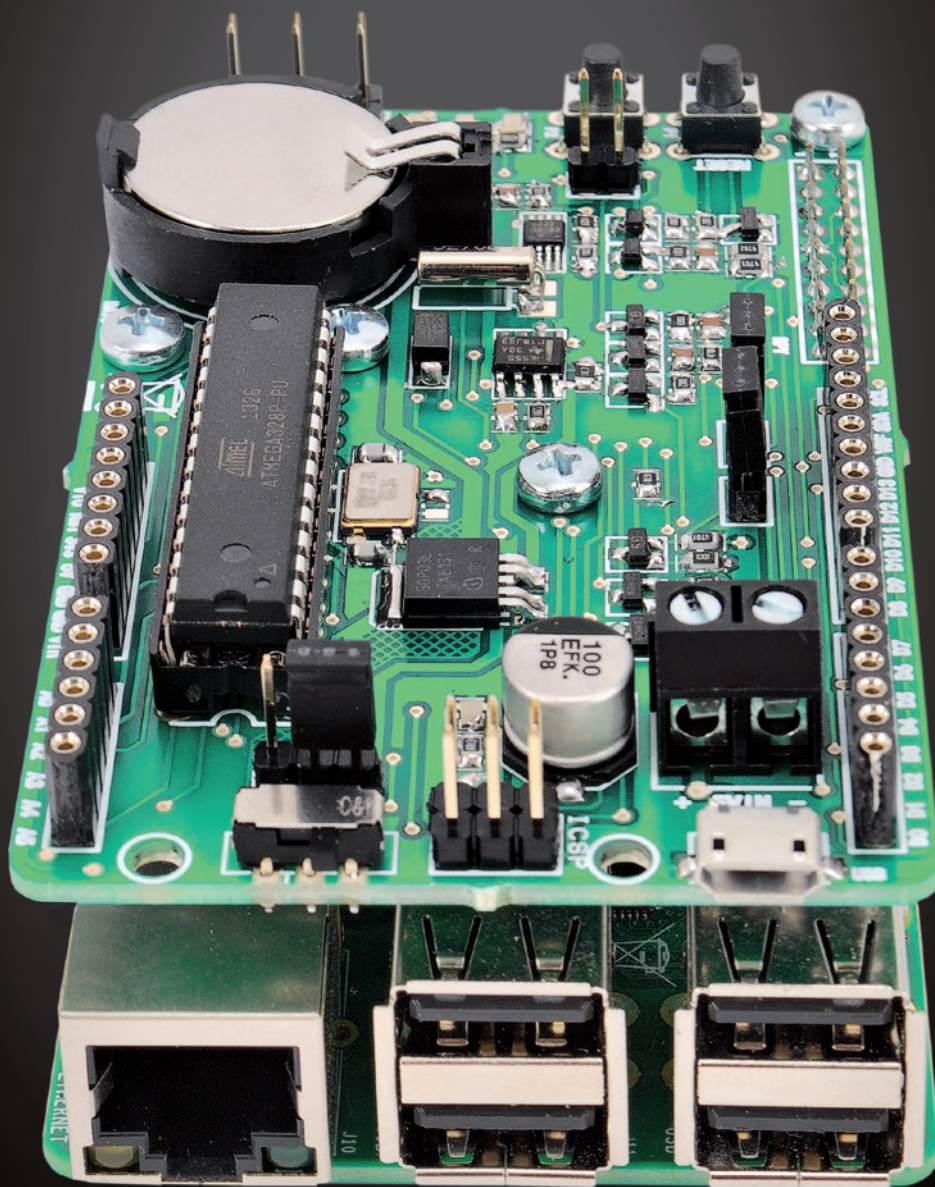
Il materiale va richiesto a:
Futura Elettronica, Via Adige 11, 21013
Gallarate (VA) • Tel: 0331-799775
Fax: 0331-792287 - www.futurashop.it

per il MATERIALE

IMPARIAMO A USARE RANDA

È giunto il momento di utilizzare la scheda che coniuga le potenzialità di Raspberry Pi alla semplicità d'uso di Arduino.

di DANIELE DENARO



I mondi di Raspberry Pi e Arduino si sono toccati, ma senza collisione: anzi, ne è scaturita una cooperazione che ha dato vita a un ibrido dalle enormi potenzialità. Ne abbiamo iniziato a parlare nel fascicolo di novembre, presentando la scheda RandA (questo nome significa **Raspberry and Arduino**) ovvero una board di prototipazione dotata di un "core" Arduino, però provvista sia dei classici strip femmina per ospitare gli shield di Arduino, sia delle connessioni per essere interfacciata al connettore di espansione di Raspberry Pi (anche della versione ultima nata del "lampone"). Nell'articolo del fascicolo n° 190 abbiamo visto le caratteristiche principali di RandA ed ora ne approfondiamo alcuni aspetti con esempi concreti di utilizzo. Diamo per scontata l'avvenuta installazione del software a corredo di RandA, che comprende l'IDE Arduino modificato da utilizzare nel proprio computer e di tutto il software previsto su Raspberry Pi; per l'installazione del caso, rimaniamo all'articolo introduttivo e ai file README presenti nella distribuzione. Ricordiamo che chi fosse interessato, potrà acquistare la scheda SD per Raspberry Pi con già l'installazione pronta.

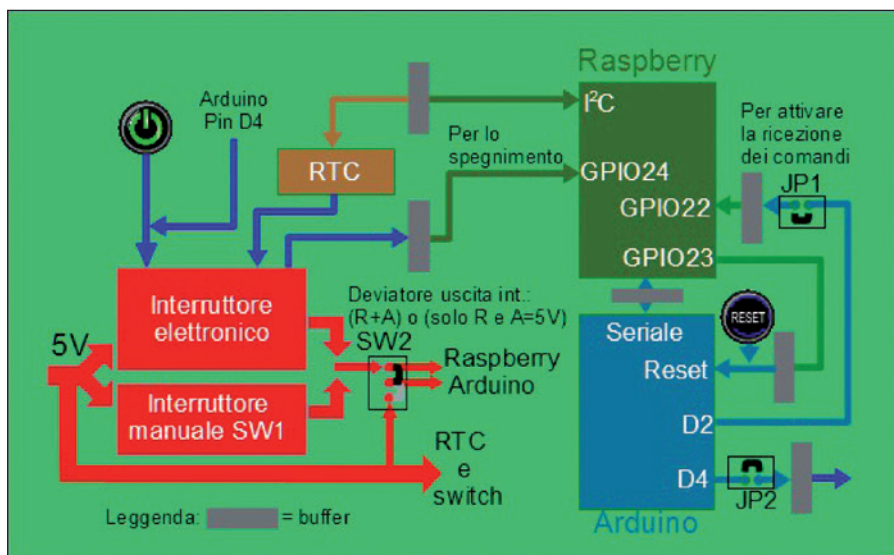


Fig. 1 - Hardware di RandA.

In **Fig. 1** riassumiamo lo schema di RandA e degli switch. Come abbiamo visto, la programmazione e l'utilizzo della scheda RandA prevede due tipi di approccio: uno Arduino-centrico, dove è Arduino a "guidare le danze", ed un approccio Raspberry-centrico, dove invece l'ambiente di riferimento è il sistema operativo Linux.

APPROCCIO ARDUINO-CENTRICO

Partiamo dall'approccio Arduino-centrico e cominciamo a sviluppare uno sketch sul nostro computer utilizzando l'IDE Arduino modificato. L'IDE risulta perfettamente compatibile con l'utilizzo tradizionale di una scheda Arduino collegata localmente sulla porta USB del nostro computer, quindi, per chi ha già utilizzato questo ambiente di programmazione non c'è alcunché da aggiungere.

La sorpresa arriva al momento di selezionare, tra quelle disponibili, la porta cui Arduino risulta collegato. Insieme ad eventuali porte locali (seriali e COM virtuali su USB), nel menu appare una nuova porta descritta da un nome complesso che contiene un indirizzo di rete. Per esempio `//192.168.10/Arduino`. Questo, ovviamente se il sistema RandA è acceso ed è

collegato in rete, tramite la porta ethernet, disponibile su Raspberry, oppure tramite una eventuale chiavetta WiFi, anche essa collegata a Raspberry Pi.

In realtà si potrebbero avere diverse porte in elenco, se ci fossero più sistemi RandA collegati in rete, per cui potremmo anche caricare sketch su sistemi diversi.

Inoltre, se non avessimo assegnato stabilmente sul router, un indirizzo di rete locale per RandA, potremmo utilizzare questo IDE modificato per rilevare l'indirizzo assegnato dal router; per poi collegarci con un client SSH come MobaXterm o via browser al server WEB installato su RandA. Per fare una prima prova, possiamo utilizzare un nostro sketch scritto per una Arduino standard, oppure uno degli esempi elencati

nell'IDE. Una volta selezionata la porta remota ed effettuato l'uploading, lo sketch sarà operativo come al solito. Per esempio, se avessimo utilizzato lo sketch di blinking, potremmo vedere il LED sulla scheda RandA lampeggiare. Ricordiamo, infatti, che la scheda RandA è completamente compatibile con Arduino Uno.

Ma se volessimo caricare uno sketch che utilizza la porta seriale? Niente paura, attivando la console dell'IDE, come usuale, vedremo i dati arrivare (o partire) come se Arduino fosse collegato localmente. Infatti, in RandA la porta seriale è stata completamente remota; quindi possiamo interagire con Arduino completamente. Fino a qui abbiamo verificato che, senza dover utilizzare appositi shield, Arduino si trova collegato in rete perchè si appoggia sulle caratteristiche di networking di Raspberry Pi. Anzi, è possibile fare l'uploading di sketch, diversamente dall'uso degli shield di rete. Anche se questo è già un buon risultato, è solo una piccola parte delle potenzialità del sistema RandA.

Sulla scheda RandA possiamo collegare uno dei numerosi shield disponibili per Arduino, per gestire o ampliare le connessioni di I/O: espansori di I/O, sistemi a relè, sistemi opto-isolati, gestione di motori ecc.

Ma si può fare molto di più e

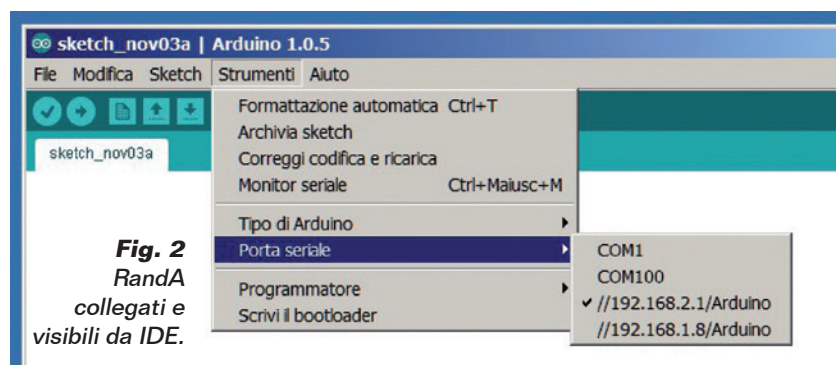


Fig. 2
RandA
collegati e
visibili da IDE.

superare, in parte, le limitazioni hardware di Arduino. Infatti si può pensare di utilizzare Raspberry Pi per fargli eseguire dei compiti. Per fare ciò è necessario che Arduino possa lanciare dei comandi su Raspberry Pi ed eventualmente leggere la risposta di questi.

Per tale scopo è stata predisposta una libreria Arduino inclusa nell'IDE modificato. Questa libreria (RAComm) prevede la possibilità di lanciare dei comandi su Raspberry e riceverne la risposta. Ma permette anche di aprire un file per scriverci o leggerci. Poiché, mentre dialoga con Raspberry Pi, Arduino impegna la porta seriale, questa non è disponibile per il dialogo con l'IDE. Per cui non è possibile utilizzare la console dell'IDE. Per questo motivo è stata predisposta la possibilità di aprire una console su Raspberry Pi per scrivere messaggi, per leggere input o semplicemente per il debugging. Però questa console, realizzata in modalità X terminal, ha bisogno di un collegamento con Raspberry (in rete con protocollo SSH oppure diretto da terminale) per essere visibile.

Vediamo allora un esempio pratico di questa sinergia tra Arduino e Raspberry Pi: usiamo un utile comando Linux che permette di fare una richiesta (GET) di una risorsa su rete, il comando "curl". Proviamo a utilizzarlo per richiedere al sito "openweathermap.org" dati atmosferici relativi ad una località. Seguendo le indicazioni del sito, sappiamo che la richiesta va fatta con il formato visibile in Fig. 3, dove compare anche la sua risposta. Nell'esempio, abbiamo utilizzato **Roma**, però la richiesta può essere fatta per qualunque località mondiale.

In modalità base, i dati vengono forniti in formato JSON; ma si possono avere anche in formato

```
curl -i http://api.openweathermap.org/data/2.5/weather?q=Roma,it
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 02 Nov 2014 14:56:22 GMT
Content-Type: application/json; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Source: redis
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: GET, POST

{"coord":{"lon":12.48,"lat":41.89},"sys":{"type":1,"id":5851,"message":0.1529,"country":"IT","sunrise":1414907053,"sunset":1414944182},"weather":[{"id":800,"main":"Clear","description":"Sky is Clear","icon":"01d"}],"base":"cmc stations","main":{"temp":292.84,"pressure":1021,"humidity":63,"temp_min":292.15,"temp_max":293.15},"wind":{"speed":3.6,"deg":280},"clouds":{"all":0},"dt":1414938000,"id":3169070,"name":"Roma","cod":200}
```

Comando Linux e risposta.

N.B. La risposta è stata troncata in record di 64 caratteri massimo, dal codice dello sketch, ma in realtà i dati in formato JSON sono senza ritorno a capo.

Fig. 3 - Comando curl in esecuzione.

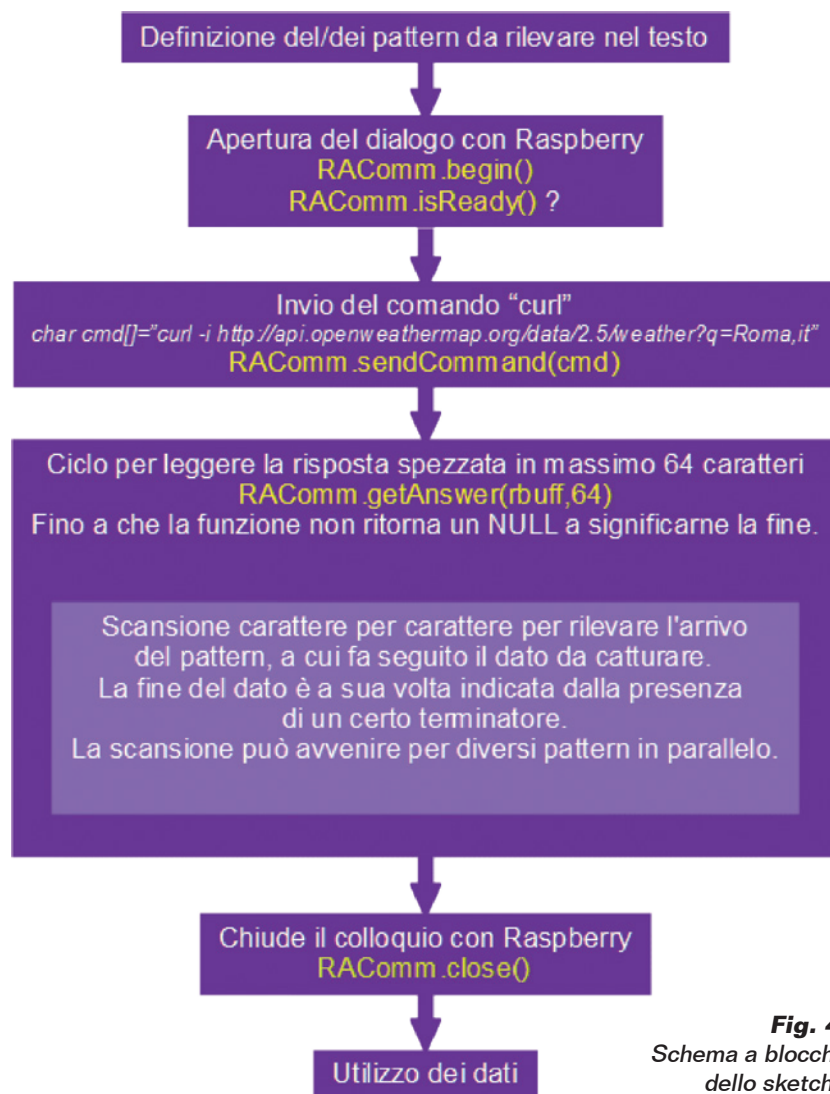


Fig. 4
Schema a blocchi
dello sketch.

Definizioni generali

```
#include "RAComm.h"      // Use Raspberry command library

#define CITY "Roma,it"

// Raspberry command to send a request to a URL
#define CMD1 "curl -i http://api.openweathermap.org/data/2.5/weather?q=" CITY

#define DEB true         // debug flag
#define LENBUFF 64       // buffer length for answer reading

RAComm Cmd;              // Cmd is the RAComm class

char rbuff[LENBUFF];     // buffer for Raspberry command answer

int flagStatus=0;        // signaling status using standard Arduino led
int led;
```

Funzioni setup e loop

```
void setup()
{
    initLed();             // standard Arduino led used as flag

    createPattArray(5);    // define an array of patterns
    setPattern(0,"temp\":"",',',10); // define each pattern
    setPattern(1,"temp_min\":"",',',10);
    setPattern(2,"temp_max\":"",',',10);
    setPattern(3,"humidity\":"",',',10);
    setPattern(4,"pressure\":"",',',10);

    getWeather();
}

void loop() // just signaling status (blinking: n times with millisecc of period)
{
    switch (flagStatus) // 1: no dialog with Raspberry ; 2: error in dialog
    {
        case 1 : blinking(50,100);break;
        case 2 : blinking(10,500);break;
    }
    delay(2000);
}
```

Fig. 5a - Corpo dello sketch.

Funzione principale

```
int getWeather()
{
    boolean ok;
    Cmd.begin();           // Raspberry command dialog initialization
    delay(500);            // delay (just for tidiness)
    for (int i=0;i<20;i++) // Max 20 checks. If not ready something is wrong!
    {
        ok=Cmd.isReady();if (ok) break;Cmd.begin();delay(100);
    }
    if (!ok) (noListen());return -1; // if no dialog no action

    Cmd.openFileWrite("/home/pi/wheather.log"); // open file to record data
    ok=Cmd.sendCommand(CMD1); // send command to get wheter
    if (!ok) (error());return -1;
    while (Cmd.getAnswer(rbuff,LENBUFF)!=NULL) // read answer 64 bytes at time
    {
        if (DEB){Cmd.writeRec(rbuff);} // if DEB write all answer on the file
        checkPattern(pattarray.ptn,rbuff); // check every patterns and collect data
    }
    // Write data on file: temperature,tem. min.,temp. max.,humidity,pressure
    char rec[30];
    float v=0;
    char sval[10];
    // read temperature as float and get new string
    // because Atmel compiler doesn't use %f format
    v=getPattValN(0);v=v-273.15;dtostrf(v,5,1,sval);
    sprintf(rec,30,"Temperature: %s C",sval);Cmd.writeRec(rec);
    v=getPattValN(1);v=v-273.15;dtostrf(v,5,1,sval);
    sprintf(rec,30,"Temp-min : %s C",sval);Cmd.writeRec(rec);
    v=getPattValN(2);v=v-273.15;dtostrf(v,5,1,sval);
    sprintf(rec,30,"Temp-max : %s C",sval);Cmd.writeRec(rec);
    sprintf(rec,30,"Humidity : %s%%",getPattVal(3));Cmd.writeRec(rec);
    sprintf(rec,30,"Pressure : %s P",getPattVal(4));Cmd.writeRec(rec);
    Cmd.close();           // close dialog
}
```

Fig. 5b - Funzione getWeather.

XML. Lo sketch, che è fornito come esempio nella libreria, si chiama "ReadWeb" e sarebbe formato da poche righe di codice se non fosse per la parte che riguarda l'estrazione dei dati dal

verboso testo restituito dal sito. Infatti, oltre al codice del protocollo HTTP, anche i dati forniti sono numerosi. A questo punto dobbiamo tenere presente la limitazione di memoria di Arduino.

Infatti non è pensabile memorizzare tutta la risposta in un buffer; inoltre il buffer della seriale è di 64 byte e non è opportuno leggere maggiori dati per evitare la perdita degli stessi. Per cui dobbiamo richiedere la risposta a piccoli pezzi e scandire carattere per carattere per rilevare l'inizio del dato che vogliamo estrarre. In Fig. 4 è rappresentato uno schema a blocchi dello sketch.

In Fig. 5a e Fig. 5b sono mostrate le parti essenziali dello sketch. Come si vede, la richiesta è fatta "una tantum", e per ripeterla bisogna resettare Arduino; infatti questo sketch è dimostrativo, mentre per un reale utilizzo, probabilmente, sarebbe necessario ripetere la richiesta ad intervalli regolari. Per fare ciò basta aggiungere un ulteriore comando che regoli la sveglia e poi un comando di "shutdown". Ovvero, per esempio, per provocare la lettura dei dati meteorologici ogni ora basta inserire:

- RAComm.
 sendCommand("SetRestartAt -sd 00")
- RAComm.sendCommand("sudo shutdown -h now")

Il primo regola la sveglia allo scade dell'ora, mentre il secondo provoca lo spegnimento del sistema. Ricordiamo che in RandA lo "shutdown -h" provoca l'interruzione dell'alimentazione alla fine del processo di chiusura.

Inoltre lo sketch scrive su un file tutta la risposta, realizzando una sorta di log. Per fare ciò si utilizzano funzioni di libreria che permettono di aprire un file (uno solo alla volta!) per scriverci o per leggerci.

I comandi utilizzati nell'esempio sono:

- RAComm.openFileWrite("/home/pi/wheather.log")
- RAComm.writeRec(rbuff)

Il file viene chiuso automaticamente dal comando di chiusura del colloquio con Raspberry (anche se, a stretto rigore, esiste un comando specializzato).

I dati estratti dalla risposta sono visibili in **Fig. 6**. Per la temperatura è stata fatta una conversione da temperatura assoluta a temperatura in gradi centigradi (T assoluta -273,15).

I dati sono salvati sul file, ma, per realizzare un'applicazione pratica, si potrebbero usare, ad esempio, per attivare o regolare un sistema di riscaldamento o raffreddamento tramite le uscite digitali o analogiche di Arduino.

Come si è detto, la parte più consistente dello sketch è quella relativa a una problematica che non riguarda l'interazione Arduino-Raspberry, quanto piuttosto il superamento delle limitazioni della RAM di Arduino. La risposta viene spezzata in record di massimo 64 caratteri, ma il pattern da rilevare (per esempio "temp":) può finire spezzato su due righe. Anche se la problematica non riguarda direttamente questo articolo, diamo qui una breve descrizione dell'algoritmo adottato.

La problematica è stata generalizzata e quindi è possibile utilizzarla in altre occasioni per leggere, per esempio, un testo HTML in cerca di pattern particolari, in una logica: chiave-valore.

Innanzitutto vengono definite una struttura "pattern" ed un array di queste. Ogni elemento dell'array contiene il pattern da ricercare che funziona come start del dato da

```
Temperature: 19.7 C
Temp-min    : 19.0 C
Temp-max    : 20.0 C
Humidity     : 63%
Pressure     : 1021 P
```

Fig. 6 - Dati metereologici estratti.

leggere, ed un carattere che indica la fine di questo. Inoltre sono definite alcune variabili usate dall'algoritmo. L'algoritmo funziona confrontando in parallelo tutti i pattern definiti, scandendo i singoli caratteri. Quando trova un pattern, inizia a leggere il dato in modo indipendente dagli altri pattern. I dati vengono memorizzati (come stringhe) in un buffer della struttura corrispondente.

Prima di passare ad un altro esempio di utilizzo della libreria RAComm, è il caso di chiarire che in *RAComm.sendCommand()* è possibile inserire tutti i comandi standard delle distribuzioni Linux (Raspbian), i programmi che abbiamo predisposto per RandA e qualunque script o eseguibile che voi implementiate.

Abbiamo già visto l'utilizzo di un comando predisposto per RandA: *SetRestartAt* (è un eseguibile C); ora è il caso di elencare brevemente tutti gli altri comandi, che potete vedere nella **Tabella 1**, ma che è altresì possibile visualizzare anche da console Raspberry Pi digitando il comando "commands". Aggiungendo -h vengono visualizzati anche tutti gli help. In verità alcuni di questi, come *ArduLoad*, *ArduIO* e *ArduInterrupt*, sono pensati per un utilizzo Raspberry-centrico, in quanto forniscono un accesso alle funzionalità Arduino da parte di script Linux.

Il secondo esempio che vedremo riguarda la spedizione di un'e-mail: supponiamo di voler rilevare la presenza di un segnale importante ed avisare di conseguenza qualcuno dell'evento. La parte relativa alla rilevazione della condizione è tipicamente un compito per Arduino. Per esempio, potremmo rilevare il superamento di una soglia analogica o una condizione ON/OFF; però spedire una e-mail con Arduino è un compito difficile, soprattutto a

Tabella 1

Comando	Significato
ArduIO	Definisce e scrive o legge i pin di Arduino
ArduLoad	Carica uno sketch compilato (.hex) salvato su Raspberry
ArduInterrupt	Si blocca fino ad una condizione su pin di Arduino
ResetRandA	Reset di Arduino
GetRTC	Legge l'orologio di RandA
SetRTC	Regola l'orologio di RandA
SetRestartAt	Definisce la sveglia per la riaccensione
ResetAlarm	Elimina la sveglia eventualmente attivata
SetSysClock	Aggiusta l'orologio di sistema utilizzando lo RTC
SendMail	Spedisce una e-mail

causa dei server SMTP che ormai utilizzano i protocolli protetti, la cui implementazione con Arduino è impossibile.

Il comando da utilizzare potrebbe essere, per esempio:

```
RAComm.sendCommand("
SendMail mailto=\"pippo@pluto.it\"
subject=\"Rilevazione\"
filemess=\" /home/pi/messaggio.txt\"
")
```

Il comando è stato spezzato su più righe, esclusivamente per comodità di stampa. Il pattern \" sta ad indicare la presenza di doppi apici all'interno di una stringa (il comando da passare come argomento alla funzione). Il testo del messaggio è automaticamente formato da "SendMail" prendendolo dal file in cui è stato appoggiato. Il comando prevede la possibilità di spedire anche semplicemente un testo corto, invece del testo appoggiato su un file, in questo modo:

```
RAComm.sendCommand("
SendMail mailto=\"pippo@pluto.it\"
subject=\"Rilevazione\"
message=\"Valore : 320\" ")
```

È anche possibile aggiungere un

Definizioni

```
#include "RAComm.h"      // Use Raspberry command library

#define analog 4          // analogic pin used
#define LOGFILE "/home/pi/values.dat" // log file name
#define TEXTMAIL "/home/pi/mailval.txt" // file containing message text

#define Q "\""             // escape character for quote(")
#define SENDTO "pippo@pluto.it" // receiver (use yours)
#define SUBJECT "Value"    // subject

//Commands:

// Raspberry command to get timestamp
#define CMD1 "GetRTC -s"

// Command that uses "tail" linux command to extract last 5 lines from log whereby making text to send
#define CMD2 "tail -n 5 " LOGFILE " " > " TEXTMAIL

// Raspberry command "SendMail" text (NB. text in quote has to be inserted using escape char)
#define CMD3 "SendMail mailto=" SENDTO " subject=" Q SUBJECT Q " filemess=" Q TEXTMAIL Q

// Raspberry command for alarm setting
#define CMD4 "SetRestartAt -sd %d %d" //used in sprintf to insert variables

// Shutdown command
#define CMD5 "sudo shutdown -h now"

RAComm Cmd;              // Cmd is the RAComm class
```

Fig. 7 - Sketch SendMail (definizioni principali).

allegato, oppure avere più destinatari. Ma per maggiori dettagli si rimanda all'help del comando. A questo punto è il caso, però, di precisare che occorre inizializzare il file `"/home/pi/bin/Mail.properties"` con i propri dati; cioè con l'utente e la password riconosciute dal server SMTP utilizzato e l'indirizzo del server stesso. Inoltre c'è da tenere presente che chi non è abituato ad utilizzare un client di posta elettronica (come Outlook o Thunderbird) potrebbe trovare il server di Gmail non abilitato a ricevere comandi di spedizione; in tal caso bisogna provvedere alle impostazioni del caso sul sito di Gmail. Nell'esempio, incluso nella libreria, chiamato `"SendMail"`, viene rilevato a scadenze periodiche un valore analogico, il quale, insieme al timestamp prelevato dall'RTC, viene memorizzato su un file e quindi inviato ad un indirizzo e-mail. Notate che non viene inviato solo l'ultimo dato letto: si inviano per e-mail gli ultimi 5 record. Per fare ciò viene utilizzato il comando Linux `"tail"` che estrae gli

ultimi `"n"` record da un file.

In Fig. 7 sono visualizzate solo le principali definizioni presenti nello sketch.

Con questi esempi mostrati, la potenzialità della libreria di collegamento con Raspberry Pi dovrebbe risultare evidente. Inoltre tenete presente che, in caso di errore logico (per esempio sketch che provoca immediatamente lo shutdown alla partenza del sistema), aprendo il ponticello JP1 è possibile disabilitare il colloquio con Raspberry Pi; JP1 ha lo scopo di segnalare a Raspberry Pi l'intenzione di dialogare.

Lo script che si mette in ascolto del segnale viene lanciato allo startup (nel file `/etc/rc.local`) e si chiama `StartListenCmd`; può essere anche lanciato con un file di log. Più semplicemente, in caso di gravi problemi di debug si può lanciare direttamente il programma C che dialoga e che si chiama `ExecSCmd` (dopo aver disabilitato `StartListenCmd`). `ExecSCmd` può essere lanciato con il flag di debug `-s`, con il quale fa l'eco sulla console di tutto ciò che passa sulla seriale.

Il sorgente di `ExecSCmd` si trova, insieme agli altri sorgenti dei programmi per RandA, in:

`"/home/pi/workspace/cworkspace"`

C'è ancora una ulteriore possibilità da sfruttare nell'approccio Arduino-centrico: si può tenere Arduino sempre alimentato, agendo sul ponticello deviatore SW2 che lo separa dalla gestione dello switch elettronico. Arduino, privato del suo alimentatore lineare e dell'USB, consuma pochissimo e si può mettere anche da solo in condizione di bassissimo consumo. In questa condizione, Arduino può accendere Raspberry Pi dando un impulso al suo pin D4 (sempre che il corrispondente ponticello JP2 sia chiuso). Un ulteriore esempio contenuto nella libreria fa vedere questo utilizzo, e accende Raspberry Pi se sul suo ingresso analogico A0 è presente un valore superiore a 300.

APPROCCIO RASPBERRY-CENTRICO

Chi ha dimestichezza con il

sistema operativo Linux, troverà sicuramente più semplice utilizzare Raspberry come base di programmazione. Per questo scopo si può utilizzare un ambiente di sviluppo per C potente come "codeblocks" che abbiamo installato su Raspberry Pi, oppure il linguaggio Python per il quale abbiamo installato l'ambiente di sviluppo "idle".

Poiché su Raspberry Pi è installato anche Java, è possibile utilizzare anche questo potentissimo linguaggio. Infatti sono già presenti dei programmi Java utilizzati in RandA, come il comando di SendMail o le servlet presenti nel Web server di cui parleremo più avanti. Non abbiamo, però, installato un ambiente di sviluppo per Java, in quanto non ancora soddisfatti delle disponibilità di IDE gratuiti per un hardware ridotto come Raspberry Pi. Benché installabile dal sito dei pacchetti Debian, Eclipse non è in una versione soddisfacente, a causa di problemi sia di installazione, sia di performance.

I programmi Java completi di sorgenti sono catalogati in:

```
"/home/pi/workspace/jworkspace"
```

I programmi (o gli script) realizzati possono essere anche utilizzati nell'approccio Arduino-centrico, come abbiamo già visto. Oppure possono utilizzare la porta seriale per dialogare con sketch predisposti a gestire la comunicazione (cioè al di fuori della libreria RAComm). In questo modo Arduino diventa una potente periferica programmabile di Raspberry Pi.

