

Mensile di progettazione elettronica, attualità scientifica, novità tecnologiche.

€ 6,00

Elettronica In

oltre l'elettronica

www.elettronica.in.it

Anno XIX - n. 178
Luglio / Agosto 2013



**IMPARA
A STAMPARE IL TUO
CONTENITORE**

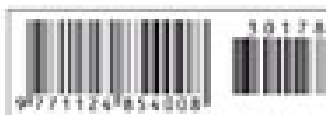
CON LA **3DAG**

- Web