Il comando "Arduload", cui aggiungere in nome del file eseguibile (.hex) con il path completo, permette di modificare di volta in volta il comportamento di Arduino, perché "installa" su Arduino

lo sketch utilizzabile dall'applicazione. Questo sketch, in formato file eseguibile, può essere stato creato con l'IDE remoto oppure con l'IDE installato localmente su Raspberry Pi; infatti ambedue le versioni dell'IDE, oltre a fare l'uploading dello sketch, salvano l'eseguibile su Raspberry Pi in due directory distinte:

```
- /home/ArduinoUpload per gli sketch elaborati localmente;  
- /home/RArduinoUploads per gli sketch elaborati in remoto.
```

Questa è un'ulteriore aggiunta all'IDE da noi modificato.

Come esempio di applicazione che tratta Arduino come periferica, basta fare riferimento al comando "ArduIO" già elencato nella **Tabella 1**. Questo programma C lo trovate anche nel "cworkspace", insieme al suo sorgente.

ArduIO vuole essere l'equivalente della utility GPIO (wiringPi) di Raspberry Pi, infatti permette di gestire i pin di Arduino da linea di comando. Ad esempio, per accendere il LED standard di Arduino (pin 13) bisogna:

1. definire come output il pin 13 con *ArduIO -set 13 out*;
2. porre ad 1 il pin 13 con *ArduIO -wrd 13 1*.

Per spegnerlo, si impartisce il comando *ArduIO -wrd 13 0*. È possibile utilizzare le solite modalità previste da Arduino: leggere pin analogici e digitali, attivare/disattivare output digitali e regolare i pin PWM. Digitando "ArduIO -h" si ottiene l'help dettagliato. Più precisamente, in **Tabella 2** sono elencate le varie opzioni.

Il programma utilizza uno sketch (*SerialRasp*) con cui collabora spedendo ad esso i comandi corrispondenti: è quest'ultimo che attiva i pin. Quindi lo sketch deve essere stato caricato su Ardui-

no precedentemente. In realtà ArduIO supera questo problema, caricandolo egli stesso, nel caso non fosse presente su Arduino.

Per consentire ciò è necessario:

- a) che possa identificare lo sketch presente su Arduino;
- b) che possa trovare lo sketch in una posizione prestabilita nel caso non fosse quello stabilito e fosse necessario l'uploading.

Il primo punto si risolve facilmente predisponendo lo sketch in modo che, se interrogato, risponda con il proprio nome; per il secondo punto è stata predisposta una directory chiamata "/home/pi/bin/sketch4cmd" da dove prelevarlo. In **Fig. 8** è evidenziata la collaborazione fra i due ambienti, con particolare riferimento al riconoscimento dello sketch ed ad un suo eventuale uploading automatico.

Ma non è proprio necessario utilizzare linguaggi compilati, come C, per realizzare applicazioni su Raspberry Pi che usino Arduino per l'input/output: si possono utilizzare anche script bash (Linux) o script Python. Basta utilizzare i comandi predisposti per RandA (**Tabella 1**) per avere il compito molto semplificato.

Nell'esempio successivo è mostra-

Tabella 2

Opzione	Significato
-set n x	Set del pin n come x=inp,out,ipl (ipl:input-pullup)
-rda n	Legge pin analogico n
-rdd n	Legge pin digitale n
-wra n v	Set dei pin PWM
-wrd n v	Set dei pin digitali (0/1) (Attenzione al pin 4: togliere il ponticello per utilizzarlo)(Il pin 4 è collegato allo switch ON/OFF)
-pou n v	Pulsazione sul pin n di frequenza v (tone)
-poi n v	Legge durata impulso su pin n (v=0/1 definisce impulso alto o basso)

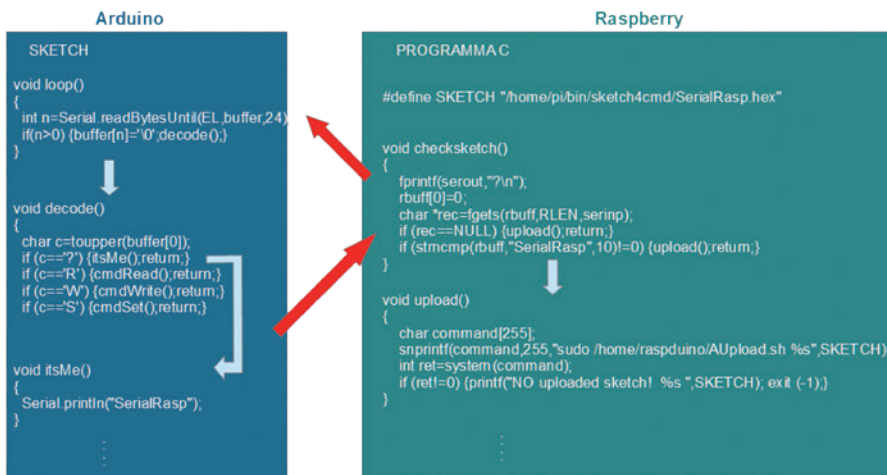


Fig. 8 - SerialRasp collabora con Arduino.

ta una piccolissima applicazione dimostrativa che legge il valore di tensione presente sul pin A1 (per esempio fornito da una fotocellula) ed attiva il LED standard di Arduino nel momento in cui il valore letto è inferiore ad una soglia. In Fig. 9a è mostrata la versione realizzata con uno script bash, mentre in Fig. 9b è visibile

l'equivalente in Python. Naturalmente lo script è per lo più didattico, ma con piccole aggiunte lo si può inserire un ciclo periodico o ci si può inserire un controllo sull'ora oppure l'invio di una e-mail di avviso. Questi due file (*ScriptExample4IO.sh* e *PyExample4IO.py*) sono presenti nella directory */home/*

pi/bin/examples. C'è un ulteriore comando che permette a Raspberry Pi di utilizzare le potenzialità di Arduino, e che può essere utilizzato all'interno degli script: *ArduInterrupt*.

Questo programma utilizza lo sketch "SerialStop", presente in *"/home/pi/bin/sketch4cmd"*, ed in sostanza si blocca fino a che una certa condizione non viene soddisfatta; la condizione può essere su un pin digitale, quando diventa 1 o diventa 0, oppure su un pin analogico quando il valore misurato è maggiore o minore di una certa soglia.

Per esempio, "ArduInterrupt -ana 2 lt 200", si blocca fino a che sul pin analogico 2 di Arduino il valore non scende sotto 200. È evidente che l'uso più opportuno di questo comando è all'interno di uno script dove le istruzioni successive realizzano un'azione. Inoltre lo script sarà verosimilmente lanciato in background.

Ma come sviluppare applicazioni su Raspberry in modo efficiente? Con l'approccio Arduino-centrico abbiamo usato una postazione remota (un qualunque PC collegato in rete locale) per sviluppare sketch che potessero utilizzare anche le potenzialità di Raspberry Pi; ora, rovesciando il punto di vista è ancora possibile utilizzare una postazione remota (sempre il nostro PC collegato in rete locale), questa volta utilizzando una console remota Linux su protocollo SSH ma con la possibilità aggiuntiva di un ambiente grafico. Per questo si può utilizzare un software che permetta di agire anche come terminale X Window (che è l'ambiente grafico di base di Linux). Uno di questi è MobaXterm (gratuito e scaricabile dal sito <http://mobaxterm.mobatek.net/>). In Fig. 10 è mostrata una situazione tipica in cui insieme alla console MobaXterm, sono sovrapposte

```

#!/bin/bash
ArduIO -set 13 out          #set LED
out="$(ArduIO -rda 1)"      #value of A1 in out variable
if [[ $out == NO* ]]       #if can't read exit
then echo "Err!"; exit 0; fi
out=${out//[0-9]/}         #delete any character not digit
echo $out                 #so, in out just a number
if [ "$out" -lt "200" ]    #if < of trheshold
then
  echo "Switch on!"
  ArduIO -wrd 13 1         #LED on
else
  ArduIO -wrd 13 0         #LED off
  echo "Nothing to do!"
fi

```

Fig. 9a - Script bash per l'accensione condizionata.

```

#!/usr/bin/python
import os
import commands
ret=os.system("ArduIO -set 13 out")
out=commands.getoutput("ArduIO -rda 1")
if out[0:1] == "NO": quit()
out=int(out)
print out
if out < 200:
  print "Switch on!"
  ret=os.system("ArduIO -wrd 13 1")
else:
  print "Nothing to do"
  ret=os.system("ArduIO -wrd 13 0")

```

Fig. 9b - Script Python equivalente a quello di Fig. 9a.

una finestra sul system file ed una finestra codeblocks IDE per la programmazione C.

Chi è più abituato ad utilizzare Linux o comunque i moderni sistemi operativi, preferirà questo approccio Raspberry-centrico, perché permette una maggiore flessibilità ed l'efficiente utilizzo delle potenzialità del sistema RandA. Chi invece è più abituato all'ambiente di Arduino, preferirà il precedente approccio.

In realtà è utile anche mischiare i due punti di vista a seconda dell'applicazione. Ma, soprattutto, riteniamo RandA un sistema ideale per l'insegnamento ed il primo approccio all'informatica e ai sistemi digitali, in quanto è possibile coprire, con esso, una incredibile quantità di argomenti e di ambienti operativi.

IL SERVER WEB

La dotazione software non si esaurisce con quello che abbiamo appena descritto; infatti abbiamo pensato di aggiungere anche un Web Server, che permette una gestione basilare del sistema anche in rete geografica. Infatti fino ad ora abbiamo circoscritto l'utilizzo dell'approccio remoto esclusivamente alla rete locale, sia per la tipologia del software di rete utilizzato, sia per motivi di sicurezza. Con un Web Server, è invece possibile superare questo limite, a patto di definire una porta di uscita sul router locale ("port forwarding"), ovvero di avere un indirizzo di rete visibile all'esterno.

Il server software utilizzato non è solo un Web server, ma piuttosto, un "Web Application Server". Questa modalità utilizza un approccio più efficacemente orientato all'interazione. Il server si basa su uno dei più conosciuti software gratuiti: Tomcat. Purtroppo un software di questo genere prevede

Il web nasce come fornitore di pagine riccamente strutturate e multimediali, ma statiche. Non a caso nasce in ambiente accademico, dove è stato pensato come un immenso libro distribuito. Ben presto, però, si è cominciato a cercare di interagire con le pagine immettendo dati e aspettandosi risposte. Per fare ciò è necessario che il server web abbia la possibilità di attivare programmi che ricevendo l'input sappiano costruire la risposta, in formato HTML, che il server possa rispedire al richiedente. I programmi che possono fare questo sono integrati in quello che viene chiamato CGI (Common Gateway Interface). Seguendo questa interfaccia il server WEB deve sapere intercettare la richiesta come richiesta di una elaborazione diversamente dalla richiesta di una pagina. Nel primo caso deve sapere quale programma attivare e passargli la richiesta; i programmi che sono utilizzati per questa funzionalità sono, in genere, interpreti di script.

L'aumento della richiesta di interazione, come per esempio l'utilizzo della tecnologia AJAX che interagisce in background, ha portato ad integrare la capacità elaborativa più internamente al Web Server; da qui il passaggio ai Web Applications Server. Le due differenti tecnologie: dot net e Java, si contendono il campo; la prima in esclusivo ambito Microsoft, mentre l'altra è più universale.

Nel caso di Java, si parla di un'integrazione formata da agili thread chiamati Servlet o di un loro più complesso utilizzo, simile agli script PHP, rappresentato dalle pagine JSP. Con Servlet e JSP, l'interazione è molto più fluida perché non ci sono interi processi da attivare, né script da interpretare (infatti le pagine JSP sono automaticamente compilate in Servlet).

Nel caso di interazione blanda, la complessità della programmazione delle Servlet può essere eccessiva rispetto al compito, ragion per cui, a volte, è conveniente l'uso della tecnologia CGI.

Con la tecnologia CGI, il programma (script) viene attivato chiamando la risorsa con il nome dello script. Questo va collocato con un riferimento preciso nella struttura del sito. Lo script riceve i parametri della richiesta tramite la variabile di ambiente QUERYSTRING ed il suo output verrà restituito dal Web Server al richiedente.

l'utilizzo di Servlet Java (o script JSP) per le applicazioni interattive. Ma la complessità del software utilizzato è compensata dalla notevole efficienza delle applicazioni: basti pensare che tutti i server professionali utilizzano questi tipi di ambiente o il suo concorrente "dot-net" (vedere riquadro in questa pagina).

Ma se non siete esperti di Servlet o JSP, niente paura, perché il server è stato predisposto per gestire anche gli script CGI, ossia script Linux oppure Python; in questo modo potete costruirvi le vostre applicazioni interattive ed utilizzarle da un browser in qualunque posto nel mondo.

Tomcat installato è alla versione 7 ed è nella directory: `"/home/apache-tomcat-7.0.47"`

In questa stessa directory sono presenti due script: `startWebS.sh` e `stopWebS.sh` per far partire e chiudere il web server. Il primo è chiamato all'interno del file di

sistema di startup `"/etc/rc.local"`. Se si vuole disabilitare la partenza automatica del Web Server, basta commentare questo comando nel suddetto file di sistema.

Il server Tomcat è stato messo in ascolto sulla porta 80 (invece che 8080) aggiornando il file: `"/home/apache-tomcat-7.0.47/conf/server.xml"`.

All'interno della directory di Tomcat è compreso tutto il Web Server: dalle applicazioni alle pagine statiche. In particolare, la directory `"/.../webapps/ROOT"` contiene la pagina di partenza `"index.html"`. Invece l'applicazione per la gestione di Raspberry Pi e Arduino è tutta contenuta nella directory `"/.../webapps/RandA"`.

Per chi volesse caricare delle proprie applicazioni o applicazioni di terze parti, la cosa più semplice da fare è utilizzare i file di archivio .war (come usualmente vengono distribuite le applicazioni web Java) e installarli tramite la sem-

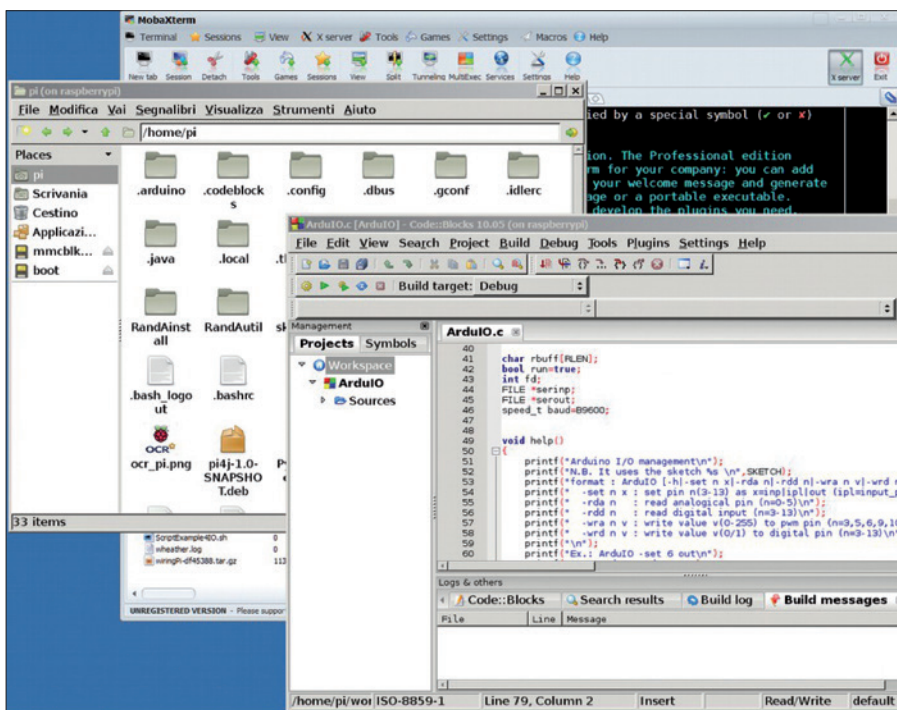


Fig. 10 - Ambiente di gestione Raspberry-centrico.

plice console del manager Tomcat. A tale console si accede dall'indirizzo "http://...../manager" ed è protetta da user name (tomcat è il predefinito) e password (tomcat è la predefinita). Il file .war può risiedere anche sul computer collegato al web server; infatti la consolle permette di fare automaticamente il downloading e l'installazione in un colpo solo. Quindi in sostanza l'installazione di applicazioni è remotizzata. Chi volesse invece utilizzare semplici pagine HTML può collocarle nella directory ROOT, dove saranno immediatamente disponibili. Ovviamente potranno essere create anche sottodirectory di ROOT.

Invece in caso di utilizzo della modalità CGI, gli script bash vanno collocati nella directory:

```
"/home/apache-tomcat-7.0.47/webapps/ROOT/WEB-INF/cgi"
```

Ma, attenzione, vanno referenziati come: <http://...../bin-cgi/nomedel-lloscript>.

Esempio: <http://192.168.1.8/>

bin-cgi/testcgi2.sh. I file "testcgi.sh" e "testcgi2.sh" presenti nella directory/ROOT/WEB-INF/cgi, sono due file di esempio che mostrano l'uso di semplici script bash per il dialogo web ed il lancio di comandi Linux.

Nel caso si voglia usare un diverso interprete di script, bisogna editare il file "/home/apache-tomcat-7.0.47/conf/web.xml" (blocco <servlet> parametro "executable").

La pagina di partenza (per visualizzarla basta inserire nel browser l'indirizzo di RandA), dà accesso all'applicazione di gestione e ad una console web (di fortuna) per l'accesso a Linux anche da rete geografica (applicazione di terze parti). L'applicazione di gestione non è per il momento protetta da password, ma si può facilmente provvedere editando il file "/home/apache-tomcat-7.0.47/webapps/RandA/WEB-INF/web.xml", e togliendo il commento al blocco di autenticazione. Mentre la console web ha l'accoppiata user-name, password: *randa,randa*. L'applicazione di

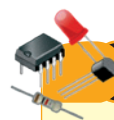
gestione dà accesso alle seguenti pagine/applicazioni:

- caricamento sketch su Arduino;
- console di Arduino;
- gestione I/O Arduino;
- definizione della sveglia;
- aggiustamento dell'orologio;
- spegnimento di RandA.

Le applicazioni sopra elencate permettono di avere un accesso remoto al sistema RandA anche da rete geografica, ma il server è stato installato pensandolo soprattutto come base per la personalizzazione da parte degli utenti, magari ricorrendo a script CGI.

CONCLUSIONI

Abbiamo cercato di fornire un sistema pronto e funzionante, disponibile ad essere utilizzato in applicazioni di vario genere, anche con l'aggiunta di hardware specializzato; il tutto, mantenendo una visione didattica, hobistica e di sviluppo, commentando il più possibile il software prodotto, per facilitare personalizzazioni e miglioramenti. ■



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>

Teoria, laboratori pratici e supporto in una sola parola, anzi due: Futura Academy, il programma formativo che stiamo mettendo a punto per voi che non siete mai stanchi di apprendere e sperimentare dal vivo.

INIZIA LA STAGIONE DELLA FORMAZIONE

a cura della REDAZIONE

Quella di tenere dei corsi è un'idea nata anni fa e che ogni tanto riaffiorava nelle nostre teste, insieme a quell'interminabile serie di visioni dalle quali scaturiscono i nostri progetti. Ora finalmente (superati anche non pochi problemi burocratici) è diventata una realtà, che prende forma in "Futura Academy". Con questo nome, scelto forse con un po' di presunzione, abbiamo battezzato un progetto di formazione a base di Corsi di Elettronica, In-

formatica e Tecnologie Innovative organizzati dalla nostra rivista in collaborazione con Futura Elettronica. Un'iniziativa che rientra a pieno titolo in quella che è la nostra "mission": divulgare e diffondere l'elettronica e delle tecnologie innovative a tutti i livelli e con tutti gli strumenti che il caso richiede. Elettronica In, non è un mistero, è da anni parte di Futura Group, una più ampia realtà che vede la cooperazione tra Futura Elettronica

-azienda impegnata nella progettazione, produzione e commercializzazione di dispositivi elettronici- e Futura Group Edizioni, che vede in prima linea Elettronica In. Questa collaborazione strettissima rende possibili sinergie che permettono di sfruttare l'offerta di Futura Elettronica al servizio della formazione; ciò significa avere a disposizione non solo apparecchiature innovative e utili dal punto di vista didattico, ma anche strumenti di

sviluppo e attrezzature varie. In questo modo, quanti frequenteranno i nostri corsi, non dovranno preoccuparsi di questo aspetto e potranno, nel caso, acquistare al termine del corso, gli stessi strumenti utilizzati per seguire le lezioni, al fine di poterli utilizzare a casa propria per continuare gli esperimenti e approfondire i concetti appresi. Elettronica In, da quasi 20 anni (li compiamo la prossima estate!) propone articoli, progetti, corsi e novità che hanno



completamente nuova area, che inaugureremo a giorni, all'interno della nostra struttura di Gallarate.

Quella dei corsi è un'iniziativa che può contare -cosa non da poco- sull'immediato accesso alle risorse hardware che solo chi ha un'attività a 360° come la nostra può garantire. Non funziona una scheda? Si è rotto un cavo? Serve un saldatore? Basta accedere al magazzino che si trova dietro l'aula per risolvere il problema. Tutto ciò renderà anche molto semplice organizzare gli "hackathon", che abbiamo già in programma per i prossimi mesi.

Per "rendere il piatto ancor più appetibile", vi diciamo anche che siamo già stati contattati da alcune Case di semiconduttori per organizzare dei corsi e dei workshop su loro prodotti; quindi, non solo teoria, ma pratica sui più recenti prodotti delle principali Case. Tra gli scopi di Futura Academy, c'è la collaborazione con le istituzioni scolastiche e gli Enti della zona per fornire quelle competenze che



per i motivi più vari, il "pubblico" non riesce a soddisfare.

Per quanto riguarda la tipologia e il livello dei corsi, prevediamo corsi di base e avanzati. I livelli base saranno dedicati a chi si avvicina per la prima volta al tema trattato o che ha ancora poca dimestichezza con la tecnologia presentata, mentre nei laboratori avanzati si avrà la possibilità di approfondire l'argomento di cui si conoscono già i principi di base, studiando e sviluppando concretamente applicazioni o progetti in tema.

Ai corsi abbiamo deciso di dare un'impostazione tale da renderli accessibili a tutti coloro

che hanno voglia di imparare e che si sentono predisposti all'apprendimento di tecnica e tecnologia.

Per quanto riguarda gli argomenti, alcuni sono già stati definiti e stiamo redigendo il relativo calendario; per gli altri abbiamo chiesto il vostro aiuto e i vostri suggerimenti attraverso un sondaggio sul web. Ai corsi è dedicata un'apposita sezione del nostro sito raggiungibile dall'indirizzo www.elettronica.it/futura_academy/, dove gli interessati potranno trovare tutti gli aggiornamenti sul nostro progetto, comprese le date e il programma formativo.

stimolato e soddisfatto la voglia di imparare di studenti, tecnici, hobbisti (makers...oggi si dice così...) e insegnanti, persone che non è difficile trovare oggi a capo di importanti aziende del settore. Col tempo, alle riviste si sono aggiunti i siti web e i blog. Con Futura Academy inauguriamo un percorso che ci consente di trasferire le nostre competenze e la nostra esperienza direttamente al pubblico. Per i corsi abbiamo creato un'apposita e



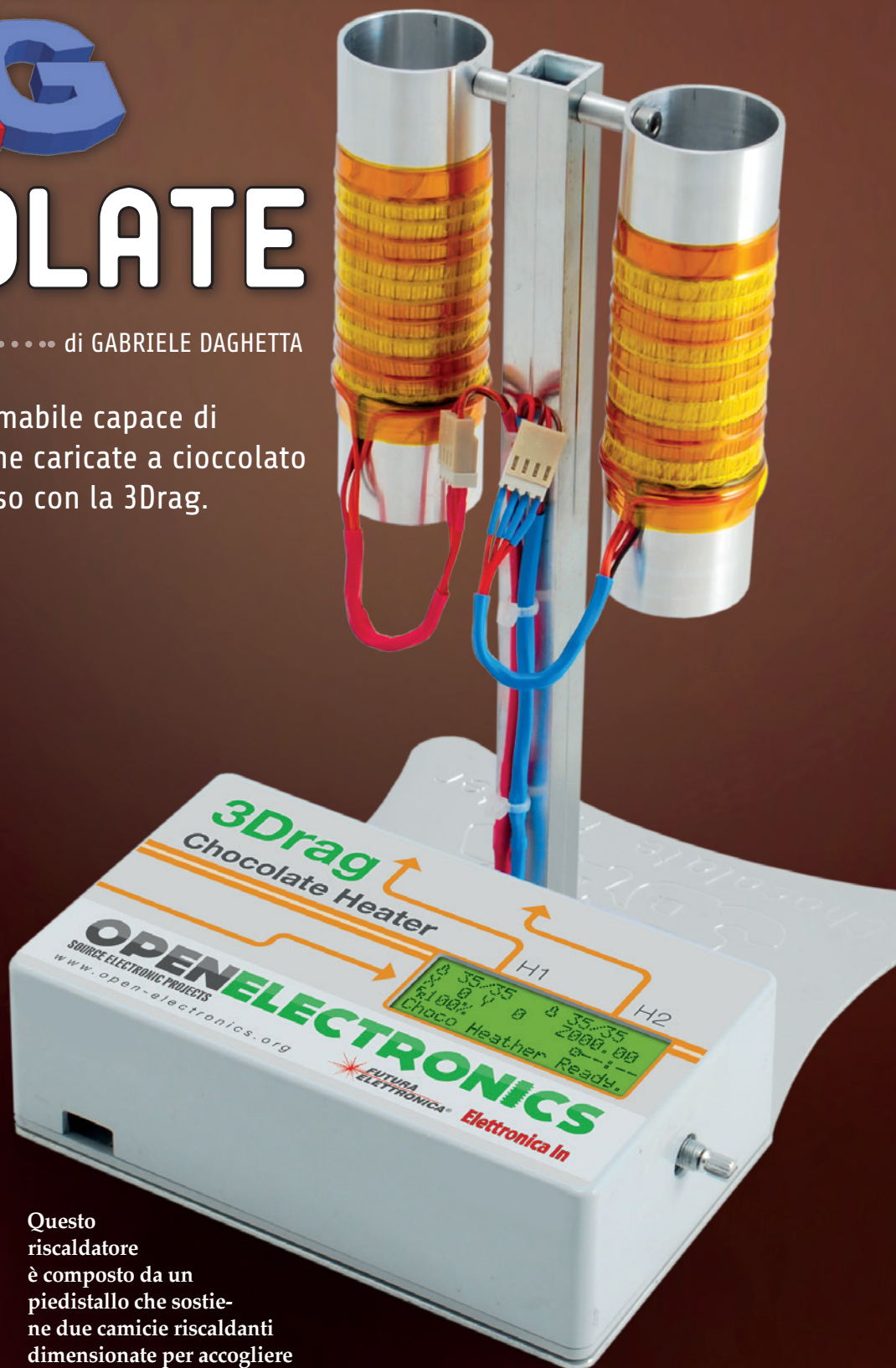
SCALDASIRINGHE PER 3DRAG CHOCOLATE

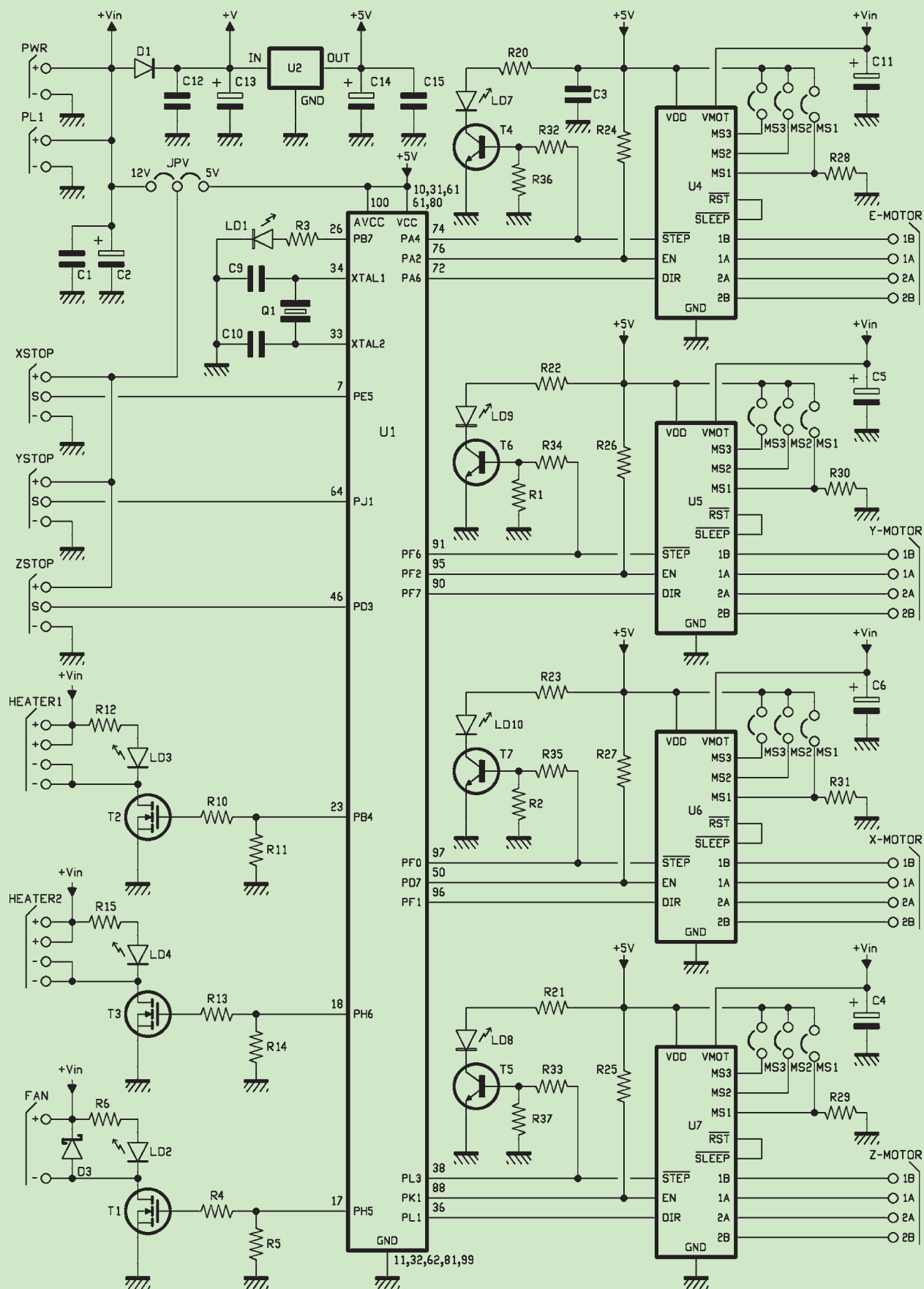
..... di GABRIELE DAGHETTA

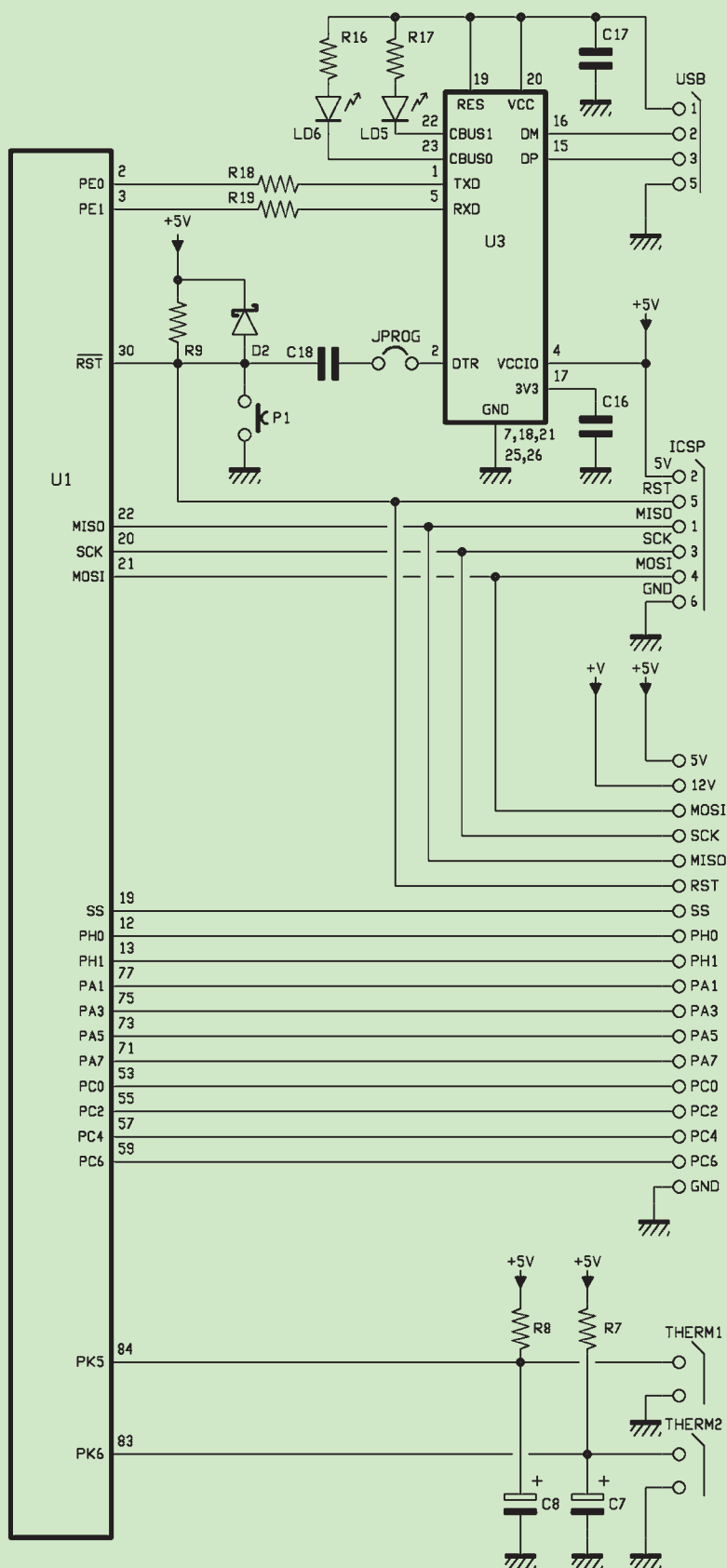
Riscaldatore programmabile capace di mantenere due siringhe caricate a cioccolato sempre pronte per l'uso con la 3Drag.

Un paio di mesi fa abbiamo cercato di stupirvi dimostrandovi un'altra delle incredibili trasformazioni della stampante 3Drag, che permette a questa macchina di stampare oggetti e decorazioni in cioccolato. Ebbene, se avete deciso di acquistare il kit di modifica equipaggiato con la testa di stampa a cioccolato, significa che in qualche modo ci siamo riusciti. Se oggi state stampando, per diletto o lavoro, particolari in cioccolato (al latte, fondente o bianco) potreste trovare interessante l'accessorio qui proposto e annunciato nel fascicolo dello scorso ottobre: si tratta del riscaldatore per le siringhe, che permette di mantenere caldo e fuso il cioccolato in modo da avere ricariche pronte all'uso, quando bisogna produrre tante stampe e non si può perdere tempo ad attendere che la siringa della stampante scaldi il prodotto alimentare.

Questo riscaldatore è composto da un piedistallo che sostiene due camicie riscaldanti dimensionate per accogliere







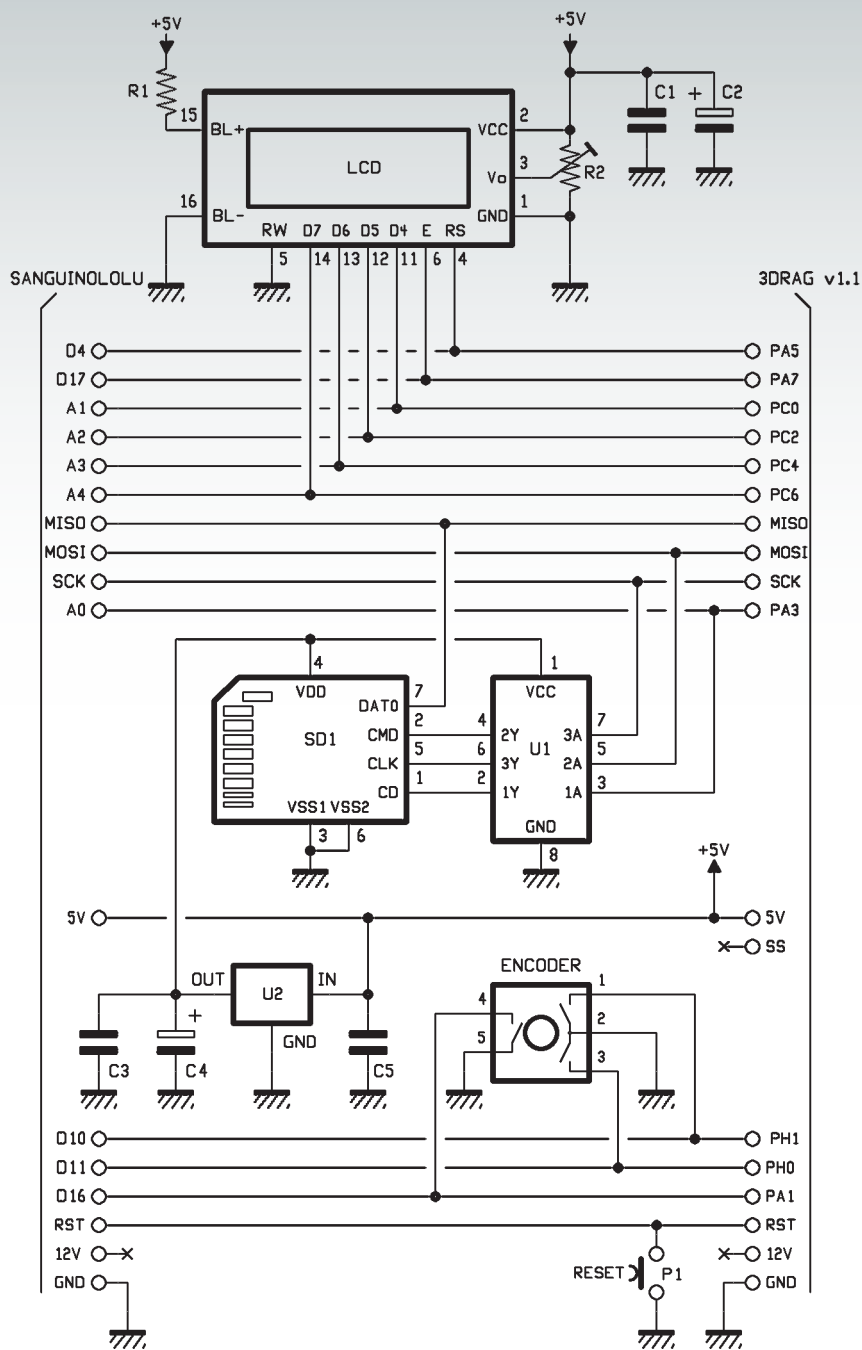
Elenco Componenti:

R1; R2: 10 kohm (0805)
 R3: 330 ohm (0805)
 R4, R10, R13: 10 ohm (0805)
 R5, R11, R14: 100 kohm (0805)
 R6: 1,8 kohm (0805)
 R7 ÷ R9: 4,7 kohm (0805)
 R12, R15: 1,8 kohm (0805)
 R16, R17: 470 ohm (0805)
 R18, R19: 10 ohm (0805)
 R20 ÷ R23: 330 ohm (0805)
 R24 ÷ R27: 10 kohm (0805)
 R28 ÷ R31: 100 kohm (0805)
 R32 ÷ R35: 4,7 kohm (0805)
 R36, R37: 10 kohm (0805)
 C1, C3: 100 nF ceramico (0805)
 C2, C4 ÷ C6: 100 µF 25 VL elettrolitico (E)
 C7, C8: 10 µF 35 VL elettrolitico (B)
 C9, C10: 22 pF ceramico (0805)
 C11, C13, C14: 100 µF 25 VL elettrolitico (E)
 C12, C15 ÷ C18: 100 nF ceramico (0805)
 D1: GF1M
 D2: BAT42W
 D3: MBRA140TRPBF
 T1 ÷ T3: BUK6215-75C
 T4 ÷ T7: BC817
 U1: ATMEGA2560-16AU
 U2: MC7805ABD2T (D2PAK)
 U3: FT232RL
 U4 ÷ U7: Driver motori cod. MD09/FT1031
 LD1: LED verde (0805)
 LD2 ÷ LD10: LED rosso (0805)
 P1: Microswitch
 Q1: Quarzo 16 MHz (C7S)

Varie:

- Plug alimentazione
- Morsettiera 2 poli
- Connettore mini-USB
- Strip maschio 2 poli (4 pz.)
- Strip maschio 3 poli (5 pz.)
- Strip maschio 4 poli (6 pz.)
- Strip femmina 32 poli (2 pz.)
- Jumper
- Circuito stampato

la siringa del cioccolato, simili a quella montata sulla testa di stampa per cioccolato ma prive della parte inferiore scalda-ago. Le due camicie sono montate ai lati di una staffa. Il sistema che usiamo consiste nel riscaldare il corpo della siringa mediante un riscaldatore elettrico a resistenza avvolto su un cilindro che ospita la siringa stessa. L'intero corpo della siringa è introdotto in un cilindro in alluminio il cui duplice scopo è sostenere la siringa e portarle il calore prodotto dal riscaldatore. Essendo l'alluminio un



Elenco Componenti:

R1: 33 ohm
 R2: Trimmer multigiri 10 kohm
 C1, C3: 100 nF multistrato
 C2: 100 μ F 16 VL elettrolitico
 C4: 100 μ F 16 VL elettrolitico
 P1: Microswitch
 SD1: Connettore SD-Card
 U1: 74HC4050
 U2: MCP1702-3302E/TO ENCODER:
 Encoder rotativo (cod. ENCODER12MM)
 LCD: Display LCD 20x4 (cod. LCD20X4)

Varie:

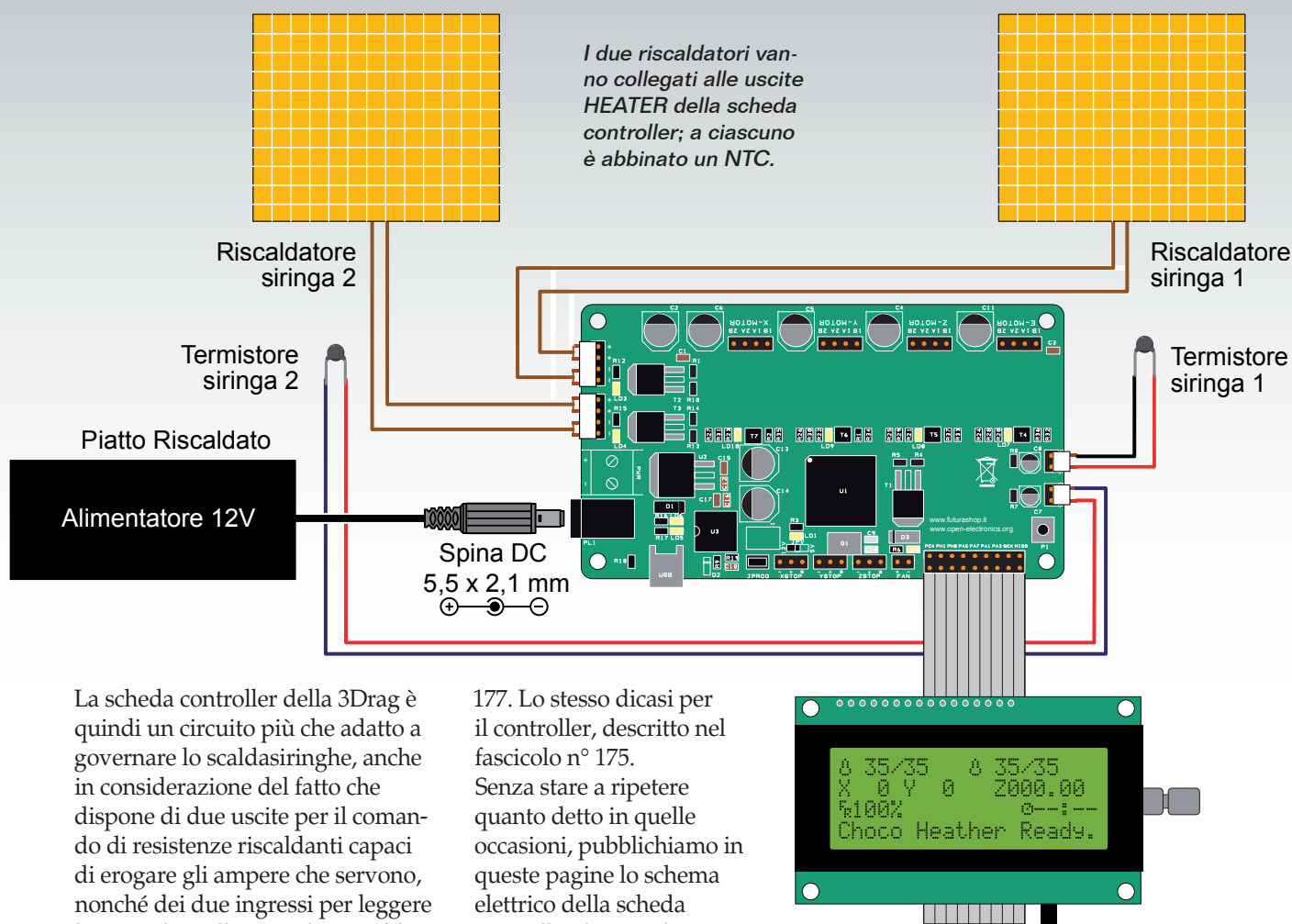
- Zoccolo 8+8 - Strip maschio 3 poli 11mm (2 pz.)
- Strip maschio 7 poli 11mm (2 pz.)
- Strip maschio 9 poli 11mm (2 pz.)
- Strip maschio 16 poli 25mm (1 pz.)
- Strip femmina 16 poli (1 pz.)
- Torretta F/F 20mm (3 pz.)
- Vite 3MA/8mm testa conica (6 pz.)
- Flat cable con connettore universale
- Circuito stampato

buon conduttore di calore, se è vero che trasmette assai bene il caldo alla parete esterna della siringa, è vero altrettanto che la sua parete esterna tende a disperdere e quindi a raffreddare; per questa ragione abbiamo pensato di avvolgere esternamente il cilindro e il riscaldatore con della pellicola di Kapton.

COME È FATTO LO SCALDASIRINGHE

Il sistema di riscaldamento e sostegno di ciascuna siringa consta, come detto, di un cilindro in alluminio rivestito da un riscaldatore flessibile che aderisce alla sua superficie esterna; il riscaldatore è del tipo a resistenza elettrica e dispone di due fili per collegarlo all'alimentazione. Ad esso, o meglio, al corpo in alluminio, è applicato un termistore che permette di rilevare la temperatura raggiunta e comunicarla all'elettronica di controllo del sistema, così da stabilizzare la temperatura del cioccolato facendo in modo che sia fuso, ma non oltre i 33 gradi tipici che costituiscono la soglia oltre la quale si rischia la perdita della tempra.

Il controllo del riscaldatore attraverso la temperatura letta viene operato da una scheda elettronica che è la stessa che governa la 3Drag V 1.2, ma priva dei moduli driver per i motori passo-passo, dato che qui non servono; ricordiamo che la scheda controller della 3Drag dispone di sei uscite, ossia quattro per il controllo di altrettanti stepper-motor e due a singolo MOSFET per alimentare in PWM i riscaldatori (quello dell'estrusore e quello del piatto riscaldato) più un'uscita per la ventola. Inoltre prevede due ingressi, letti dall'ADC del microcontrollore on-board, con cui vengono acquisiti i valori dei due termistori NTC, che nella stampante sono quelli per monitorare la temperatura dell'estrusore e del piatto riscaldato.



La scheda controller della 3Drag è quindi un circuito più che adatto a governare lo scaldasiringhe, anche in considerazione del fatto che dispone di due uscite per il comando di resistenze riscaldanti capaci di erogare gli ampere che servono, nonché dei due ingressi per leggere le NTC che nelle custodie riscaldanti per le siringhe servono a monitorare la temperatura raggiunta, fornendo il feedback al controller. Basta quindi realizzare la versione base, senza montare i moduli driver degli stepper-motor, caricare nel microcontrollore il firmware che abbiamo preparato apposta, ed il gioco è fatto.

Siccome ci è sembrato giusto far gestire il sistema in stand-alone da una persona, senza passare da un PC, lo abbiamo dotato di un pannello di controllo provvisto di display LCD a 4 righe per 20 caratteri e di un encoder rotativo con pulsante centrale sulla manopola. Il pannello è lo stesso che equipaggia la 3Drag nella versione a stampa autonoma (VM8021) e può essere acquistato presso Futura Elettronica insieme alla scheda controller, ovvero autocostruito seguendo le indicazioni fornite nel fascicolo n°

177. Lo stesso dicasi per il controller, descritto nel fascicolo n° 175. Senza stare a ripetere quanto detto in quelle occasioni, pubblichiamo in queste pagine lo schema elettrico della scheda controller, limitandoci a descrivere le parti che riguardano la gestione delle uscite a MOSFET e dei termistore, nonché l'interfacciamento con il pannello di controllo.

LA SCHEDA CONTROLLER

Il circuito si basa su un microcontrollore ATmega2560 ed è predisposto per ospitare quattro moduli driver per motori passo-passo, che non monteremo; delle tre uscite a MOSFET, utilizziamo HEATER 1 e HEATER 2, mentre ignoriamo la terza, che nella 3Drag è destinata al controllo della ventola a bassa tensione; lo schema è completato da un converter USB/seriale per interfacciare l'ATmega con il computer e ad uno stadio di alimentazione. Nel microcontrollore ATmega2560 è caricato il firmware "Marlin" opportunamente configurato per pilotare i riscaldatori, leggere i ter-

mistore e dialogare con il pannello di controllo.

I MOSFET utilizzati per alimentare i riscaldatori sono dei BUK6215-75C, a canale N enhancement mode, capaci di erogare una corrente di drain che raggiunge 57 A e sopportare una V_{DS} in stato di interdizione, pari a 75 V; la loro bassissima $R_{DS(on)}$ (15 milliohm max a 15 A di I_D) consente di minimizzare la potenza dissipata e perciò di saldarli direttamente ad una pista dello stampato, che funge da dissipatore. I MOSFET sono incapsulati in case SOT428, per montaggio superficiale, ideale per saldare l'aletta metallica al rame dello stampato, e non richiedono radiatore perché vengono utilizzati a correnti molto ridotte: nella 3Drag si parla di 2÷2,5 ampere per il riscaldatore dell'estrusore e di 5÷6 A per il piatto riscaldato,

Cilindro

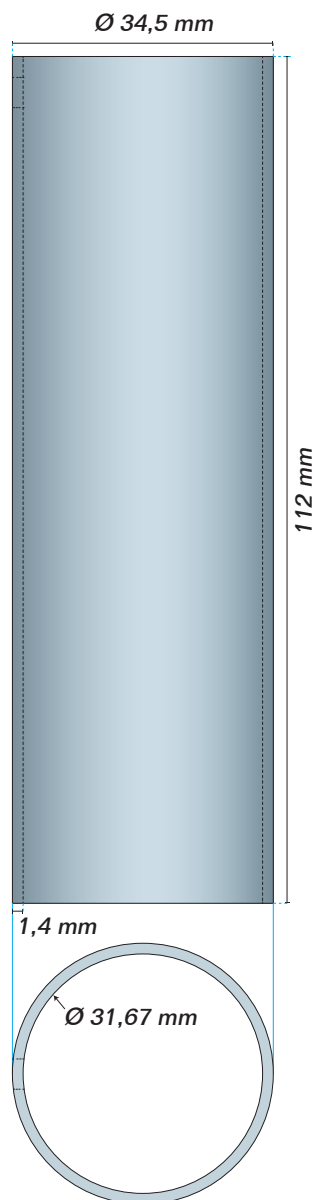


Fig. 1 - Dimensioni del cilindro portasiringa.

mentre nel controllo dello scaldasiringhe si parla di circa 0,9 A per ciascun riscaldatore. I MOSFET vengono pilotati tutti con un segnale PWM, generato con la frequenza opportuna dall'ATmega, così da permettere una precisa regolazione della potenza ceduta agli utilizzatori, che si traduce in un controllo ottimale del calore generato dalle resistenze dei riscaldatori; questi ultimi, pur avendo tensione nominale di 5 V, ricevono impulsi a

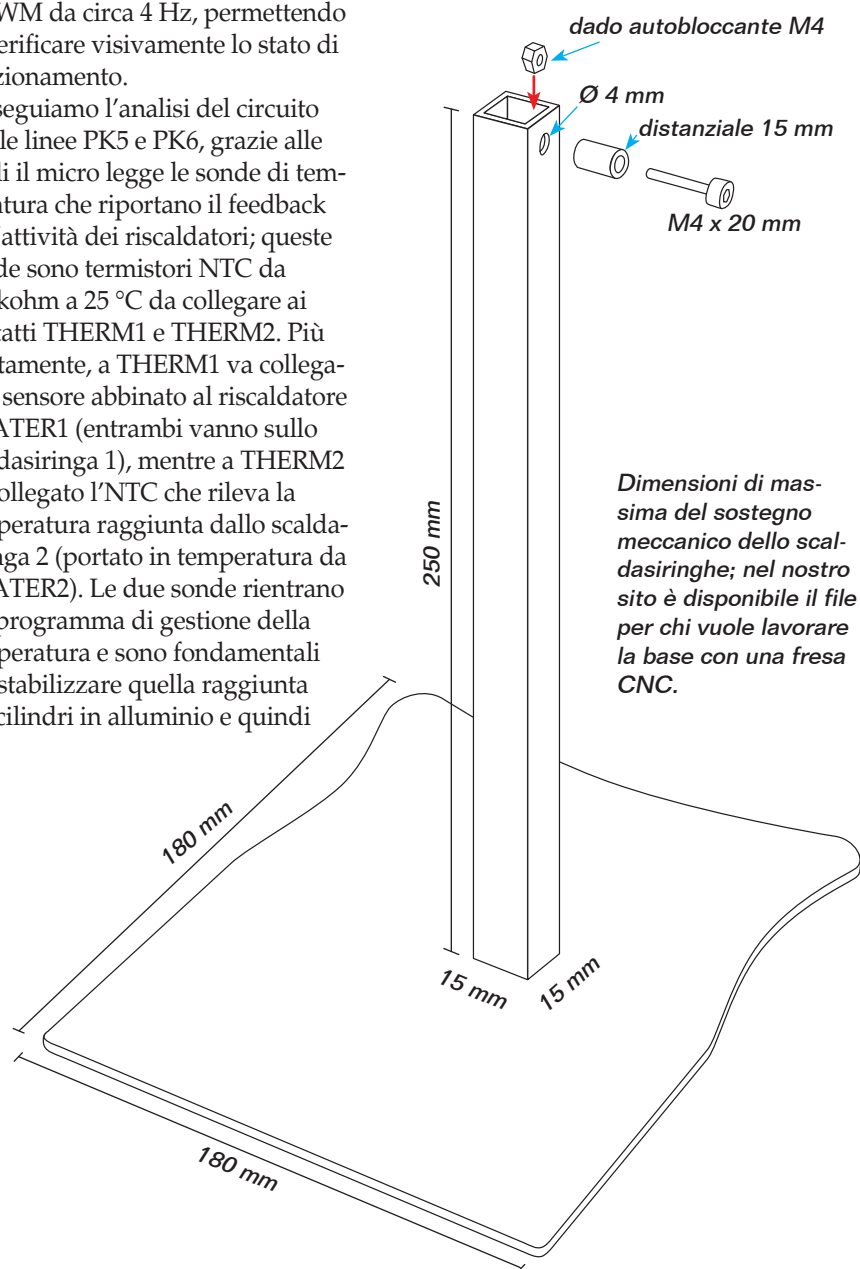
12 V, ma non subiscono danni perché giocando sulla larghezza degli impulsi si ottiene una tensione di valor medio pari a circa metà. L'uso del PWM consente altresì una regolazione a bassissime perdite, dato che i MOSFET lavorano in modo on-off pur variando la tensione effettivamente applicata ai carichi che alimentano.

Ciascuna delle uscite riservate ai riscaldatori è provvista di un LED che pulsa insieme al relativo segnale PWM da circa 4 Hz, permettendo di verificare visivamente lo stato di funzionamento.

Proseguiamo l'analisi del circuito con le linee PK5 e PK6, grazie alle quali il micro legge le sonde di temperatura che riportano il feedback sull'attività dei riscaldatori; queste sonde sono termistori NTC da 100 kohm a 25 °C da collegare ai contatti THERM1 e THERM2. Più esattamente, a THERM1 va collegato il sensore abbinato al riscaldatore HEATER1 (entrambi vanno sullo scaldasiringa 1), mentre a THERM2 va collegato l'NTC che rileva la temperatura raggiunta dallo scaldasiringa 2 (portato in temperatura da HEATER2). Le due sonde rientrano nel programma di gestione della temperatura e sono fondamentali per stabilizzare quella raggiunta dai cilindri in alluminio e quindi

dal cioccolato; infatti permettono al microcontrollore di regolare la corrente erogata alla resistenza che riscalda il cilindro in modo da mantenere la temperatura impostata indipendentemente dalle vicende termiche dell'ambiente in cui lo scaldasiringhe lavora.

Senza questa retroazione, non si potrebbe stabilizzare la temperatura, in quanto anche ammesso di lavorare a corrente costante e di supporre che a quella corrisponde



una certa temperatura, le variazioni ambientali e operative potrebbero alterare la temperatura alla quale viene portato il cioccolato, col risultato che se diventa troppo bassa poi la 3Drag impiega più tempo a iniziare il lavoro (perché deve riscaldare ancora il cioccolato), mentre se è troppo alta il cioccolato perde la tempa e la stampa sarà priva di consistenza.

Rilevando la temperatura del riscaldatore attraverso la tensione presente ai capi THERM1, il microcontrollore sa qual è la temperatura raggiunta da HEATER1 e verifica il dato con quello imposto dal processo di stampa: se la temperatura eccede quella impostata, riduce la corrente nel filamento riscaldatore, mentre in caso contrario la aumenta. Analogamente il discorso per quanto riguarda il riscaldatore HEATER2. Notate che per leggere le temperature colleghiamo i termistori al partitore resistivo con altrettanti resistori connessi al positivo di alimentazione, i cui valori sono stati opportunamente calcolati; avendo a che fare con dei termistori NTC, è chiaro che ad un aumento della temperatura rilevata corrisponde un calo di tensione e viceversa. I termistori utilizzati sono NTC da 100 kohm a 25 °C.

Bene, passiamo adesso alla sezione di comunicazione con il computer, utilizzata solo per caricare o aggiornare il firmware; essa fa capo alle linee PE0, PE1 ed RST del microcontrollore: alle prime due viene assegnato l'UART interno, che dialoga con il convertitore USB/seriale siglato U3. Quest'ultimo è il canonico FT232RL della FTDI, e contiene la logica che occorre a trasformare i dati dal formato seriale TTL all'USB, smistando i dati in arrivo da TXD ed RXD sui piedini DP e DM. L'integrato viene alimentato direttamente dai 5 volt dell'USB quando la scheda controller viene connessa al computer.

I LED siglati LD5 ed LD6 indicano localmente l'attività delle linee di trasmissione e ricezione dei dati. Notate che l'integrato è provvisto di piedini per alcuni dei segnali di controllo dello standard RS232; di questi, usiamo il DTR per far avviare il bootloader quando serve caricare il firmware nel microcontrollore. Questa prassi è ormai consolidata nello standard Arduino, in quanto una caratteristica dei microcontrollori dell'architettura Arduino è di essere programmati con un minimo di firmware, detto bootloader, che possiamo considerare come il BIOS del computer e che, opportunamente stimolato da un evento, avvia l'acquisizione dal computer dei dati che rappresentano il firmware e lo carica nella Flash di programma. Nel nostro caso, quando l'IDE invia l'apposito comando di programmazione, la linea DTR dell'FT232RL fornisce un impulso a livello logico basso, che attraversa il condensatore C18 e porta momentaneamente allo stato zero il piedino /RST dell'ATmega; il bootloader, che è il firmware di base che gira nel microcontrollore, riconosce questa condizione come la richiesta di comunicazione con il computer (via USB) per iniziare il download del firmware che dovrà poi essere eseguito normalmente, ossia quello che controllerà la stampante 3D.

Quando DTR torna a livello logico alto, ovvero il condensatore C18 si è caricato e smette l'impulso di reset, lo stesso C18 si scarica attraverso il diodo Schottky D2. Pari effetto si può ottenere premendo localmente il pulsante di reset, esattamente come avviene nelle schede Arduino.

Passiamo oltre e

vediamo il connettore ICSP, che praticamente non ha bisogno di spiegazioni: serve a voi per fare la programmazione di base dell'ATmega2560, ovvero per caricargli il bootloader Arduino; questo connettore riporta l'alimentazione a 5 volt, nonché le linee MISO, MOSI ed SCK dell'SPI, nonché il RESET del microcontrollore. Le stesse linee le ritroviamo, insieme agli I/O dell'ATmega2560 rimasti inutilizzati, sul pin strip di espansione che abbiamo previsto per la connessione con il pannello di controllo.

IL PANNELLO DI CONTROLLO

Il pannello si collega al connettore di espansione della scheda controller della stampante e dispone di un encoder rotativo facente capo a una manopola e di un pulsante utilizzabile per il reset, oltre a un display LCD da 20 colonne per 4 righe e ad un adattatore per SD-Card. L'encoder è tipo quello che si trova nei controlli multifunzione di autoradio, impianti hi-fi e vari altri apparati; nel nostro caso serve a scorrere tra le funzioni permesse dal pannello, in base al menu attivo al momento. Siccome l'encoder dispone della funzione "push", premendo il perno invece di ruotarlo, si dà

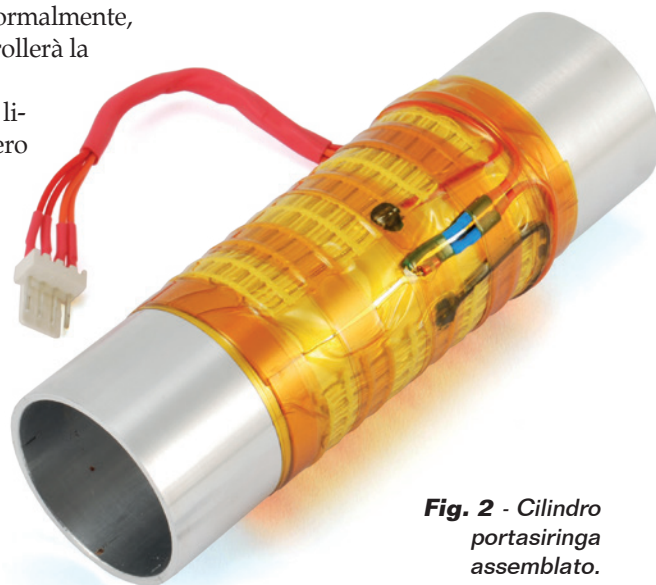


Fig. 2 - Cilindro portasiringa assemblato.

Listato 1

```
void manage_heater()
{
    float pid_input;
    float pid_output;

    if(temp_meas_ready != true)    //better readability
        return;

    updateTemperaturesFromRawValues();

    for(int e = 0; e < EXTRUDERS; e++)
    {

#ifdef PIDTEMP
        pid_input = current_temperature[e];
        pid_output = constrain(target_temperature[e], 0, PID_MAX);
    #else /* PID off */
        pid_output = 0;
        if(current_temperature[e] < target_temperature[e]) {
            pid_output = PID_MAX;
        }
    #endif

        // Check if temperature is within the correct range
        if((current_temperature[e] > mintemp[e]) && (current_temperature[e] < maxttemp[e]))
        {
            soft_pwm[e] = (int)pid_output >> 1;
        }
        else {
            soft_pwm[e] = 0;
        }
    } // End extruder for loop

#ifdef PIDTEMPBED
    if(millis() - previous_millis_bed_heater < BED_CHECK_INTERVAL)
        return;
    previous_millis_bed_heater = millis();
#endif

    #if TEMP_SENSOR_BED != 0

#ifdef PIDTEMPBED
        pid_input = current_temperature_bed;
        pid_output = constrain(target_temperature_bed, 0, MAX_BED_POWER);
```

Continua

conferma della voce attualmente selezionata; insomma, questo encoder è praticamente un joystick. Per quanto riguarda le connessioni, alimentazione e linee dati passano su un unico cavo di tipo flat con due connettori agli estremi. Il flat termina su un connettore a 18 poli (due file da 9). Dando uno sguardo al rispettivo schema elettrico, notiamo che sul pannello si trovano il lettore di SD-Card, il display e l'encoder, più un traslatore di livelli che serve ad adattare i segnali 0/5V con cui comunica la scheda controller a quelli 0/3,3V richiesti dalla logica del pannello. La comunicazione dati, scandita dal segnale di clock che il micro del controller manda sul filo SCK, avviene con il bus SPI tramite il canale MOSI (Master Output Slave Input) che

trasporta i dati dal microcontrollore ATmega2560 al pannello e il MISO (Master Input Slave Output) dove i dati in arrivo dal pannello vanno all'ingresso RX dell'ATmega2560. Questo, per quanto riguarda l'accesso alla SD-Card, funzione che qui non viene usata. Ciò che ci interessa è invece il display, che viene gestito nella modalità a 4 bit di dati, dalle linee A1, A2, A3, A4 della scheda controller, della quale D4 controlla l'RS (definisce quando i dati sono riferiti a caratteri o sono comandi di gestione del cursore ecc.) e D17 l'E (enable); il piedino R/W del display è fisso a massa perché in questa applicazione il display deve solo ricevere comandi di caratteri da visualizzare. Andiamo adesso all'encoder, del quale il controller legge i due

switch che indicano i passi nei due versi (D10 e D11) e il pulsante centrale dell'Enter (D16). Sul pannello di controllo è presente anche un pulsante collegato alla linea RST, utilizzabile per resettare la scheda controller quando serve aggiornare il firmware (il pulsante risulta in parallelo a quello della scheda controller).

Tutta la logica di bordo è alimentata a 3,3 volt mediante il regolatore LDO U2, che prende tensione dalla linea +5V del controller.

REALIZZAZIONE PRATICA

Per realizzare lo scaldasiringhe occorre preparare o procurarsi le schede elettroniche, quindi mettere insieme la parte meccanica provvista di elettroriscaldatori. Partiamo con l'elettronica, le cui schede

Listato 1 (segue)

```

        if((current_temperature_bed > BED_MINTEMP) && (current_temperature_bed < BED_MAXTEMP))
        {
            soft_pwm_bed = (int)pid_output >> 1;
        }
        else {
            soft_pwm_bed = 0;
        }

#elif !defined(BED_LIMIT_SWITCHING)
// Check if temperature is within the correct range
if((current_temperature_bed > BED_MINTEMP) && (current_temperature_bed < BED_MAXTEMP))
{
    if(current_temperature_bed >= target_temperature_bed)
    {
        soft_pwm_bed = 0;
    }
    else
    {
        soft_pwm_bed = MAX_BED_POWER>>1;
    }
}
else
{
    soft_pwm_bed = 0;
    WRITE(HEATER_BED_PIN, LOW);
}
#else // #ifdef BED_LIMIT_SWITCHING
// Check if temperature is within the correct band
if((current_temperature_bed > BED_MINTEMP) && (current_temperature_bed < BED_MAXTEMP))
{
    if(current_temperature_bed > target_temperature_bed + BED_HYSTERESIS)
    {
        soft_pwm_bed = 0;
    }
    else if(current_temperature_bed <= target_temperature_bed - BED_HYSTERESIS)
    {
        soft_pwm_bed = MAX_BED_POWER>>1;
    }
}
else
{
    soft_pwm_bed = 0;
    WRITE(HEATER_BED_PIN, LOW);
}
#endif
#endif
}

```

possono essere sia acquistate già pronte (è consigliato, almeno per la controller 3Drag, dato che è formata quasi esclusivamente da componentistica SMD) sia autocostruirle, scaricando allo scopo tutti i file necessari dal nostro sito. Ovviamente sono necessari degli strumenti e delle attrezzature specifici per il montaggio superficiale, soprattutto di componenti con pin ad elevata densità come il microcontrollore ATmega2560 che deve essere saldato usando gli strumenti opportuni. Ricordiamo che nella scheda controller non vanno inseriti i moduli driver, quindi se la acquistate procuratevi la sola scheda senza moduli, che si trova da Futura Elettronica, www.futurashop.it, con il codice prodotto 7350-3DCONTROLLER. Il pannello di controllo si trova

anch'esso già pronto, sempre su www.futurashop.it, con il codice prodotto 8220-VM8201 (ignorare i file .STL forniti per stampare in 3D il contenitore: quello usato in questo progetto è un Teko Coffe 3). Per poter utilizzare la scheda bisogna caricare il bootloader (cioè il firmware di base) nell'ATmega 2560; la scheda acquistata da Futura Elettronica è già programmata, mentre se è stata realizzata autonomamente, va programmata direttamente sulla scheda. Allo scopo dovete usare il pin-strip a 3+3 pin siglato ICSP ed un programmatore per ATmega, ma nulla vieta di caricare il bootloader usando come programmatore un Arduino, secondo quanto spiegato negli articoli che abbiamo dedicato, in passato, alla programmazione stand-alone (in

particolare, sulla tecnica descritta nel fascicolo n° 159). In quest'ultimo caso l'operazione è abbastanza semplice, in quanto nell'IDE di Arduino trovate già la voce ATmega2560 e il rispettivo bootloader. Il riscaldatore di una siringa deve essere collegato, senza badare alla polarità, ai contatti HEATER1, mentre l'NTC corrispondente va a THERM1; quello dell'altra siringa va connesso con le stesse modalità a HEATER2 e il relativo NTC va su THERM2. Fatto ciò, alimentate la scheda di controllo con un alimentatore dotato di plug adatto a quello montato sulla scheda: va bene un alimentatore di quelli AC/DC per portatili, che fornisca 12 volt c.c. ed una corrente dell'ordine dei 4 ampere. In alternativa, potete utilizzare un alimentatore da rete multiten-



Robot autoreplicanti stampati in 3D

Una delle cose che l'uomo si aspetta dai robot è che svolgano compiti pericolosi o gravosi e che lo facciano in autonomia; purtroppo ad oggi, anche per i limiti della capacità di movimento, gli automi possono, quando operano in un ambiente sconosciuto, trovarsi in difficoltà perché non dispongono dei mezzi per superare un ostacolo o rimuovere un oggetto. La soluzione ottimale sarebbe disporre di robot in grado di adattare il proprio corpo alla situazione, ad esempio dotandoli di una stampante 3D che prepari sul posto protesi, attrezzi, parti da aggiungere, che i robot monteranno. Non è fantascienza, perché i ricercatori dell'Università di Oslo (Norvegia) stanno sviluppando un sistema in grado di stampare robot personalizzati per affrontare le situazioni impreviste. I ricercatori hanno implementato algoritmi evolutivi da applicare ai robot, i quali, presa visione della difficoltà sul posto, si autoreplicheranno fino a trovare la soluzione che gli faccia guadagnare l'autonomia.

sione, collegandone i cavi di uscita positivo e negativo rispettivamente ai morsetti + e - PWR.

Per caricare il firmware, collegate la scheda al PC mediante un cavo USB A/B-mini e avviate l'IDE Arduino, quindi aprite il file corrispondente (.ino) e avviate l'upload con il pulsante *Carica*. Caricato il firmware, staccate il PC e collegate tra loro la scheda controller e il pannello di controllo, inserendo le due estremità del cavo flat nei rispettivi connettori e curandovi che ci sia corrispondenza fra i pin (fate in modo che il filo colorato sia dal lato di MISO e 5V sia sulla scheda del pannello di controllo sia sulla controller).

COSTRUZIONE MECCANICA

Lo scaldasiringhe è composto, oltre che dal sostegno, dai due gruppi che sostengono e scaldano le siringhe, ognuno formato da un corpo

cilindrico in alluminio con diametro esterno di 35 mm ed interno di 31,6 mm (**Fig. 1**), opportunamente forato, su cui viene avvolta una cartuccia riscaldante di tipo "flat" della Futura Elettronica (codice 7350-COM11288).

Per assemblarlo, prendete il cilindro in alluminio porta-siringa e applicate alla sua superficie esterna il termistore NTC, quindi fissatelo con del nastro in Kapton; avvolgete sul cilindro anche la cartuccia riscaldante tipo "flat" fissandola con il solito nastro in Kapton, in modo che l'insieme appaia come in **Fig. 2**. Fissare quindi i due cilindri completi al relativo supporto mediante due viti M4x20 e due distanziali a tubo da 15 mm. A cablaggio finito, il pannello di controllo completo va inserito nel contenitore plastico Coffe 3, del quale forare la parete frontale in modo da far fuoriuscire

il display; lateralmente bisogna praticare il foro per far uscire il perno dell'encoder rotativo.

Installato il pannello, se tutto è a posto sull'LCD avrete le scritte; se il display è vuoto o mostra caratteri privi di significato, ruotate il cursore del trimmer sulla scheda per regolare il contrasto (se appaiono le scritte è tutto ok). Se il pannello non interagisce con la scheda, significa che il cavo di collegamento è stato inserito male. Con il display collegato ed attivato, il firmware deve presentare le informazioni base.

IL FIRMWARE

Uno dei motivi per cui la 3Drag adotta un firmware Marlin è che comprende il modulo dedicato al pilotaggio degli LCD. Per lo scaldasiringhe abbiamo mantenuto il firmware Marlin, anche perché costituisce un'ottima base ed incorpora la gestione USB, compatibilità Repetier-Host e il PID per la gestione temperatura. Inoltre permette di utilizzare vari tipi di termistore, integrando i rispettivi parametri da introdurre nell'equazione di Steinhart Hart.

Nella scheda controller è caricato un firmware che sostanzialmente è il Marlin per la 3Drag V 1.2 dotata di pannello di controllo per stampa autonoma (può essere scaricato direttamente dalla pagina web <http://3dprint.elettronica.it/2012/09/06/software/> in cui sono raccolte tutte le versioni disponibili e dove troverete anche la relativa procedura di aggiornamento) alleggerito, perché è stata tolta tutta la parte di gestione degli stepper motor; in esso vengono conservate le routine di interfaccia col pannello di controllo e quelle della gestione delle due uscite per i riscaldatori e della lettura degli ingressi relativi ai termistori usati per la retroazione necessaria al controllo della temperatura. Nel **Listato 1** riportiamo la porzione di codice riguardante la

```

0 35/35 0 35/35
X 0 Y 0 Z000.00
F100% 0--!--
Choco Heater Ready.

```

Schermate del menu del pannello di controllo: le voci si confermano premendo la manopola dell'encoder, mentre per evidenziare quella interessata ci si sposta con la rotazione della stessa manopola.

```

#Info screen #
Prepare →
Control →

```

```

#Main #
Preheat Choco →
Cooldown
Switch Power off

```

```

#Prepare #
Preheat H1
Preheat H2
Preheat All Choco

```

gestione della temperatura. Spendiamo qualche parola per spiegare che cosa è stato cambiato nel firmware Marlin: per operare col cioccolato è necessario mantenere il riscaldatore a temperature molto più basse rispetto a quelle richieste nell'uso con i materiali plastici. Ricordiamo che il firmware Marlin nasce per il controllo di stampanti 3D a plastica e include tutta una serie di protezioni tra le quali quella che impedisce di estrarre il materiale al di sotto dei 170°C. Dovendo sciogliere il cioccolato ad una temperatura di circa 33°C, l'istruzione:

```
#define EXTRUDE_MINTEMP 170
```

è stata trasformata in:

```
#define EXTRUDE_MINTEMP 10
```

Questa modifica sposta la soglia di protezione da 170°C a 10°C e consente di scendere a 33 °C. Un'ulteriore modifica riguarda il PID per il controllo della temperatura: per evitare che questa oscilli eccessivamente, modifichiamo il parametro P, ovvero l'istruzione:

```
#define DEFAULT_Kp 22.2
```

trasformandola in:

```
#define DEFAULT_Kp 32.2.
```

Questo valore è stato calcolato empiricamente, ma i test da noi effettuati hanno dimostrato che è

quello ottimale.

Per gestire il pannello bisogna utilizzare la parte di firmware con la gestione dell'LCD e la lettura dell'encoder; il tutto è contenuto nel firmware completo che abbiamo previsto per lo scaldasiringhe.

CONSIGLI D'USO

Le siringhe vanno posizionate con il pistone in giù, senz'ago, e ciò per due ragioni: la prima è che altrimenti, man mano che il cioccolato si riscalda e fonde, colerebbe; la seconda è che introducendo scaglie o pezzi di tavoletta, restano spazi vuoti all'interno della siringa e una volta fusi il cioccolato, si formano bolle d'aria. In questo caso, se la siringa è a testa in giù l'aria viene spinta in alto dalla pressione del peso del cioccolato ed esce, come avviene nello sfato dei caloriferi, mentre se è a testa in su, le bolle rimaste creano dei problemi perché quando la siringa viene caricata nella stampante e il meccanismo di pressione dello stantuffo si abbassa, l'aria esce insieme al cioccolato causando vuoti di alimentazione e una deposizione irregolare del cioccolato sul piatto di stampa. Ogni siringa può essere caricata in due modi: colando all'interno della siringa cioccolato già fuso a bassa temperatura, quindi inserendola nell'apposito cilindro; inserendo nella siringa scaglie di cioccolato che poi fonderanno grazie all'a-

```

#Control #
Heater 1: 35
Heater 2: 35
Autotemp: Off

```

```

0 Min: 10
0 Max: 50
0 Fact: 000.10
>PID-P: +032.20

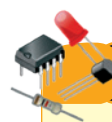
```

```

PID-I: +001.00
PID-D: +114.00
PID-C: 1
>Preheat Choco con+

```

zione del riscaldatore, una volta inserita la siringa nel cilindro. ■



per il MATERIALE

Una parte del materiale necessario per realizzare questo progetto può essere acquistata presso Futura Elettronica:

- Controller per stampante 3Drag (Scheda da programmare con firmware scaricabile cod. 7350-3DCONTROLLER - 48,00 Euro)
- Scheda per stampa autonoma (cod. 8220-VM8201 - 64,00 Euro)
- Riscaldatore flat (cod. 7350-COM11288 - 6,00 Euro)
- Termistore (cod. 7350-NTC100K - 6,00)

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

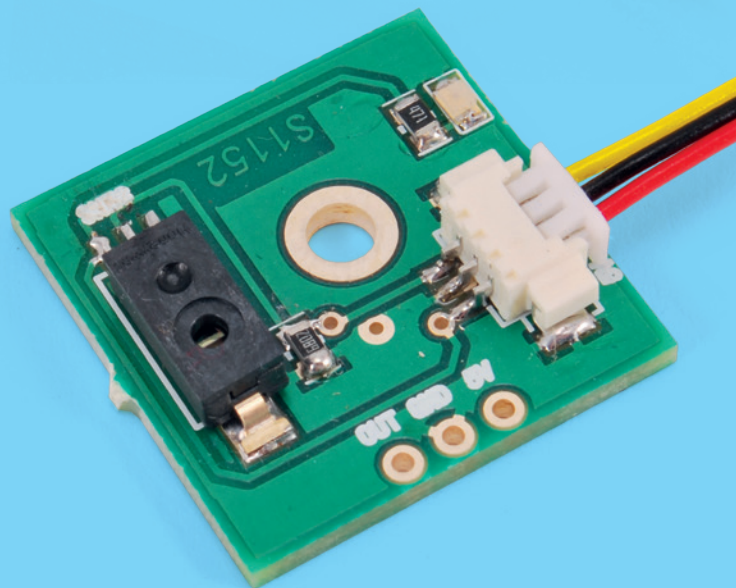
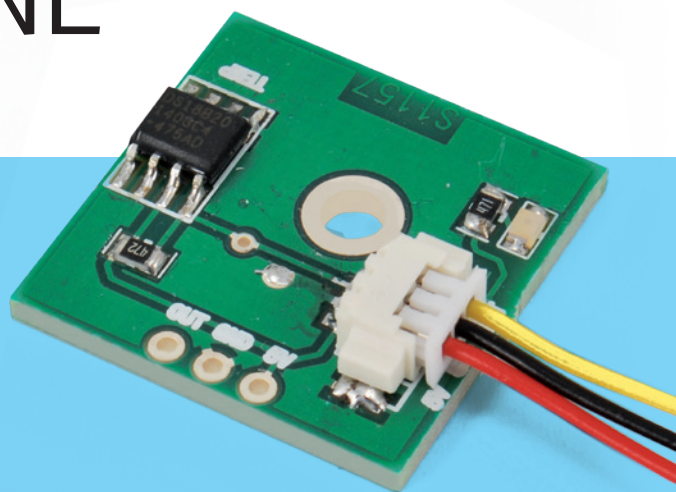
Volete utilizzare i più recenti componenti elettronici ma li trovate difficili da montare perché sono in SMD, piccolissimi e richiedono speciali attrezzature? Non preoccupatevi, da oggi potete trovarli su schede di prototipazione pronte all'uso e il cui collegamento è alla portata di tutti.

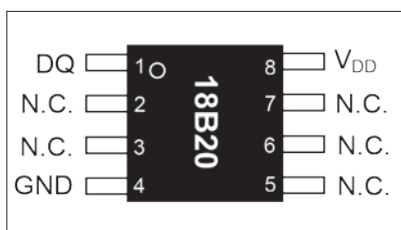
BREAKOUT BOARD MULTIFUNZIONE

di FRANCESCO DONI

L'elettronica ha fatto e continua a fare progressi con una rapidità esponenziale e se è vero che nell'integrazione delle funzioni i passi sono stati "da gigante", le dimensioni dei nuovi componenti sono sempre più "da nani", anzi microscopiche. L'esigenza di creare dispositivi sempre più minuscoli è dettata dalle richieste dell'industria dei dispositivi portatili, che necessita sempre più funzioni a parità di ingombro e peso. Succede così che in commercio i componenti elettronici tradizionali, quelli a montaggio a terminale passante (THT) per intenderci, stanno progressivamente scomparendo a vantaggio degli SMD;

ed anche che questi ultimi, dalle iniziali dimensioni paragonabili a quelle della componentistica tradizionale, diventano ogni anno più piccoli e sono ormai minuscoli, oltretutto caratterizzati da piedinature con contatti fittissimi a passo anche di 1 mm (difficile quindi saldarli senza unirne due o più con la lega saldante). Per questa ragione chi non ha particolari attrezzature per la saldatura e il rework degli SMD (saldatori a punta finissima, stazioni di saldatura con pinze termiche per SMD o stazioni ad aria calda) si sente escluso e se anche volesse tentare l'approccio ai componenti più moderni, per sfruttarne l'indubbia convenienza





Piedinatura della sonda termica DS18B20 in contenitore µSOP.

in fatto di funzioni svolte, non saprebbe come fare. Lo stesso discorso vale per chi, pur pronto a dotarsi delle apparecchiature da laboratorio del caso, comincia ad avere la mano non proprio ferma e la vista da vicino un po' incerta, tanto da non riuscire a portare a termine un montaggio del genere con successo. In questi casi vengono in aiuto le cosiddette breakout board, ovvero schede di prototipazione contenenti il componente interessato già saldato, le cui connessioni sono portate all'esterno delle basette su connessioni facilmente utilizzabili per passo e terminazione; di solito i collegamenti sono piazzole a passo 2,54 mm, come quelle dei classici integrati DIP che per decenni sono stati alla portata di tutti.

Per consentire a chi, tra voi, vorrebbe utilizzare componentistica di un certo livello ma non ha i mezzi o le qualità per saldarla, abbiamo identificato un certo numero di integrati (comprendenti sensori, alimentatori switching, caricabatterie, amplificatori lineari ecc.) e li abbiamo montati su basette pronte all'uso; queste breakout board (di cui in questo articolo presentiamo le prime tre) sono sia un ausilio per chi desidera avere già pronto in un formato "tradizionale" il meglio dell'elettronica SMD, sia per chi -pur riuscendo a lavorare con componenti SMD- necessita di avere tali componenti disponibili su schede di prototipazione per applicarle a circuiti esistenti e fare prove, ovvero per realizzare prototipi che

integrino le funzioni dei relativi integrati, prima di realizzare il circuito stampato definitivo di un'apparecchiatura. Ad esempio, se bisogna sostituire una sonda di temperatura o aggiungerla a un apparato esistente, la breakout board con il sensore di temperatura Dallas DS18B20 fornisce il sensore già pronto per l'uso e le connessioni di alimentazione e di uscita facilmente accessibili per la prototipazione.

Tutte le breakout board proposte si possono acquistare già pronte presso Futura Elettronica - www.futurashop.it - ovvero, per chi se la sentisse, realizzarle da sé: per questa ragione pubblichiamo i relativi piani di montaggio e (sul nostro sito www.elettronica.in.it) le tracce lato rame.

In questo articolo iniziamo a descrivere le prime tre breakout board: quella con il sensore di temperatura, quella con il trasduttore di pressione e quella con il sensore di umidità; nei prossimi fascicoli proporremo via-via le altre, fino a completare la serie.

SCHEMA CON DS18B20

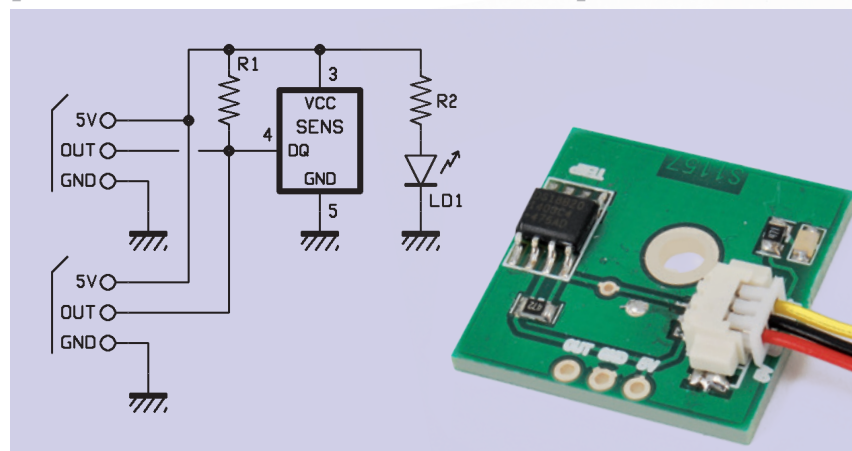
Il componente chiave di tale breakout board, della quale in fondo a questa pagina trovate lo schema elettrico, è la sonda a semi-

conduttore DS18B20 della Maxim Dallas; si tratta di un componente ormai molto diffuso in svariate apparecchiature fisse e portatili, che avete trovato negli ultimi anni anche tra le pagine di Elettronica In, ma in versione TO-92. In questo progetto impieghiamo, invece, la versione in case TSSOP miniaturizzata, chiamata µSOP.

Il sensore fornisce un segnale di uscita di tipo digitale, ragion per cui deve essere interfacciato ad un microcontrollore o un microprocessore che provveda a interrogarlo secondo il protocollo previsto. Le connessioni del sensore termico DS18B20 sono essenzialmente tre: i due pin d'alimentazione, quindi Vcc e GND, e la linea dati, che è bidirezionale e fa capo al pin che resta (DQ). Infatti la sonda della Dallas comunica sfruttando il protocollo One-Wire, che contempla la comunicazione dati mediante un solo filo (riferito al piedino di massa, ovviamente).

Affinché tutto funzioni correttamente, il canale dati necessita di una resistenza di pull-up, che in questo caso, a differenza di altre nostre realizzazioni, è implementata nel microcontrollore. Essendo la comunicazione con il sensore di tipo digitale, si può sistemare la sonda anche molto

[schema elettrico DS18B20]



	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
MS Byte	S	S	S	S	S	2^6	2^5	2^4

Fig. 1 - Struttura dei due bit emessi in sequenza dal DS18B20 ogni volta che viene interrogato: il byte LS è quello di peso minore e l'MSB è quello più significativo. Questa struttura si riferisce alla rappresentazione a 11 bit più segno.

lontano dal circuito senza preoccuparsi di disturbi ed interferenze provenienti dall'ambiente circostante.

Il DS18B20 misura tra -55 e +125°C ed esprime la temperatura rilevata con una risoluzione di 11 bit o 12 bit; inoltre presenta un tolleranza nella misura di appena 0,5 °C in un range che va da -10°C a +85°C. Per interrogarlo si utilizza un primo codice comando "Convert T" (codice 44h), corrispondente all'avvio dell'operazione di conversione della misura. La sonda risponde con uno 0 mentre esegue la conversione e con un 1 ad operazione terminata. A questo punto nei registri del componente sono disponibili due byte corrispondenti al valore della temperatura rilevata in gradi Celsius. Quindi, non resta che leggerli con un codice comando "Read ScratchPad (codice BEh)". Il registro a 2 byte, per un campionamento a 11 bit, risulta essere strutturato come mostra la Fig. 1.

Nella modalità ad 11 bit, dei 16 bit restituiti dal DS18B20 ogni volta che il microcontrollore lo interroga, i primi 11 (tutto il byte meno significativo e primi tre bit di quello più significativo) esprimono il valore della temperatura e gli ultimi cinque il segno, che viene indicato con tutti 0 se la temperatura rilevata è positiva e tutti 1 se, invece, la misura si riferisce a una temperatura negativa (sotto lo zero).

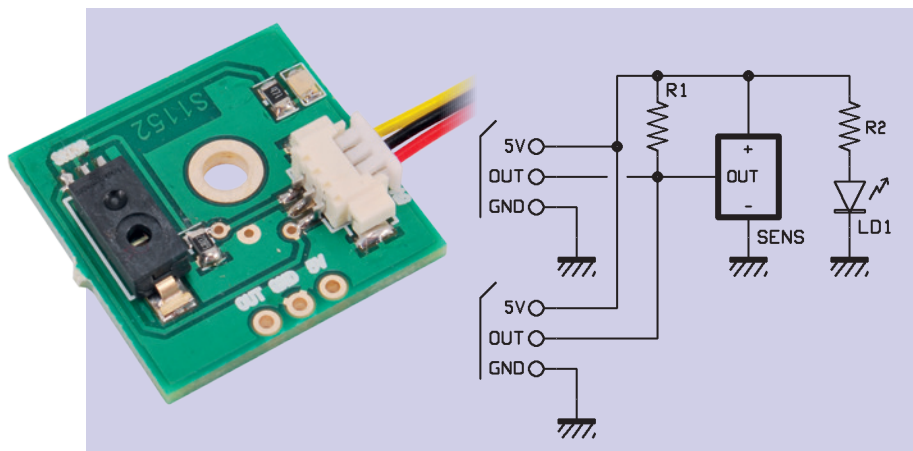
La particolarità del componente Dallas è che può esprimere la temperatura con i decimali, infatti, se guardate la Fig. 1 vi accorgete che i primi quattro bit meno significativi si riferiscono a valori minori di 1, dato che il primo vale 2 alla -4, ossia 1/16, il secondo vale 0,125, il terzo 0,25 e il quarto 0,5. Così il sensore Dallas riesce ad esprimere praticamente tutti i valori di temperatura che può misurare. Per fare un esempio chiarificatore, immaginiamo di misurare 64,25°C positivi; all'uscita dell'integrato

DS18B20 avremo due byte, il primo dei quali sarà (partendo dal bit più significativo) 00000100 e il secondo (sempre tenendo a sinistra il bit maggiormente significativo) 00000100. Analizzando i dati, il byte LSB dà il valore di 0,25 e il secondo byte dà +64 gradi. Volendo considerare il dato nell'insieme, ossia il valore complessivo formato dai due byte, possiamo dire che i possibili valori esprimibili dall'uscita del DS18B20 sono compresi tra 111110110110000 (-55 °C) e 000001111010000 (+125 °C). Trasferendo questi valori in formato esadecimale, possiamo dire che l'escursione è compresa fra AB70h e 07D0h. Sempre ragionando in esadecimale, vediamo che una temperatura di 10,125 °C corrisponde a 00A2h e che una di -20 °C si esprime con BCC0h. Oltre alla modalità descritta, il DS18B20 ne implementa una seconda, dove il valore della temperatura è espresso da 12 bit e il segno da quattro; le procedure di interrogazione e il formato dei dati sono simili a quella appena descritta, con la differenza che dei due byte ottenuti, uno riporta i 8 bit di temperatura e il secondo i restanti quattro più quattro di segno.

SCHEDA CON SENSORE DI UMIDITÀ

Questa seconda breakout board (trovate lo schema qui sopra) monta un sensore igrometrico, quindi si presta a rilevare l'umidità presente nell'ambiente in cui viene posta. Il sensore è prodotto dalla Honeywell e siglato HIH-

[schema elettrico HIH-5030-001]



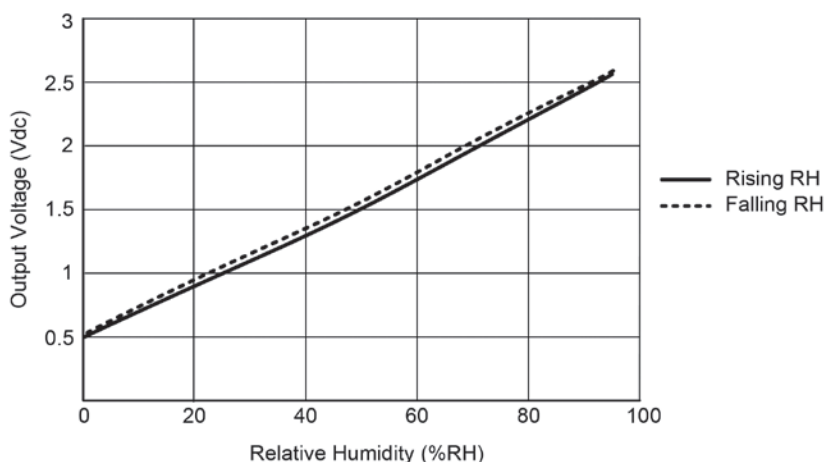


Fig. 2 – Tensione di uscita del sensore HIH-5030-001 in funzione dell'umidità relativa rilevata.

5030-001: funziona a bassissima tensione (opera correttamente anche a meno di 2,7 Vcc e fino a 5,5 Vcc) assorbendo tipicamente 200 microampere, quindi risulta ideale anche nelle applicazioni alimentate a pile e dove sia richiesto un assorbimento minimo.

Il sensore HIH-5030-001 ha il grande pregio di fornire, a differenza del DS18B20, un segnale di uscita analogico, che consiste in una tensione continua che varia in modo quasi lineare al variare dell'umidità relativa (RH); può quindi essere interfacciato sia a circuiti analogici che controllano la portata d'aria di un impianto di deumidificazione, sia essere letto da un A/D converter o dall'ingresso di un microcontrollore dotato di convertitore analogico/digitale.

L'elemento sensore del dispositivo è di tipo capacitivo, composto da polimeri termoindurenti tagliati a laser; nell'integrato è presente la circuiteria di condizionamento del segnale integrata su chip. La costruzione multistrato dell'elemento sensibile fornisce un'eccellente resistenza alla maggior parte dei rischi di applicazioni come ad esempio la formazione di condensa, polvere, sporco, oli e prodotti chimici comuni presenti nell'ambiente di misura.

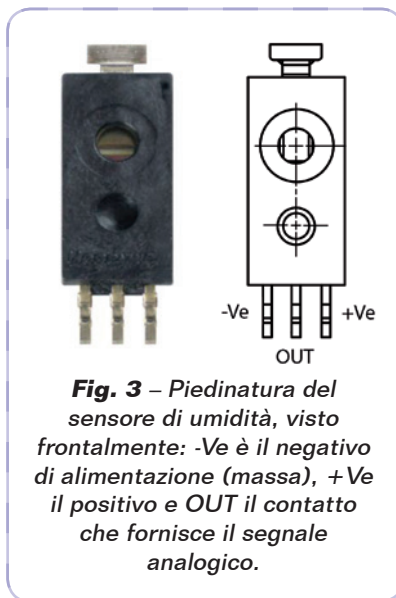


Fig. 3 – Piedinatura del sensore di umidità, visto frontalmente: -Ve è il negativo di alimentazione (massa), +Ve il positivo e OUT il contatto che fornisce il segnale analogico.

L'accuratezza della misura, considerata tra l'11% e l'89% di umidità relativa è di $\pm 3\%$ e il tempo di risposta alle variazioni di umidi-

tà è di circa 5 secondi. La curva di variazione della tensione di uscita del dispositivo in funzione dell'umidità relativa, a 3,3 volt di alimentazione e a una temperatura ambiente di 25 °C, è illustrata in Fig. 2.

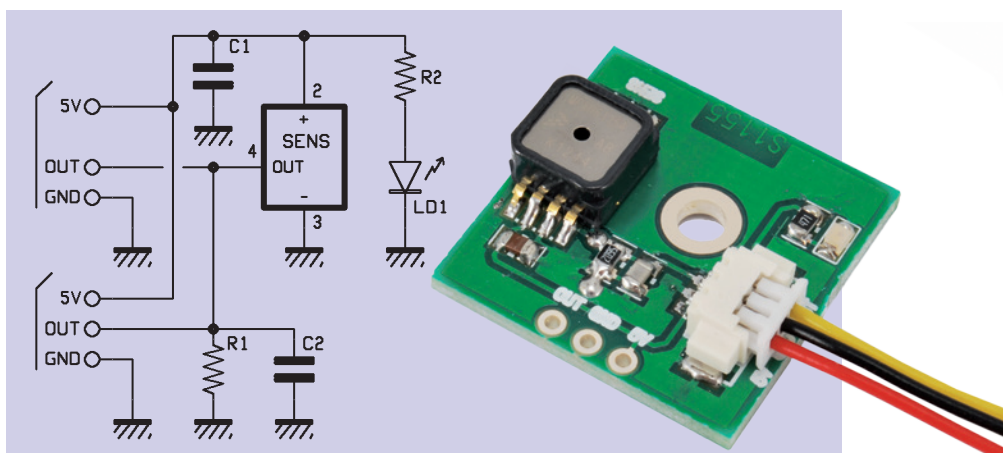
L'uscita del componente è ad alta impedenza e può pilotare circuiti che abbiamo non meno di 65 kohm di impedenza; diversamente la tensione fornita in funzione dell'umidità potrà non rispondere alla curva di Fig. 2, ma varierà diversamente.

Oltre che a un A/D converter, l'uscita del dispositivo si può collegare a un microamperometro a lancetta o a un voltmetro opportunamente tarato, così da realizzare un vero e proprio igrometro dal quale avere un'indicazione diretta dell'umidità relativa.

SCHEDA CON SENSORE DI PRESSIONE

Concludiamo l'analisi delle breakout board con quella che monta il trasduttore di pressione, un componente prodotto dalla Freescale e siglato MPXH6115A6U; come quello di umidità relativa, questo sensore fornisce un segnale di uscita analogico, ragion per cui può essere utilizzato per pilotare direttamente strumentini

[schema elettrico MPX6115A6U]



a lancetta e ottenere un'indicazione immediata della pressione che si sta misurando, per esempio in un tubo dove passa aria o in un contenitore a tenuta.

L'MPX6115A6U è un sensore integrato che su un unico chip comprende il trasduttore di pressione, un amplificatore operazionale in tecnologia bipolare e una rete di resistori a film sottile che permette di fornire un segnale di livello apprezzabile e la compensazione dell'errore introdotto dalle variazioni di temperatura; in pratica l'elemento trasduttore di pressione corredato dalla circuiteria necessaria al condizionamento del segnale. Il trasduttore all'interno del componente è di tipo piezoresistivo.

L'insieme fornisce una precisione dell'ordine dell'1,5% in un campo di temperatura operativa compreso tra 0 e 85 °C, la compensazione in temperatura fra -40 e +125 °C e un segnale di uscita analogico piuttosto lineare, in un campo di misura della pressione che va tra 15 e 115 kPa, che in bar significa poter misurare tra 0,15 e 1,15 bar (tra 0,1529 e 1,1726 kg/cm²).

La tensione di uscita fornita varia abbastanza linearmente con un rateo di salita di 45 mV/kPa; l'uscita presenta un'escursione utile (con alimentazione di 5 volt e un range di temperatura operativa fra

0 e 85 °C) di 4,5 V, con un offset pari a 0,2 V alla minima pressione misurabile e un massimo di 4,7 V a 115 kPa. La massima pressione tollerabile dal trasduttore (superata la quale il dispositivo può essere compromesso) è 400 kPa (4 bar). Il tempo di risposta alle variazioni di temperatura è dell'ordine del millisecondo, mentre il tempo di avvio (affidabilità dell'indicazione fornita in uscita dopo l'applicazione dell'alimentazione) è tipicamente 20 ms.

Fig. 4 - Aspetto esterno e schema a blocchi del sensore di pressione; i piedini 1, 5, 6, 7, 8 non sono collegati.

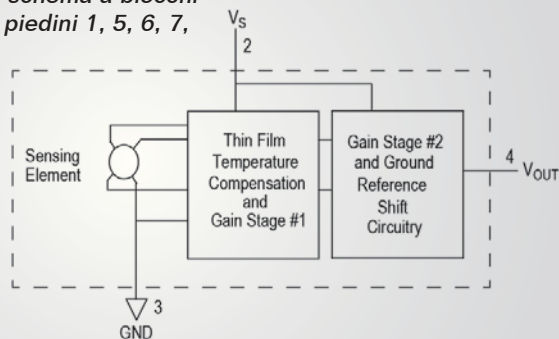
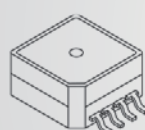
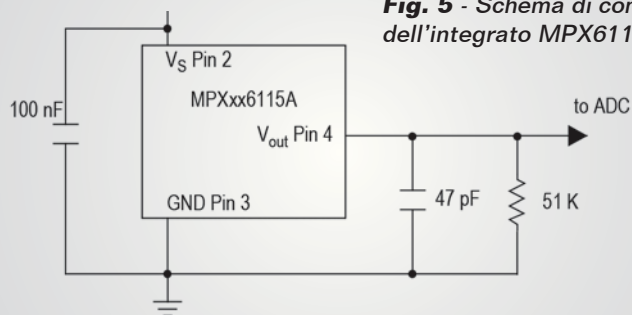


Fig. 5 - Schema di connessione dell'integrato MPX6115A6U.



Il sensore è disponibile in varie esecuzioni; per la nostra scheda di prototipazione abbiamo scelto quella in contenitore SSOP, ovviamente pensato per il montaggio in SMD (Fig. 4).

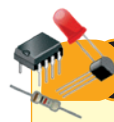
Gli utilizzi dell'MPX6115A6U vanno dai rilevatori barometrici agli altimetri per aeromodelli e aerei veri, ai misuratori di pressione in circuiti pneumatici di vario genere.

Per il collegamento del sensore a circuiti esterni, si può fare riferimento alla Fig. 5, rammentando che la massima corrente erogabile dal contatto di uscita è 0,5 mA (a 4,7 V di uscita e 5 V di alimentazione) e che l'impedenza di carico dell'uscita non deve scendere sotto i 10 kohm, almeno se si desidera mantenere una corretta proporzione tra la pressione rilevata e la tensione erogata.

REALIZZAZIONE PRATICA

Bene, descritti gli schemi elettrici e spiegato in che modo funzio-

nano i sensori alla base delle tre breakout board, 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. Ottenuto il c.s. disponetevi i componenti iniziando dal sensore;



per il MATERIALE

Le schede vengono fornite assemblate e comprese di cavetto.

Possono essere acquistate in Italia presso Futura Elettronica.

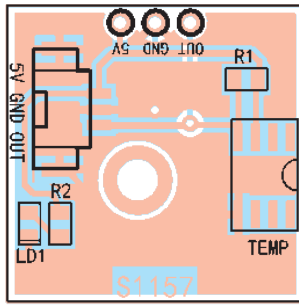
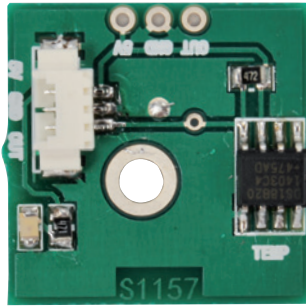
- Sensore di temperatura Euro 13,50 (cod. 7100-BREAKOUT001)
- Sensore di umidità Euro 18,50 (cod. 7100-BREAKOUT002)
- Sensore di pressione Euro 20,50 (cod. 7100-BREAKOUT003)

Il materiale va richiesto a:

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

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

[piano di montaggio **DS18B20**]



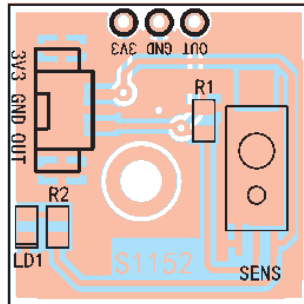
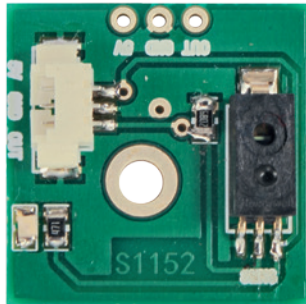
Elenco Componenti:

R1: 4,7 kohm (0805)
R2: 470 ohm (0805)
LD1: LED verde (0805)
U1: DS18B20Z+

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 **HIH-5030-001**]



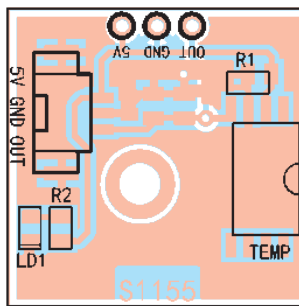
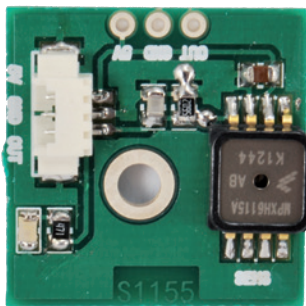
Elenco Componenti:

R1: 68 kohm (0805)
R2: 470 ohm (0805)
LD1: LED verde (0805)
U1: HIH-5030-001

Varie:

- Strip maschio 3 vie 90°
- Connettore JST 1,25mm 3 vie volante
- Connettore JST 1,25mm 3 vie da CS
- Circuito stampato

[piano di montaggio **MPX6115A6U**]



Elenco Componenti:

R1: 68 kohm (0805)
R2: 470 ohm (0805)
C1: 100 nF ceramico (0805)
C2: 47 pF ceramico (0805)
LD1: LED verde (0805)

U1: MPXH6115A6U

Varie:

- Strip maschio 3 vie 90°
- Connettore JST 1,25mm 3 vie da CS
- Connettore JST 1,25mm 3 vie volante
- Circuito stampato

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 flussante per agevolare la saldatura dei terminali.

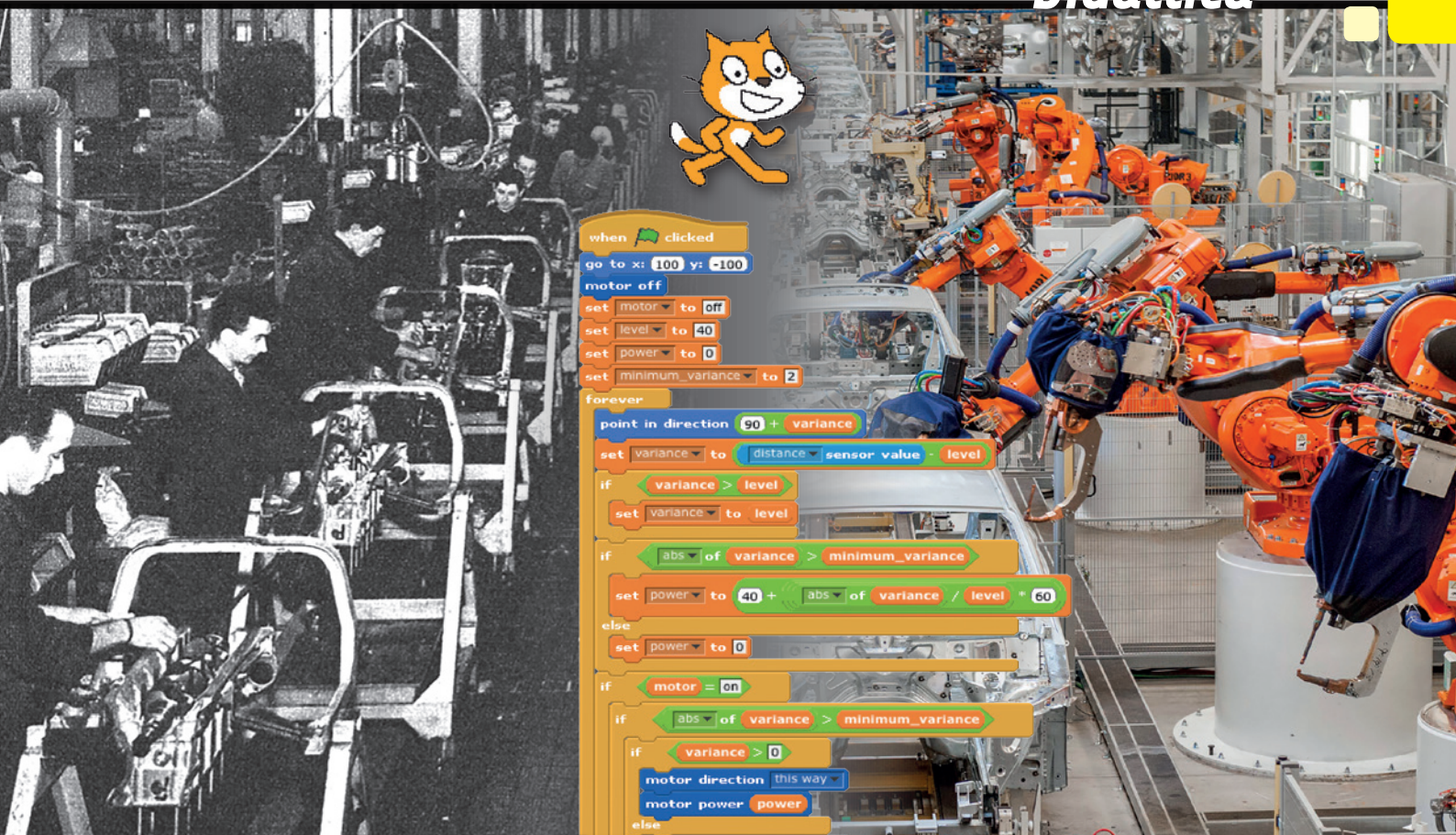
La disposizione e l'orientamento dei sensori sono visibili nei disegni di montaggio e nelle foto dei prototipi che vedete in queste pagine.

Per la breakout board contenente il DS18B20, completate il circuito con le resistenze, il connettore e il LED in SMD (la tacca su di esso contrassegna il catodo). Per quella contenente il sensore di umidità, montate, dopo il trasduttore, i resistori, il connettore e il solito LED in SMD.

Per la scheda del sensore di pressione, completate il circuito con i condensatori, il connettore 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 da tre poli: 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.



SCRATCH: LA PROGRAMMAZIONE RESA FACILE

..... di MARCO MAGAGNIN

In questi ultimi anni abbiamo presentato una serie di articoli dedicati al mondo embedded, con particolare attenzione alle soluzioni basate su GNU/Linux. Il filo conduttore di questa scelta è stato “facilitare” il passaggio a queste tecnologie a chi ha un background “elettronico”, ed è eventualmente già avvezzo al mondo informatico per aver utilizzato Arduino o altri microcontrollori “programmabili”. Sempre in base a questa “visione” abbiamo approfondito gli aspetti legati al primo impatto con tali sistemi, ossia la “messa in moto” iniziale, la configurazione e la personalizzazione per un utilizzo il più immediato possibile, senza partire con spiegazioni teoriche sull’architettura dei calcolatori, le reti, i sistemi operativi, i Web Server o i database. Per questi aspetti abbiamo proposto percorsi di confi-

Ambiente di sviluppo studiato particolarmente per chi vuole muovere i primi passi nel mondo della programmazione. È rivolto a tutti, senza limiti di età: conta solo la voglia di imparare divertendosi. Prima puntata.

gurazione guidata passo-passo. Quando necessitava la scrittura di qualche programma abbiamo fornito il listato, commentato con le indicazioni necessarie all'installazione e all'utilizzo. Per lo stesso motivo ci siamo sempre tenuti lontani dal fornire indicazioni sul come "programmare" da sé le proprie applicazioni. In compenso abbiamo sempre proposto l'utilizzo di linguaggi e ambienti alla portata di tutti, pur diffusamente presenti nello sviluppo delle applicazioni professionali d'avanguardia. D'altra parte, affrontare la spiegazione delle specifiche di ciascuno dei linguaggi utilizzati avrebbe occupato uno spazio inaccettabile in una rivista, senza peraltro essere di grande aiuto per chi non ha ancora "digerito" i fondamenti della programmazione. Un elenco di "referenze" non aiuta molto a capire come combinarle tra di loro per risolvere un problema in modo compiuto...

È anche vero che per imparare a programmare, così come a leggere e a scrivere, bisogna pur fare riferimento ad un qualche linguaggio: leggere significa portare a sé un messaggio creato da altri, mentre scrivere vuol dire esprimere in modo codificato un proprio pensiero, materializzarlo perché possa giungere ad altri ed essere compreso.

Una volta il concetto di programmazione si applicava pressoché unicamente alla programmazione dei calcolatori elettronici; oggi è esteso ad un'infinità di entità che compongono l'ambiente nel quale viviamo. Già si dice programmare il videoregistratore, piuttosto che la radiosveglia, o l'impianto di irrigazione, la gestione dell'energia nella casa, la sicurezza e anche i calcolatori. Qualcuno obietterà che un conto è programmare un elettrodo-

mestico con la pressione di una sequenza di tasti, altro è scrivere un videogioco interattivo sulla rete: vero, ma è anche vero che sono sulla bocca di tutti termini come domotica, teleassistenza sanitaria, digitalizzazione, Internet delle cose, innovazione, automazione, robotica e simili.

Tutti questi neologismi, pur non avendo ancora una definizione formalizzata, che permetta di riferirsi e individuare qualcosa di ben definito, nascondono una ben precisa realtà: spiegare a qualcosa che assomiglia molto ad un computer, cosa deve fare per ottenere ciò che abbiamo in mente; e questo si ottiene, almeno per ora, con la programmazione, che come qualsiasi attività espressiva, 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, fossero anche "accendere il barbeque di domenica alle 11 dentro il garage di fianco all'automobile"!

Ora si stanno affacciando sul mercato i primi sistemi che si ispirano alla filosofia dell'Internet delle cose. I più recenti modelli di lavatrici hanno a bordo un sistema operativo simil-android, che permette il collegamento in WiFi ed il controllo remoto dallo smartphone o da qualsivoglia dispositivo collegato in rete.

Nella "nuvola" fanno capolino i primi servizi che permettono di elaborare le cosiddette "scene" ovvero risposte attive condizionate dallo stato complessivo e ragionato di diversi "sensori" (per

esempio avvisare di spegnere l'asciugacapelli se la lavatrice è in funzione). Le istruzioni sono del tipo "IF THEN ELSE", "AND", "OR" e sono associate ad eventi di tipo cronologico (calendario) o di stato fisico come, alla mattina (?), se piove o peggio, se non piove, e simili.

Avere dimestichezza con i concetti base della programmazione non fa certo male anche in questi casi, che saranno sempre più diffusi. E aiuteranno molto anche chi vorrà o dovrà vedersela con i sistemi computerizzati "veri". Se analizziamo la crisi che stiamo attraversando, in compagnia del resto del mondo, è ascrivibile anche se non unicamente alla distanza esistente tra le capacità medie della popolazione attuale e le esigenze di conoscenza e formazione richieste dall'evoluzione del mondo produttivo e dei servizi.

In un passato non troppo remoto (che sarà mai un secolo...?) per vivere una vita normale, secondo gli standard di allora, non era necessario sapere leggere e scrivere: quello che serviva lo si imparava sul "campo" (In senso sia figurato, sia agricolo!).

Poi la rivoluzione industriale ha richiesto, a chi lavorava in fabbrica di sapere leggere e scrivere: in realtà, più leggere che scrivere. La crescita degli addetti alle macchine di produzione non poteva essere più coordinata con il metodo dell'esempio sul campo, ma bisognava essere in grado di leggere ordini di lavorazione, impostazioni di macchine, numeri di riferimento, eccetera. A seguire, scolarizzazione di massa, trasmissioni televisive "Non è mai troppo tardi", corsi delle 150 ore e via di seguito, per risolvere il problema "nazionale".

Ora siamo da capo: la nuova esigenza non è più avere "la-

voratori" che sappiano leggere ed eseguire disposizioni scritte, oltre naturalmente a saper "fare" ciò che viene richiesto; la nuova frontiera del lavoro, almeno per una parte non marginale, richiede "lavoratori" in grado di "spiegare" ai robot o più in generale a macchine e sistemi, come si devono comportare.

Il problema che si pone è come insegnare a "insegnare" alle macchine ed agli altri umani secondo modalità del tutto nuove e non ancora ben definite ed esplorate e, più in generale, a diventare parte attiva dei nuovi processi di insegnamento e produttivi e non solo "spettatori" o meri "clienti" dei processi stessi.

In pratica, gli obiettivi del "nuovo" modo di insegnare e di affrontare le esigenze dei nuovi scenari di lavoro si possono riassumere nei seguenti:

- riconoscere la "natura" dei problemi e scoprire diversi modi di trovare soluzioni che risolvano i problemi stessi;
- ragionare in modo astratto e "quantitativo" per poi trasformare ciò in processi concreti mediante l'individuazione di "variabili" che rappresentino le grandezze trattate ed algoritmi che ne rappresentino le relazioni (in breve: adottare la simulazione matematica alla risoluzione dei problemi);
- realizzare modelli che riproducano il comportamento di entità reali;
- prestare attenzione a tutti i dettagli di un problema implementandoli accuratamente negli algoritmi;
- adottare approcci strutturati alla progettazione e realizzazione delle soluzioni individuando i moduli ripetitivi, scambiando esperienze e codice con le comunità di ricerca e sviluppo, costruendo progetti

sempre più complessi utilizzando componenti modulari.

PERCHÈ SCRATCH

Questa lunga premessa serve a spiegare perché abbiamo deciso di proporre come "campo" di studio e di divertimento per apprendere l'arte della programmazione l'ambiente di sviluppo "Scratch".

Scratch è il risultato di lunghi anni di ricerche e di "confronti" filosofico-tecnologici sul tema dell'insegnamento della tecnologia informatica al più alto numero di persone possibile.

Scratch nasce nel Lifelong Kindergarten research group dei Media Lab dell'MIT, sulla base delle esperienze del Prof. Seymour Papert, uno dei fondatori del LAB stesso e realizzatore negli anni '60, del linguaggio LOGO, oltre che collaboratore della LEGO nello sviluppo dei LEGO Mindstorm.

Scratch è un linguaggio di programmazione didattico ed insieme al suo ambiente di sviluppo, ispirato alla teoria costruttivista dell'apprendimento, è stato progettato per l'insegnamento della programmazione tramite primitive visive. Scratch e il suo ambiente di sviluppo sono adatti a studenti, insegnanti e genitori, e possono essere utilizzati per progetti pedagogici e di intrattenimento che spaziano dalla matematica alla scienza, consentendo la realizzazione di simulazioni, la visualizzazione di esperimenti, l'esecuzione di animazioni e musica, l'arte interattiva, semplici giochi e, nel nostro caso, anche di interfacciarsi con sensori e circuiti elettronici.

CARATTERISTICHE DEL LINGUAGGIO SCRATCH

Al di là del suo aspetto "infantile" Scratch è un ambiente di svi-

luppo di tutto rispetto. Prevede un approccio, in un certo senso, orientato agli oggetti, dove gli "oggetti" sono fisicamente rappresentati da "spiritelli" "sprite", dotati di struttura (forme) metodi e costumi.

Presenta la possibilità di gestire le strutture logiche fondamentali, la programmazione ad eventi e la gestione di "messaging" tra diversi processi in esecuzione contemporanea. Scratch è un linguaggio di programmazione che consente di elaborare storie interattive, giochi, animazioni, arte e musica. Inoltre permette di condividere i progetti con altri utenti del web.

L'idea ispiratrice di questo linguaggio è che anche i bambini o le persone inesperte di linguaggi di programmazione possano arrivare ad imparare importanti concetti di calcolo matematico, ragionare in modo sistematico, pensare in modo creativo e anche lavorare in gruppi partecipativi. Scratch è caratterizzato da una programmazione "grafica" con blocchi di costruzione (blocchi grafici) creati per adattarsi l'un l'altro, ma solo se inseriti in una corretta successione; questa importante caratteristica evita errori nella sintassi.

Ambiente

L'ambiente Scratch è disponibile sia via web, sul sito ufficiale, sia come software installabile ed utilizzabile offline, disponibile per le piattaforme Windows, Mac OS X e GNU/Linux. Su Raspberry Pi ed altri microPC è incluso nella distribuzione ufficiale. L'ultima versione stabile è la 1.4.

Comunità

Lo slogan della comunità online di Scratch, che recita "Immagina, Programma, Condividi", sottolinea l'importanza della condivi-

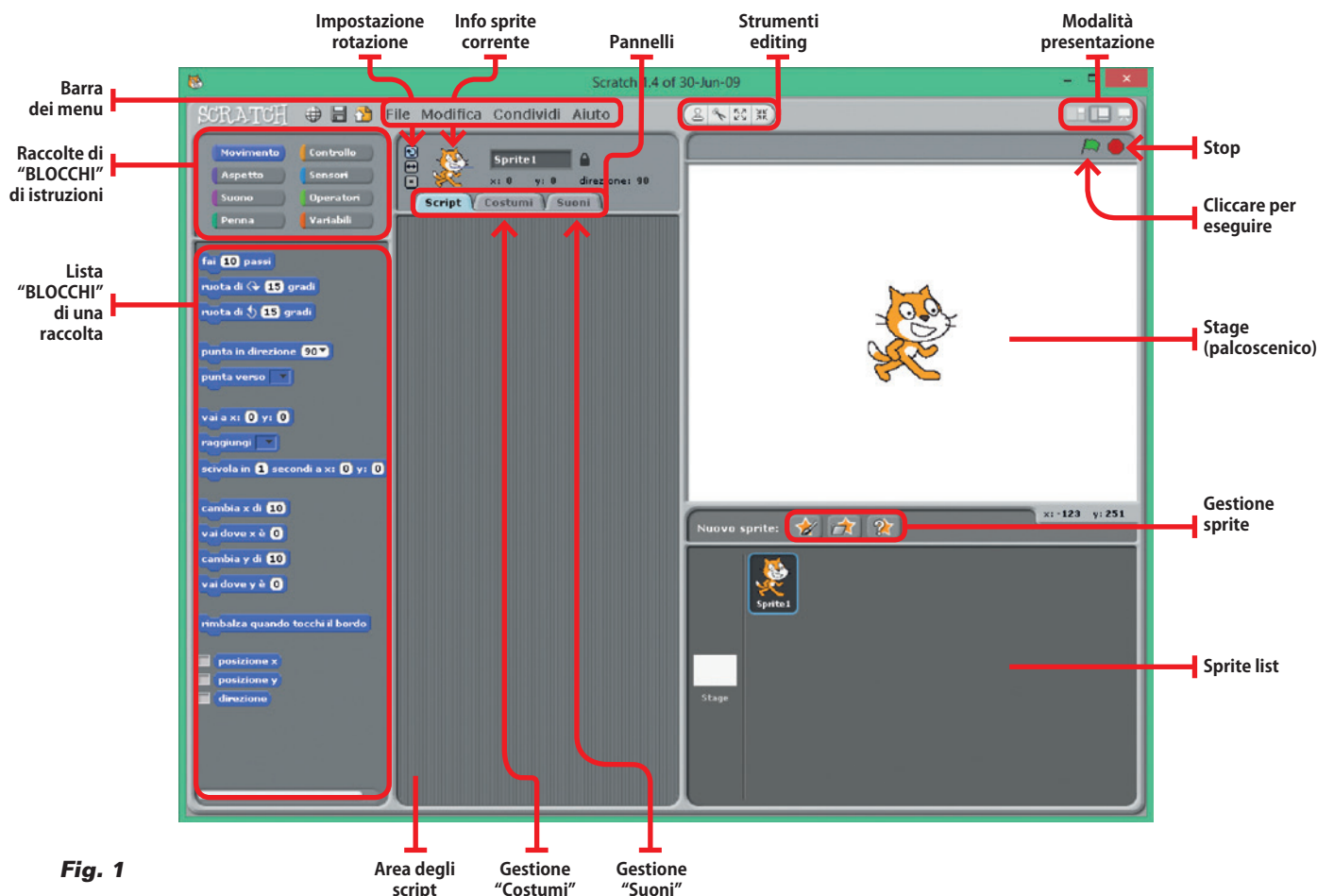


Fig. 1

sione e degli aspetti sociali della creatività nella filosofia alla base di Scratch.

I progetti Scratch, essendo a codice aperto, sono liberamente modificabili e utilizzabili per crearne di nuovi.

I progetti possono essere inviati direttamente dal programma al sito web di Scratch, e qualsiasi membro della comunità può scaricarne il codice per studiarlo o modificarlo in un nuovo progetto. I membri possono inoltre creare gallerie di progetti, commentare, "taggare" e aggiungere

progetti ai "preferiti". Tutti i progetti sul sito sono condivisi con licenza Creative commons "Share-Alike" e riprodotti su un browser (utilizzando Flash Player).

Il sito web riceve quasi 10 milioni di visite al mese e i membri registrati sono oltre 1.400.000 (di cui oltre 400.000 hanno condiviso progetti), per un totale di più di 4.200.000 progetti condivisi (più di un progetto inviato al minuto). Il sito web organizza periodicamente lo "Scratch Design Studio", competizione per incoraggiare la creazione e condivisione di progetti dalla grafica elementare. Nel 2008, la piattaforma della comunità online di Scratch (denominata "ScratchR") ha ricevuto una menzione onoraria all'Ars Electronica Prix.

Esiste anche una comunità online per educatori, denominata ScratchEd.

DOWNLOAD E INSTALLAZIONE DI SCRATCH

Per quanto riguarda il download e l'installazione, niente di più semplice: nell'ambito Windows basta scaricare il programma dall'indirizzo http://scratch.mit.edu/scratch_1.4/ ed eseguire il file .exe una volta terminato il download. Per quanto riguarda l'installazione nel sistema operativo Raspbian (Raspberry Pi) si utilizzano i soliti comandi:

```
apt-get update
apt-get install scratch
```

Questi valgono ovviamente anche per installare Scratch sulle distribuzioni Debian del sistema operativo GNU/Linux per altri microPC. Comunque nelle distribuzioni recenti, in genere, Scratch è già installato. In tutti i casi, per eseguire Scratch dal desktop basta cliccare

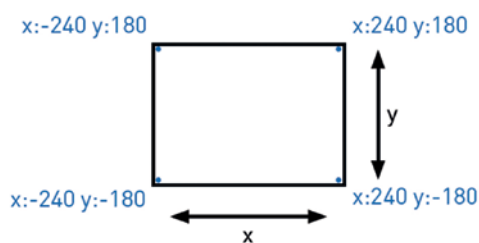


Fig. 2

sull'icona del collegamento che riproduce il muso di un gattino (la mascotte di Scratch).

GLI "INGREDIENTI" DI UN PROGETTO SCRATCH

In genere i progetti Scratch sono realizzati attorno ad oggetti grafici chiamati "Sprite"; ciascun oggetto può assumere diversi aspetti rappresentati da differenti "Costume" (costumi). Nel caso elettronico potremmo avere uno sprite "LED" con due costumi: "Acceso" e "Spento". Con gli sprite si può rappresentare pressoché qualunque entità: persone, mezzi di trasporto, componenti elettronici, animali o qualsiasi altra cosa. Come "costumi" si possono usare immagini di qualsiasi tipo, create con un editor di immagini, importate da un disco fisso, scaricate da un sito web o riprese da una fotocamera. Gli "sprite" possono essere "istruiti" ad eseguire una quantità di diverse azioni: si possono muovere, cambiare di aspetto, possono emettere suoni e musica o interagire con altri sprite. I "comandi" che guidano il comportamento degli sprite sono organizzati in veri e propri programmi, realizzati componendo moduli grafici che rappresentano le diverse tipologie di "istruzioni" e che hanno forme tali che possono essere "incastrati" tra di loro in modo da rispettare una corretta logica di programmazione. I programmi costruiti con i blocchi grafici sono definiti "script". Per eseguire uno script basta cliccarci sopra due volte con il tasto sinistro del mouse.

In seguito vedremo altre modalità di esecuzione degli script.

L'INTERFACCIA PER L'UTILIZZATORE

Nella Fig. 1 è rappresentata l'interfaccia di sviluppo dei progetti

"Scratch", un vero e proprio IDE di sviluppo grafico. Nella figura sono indicati le principali finestre e i comandi essenziali per interagire con l'ambiente di sviluppo.

Stage (Palcoscenico)

Partiamo dallo *stage*, che è la finestra di lavoro principale: il "palcoscenico" della nostra applicazione. In questa finestra viene realizzata l'interfaccia utente del progetto dove si inseriscono gli sprite, le animazioni e i comandi del progetto. Il posizionamento degli oggetti nello stage avviene con riferimento alla griglia grafica che caratterizza lo stage stesso: la relativa finestra è composta da 480 "unità" di larghezza e "360" "unità" di altezza; ciascuna locazione è individuata nella finestra con riferimento agli assi x, y secondo lo schema visibile in Fig. 2. Il centro della finestra è caratterizzato dalle coordinate 0,0. Per individuare visivamente le coordinate di una data locazione, basta posizionare il puntatore del mouse sulla locazione desiderata e leggere le coordinate nell'indicatore in basso a destra

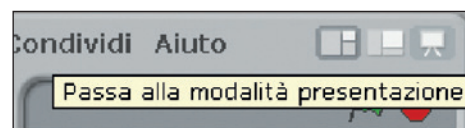


Fig. 3

della finestra. Cliccando sul pulsante "Passa alla modalità presentazione" visibile in Fig. 3, è possibile vedere la finestra di stage a schermo intero.

Per uscire dalla presentazione a schermo intero basta utilizzare il tasto "Esc".

Gli altri due tasti a fianco permettono di allargare o restringere l'area dello stage.

Gestione degli sprite nello stage

Quando si apre un nuovo progetto, la finestra di stage si presenta con all'interno lo sprite del gatto. Per gestire gli sprite si utilizzano i pulsanti (Fig. 4 e Fig. 5):

- **Disegna il tuo sprite**; apre l'editor grafico in stile "Paint" con il quale è possibile dare vita ai propri "sprite".
- **Scegli un costume per uno**



Fig. 4

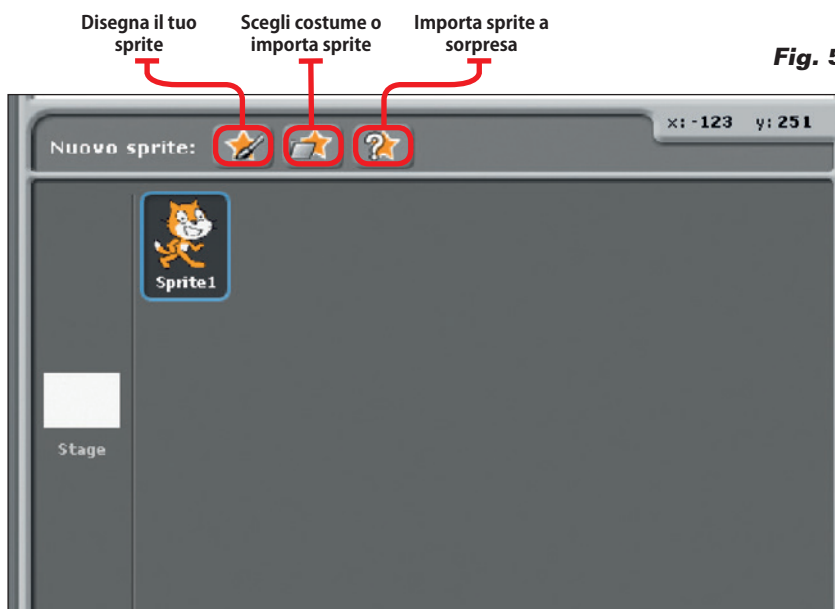


Fig. 5

sprite o importa uno sprite

completo; permette di scegliere ulteriori costumi per uno sprite già incluso nel progetto, oppure permette di importare uno sprite con i suoi costumi associati, in genere "riutilizzato" da un progetto esistente;

- **Importa uno sprite a "sorpresa";** aggiunge al progetto uno sprite a "caso" tra quelli presenti nella galleria degli sprite; se si desidera cancellare uno sprite, basta selezionare la funzione "forbici" dalla barra degli strumenti e cliccare sullo "sprite" da eliminare (un altro modo di eliminare uno sprite è cliccarvi sopra con il tasto destro del mouse e selezionare "delete" dal menu a tendina).

Galleria degli sprite

Nell' IDE di Scratch esiste una nutrita galleria di sprite preconfezionati, completi di costumi ed effetti sonori.

La funzione "Sprite List" mostra le anteprime di tutti gli sprite disponibili. Per ciascuno sprite vengono indicati il nome, da quanti script è utilizzato e quanti costumi sono disponibili. Per visualizzare e modificare gli script, i costumi o gli effetti sonori associati ad uno sprite; se siete nella Sprite List cliccate sull'anteprima dello sprite, oppure, nella finestra di stage, cliccate due volte direttamente sullo sprite. Lo sprite selezionato verrà evidenziato con un contorno blu.

Nella finestra "Sprite List" è possibile organizzare gli sprite a piacere, semplicemente cliccandovi sopra e trascinandoli nelle posizioni desiderate.

È possibile personalizzare l'aspetto di uno sprite assegnando costumi differenti, ma anche modificare gli sfondi dell'area di stage.

Fig. 6

fai 10 passi

Fig. 7

posizione

Fig. 8

passa a strumento 1

Per visualizzare e modificare gli script, gli sfondi e gli effetti sonori associati allo stage cliccate sull'icona "Stage" sulla sinistra della finestra "Sprite List".

Palette blocchi di istruzioni e area degli script

Per animare uno sprite si deve creare un programma (script) trascinando i blocchi grafici corrispondenti alle istruzioni richieste, dalla Blocks Palette nella Scripts Area.

Per eseguire un blocco di istruzioni basta farvi doppio clic. Per creare un programma bisogna impilare i blocchi grafici uno sull'altro, rispettando la loro "forma" e la possibilità che si "incastrino" correttamente tra di loro; in questo modo si ha un controllo preliminare sul rispetto della "sintassi" del linguaggio. In altre parole il programma è corretto dal punto di vista della sequenza e coerenza logica delle istruzioni. Non necessariamente, il programma eseguirà quello che noi desideriamo. Per questo dovremo eseguire i cicli di debug fino a quando otterremo i risultati voluti. Facendo doppio clic su un blocco di istruzioni si avvia l'esecuzione delle istruzioni stesse, dall'alto verso il basso, rispettando

- (1) Gran Piano Acustico
 - (2) Piano Brillante Acustico
 - (3) Gran Piano Elettrico
 - (4) Piano Honky-tonk
 - (5) Piano Elettrico 1
 - (6) Piano Elettrico 2
 - (7) Clavicembalo
 - (8) Clavinet
 - (9) Celesta
 - (10) Glockenspiel
 - (11) Carillon
 - (12) Vibrafono
 - (13) Marimba
 - (14) Xilofono
 - (15) Campane Tubolari
 - (16) Saltèrio
 - (17) Organetto
 - (18) Organo a percussione
 - (19) Organo Rock
 - (20) Organo da Chiesa
 - (21) Organo ad ancia
 - (22) Fisarmonica
 - (23) Armonica
 - (24) Fisarmonica da Tango
 - (25) Chitarra con corde in nylon
 - (26) Chitarra con corde in acciaio
 - (27) Chitarra Elettrica Jazz
 - (28) Chitarra elettrica
 - (29) Chitarra Elettrica Stoppata
 - (30) Chitarra con Overdrive
 - (31) Chitarra con distorsore
 - (32) Armonici di Chitarra
 - (33) Basso acustico
- altri...

le iterazioni e le istruzioni di controllo presenti. Per approfondire le funzionalità di un blocco basta cliccarvi sopra col tasto destro del mouse e selezionare la voce "Aiuto" dal menu a tendina. Quando si trascina un blocco all'interno della "Script Area" una linea bianca luminosa indica se un blocco può essere o no inserito nella posizione in cui si trova per costituire una connessione valida con i blocchi circostanti; la forma dei blocchi aiuta a capire se possono essere incastrati tra loro oppure se sono incompatibili.

Per spostare uno script bisogna "afferrarne" il centro del blocco superiore; se si "stacca" un blocco che si trova all'interno di una sequenza di istruzioni, verranno

staccati anche tutti i blocchi sottostanti. Alcuni blocchi presentano dei campi bianchi al loro interno (Fig. 6) che possono essere utilizzati per introdurre quelle che nei linguaggi di programmazione tradizionali sono definite “costanti” e “variabili”. Per modificare il valore della costante nel blocco bisogna cliccare sul campo bianco col tasto destro del mouse ed inserire un nuovo valore. Per utilizzare il campo come una variabile, occorre trascinare un blocco di forma simile a quello in Fig. 7 nel campo bianco.

È possibile utilizzare “variabili” predefinite, oppure crearne di nuove. Vedremo in seguito come creare e utilizzare le variabili che ci necessitano per i nostri progetti.

Alcuni blocchi presentano dei menu a tendina come quello in Fig. 8. Basta cliccare sul menu per aprire la tendina e cliccare una seconda volta sulla selezione desiderata.

La pagina che si apre cliccando sulla linguetta “Costumi” permette di vedere e modificare i costumi degli sprite. Nella Fig. 9 sono visibili i due costumi dello sprite “gatto”; come si vede, quello corrente appare evidenziato. Per impostare per lo sprite un costume differente da quello attuale, è sufficiente cliccare sul costume desiderato. Per creare nuovi costumi si può ricorrere a tre modalità:

- cliccare sul pulsante “Disegna” per disegnare un nuovo costume nell’ “Editor di immagini” (Fig. 10);
 - cliccare sul pulsante “Importa” per importare un’immagine da un file (Fig. 11);
 - cliccare sul pulsante “Webcam” per importare un’immagine da webcam (Fig. 12).
- Per quel che riguarda l’impor-

tazione, l’ambiente di sviluppo Scratch riconosce diversi formati di file d’immagine: JPG, GIF, BMP e PNG.

I costumi possono essere disposti a piacere nel pannello trascinando e riposizionando le relative immagini. Cliccando su un costume con il tasto destro del mouse, si può trasformarlo in un nuovo sprite oppure esportarlo in un file separato. Cliccando sulla linguetta “Suoni” si apre la pagina di gestione degli effetti sonori. Cliccando sui rispettivi pulsanti, si possono registrare suoni in diretta mediante i dispositivi audio in dotazione al sistema, oppure importarli da fonti esterne (Fig. 13); a tal proposito va detto che scratch accetta file codificati secondo gli standard MP3, WAV (non compresso), AIF ed AU campionati ad 8 o 16 bit (non è accettata la codifica a 24 bit).

In Fig. 14 è raffigurata la porzione dell’ambiente di sviluppo che riassume le informazioni principali sullo sprite: il nome, la posizione (espressa in coordinate x e y) e la direzione. In questo pannello è possibile modificare il nome dello sprite digitandolo nell’apposito campo.

Il campo “direzione” indica in quale direzione si muoverà lo sprite a fronte dell’esecuzione di un blocco “Muovi”: in particolare, il valore “0” indica un movimento verso l’alto, 90 verso destra, 180 in basso e -90 verso sinistra. La stessa informazione è evidenziata graficamente dall’orientamento della linea blu sull’immagine dello sprite. È possibile afferrare e trascinare la linea blu per modificare la direzione di spostamento dello sprite. Cliccando sull’immagine dello sprite si ripristina la direzione allo stato originale (direzione=90).



Fig. 9



Fig. 10



Fig. 11



Fig. 12



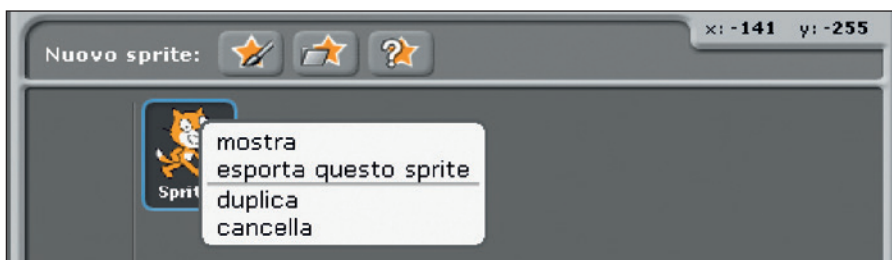
Fig. 13



Fig. 14

Per verificare come si comporta lo sprite durante i movimenti di rotazione utilizzate i pulsanti Rotation style:

- **Può ruotare** = il costume ruota in modo coerente con il cambiamento di direzione dello sprite;
- **Voltati solo a destra-sinistra** = il costume si gira verso destra o sinistra a seconda della direzione;



- **Non ruotare** = il costume non ruota durante i cambiamenti di direzione.

Per salvare lo sprite come un file separato, in modo da poterlo utilizzare in un altro progetto, cliccate sulla figura dello sprite nella "Sprite list" e selezionate l'opzione "Esporta" dal menu a tendina (Fig. 15).

La barra degli strumenti permette di eseguire le operazioni di editing sui diversi oggetti (Fig. 16):

- **Freccia** = modalità di utilizzo normale, afferra e sposta sprite, blocchi e oggetti in genere, si attiva cliccando fuori dalla barra degli strumenti;
- **Duplica** = esegue una copia dello sprite (selezionare duplica e, con il mouse a forma di timbro, cliccare sullo sprite, nella "sprite list", che si vuole duplicare);
- **Cancella** = elimina uno sprite dalla "sprite list";
- **Espandi sprite** = aumenta le dimensioni dello sprite;
- **Riduci sprite** = diminuisce le dimensioni dello sprite.

In Fig. 17 vediamo le diverse opzioni della barra del menu generale dell'ambiente di sviluppo, che permette la gestione dei progetti e la configurazione e personalizzazione dell'ambiente stesso, e la gestione del debug passo passo del programma. Sono presenti anche le opzioni



Fig. 16

di esportazione e condivisione del progetto.

Il pulsante con la bandierina verde fornisce un modo comodo per mettere in esecuzione diversi script contemporaneamente. Infatti il pulsante con la bandiera verde mette in esecuzione tutti gli script che hanno come "cappello" il blocco visibile in Fig. 18.

Nel modo "Presentation" la bandierina verde appare come un'icona nell'angolo in alto a destra dello schermo. Premere il tasto "Enter" ha lo stesso effetto di cliccare sulla bandierina; in questo modo si possono realizzare progetti composti da molti task che lavorano in concorrenza, in grado di gestire molteplici eventi e processi contemporanei. Già da questa possibilità, possiamo comprendere che siamo di fronte ad un ambiente di sviluppo didattico, orientato a imparare a programmare in modo grafico ed interattivo, e non certo ad un "giocattolo". Con questo ambiente possiamo realizzare videogiochi interattivi, ed anche applicazioni in grado di interfacciare sensori e moduli di ingresso e uscita elettronici. Passiamo a descrivere i diversi tipi di "mattoni" o blocchi che utilizzeremo per "costruire" i nostri programmi e progetti. Semplificando molto, nelle raccolte di blocchi (Block Palette) vi sono i quattro seguenti tipi principali di "mattoni".

- **Controllo**: questi blocchi permettono di realizzare la logica principale dei programmi e riproducono graficamente le tre principali strutture logiche della

Fig. 15

programmazione: sequenza, alternanza (if, then else oppure case) e iterazione (for, while). I blocchi hanno degli incavi nella parte superiore e delle protuberanze nella parte inferiore che fungono da "chiavi" per il loro corretto "impilaggio". I blocchi, come abbiamo già visto, hanno al loro interno delle ulteriori "finestre" di forme diverse dove vengono inseriti gli "argomenti" delle diverse "istruzioni", definiti in Scratch come "reporters". In alternativa, alcune istruzioni mostrano un menu a tendina che offre una serie di scelte

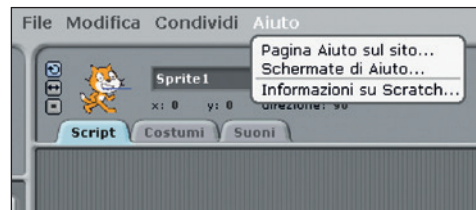
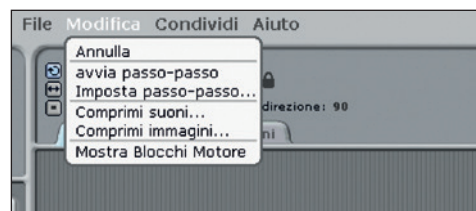


Fig. 17



Fig. 18



Fig. 19



Fig. 20



Fig. 21



Fig. 22

preconfezionate per “personalizzare” il comportamento dell’istruzione stessa. I blocchi di “controllo” della sequenza di elaborazione hanno forme a “C”, come le istruzioni di controllo delle iterazioni ed a “E” come nelle istruzioni di controllo delle alternanze soggette a condizioni che possono verificarsi o meno. Nel caso vero, viene eseguita la sequenza di istruzioni contenute nella “bocca” superiore della “E”, in caso contrario il blocco di istruzioni contenute nella “bocca” inferiore. Anche gli interni delle “bocche” sono conformati graficamente in modo da poter accogliere solo blocchi di forma che combaciano con la forma dell’alloggiamento (Fig. 19).

- *Cappelli (Hats)*: questi blocchi (Fig. 20) hanno la parte superiore arrotondata e sono utilizzati per indicare l’inizio di ciascun “blocco” di istruzioni. Nella stessa area di script è possibile inserire più blocchi, ciascuno con il proprio “cappello” che resta in attesa dell’evento in

base al quale dare inizio alla sequenza di istruzioni sottostante. Si va dal semplice clic sulla bandierina verde all’attesa della pressione di un tasto sulla tastiera, all’azione su uno sprite, ad un messaggio da un altro blocco o ad un evento legato ad un sensore esterno, come vedremo nelle prossime puntate. Grazie a questi blocchi, utilizzando Scratch si possono realizzare applicazioni “basate su eventi”, come si suole dire e strutturare i programmi in modo molto simile a quella che viene definita “programmazione orientata agli oggetti”. Ottenere lo stesso risultato didattico, utilizzando linguaggi tradizionali, richiederebbe uno sforzo di logica ed astrazione di dimensioni tali che rappresenta la principale causa di “abbandono” da parte di chi affronta per la prima volta la programmazione secondo il metodo “tradizionale”.

- *Comandi (Istruzioni nei linguaggi classici)*: rappresentano le singole azioni che devono essere eseguite per completare una funzione di senso compiuto (Fig. 21).
- *Reporters*: questi blocchi, dalle forme simili a quelle visibili in Fig. 22 sono predisposti per “incastrarsi” negli appositi spazi all’interno di altri blocchi. Ovviamente questi blocchi possono essere incastrati unicamente in aree della stessa forma. I blocchi di forma arrotondata “riportano” il valore delle variabili espresse dal nome del blocco stesso mentre i blocchi con le

estremità appuntite riportano la condizione di “vero” o “falso” dell’espressione contenuta al loro interno.

Alcuni blocchi di questa “famiglia” hanno un campo di spunta (check box) affiancato alla forma grafica che li rappresentano. Cliccando sul campo di spunta (attivandolo), si fa apparire nell’area di stage un elemento di visualizzazione che evidenzia di volta in volta il valore assunto dalla variabile. Il campo di visualizzazione può assumere tre formati differenti con i significati visibili nella Fig. 23.

Per passare da un formato all’altro del campo cliccate due volte sul campo stesso oppure una sola volta col tasto destro del mouse. Cliccando col tasto destro del mouse su un indicatore a slitta è possibile modificare i valori di minimo e massimo.

UN PRIMO ESEMPIO

Prima di concludere questa prima puntata, presentiamo un semplice esempio per iniziare a comprendere la “logica” sottostante l’ambiente di sviluppo ed il linguaggio Scratch. Per questa volta non rendiamo disponibile il listato sul sito di Elettronica In. Il progetto in questione consiste nel creare uno sprite a forma di “palla” da fare muovere all’interno dell’area di stage facendolo “rimbalzare” quando tocca uno dei bordi. Il tipo di rimbalzo è simile a quello che avviene su un tavolo da biliardo. Per prima cosa ci serve una palla, non il gatto, quindi clicchiamo



Una casella con il nome ed il valore della variabile



Una casella di dimensioni maggiori con il solo valore della variabile



Una casella con nome, valore ed un cursore che permette di modificare il valore della variabile, chiaramente utilizzabile per le sole variabili in input

Fig. 23

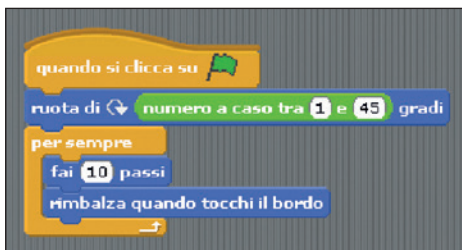


Fig. 24

col tasto destro del mouse sullo sprite del gatto nella "Sprite list" e selezioniamo "Cancella". Ora selezioniamo "Seleziona nuovo sprite da file" e dalla cartella "Things" scegliamo una "Palla". Rinominiamo il nostro sprite in "Palla".

Ora creiamo uno script per fare rimbalzare la palla all'interno dell'area di stage. Trasciniamo nell'area degli script un "Capello" del tipo, "Quando si clicca su bandierina verde". Per fare in modo che, ogni qualvolta eseguiamo il programma, la palla esegua un percorso sempre diverso, impostiamo un angolo di rotazione differente per ogni volta.

Trasciniamo un blocco blu con l'istruzione "ruota di ... gradi" e nel campo bianco della variabile trasciniamo un blocco "nume-

ro a caso tra ..." e fissiamo gli estremi dell'intervallo tra 1 e 45. "Numero a caso" (Random) è una funzione presente in pressoché tutti i sistemi di calcolo per generare un numero casuale in assoluto o all'interno di un intervallo, se quest'ultimo è indicato, come nel nostro caso.

Inseriamo sotto al blocco blu un blocco che racchiuda le istruzioni che devono essere eseguite in continuazione, ovvero quelle che costituiscono il corpo principale del programma. Trasciniamo il blocco a "C" "per sempre". Al suo interno inseriamo un blocco di avanzamento "fai <x> passi" ed indichiamo in "10" i passi di avanzamento ogni volta che viene eseguito il blocco. Lo trasciniamo all'interno della "bocca" della "C".

Ora dobbiamo fare in modo che quando la "palla" tocca un bordo, rimbalzi tornando all'interno dell'area di stage. Abbiamo un blocco che fa al caso nostro "rimbalza quando tocchi il bordo". Potete eseguire il "programma"



Fig. 26



Fig. 27

cliccando sulla bandierina verde in alto a destra (Fig. 25).

Carino, no?

Con una manciata di blocchi abbiamo scritto un programma da far venire il mal di testa se avessimo utilizzato un linguaggio "tradizionale".

In realtà in questi pochi blocchi sono presenti due delle "strutture" di un linguaggio di programmazione (sequenza e iterazione) oltre ad una prima funzione basata su eventi. Divertitevi a trovare funzionalità da aggiungere al giochino e scoprire come fare. Per eliminare un'istruzione dallo script bisogna prima "staccarla" dallo script, trascinandola in una zona libera dell'area degli script, poi cliccarvi sopra col tasto destro del mouse e selezionare "Cancella".

Un piccolo suggerimento: aggiungete nell'area degli script il modulo visibile in Fig. 26; così facendo, ogni volta che riuscite a cliccare sulla palla in movimento, questa cambierà "colore". Non vedete cambiare colore? Rallentate il movimento della palla come visibile in Fig. 27 e riproverete, cliccando il centro della palla.

Già avete implementato un micro video gioco. E questo è solo l'inizio.

Per ora ci lasciamo, in attesa di tornare a parlare presto di Scratch e delle sue magnifiche possibilità di utilizzo.

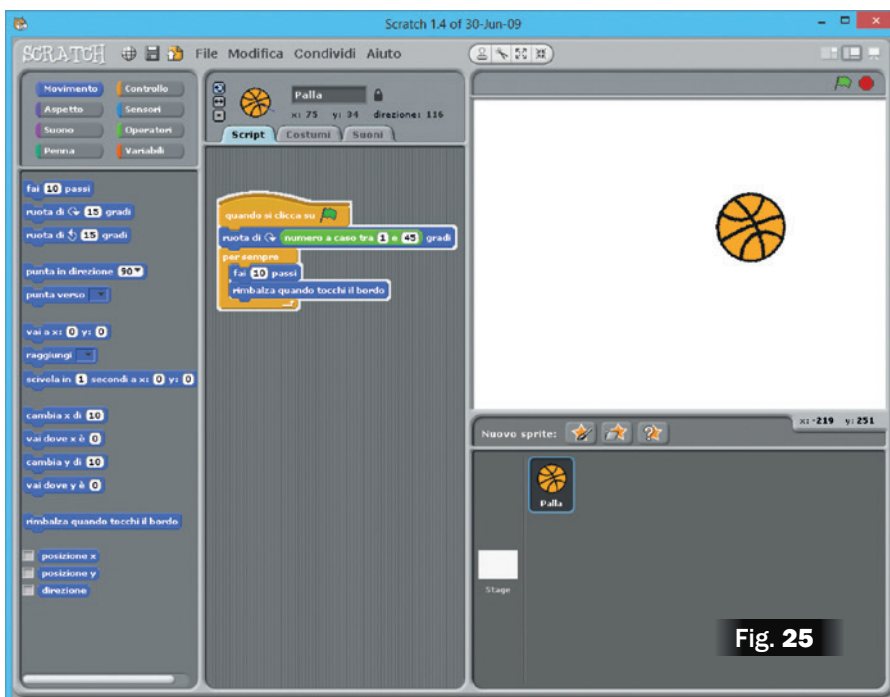


Fig. 25

SCANSIONI 3D SMART CON KINECT

Le ultime versioni del software ReconstructMe consentono di impiegare Microsoft Kinect e sensori simili per ottenere file di scansioni tridimensionali con molti meno passaggi e senza bisogno di riparazioni. Vi spieghiamo come.



di DAVIDE SCULLINO

La scansione di oggetti solidi, meglio nota come scansione 3D, consente di ottenere modelli a computer con i quali creare scenari virtuali digitali, personaggi da integrare in animazioni e videogiochi, ma anche e soprattutto file da utilizzare per la stampa 3D e per la lavorazione mediante frese CNC. Da quando le stampanti 3D

sono state sdoganate verso il grande pubblico, poter scannerizzare un viso, un busto, ma anche un oggetto è diventato un'ambizione anche di molti hobbisti, ma purtroppo le apparecchiature per farlo, ovvero gli scanner 3D, sono molto costose e indubbiamente fuori dalla portata delle tasche dell'utente comune. Qualche mese fa abbiamo descritto il progetto di uno scanner 3D per



Fig. 1 - Da destra, Microsoft Kinect e il suo alter-ego prodotto da Asus.

oggetti basati su una telecamera, due laser lineari e un piatto girevole motorizzato, il tutto governato dal software LabVIEW di National Instruments; si tratta di qualcosa di molto complesso, anche se più abbordabile dei prodotti professionali, tuttavia per la scansione 3D esistono soluzioni più pratiche, che non richiedono di realizzare hardware: l'esempio più immediato è Kinect di Microsoft. In queste pagine spieghiamo proprio come utilizzare Kinect (il sensore di movimento dei giocatori della Xbox) per acquisire la forma di oggetti mediante software disponibili anche in Internet in forma gratuita. La scansione siffatta non è una novità, infatti l'avevamo descritta già nel fascicolo n° 173; tuttavia, in passato la procedura per ottenere modelli utilizzabili nella stampa 3D era molto laboriosa e richiedeva tanti passaggi, per ognuno dei quali serviva un software specifico. Oggi ottenere modelli 3D con Kinect è molto più semplice, perché si fa tutto con due soli programmi. In queste pagine vi spiegheremo come procedere.

ACQUISIRE CON KINECT PER XBOX360

Vediamo come utilizzare Kinect per Xbox360 per acquisire immagini e creare modelli 3D: occorrono un Personal Computer dotato di scheda grafica accelerata ATI, NVidia o integrata nei chipset Intel, ed il software ReconstructMe con gli opportuni driver per il sensore utilizzato. La versione di ReconstructMe non è quella di

cui abbiamo parlato nel fascicolo n° 173, ma uno strumento molto potente che permette di controllare Kinect per ottenere da esso dei modelli 3D senza discontinuità in superficie.

ReconstructMe è un software reso disponibile sia in forma professionale (a pagamento) sia in versione gratuita (per l'uso non commerciale) con alcune limitazioni. Un programma alternativo a ReconstructMe è Kscan3D, ma in questo caso la versione Trial non permette di esportare il modello e quindi non consentirebbe di arrivare alla stampa (serve solo a valutare le funzionalità). Con ReconstructMe si può utilizzare, come sensore di acquisizione, oltre a Kinect, il dispositivo Asus Xtion Pro oppure il PrimeSense Carmine Apple.

Come esempio di scansione, spieghiamo come riprendere il busto di una persona e poi stamparlo. Partiamo dalla considerazione che siccome Kinect è stato progettato per acquisire l'intera persona e individuarne i movimenti, le ottiche e il firmware che produce i dati di profondità sono stati tarati su un volume schematizzato nella Fig. 3.

Kinect permette anche il riconoscimento delle espressioni facciali, piuttosto che di alcune caratteristiche del volto, ferma restando la definizione della mappa di profondità, che è di 320x240 punti. Per migliorare il risultato, si possono utilizzare varie tecniche di smoothing (smussamento degli spigoli) e campionamento multiplo,

sfruttando anche il fatto che il soggetto viene "esplorato" e non rimane immobile.

Si tratta di un approccio comunque molto impegnativo sotto il profilo dei calcoli, non a caso il software ReconstructMe sfrutta, per i calcoli, i processori grafici (GPU) delle schede video, così da ottenere un risultato accettabile in termini di frame al secondo. Il software, infatti, deve fare il tracking degli oggetti, aggiungendo alla nuvola di punti le nuove misurazioni e correlandole a quelle precedenti. In pratica, quando un soggetto ruota, il software riconosce la struttura della superficie già acquisita anche nella nuova mappa di profondità e quindi riesce ad "agganciare" i punti comuni, mentre aggiunge quelli non ancora gestiti.

Il software prevede due modalità di funzionamento: normale (Standard, con un volume di 1x1x1m suddiviso in 256 elementi per ciascuna dimensione) oppure ad alta risoluzione (Hi-res, con il medesimo volume, ma 512 elementi). Siccome ReconstructMe lavora con la scheda grafica e non con la CPU, è compatibile solo con quelle previste dagli sviluppatori, elencate nel software



Fig. 2 - Il cavo di collegamento di Kinect sembra un USB ma non lo è: va inserito nella femmina del cavo USB fornito con l'apparato, il quale termina dall'altro lato sull'alimentatore da rete.

Il dispositivo Kinect

e nel sito web del programma; comunque anche se una scheda è dichiarata compatibile, potrebbe non esserlo in entrambe le modalità di funzionamento. Solo il test sul campo può dare certezze.

QUALE SOFTWARE INSTALLARE

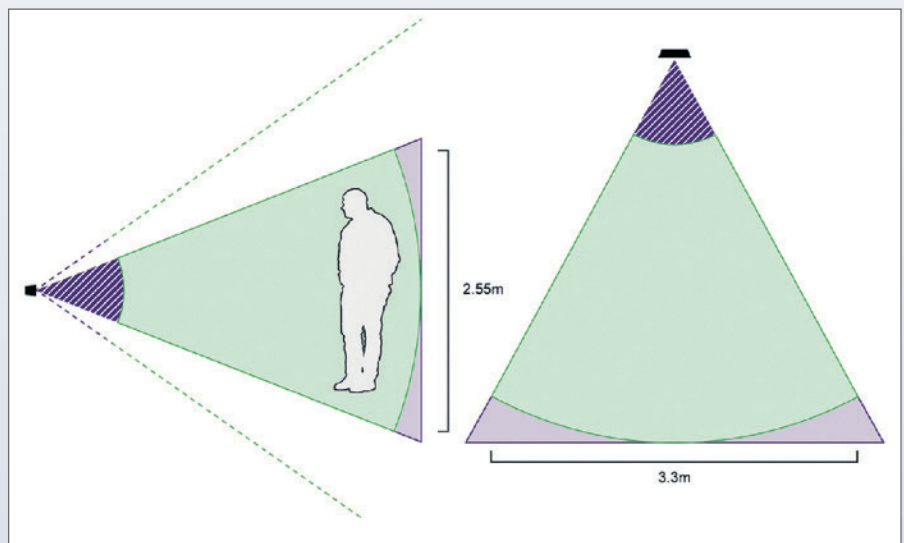
Per creare il nostro modello 3D dobbiamo aver installato il software ReconstructMe e l'SDK per la Kinect scaricabile dal sito Internet Microsoft (www.microsoft.com/en-us/kinectforwindows/), il quale fornisce una serie di driver che permettono di utilizzare le varie versioni di Kinect. ReconstructMe si scarica dal sito web <http://reconstructme.net>, facendo clic sul pulsante *Download* visibile a sinistra della pagina corrispondente; il software include una serie di profili predefiniti. In pratica, scaricate il file ad estensione .msi e procedete con l'installazione. L'installer di ReconstructMe, sostanzialmente vi chiede una directory in cui installare il tutto.

Come accennato, la versione liberamente utilizzabile di ReconstructMe (abbiamo provato la 2.1) presenta una limitazione: il file prodotto da ReconstructMe viene "macchiato" dal logo del programma e da una serie di sfere disposte a reticolo intorno al soggetto; ma di questo non preoccupatevi, perché tra breve spiegheremo come ottenere il soggetto "pulito". Per la parte relativa alla scheda video, dovete avere dei driver

Malgrado sia commercializzato come "sensore" per la Xbox, Kinect è qualcosa di più, infatti contiene una telecamera RGB con risoluzione di 640x480 pixel, abbinata a un'ulteriore telecamera, dalla risoluzione inferiore (320x240 pixel) sensibile solo agli infrarossi; questa seconda telecamera serve a rilevare le immagini prodotte dalla riflessione dei raggi IR proiettati sulla scena da osservare mediante l'apposito proiettore a infrarossi integrato nel pannello frontale del dispositivo. Oltre ai sensori video, in Kinect è presente un accelerometro triassiale per rilevare eventuali vibrazioni e movimenti. Ma non è tutto, perché Kinect mette a disposizione quattro microfoni, disposti radialmente, utilizzati, sfruttando la riflessione delle onde sonore all'interno della stanza, per la calibrazione dell'ambiente in cui si trova. Con tale soluzione (che qui non serve...) è possibile filtrare impercettibili eco e rumori di fondo durante l'utilizzo dei comandi vocali, in modo da consentire un comando a voce davvero efficace e immediato. Oltre ad avere a disposizione questa vastità di sensori, Kinect integra un motore utilizzato per farlo muovere in su e in giù allo scopo di seguire meglio i movimenti dei giocatori. Per la connessione con il mondo esterno, Microsoft Kinect uti-

lizza un cavo simile, per sagoma, a quello USB, dal quale differisce per il numero dei collegamenti: infatti il cavo USB è composto da 4 fili (due per l'alimentazione e altri due per la comunicazione) mentre quello di Kinect è dotato di 9 poli; per collegarsi ad esso è dunque necessario un adattatore che fornisca l'alimentazione ausiliaria a 12 V, oltre ai normali collegamenti dell'USB. Per usare Kinect come scanner 3D, l'hardware interessato occorrente è solo quello riguardante le telecamere e il proiettore all'infrarosso. Kinect funziona così: il proiettore a infrarossi proietta nello spazio di fronte ad esso un pattern ben preciso costituito da una miriade di punti equidistanti; analizzando l'immagine ottenuta attraverso la telecamera sensibile a tali raggi IR, Kinect determina le distanze tra i vari punti ricavandone distanza e inclinazione dell'oggetto illuminato. Siccome il fascio IR parte da un'area ristretta e si allarga man mano che la distanza aumenta (è all'incirca conico...), dove i punti risultano molto vicini significa che l'oggetto è prossimo al sensore; viceversa, se la distanza tra i punti è notevole, vuol dire che l'oggetto si trova più lontano. Unendo la totalità delle informazioni si riesce a ricavare la forma; per capire meglio la cosa, immaginate di appoggiare su un oggetto un lenzuolo a pallini: è possibile dedurre la forma di tale oggetto in base a come si dispongono i pallini man mano che il lenzuolo avvolge l'oggetto stesso.

Fig. 3 - L'area scansionata da Microsoft Kinect: per riprendere una persona, il costruttore consiglia di posizionare l'apparato a 1,8 metri da essa; per i busti basta meno.



Modello	Standard	Hi-res
ATI FirePro V3800	3	1
ATI FirePro V4800	3	1
ATI FirePro V5900	3	3
ATI Radeon HD 2600 Pro	2	1
ATI Radeon HD 4570M	1	1
ATI Radeon HD 4870	1	1
ATI Radeon HD 5145M	1	1
ATI Radeon HD 5650M	3	2
ATI Radeon HD 5700	3	3
ATI Radeon HD 6490M	2	1
ATI Radeon HD 6570	3	2
ATI Radeon HD 6850	3	3
ATI Radeon HD 7640G	2	1
ATI Radeon HD 7850	3	3
INTEL Dual Quadcore	2	2
INTEL HD Graphics 3000	2	1
NVIDIA Geforce 9500GT	2	1
NVIDIA Geforce 9800GT 3D	3	3
NVIDIA Geforce 9800GTX	3	3
NVIDIA Geforce G 102M	2	1
NVIDIA Geforce GT 240	3	2
NVIDIA Geforce GT 330M	2	1
NVIDIA Geforce GT 430	3	3
NVIDIA Geforce GT 440	3	3
NVIDIA Geforce GT 555M	3	3
NVIDIA Geforce GT 630M	3	3
NVIDIA Geforce GTX 220	2	2
NVIDIA Geforce GTX 280	3	2
NVIDIA Geforce GTX 295	3	1
NVIDIA Geforce GTX 460	3	3
NVIDIA Geforce GTX 550Ti	3	3
NVIDIA Geforce GTX 560	3	3
NVIDIA Geforce GTX 560M	3	3
NVIDIA Geforce GTX 570	3	3
NVIDIA Geforce GTX 590	3	3
NVIDIA Geforce GTX 660M	3	3
NVIDIA Geforce GTX 670M	3	3
NVIDIA Geforce GTX 680	3	3
NVIDIA GT 540M	3	3
NVIDIA Quadro 2000	2	2
NVIDIA Quadro 3000M	2	2
NVIDIA Quadro 4000	3	2
NVIDIA Quadro FX 2800M	3	1
NVIDIA Quadro FX 3500M	3	1
NVIDIA Quadro FX 4800	3	3
NVIDIA Quadro FX 580	3	1
NVIDIA Quadro NVS 140M	1	1
NVIDIA Quadro NVS 295	2	1
NVIDIA Quadro NVS 3100M	2	1
NVIDIA Quadro NVS 4200M	2	1

Tabella 1
Compatibilità
con OpenCL
delle schede
video più
comuni:
1) non
supportata;
2) ottiene una
visualizzazione
con irregolarità
in superficie;
3) ottiene una
visualizzazione
fluida.

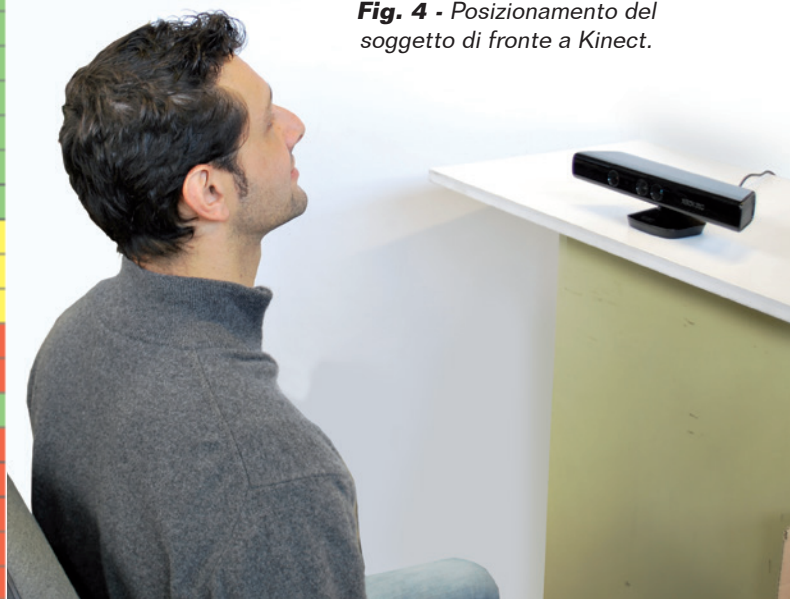
che supportano OpenCL, perciò consigliamo di visitare il sito del produttore della scheda o del computer per scaricare i driver più recenti. Come indicato nella tabella di compatibilità (**Tabella 1**), solo alcuni modelli di schede video dispongono di questi driver; per altre e più recenti non è certa e conviene consultare l'apposita sezione del sito <http://reconstructme.net>, che viene periodicamente aggiornata.

Se la vostra scheda video non è in elenco ma dispone di OpenCL, provate a usare ReconstructMe nelle due modalità disponibili, per scoprire cosa funziona e qual è il frame-rate cui potete lavorare. Eventualmente provate a cimentarvi nella modifica dei parametri dei file di configurazione seguendo le indicazioni della documentazione disponibile sul sito, magari definendo un diverso volume di scansione.

EFFETTUARE LA SCANSIONE

Per ottenere delle buone scansioni bisogna allestire un "set" con determinate caratteristiche. Va detto che rispetto alle versioni

Fig. 4 - Posizionamento del soggetto di fronte a Kinect.



del passato, quella più recente, quindi quella scaricabile dal sito al momento della stesura di questo volume, è più facilmente utilizzabile e permette di farsi la scansione da sé: basta sedersi su uno sgabello rotante e posizionare Kinect su un cavalletto stabile in modo che sia all'altezza del viso o comunque del busto. Se dovete riprendere un oggetto, usate un piatto rotante o il solito sgabello rotante e posizionate Kinect di fronte all'oggetto stesso. Se il cavalletto è regolabile in altezza è meglio, così è possibile provare a fare la scansione del soggetto da due angolazioni verticali diverse. Lo sgabello dovrebbe avere alle spalle una situazione il più neutra e piana possibile (un muro un po' distante è l'ideale).

Non serve una particolare illuminazione, perché in realtà Kinect effettua l'acquisizione analizzando la luce, prodotta dal proprio proiettore all'infrarosso, riflessa dalla superficie della persona. Qualche prova preliminare permetterà di prendere confidenza con il sistema di scansione, con il volume gestito, con la velocità di rotazione del soggetto e con i comandi che ReconstructMe permette di impartire.

Infine, affinché la scansione riesca bene il soggetto dovrebbe evitare di indossare capi con una superficie poco liscia (maglioni di lana o colletti di pelo). Notate altresì che i dettagli sottili come una montatura di occhiali non vengono rilevati, mentre degli occhiali da sole con una montatura spessa e lenti scure, sì.

In generale, ricordate che la profondità viene gestita proiettando una matrice di punti molto complessa e fitta, con luce infrarossa; ciò che non riflette in modo deciso la matrice, crea conseguentemente dei problemi

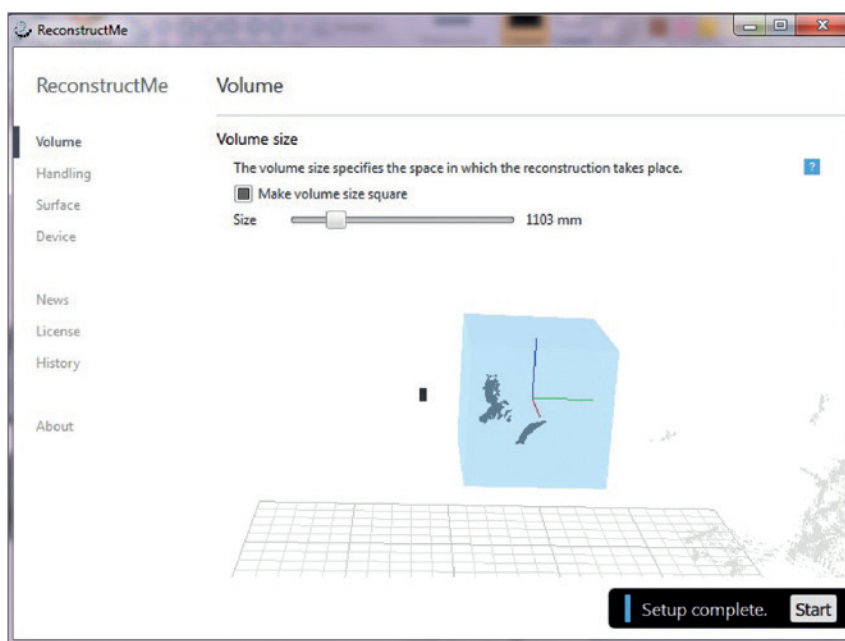


Fig. 5 - Interfaccia utente di ReconstructMe versione 2.1.

di acquisizione al sistema. Avviando ReconstructMe, si presenta la finestra che ne costituisce l'interfaccia grafica (**Fig. 5**), a sinistra della quale sono riportati i comandi: *Device* apre una finestra di dialogo dalla quale scegliere se far rilevare automaticamente al computer il dispositivo di acquisizione oppure (deselezionando il pulsante d'opzione *Automatically detect sensor*) scegliere manualmente il relativo driver (con *Browse* potete selezionare il percorso del relativo driver). Il comando *Volume* apre una finestra che visualizza un cursore con il quale definire la dimensione dell'area in cui effettuare la scansione. *Surface* apre la finestra dalla quale potete definire se desiderate una scansione colorata (per ottenere un modello da stampa 3D, non va scelta questa opzione) e se intendete ricavare un modello con impostazioni per la stampa 3D: questa è la nostra opzione, quindi selezionate *Selfie 3D Scan*.

Per avviare la scansione, basta

fare clic sul pulsante *Start* posizionato in basso a destra nella finestra principale; a questo punto il computer emette una serie di "bip" che invita a prendere posizione. Finiti i suoni, ha inizio la scansione e bisogna cominciare a girare lentamente sullo sgabello finché, completato un giro, il programma se ne accorge ed emette una combinazione di note diverse per comunicare che ha ripreso interamente la superficie del soggetto.

Notate che il tempo per mettersi in posizione può essere definito facendo clic su *Handling* e quindi trascinando con il mouse il cursore *Start Delay*; se lo variate, noterete che una volta fatto clic su *Start* cambierà la durata della serie di note acustiche.

L'aspetto più importante di questa fase, è trovare la posizione in cui il sensore lavora al meglio e riesce a produrre un ragionevole dettaglio di quanto inquadrato. Fate pratica provando a ruotare il soggetto ripreso e riprendendo da varie angolazioni verticali. La

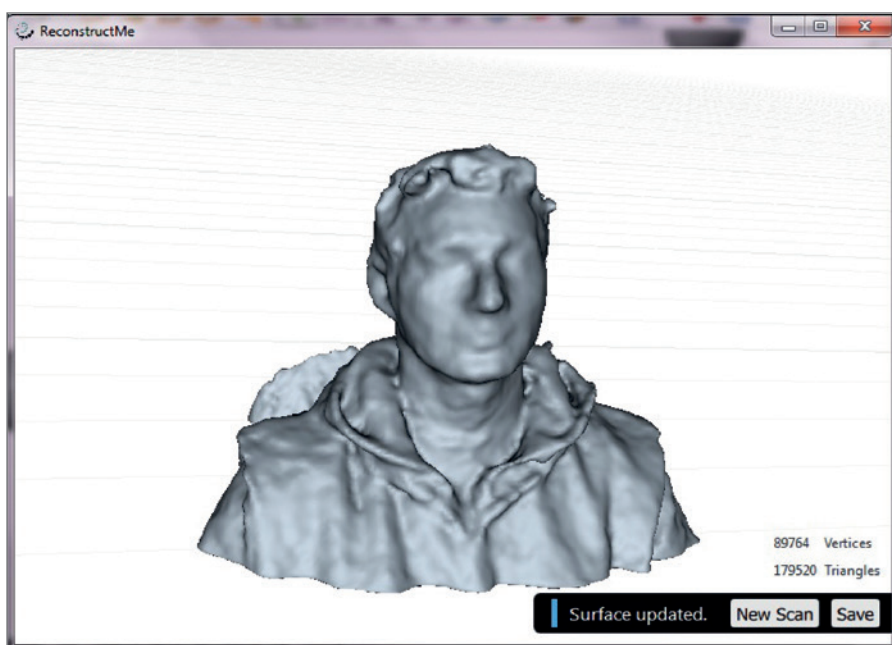


Fig. 6 - Il risultato della scansione effettuata con Kinect.

distanza minima cui posizionare il soggetto è una settantina di centimetri, il che è coerente con il fatto che Microsoft consiglia, per Kinect, una distanza dal giocatore di cui rilevare i movimenti, tipicamente di 1 metro e 80 centimetri. Il soggetto deve ruotare con lo sgabello a velocità possibilmente costante e non troppo rapida-

mente (un giro in una decina di secondi va benissimo). Ripetete questa operazione per produrre due o tre versioni del soggetto e scegliete quella venuta meglio. Ogni scansione completa mostra il risultato nella finestra di dialogo dell'interfaccia grafica (Fig. 6); in basso, la scritta *Surface updated* comunica il completamento della scansione; se il risul-

tato vi aggrada, fate clic su *Save*, altrimenti ripetete la scansione con *New Scan*. Il salvataggio genera un file ad estensione .ply cui il programma vi chiede di assegnare un nome e un percorso dove collocarlo.

CONVERSIONE DEL FILE

Il file ottenuto con questo procedimento non è stampabile in 3D, almeno che venga prima elaborato e poi convertito in STL con Meshlab; quest'ultimo software è disponibile gratuitamente per il download all'indirizzo <http://meshlab.sourceforge.net/>.

La prima cosa da fare dopo aver installato il programma è lanciarlo ed aprire in esso (con il comando *File>Open*) il file .ply contenente la scansione; apparirà qualcosa come quello mostrato in Fig. 7. Adesso bisogna ripulire l'immagine per lasciare solo il soggetto; ciò si ottiene facendo clic sull'icona *Select Connected Components in a Region* (lo stesso comando si può impartire dal menu *Edit*) e delineando con il mouse l'area che interessa tenere. Fatto ciò, dal menu *Filters*, fate clic su *Selection* e impartite il comando *Invert Selection* dal sottomenu che si apre; in tal modo si riesce a selezionare tutti gli elementi estranei in un solo colpo (altrimenti dovrete selezionarli con il mouse uno per uno e cancellarli individualmente). Premendo il tasto *Canc*, adesso eliminate le parti estranee e vedrete rimanere sullo schermo il solo soggetto ripreso (Fig. 9). Ora potete salvare il file esportandolo (comando *File>Export Mesh As*) in STL.

Notate che pur disponendo, Meshlab, di un filtro (si trova sotto il menu *Filters*) per "chiudere" i buchi del modello, non servono in quanto la versione di ReconstructMe disponibile da

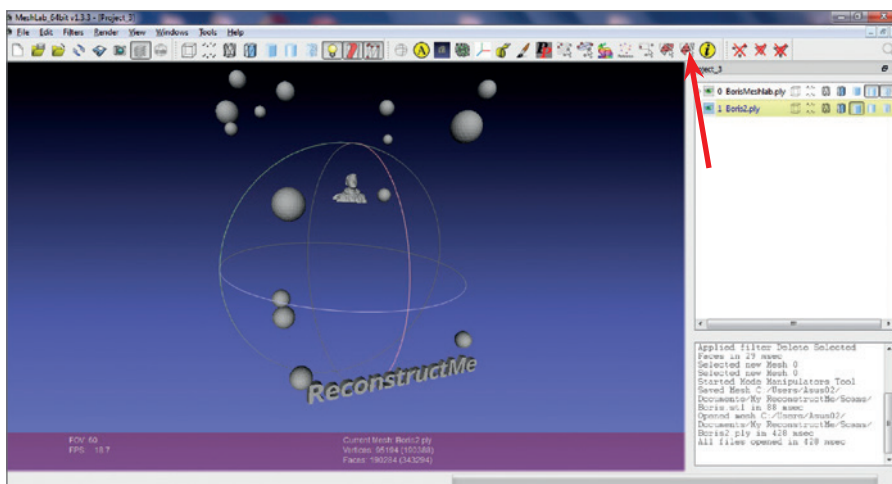


Fig. 7 - Il ritocco con Meshlab del file .ply generato da ReconstructMe: notare il logo e le sfere che "sporcano" il soggetto. La freccia indica l'icona del comando *Select Connected Components in a Region*.

un anno a questa parte provvede automaticamente a generare file .ply già sigillati da cui ricavare STL stampabili senza che sopravvengano errori. Possiamo invece usare, se il modello ci sembra un po' spigoloso, il filtro per smusarlo (*Filters>Smoothing*) come se fosse passato con della carta vetrata.

È possibile elaborare ulteriormente il modello ottenuto mediante un programma come Sculpttris (<http://pixologic.com/sculpttris/>) che permette, ad esempio, di limare parti sdoppiate (a volte nella scansione con Kinect protuberanze come il naso escono sdoppiate) o rilievi creati erroneamente durante la scansione, ovvero di aggiungere e modellare pezzi come se si lavorasse con della creta.

Il modello ottenuto da Meshlab è già stampabile e, siccome ReconstructMe genera file contenenti già una loro base, teoricamente non serve tagliarlo e quadrarlo per ottenere la base di partenza del primo strato della stampa 3D. Tuttavia se desiderate tagliare il modello a un'altezza diversa, dovete aprire il file .STL con Netfabb Studio (www.netfabb.com) potete ricollocare la base di appoggio all'altezza voluta (Fig. 10) ed eventualmente correggere l'orientamento del modello, cosa utile perché durante la scansione è facile che il soggetto non sia con la schiena dritta, ma piuttosto un po' incassato e curvo, perché per restare immobile ha scelto una posizione comoda. Con Netfabb si può correggere la posa ruotando di un certo numero di gradi il modello sull'asse X, ricordando che se il mento non risulta orizzontale, ma è leggermente piegato verso l'alto, sarà possibile stampare senza supporti, mentre se è dritto, sarà necessario attivare i supporti

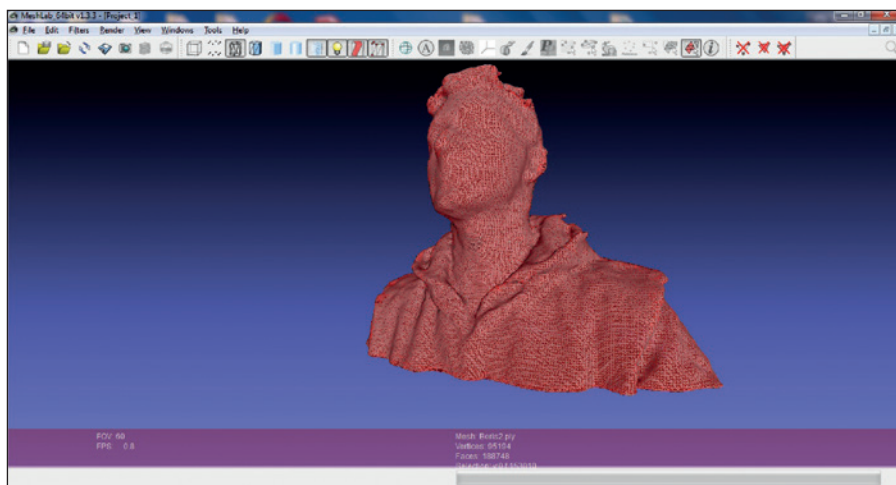


Fig. 8 - L'area selezionata in Meshlab.

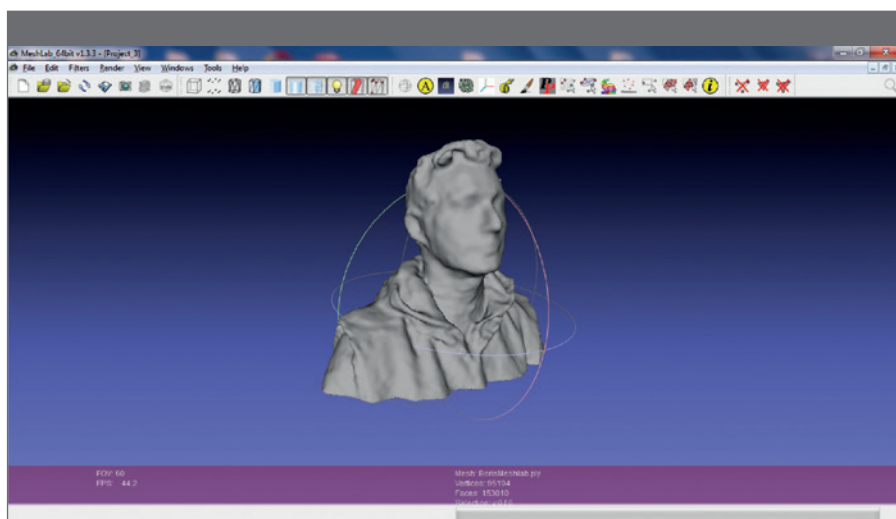


Fig. 9 - Il busto rimasto dopo la cancellazione del logo e delle sfere creati da ReconstructMe.

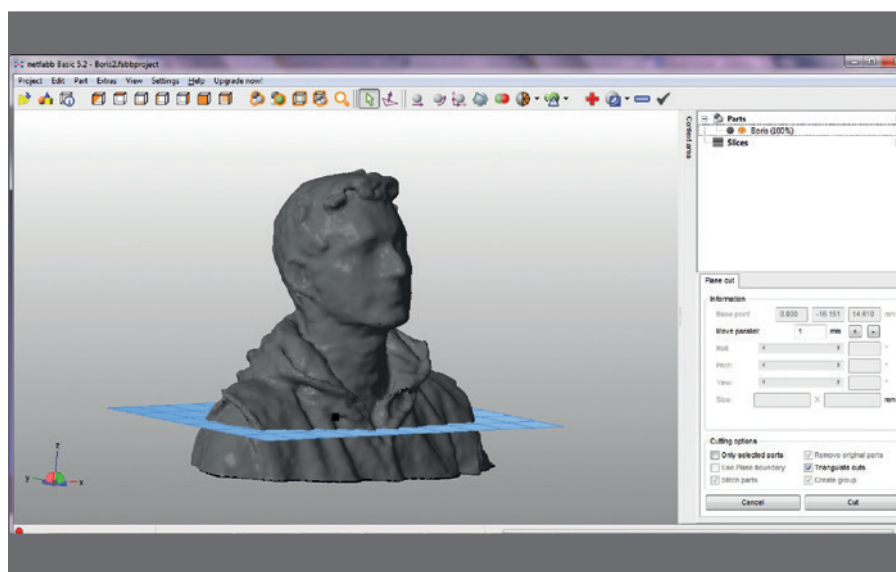


Fig. 10 - Taglio, in Netfabb Studio, della base del busto a un'altezza diversa da quella originaria.

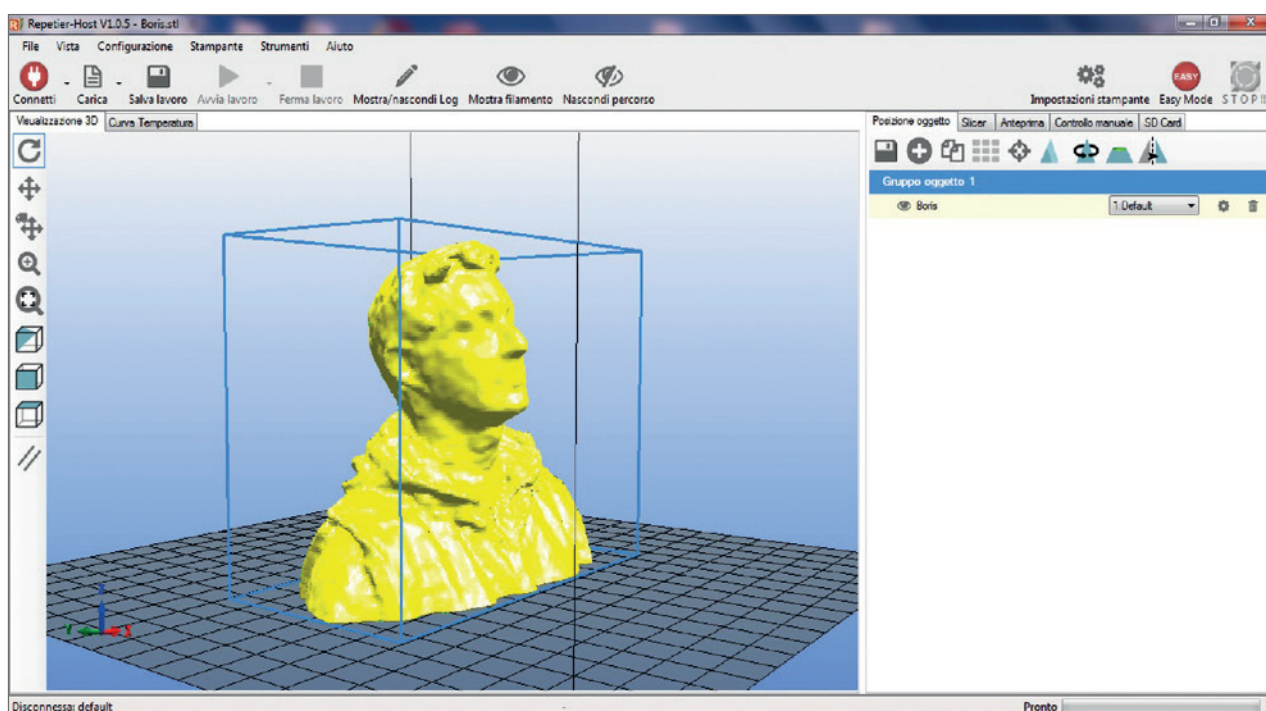


Fig. 11 - Importazione del modello in Repetier-Host e posizionamento sul piatto per la stampa.

di stampa. Con Netfabb Studio si può dare un'ulteriore "sistemata" al modello mediante la funzione di riparazione; ciò non serve se prendete l'.STL generato da Meshlab, ma se operate con Sculptiris e per errore "bucate" la superficie del pezzo. Netfabb Studio segnala con il punto esclamativo l'esistenza di problemi potenziali per la stampa: in questo caso, il software propone automaticamente la riparazione in fase di esportazione. Solo dopo aver effettuato lo slicing, però, si saprà se è tutto a posto o meno.

ADATTARE LA SCANSIONE ALLA STAMPA

Con il file STL si può passare alla stampa mediante Repetier Host. Se avete acquisito il modello con l'impostazione Selfie 3D Scan, il modello si potrà subito stampare con la vostra stampante 3D, dato che sarà grande al massimo 20x20x20 cm; senza impostare

tale opzione, il modello potrebbe essere più grande del massimo volume stampabile e in questo caso Repetier-Host lo presenta in un colore di "allarme" e lo fa pulsare (in tal caso utilizzate la funzione di ridimensionamento del modello e impostate la scala in modo da farlo rientrare nelle misure ammesse).

Usate il pulsante *Centra oggetto* per mettere il modello al centro del piatto di stampa (l'appoggio alla base del piano avviene automaticamente). La rotazione per mettere "in piedi" il busto va scelta, in gradi, per porre la base parallela al pianoX. Ripetete la centratura e l'allineamento. A seconda della fisionomia del soggetto, potete infatti avere dimensioni e proporzioni diverse (Fig. 11).

Ispezionate il modello da stampare utilizzando la rotazione della telecamera attorno al piano (la prima icona in alto a sinistra con la freccia circolare) facendo

soprattutto caso alle inclinazioni dietro alla nuca e sotto al mento per capire se ci sono problemi sulla stampa. Il passaggio successivo, quello dello slicing, potrebbe richiedere anche una quindicina di minuti, dato che il modello avrà sicuramente un elevato numero di triangoli. Per le impostazioni, vi suggeriamo di limitare i perimetri a 2 o 3 e di tenere il fill a un valore contenuto, tipo 15 o 20% per non perdere troppo tempo nel riempimento del busto. Al termine dello slicing, conviene ispezionare nuovamente il risultato e verificare che nelle zone critiche di nuca e mento non ci siano dei perimetri che si appoggiano al nulla. Se ne trovate, potete diminuire lo spessore degli strati, aumentando però il tempo di stampa, oppure tornare a Netfabb Studio per modificare l'inclinazione del busto e la conseguente angolazione del mento o della nuca.

CORSO

MPLAB X



di FRANCESCO FICILI
e VINCENZO GERMANO

Continuiamo il nostro viaggio alla scoperta di MPLab X, il nuovo ambiente di sviluppo integrato prodotto e distribuito da Microchip Technology. Iniziamo anche a conoscere i microcontrollori PIC32, i primi dispositivi a 32-bit prodotti da Microchip. Seconda puntata.

Nella puntata precedente abbiamo introdotto MPLab X, il nuovo ambiente di sviluppo lanciato da Microchip per lo sviluppo di applicazioni embedded sulle sue piattaforme ad 8, 16 e 32-bit, e basato sull'IDE cross-platform ufficiale di Oracle Netbeans. Questo ambiente, nei piani di Microchip, deve andare a sostituire MPLab IDE 8.x, che è stato l'ambiente di sviluppo ufficiale di Microchip negli ultimi anni ed è ormai in obsolescenza. A corredo del nuovo ambiente, Microchip ha lanciato una nuova linea di compilatori, XC8, XC16 ed XC32, che va a soppiantare la vecchia toolchain, costituita dai compilatori MPLab C18, C30 e C32.

In questa seconda puntata concentriamo maggiormente la nostra attenzione sui

microcontrollori PIC32, che costituiranno la piattaforma hardware principale di questo corso.

I MICROCONTROLLORI PIC32

Si tratta della più recente famiglia di microcontrollori prodotta dalla Microchip Technology ed è stata introdotta nel mercato nel novembre 2007. Questi potenti microcontrollori sono basati sul core a 32-bit MIPS M4K (licenziato dalla produttrice di IPcores inglese Imagination Technology), che gira ad una frequenza di 80 MHz, dotato di una pipeline a 5 stadi e di una ALU a 32-bit. I primi modelli prodotti sono stati mantenuti pin-to-pin compatibili con quelli della famiglia PIC24FxxGA0xx, in modo da semplificare eventuali migrazioni.

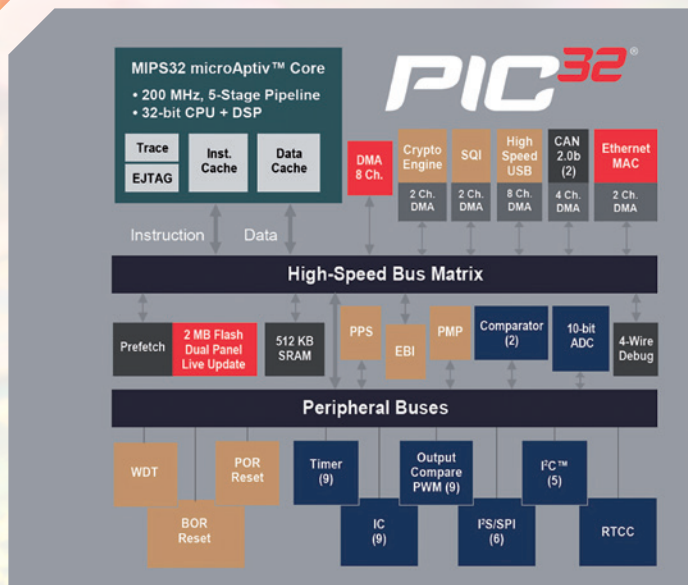


Fig. 1 - Schema a blocchi di un'architettura PIC32MZ.

I microcontrollori PIC32 costituiscono il prodotto più di alto livello commercializzato da Microchip e i chip prodotti inglobano al loro interno periferiche di elevata complessità, come USB Host/OTG/Device, Ethernet controllers, CAN, Capacitive Touch, ecc.

Attualmente esistono due famiglie principali di microcontrollori PIC32: la famiglia PIC32MX (la prima ad essere introdotta) e la famiglia PIC32MZ (attualmente la più performante). Le caratteristiche principali dei microcontrollori che vi appartengono sono:

- velocità di esecuzione 330 DMIPS @200 MHz;
- tagli di memoria Flash fino a 512k;
- una istruzione per ciclo di clock (DSP);
- possibilità di esecuzione del codice direttamente dalla memoria RAM;
- controller DMA;
- PPS (Peripheral Pin Select) per rimappare i pin del microcontrollore;
- interfaccia di programmazione/debug JTAG o 2-wire Microchip;
- supporto Execution Trace;
- Timer, Watchdog, RTCC;
- High-Speed USB (Host/Dual Role/Device), Ethernet (con MAC integrato), CAN, mTouch;

- I²C, SPI, UART, PMP;
- ADC e CCP.

Come è possibile vedere, la dotazione di periferiche di questi dispositivi è estremamente ricca, il che consente di sviluppare applicazioni embedded anche molto complesse impiegando un singolo microcontrollore e riducendo al minimo la circuiteria esterna.

I TIMER

Analizziamo adesso una delle periferiche base dei PIC32, ossia la periferica Timer. Un timer è essenzialmente una periferica hardware deputata a compiti di conteggio. Essendo svincolati dal processore, il conteggio dei timer non influenza il tempo di esecuzione della CPU e, viceversa, un timer hardware esegue un conteggio preciso che non è influenzato dal carico di esecuzione del sistema. Esistono vari tipi di timer: ad esempio, un tipo di timer specializzato per una particolare applicazione è l'RTCC (Real Time Clock Calendar), utilizzato per l'implementazione degli orologi di sistema. Un altro esempio è il timer watchdog, che viene spesso utilizzato per resettare il sistema in seguito a situazioni di stallo. Una caratteristica molto importante dei timer è che, essendo a tutti gli effetti periferiche hardware, possono essere associati ad eventi di interrupt di diverso tipo (in genere overflow o period match con un registro di appoggio, denominato appunto period register) e quindi sono spesso utilizzati per generare interrupt con tempistiche molto precise (gli scheduler, ad esempio, sono in genere basati su un interrupt generato da un timer). Tratteremo più in dettaglio gli interrupt (e la loro possibile associazione ai timer) più avanti, nel paragrafo dedicato. I PIC32 dispongono di due tipi di timer general purpose (in questa categoria non rientra il watchdog, che, data la sua natura, non può essere considerato un general purpose timer), il Timer A ed il Timer B. Il timer A è un timer a

Fig. 2 - Tabella comparativa tra Timer A e Timer B.

Available Timer Types	Secondary Oscillator	Asynchronous External Clock	Synchronous External Clock	16-Bit Synchronous Timer/Counter	32-Bit ⁽¹⁾ Synchronous Timer/Counter	Gated Timer	Special Event Trigger
Type A	Yes	Yes	Yes	Yes	No	Yes	No
Type B	No	No	Yes	Yes	Yes	Yes	Yes

16-bit che può essere anche associato ad un clock esterno (ad esempio all'oscillatore secondario) in modalità asincrona e può operare anche quando il microcontrollore lavora in sleep mode (adatto quindi all'implementazione di RTCC), dotato di un prescaler a 4 valori (1:1, 1:4, 1:8, 1:256). Il Timer B invece non può funzionare in modalità asincrona rispetto al processore (funziona quindi con il clock di sistema o sfruttando direttamente l'oscillatore primario), ma ha la caratteristica di poter essere associato ad un secondo Timer B per formare un contatore a 32-bit. Inoltre dispone di un prescaler con più valori di scala rispetto al Timer A e può generare eventi di trigger speciali (come ad esempio un trigger per l'ADC, ideale per applicazioni di acquisizione dati). Una tabella comparativa delle due tipologie di timer è riportata in Fig. 2. In genere i PIC32 dispongono di almeno un Timer A (in genere il Timer 1) e di un numero variabile (generalmente 4) Timer B.

In Fig. 3 è riportato lo schema a blocchi di un Timer A. Come si può vedere, esiste la possibilità di associare il Timer all'oscillatore secondario a bassa potenza (ideale per il mantenimento di un Real Time Clock quando il sistema si trova in sleep mode, con consumo minimo di energia), tramite il multiplexer comandato dal campo TCS (Timer Clock Select) del registro TCON.

In Fig. 4 è invece rappresentato lo schema a blocchi di un Timer B. In questo caso non c'è la possibilità di utilizzare una sorgente asincrona, come nel caso del Timer A, ma deve essere utilizzato o il clock delle periferiche (PBCLK) oppure può essere impiegato direttamente l'oscillatore primario.

Come accennato in precedenza, il prescaler presenta un numero maggiore di possibili selezioni (1:1, 1:2, 1:4, 1:8, 1:32, 1:64, 1:256), rendendo questo oggetto capace di risoluzioni maggiori rispetto al Timer A, e quindi più "general Purpose".

HELLO WORLD CON TIMER 3

Vediamo adesso un semplice esempio su come utilizzare un Timer per eseguire un conteggio di una certa precisione: eseguiamo lo stesso esercizio presentato nel corso della prima puntata, ma stavolta, anziché utilizzare un contatore a carico della CPU, usiamo un timer di tipo B, ed in particolare il Timer 3.

Creiamo un secondo progetto da MPLab X e nominiamolo Ex2, all'interno del progetto creiamo

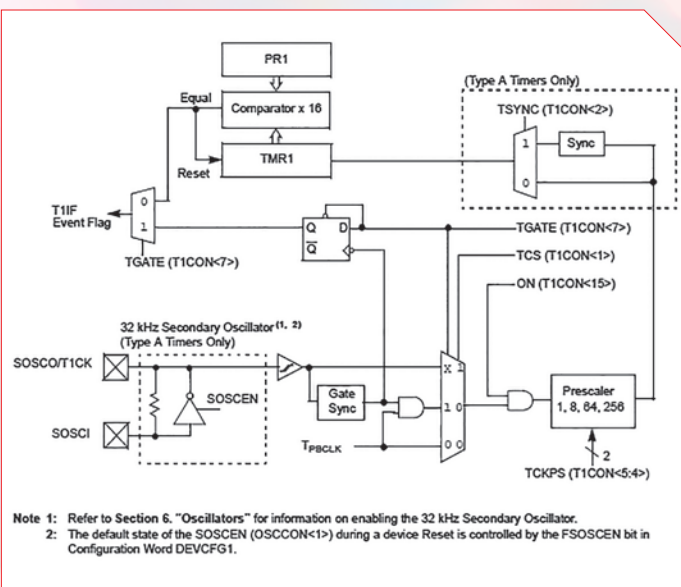


Fig. 3 - Schema a blocchi di un Timer A.

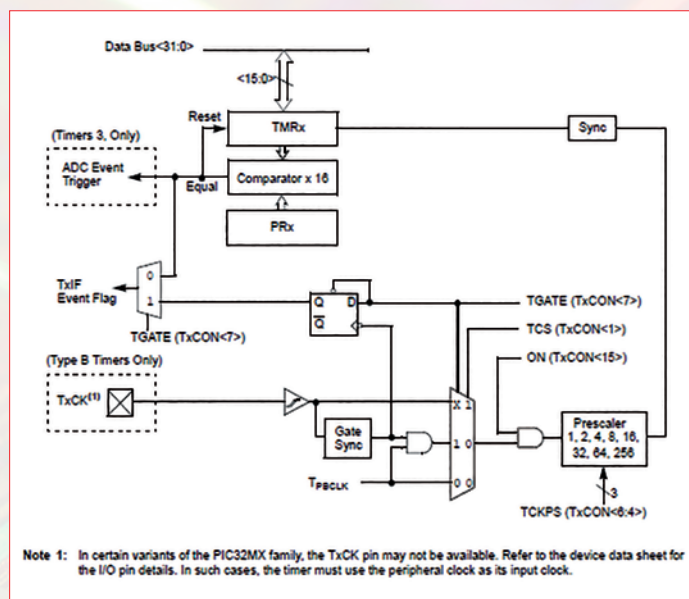


Fig. 4 - Schema a blocchi di un Timer B.

un file C di tipo main ed apriamolo all'interno dell'editor. Per prima cosa vanno impostate una serie di define che ci servono per settare i valori del registro di controllo del Timer 3. Il codice da scrivere è riportato all'interno del Listato 1.

A questo punto, all'interno della funzione main, configuriamo il registro di controllo del Timer 2, T2CON. Ciò che vogliamo ottenere è un lampeggio del nostro LED alla frequenza di 1 Hz, e quindi un intervallo tra ogni toggle pari ad un secondo. Partendo da una frequenza di funzionamento di 8 MHz e usando un prescaler pari ad 8, il timer opera ad 1 MHz e quindi esegue 1 milione di conteggi al secondo. Pertanto,

Listato 1

```

/* -- defines -- */
#define FOREVER          1
#define SYS_CLOCK        8000000
#define TMR_ON           1
#define TMR_16_BIT_MODE  0
#define TMR_32_BIT_MODE  1
#define TMR_CS_PBCLK      0
#define TMR_PRESCALER_8   3
#define TMR_PRESCALER_VALUE 8
#define TMR_MATCH_VALUE  SYS_CLOCK/TMR_PRESCALER_VALUE

```

Listato 1 - Definisce generiche e per il settaggio dei registri di controllo del Timer 2.

una volta configurato il timer, sarà sufficiente controllare quando il registro raggiunge il valore di TMR_MATCH_VALUE per essere sicuri che sia trascorso un secondo.

Impostiamo quindi T2CON con un prescaler di 8, in modalità 32-bit (per poter contare fino ad un milione), sfruttando il clock delle periferiche come clock del timer (T2CONbits.TCS = TMR_CS_PBCLK) e accendiamo la periferica (T2CONbits.ON = TMR_ON), come illustrato nel Listato 2.

A questo punto non ci rimane altro da fare che scrivere in C quanto detto prima in linguaggio naturale: "verificare quando il registro timer arriva al valore di TMR_MATCH_VALUE" e, una volta raggiunto questo valore, resettare il registro ed invertire lo stato del pin RD0, chiaramente il tutto all'interno del solito ciclo infinito while (FOREVER). Per l'implementazione guardate il Listato 3.

Per testare il funzionamento possiamo mettere il solito breakpoint sull'istruzione che esegue il toggle del pin, oppure se disponiamo di una scheda sui cui provare possiamo verificare la periodicità usando un oscilloscopio.

INTERRUPT

Trattiamo adesso un argomento un po' più complesso rispetto ai precedenti, ma che costituisce uno dei concetti più importanti nello sviluppo

Listato 2

```

/* set ddr register */
TRISD = 0x0000;
/* configure the PIC32 core for the best performance */
SYSTEMConfigPerformance(SYS_CLOCK);

/* set T1CON register */
T2CONbits.ON = TMR_ON;
T2CONbits.TCKPS = TMR_PRESCALER_8;
T2CONbits.T32 = TMR_32_BIT_MODE;
T2CONbits.TCS = TMR_CS_PBCLK;

```

Listato 2 - Configurazione di T2CON

Listato 3

```

/* endless loop */
while(FOREVER)
{
    /* if TMR exceed the match value */
    if (TMR2 > TMR_MATCH_VALUE)
    {
        /* clear timer */
        TMR2 = 0;
        /* toggle RD0 */
        PORTDbits.RD0 = ~PORTDbits.RD0;
    }
}

```

Listato 3 - Hello World con Timer 2

di software per applicazioni embedded, ossia il concetto di interrupt. Prima di tutto cerchiamo di chiarire che cos'è realmente un interrupt, e lo facciamo con un parallelismo con il mondo reale, nel quale anche noi siamo spesso soggetti a delle situazioni che possono essere modellizzate come degli interrupt. Immaginate di stare leggendo un libro e, arrivati ad una certa pagina, di ricevere una chiamata al telefono. A questo punto ponete un segno sul vostro libro e lo chiudete, rispondete al telefono e, una volta

```

63  /* endless loop */
64  while(FOREVER)
65  {
66      /* if TMR exceed the match value */
67      if (TMR2 > TMR_MATCH_VALUE)
68      {
69          /* clear timer */
70          TMR2 = 0;
71          /* toggle RD0 */
72          PORTDbits.RD0 = ~PORTDbits.RD0;
73      }
74  }
75
76  /* execution should never reach this line */
77  return (EXIT_SUCCESS);
78

```

Fig. 5 - Verifica del funzionamento dell'esercizio 2.

conclusa la telefonata, riaprite il libro alla pagina contrassegnata e ricominciate a leggere. Quello che avviene all'interno di un microcontrollore quando la CPU riceve un interrupt è più o meno la stessa cosa: durante l'esecuzione, se viene segnalato un interrupt, nella CPU si eseguono le seguenti operazioni.

1. Viene salvato il contesto di esecuzione corrente (in particolare il valore corrente del Program Counter e tutti i dati necessari a riprendere correttamente l'esecuzione del processo interrotto). Questa operazione viene proprio detta "Salvataggio del contesto di esecuzione".
2. Viene interrogata una tabella, chiamata in genere Interrupt Vector Table, che contiene le

locazioni delle funzioni che servono le varie possibili richieste di interruzioni. Le richieste di interrupt sono chiamate IRQ (Interrupt ReQuest), mentre le funzioni che servono le richieste sono chiamate ISR (Interrupt Service Routines).

- Viene eseguita la ISR richiamata dalla Interrupt Vector Table. Tale funzione, in genere, è di breve durata, in modo da interferire il meno possibile con l'esecuzione corrente.
- Una volta terminata l'esecuzione della ISR, l'interrupt si definisce Servito e il processore recupera il contesto di esecuzione pre-interrupt, aggiorna il program counter con la locazione corretta e ripristina tutti i dati necessari allo svolgimento dell'attività interrotta e va avanti.

Il parallelismo con l'esempio riportato in precedenza è ora evidente: voi che leggete siete il processore, la lettura del libro è il processo in esecuzione, il segno apportato sul libro per ricordare la pagina è il salvataggio di contesto e la telefonata è l'interrupt.

Gli interrupt sono spesso utilizzati nel mondo embedded in una moltitudine di situazioni: ad esempio possono servire alla gestione di tutti gli eventi asincroni rispetto al processo di esecuzione (per citare solo alcuni esempi: la pressione di un pulsante in un'interfaccia HMI, o l'arrivo di un dato su una linea seriale).

Tuttavia questo contesto non è l'unico nel quale l'uso degli interrupt può risultare vantaggioso; ad esempio un altro possibile utilizzo degli interrupt in congiunzione con le periferiche di tipo timer, per la generazione di eventi di interrupt periodici (si pensi ad un timer che va in overflow o in period match ogni millisecondo, generando una interruzione: questo tipo di interruzione ciclica può essere utilizzato come segnale di temporizzazione del sistema).

È inoltre possibile utilizzare gli interrupt per la gestione di situazioni di errore particolari (eccezioni), demandando quindi ad una ISR la gestione del caso di errore particolare che viene a verificarsi (anche se in questo caso si parla in genere di trap e non di interrupt).

In sostanza gli impieghi degli interrupt sono molteplici, ma comunque tutti strettamente legati alle periferiche: virtualmente, quasi ogni periferica all'interno di un microcontrollore è in grado di generare uno o più interrupt.

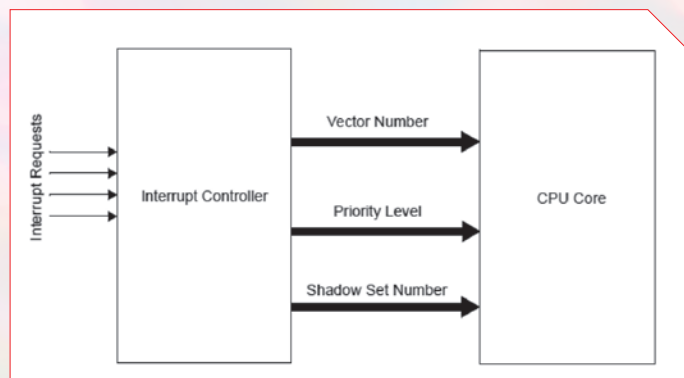


Fig. 6 – Schema a blocchi del modulo interrupt di un PIC32.

L'interrupt controller dei micro della famiglia PIC32, il cui schema a blocchi è riportato in Fig. 6, è un sistema di una certa complessità e dotato delle seguenti caratteristiche:

- fino a 96 sorgenti di interrupt;
- fino a 64 interrupt vectors;
- modalità single-vector e multi-vector;
- fino a 5 sorgenti esterne di interrupt;
- 7 livelli di priorità (priority level);
- 4 livelli di sub-priorità (sub-priority level);
- possibilità di generare interrupt da software.

La gestione e la configurazione degli interrupt è gestita tramite un particolare set di registri di controllo e di stato. Analizzeremo più in dettaglio i registri quando tratteremo l'esempio pratico, per il momento si rimandano i lettori più interessati al PIC32MX/MZ Family Reference Manual per maggiori dettagli.

HELLO WORLD CON INTERRUPT

Vediamo adesso come sia possibile riscrivere l'esempio che abbiamo visto in precedenza utilizzando una gestione ad interrupt. Quello che vogliamo fare è, nuovamente, far oscillare il pin RD0 alla frequenza di 1 Hz. Utilizzeremo anche questa volta un timer, ma anziché impiegare il processore per determinare se il conteggio del ciclo corrente è terminato (l'istruzione if dell'esempio precedente), ci serviremo di un interrupt generato dal timer stesso, e gestiremo il cambio di stato del pin all'interno di un'apposita ISR. Inoltre vogliamo complicare leggermente l'esercizio, imponendo di utilizzare il timer in modalità 16-bit con lo stesso prescaler utilizzato in precedenza (3), in modo da dover aggiungere qualche ulteriore operazione di conteggio per arrivare alla frequenza desiderata.

Creiamo quindi un nuovo progetto MPLab X, che chiamiamo MLX_EX_3, ed un nuovo file main al suo interno. In questo esempio avremo bisogno di utilizzare qualche funzione presente

Listato 4

```
#define CYCLE_PER_SEC      20
#define INT_MATCH_VALUE    ((SYS_CLOCK/TMR_PRESCALER -
                           VALUE)/CYCLE_PER_SEC)

#define T2_INT_DISABLE     0
#define T2_INT_ENABLE     1
#define T2_INT_PRIO_1     1
#define T2_INT_SUB_PRIO_0 0

#define LED                PORTDbits.RD0
```

Listato 4 – Define per esercizio 3

nella libreria delle periferiche, quindi per prima cosa inseriamo la seguente inclusione all'interno del codice:

```
#include <plib.h>
```

Ricopiamo le stesse define utilizzate nell'esempio precedente, aggiungendo quelle riportate nel **Listato 4**.

Il Timer 3 dei PIC32 ha la capacità di generare un interrupt in caso di period match con il relativo registro PR2. Impostiamo quindi il timer in modalità 16-bit ed imponiamo il valore di PR2 pari alla define INT_MATCH_VALUE (che abbiamo calcolato per avere un periodo di interrupt pari a 50ms).

A questo punto bisogna impostare le priorità (non ha particolare importanza in questo caso, in quanto non ci sono altre sorgenti di interrupt attive al momento, decidiamo quindi di impostare un livello di priority pari ad 1 e subpriority pari a 0). Ora è sufficiente abilitare la sorgente di interrupt (Timer 3), tramite il bit T2IE del registro IEC0 ed abilitare globalmente gli interrupt (abbiamo utilizzato la funzione INTEnableSystemMultiVectoredInt presente all'interno della libreria delle periferiche).

Ora periferica e relativo interrupt sono corretta-

Listato 5

```
/* set T1CON register */
T2CONbits.ON = TMR_ON;
T2CONbits.TCKPS = TMR_PRESCALER_8;
T2CONbits.T32 = TMR_16_BIT_MODE;
T2CONbits.TCS = TMR_CS_PBCLK;
PR2 = INT_MATCH_VALUE;

/* set interrupt register */
IPC2bits.T2IP = T2_INT_PRIO_1;
IPC2bits.T2IS = T2_INT_SUB_PRIO_0;
IEC0bits.T2IE = T2_INT_ENABLE;

INTEnableSystemMultiVectoredInt();
```

Listato 5 – Configurazione registri timer ed interrupt.

Listato 6

```
int main(void)
{
    /* set ddr register */
    TRISD = 0x0000;
    /* configure the PIC32 core for the best performance */
    SYSTEMConfigPerformance(SYS_CLOCK);

    /* set T1CON register */
    T2CONbits.ON = TMR_ON;
    T2CONbits.TCKPS = TMR_PRESCALER_8;
    T2CONbits.T32 = TMR_16_BIT_MODE;
    T2CONbits.TCS = TMR_CS_PBCLK;
    PR2 = INT_MATCH_VALUE;

    /* set interrupt register */
    IPC2bits.T2IP = T2_INT_PRIO_1;
    IPC2bits.T2IS = T2_INT_SUB_PRIO_0;
    IEC0bits.T2IE = T2_INT_ENABLE;

    INTEnableSystemMultiVectoredInt();

    /* endless loop */
    while(FOREVER)
    {
        /* interrupt mode - do nothing */
    }

    /* execution should never reach this line */
    return (EXIT_SUCCESS);
}
```

Listato 6 – main function

mente configurati. Se tutto è stato eseguito correttamente, la funzione main dovrebbe assomigliare a quella riportata all'interno del **Listato 6**. Ora che la configurazione del sistema è completata, passiamo all'analisi della ISR, il cui codice è riportato all'interno del **Listato 7**.

In XC32 le ISR si identificano usando la macro `__ISR`, che predende come parametri di ingresso il vettore relativo all'interrupt e la priorità (iplx,

Listato 7

```
void __ISR(_TIMER_2_VECTOR, ipl1)
Timer2InterruptHandler (void)
{
    /* interrupt counter */
    static unsigned int counter;

    /* increment counter */
    counter++;
    /* clear interrupt flag */
    IFS0bits.T2IF = 0;

    if (counter >= CYCLE_PER_SEC)
    {
        /* reset counter */
        counter = 0;
        /* toggle LED */
        LED = ~LED;
    }
}
```

Listato 7 – Interrupt Service Routine

in questo caso abbiamo impostato `ipl1`, come da configurazione). Dopodichè si può scegliere l'identificativo che si desidera per la ISR, in questo caso noi abbiamo scelto `Timer2InterruptHandler`. All'interno della ISR definiamo una variabile statica di conteggio (è importante che la variabile sia statica, in quanto si deve mantenere il conteggio tra un'interruzione e l'altra). Il corpo della funzione è molto semplice: sostanzialmente ad ogni occorrenza si incrementa il contatore, si resetta il flag relativo all'interrupt (`IFS0bits.T2IF = 0`, quest'operazione deve essere eseguita via software, in quanto il flag non si resetta automaticamente in hardware), e si controlla se il numero di occorrenze ha superato la soglia impostata per determinare il raggiungimento del tempo di un secondo (`if (counter > CYCLE_PER_SEC)`), e, in caso positivo, si resetta il contatore e si esegue il toggle del pin ed il ciclo ricomincia dall'inizio. Se lanciamo in esecuzione il codice con il simulatore e posizioniamo un breakpoint all'interno della ISR, dopo 50ms vedremo che l'esecuzione del codice salta all'interno della routine di interrupt, per effetto del period match tra il registro timer e PR2, come visibile in Fig. 7. Se inseriamo un secondo breakpoint all'interno dell'if, noteremo che dopo 20 occorrenze si entra all'interno di questo ramo dell'esecuzione e viene effettivamente eseguito il toggle del pin (Fig. 8).

CONCLUSIONI

In questa seconda puntata ci siamo addentrati

```

93 void __ISR(_TIMER_2_VECTOR, ipl1) Timer2InterruptHandler (void)
94 {
95     /* interrupt counter */
96     static unsigned int counter;
97
98     /* increment counter */
99     counter++;
100     /* clear interrupt flag */
101     IFS0bits.T2IF = 0;
102
103     if (counter > CYCLE_PER_SEC)
104     {
105         /* reset counter */
106         counter = 0;
107         /* toggle LED */
108         LED = ~LED;
109     }
110 }

```

Fig. 7 – Breakpoint all'ingresso della ISR, sono trascorsi 50ms dall'inizio dell'esecuzione.

```

92 void __ISR(_TIMER_2_VECTOR, ipl1) Timer2InterruptHandler (void)
93 {
94     /* interrupt counter */
95     static unsigned int counter;
96
97     /* increment counter */
98     counter++;
99     /* clear interrupt flag */
100     IFS0bits.T2IF = 0;
101
102     if (counter >= CYCLE_PER_SEC)
103     {
104         /* reset counter */
105         counter = 0;
106         /* toggle LED */
107         LED = ~LED;
108     }

```

Fig. 8 – Breakpoint all'interno dell'if.

più in profondità della scrittura di software embedded per PIC32 tramite MPLab X, analizzando i timers e prendendo confidenza con il concetto, assolutamente fondamentale nel mondo embedded, di interrupt.

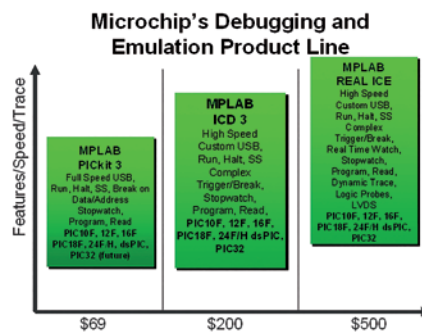
Nella prossima puntata continueremo il nostro viaggio analizzando alcune periferiche analogiche dei microcontrollori PIC32, come il convertitore analogico\digitale ed il modulo PWM. ■

EMULATORI E DEBUGGER

A supporto dei microcontrollori che produce, la Microchip commercializza una gamma completa di emulatori e debugger, in grado di soddisfare le esigenze più disparate per quanto concerne prezzi dei tool e prestazioni. Il prodotto entry level per il debugging è il PicKit, ormai arrivato alla sua terza edizione (PicKit3); si tratta di un dispositivo con interfaccia USB che funziona sia da programmatore

che da debugger di base (comunque in grado di supportare breakpoint, lettura memoria, Stopwatch, ecc). Il fratello maggiore del PicKit è l'ICD (In-Circuit Debugger), anch'esso arrivato alla versione 3; si tratta di un dispositivo più performante rispetto al precedente, che supporta un set avanzato di funzioni di Breakpoint/Trigger ed è più veloce e protetto rispetto al pickit. Infine, il dispositivo di

Fig. 11
Linea prodotti
Microchip per
Emulazione e
Debugging.



più alta gamma fornito dalla Microchip per il debugging è il Real ICE (In-Circuit Emulator). Questo è un dispositivo

di livello professionale, che supporta oltre alle funzioni tipiche dei debuggers anche la funzione di trace.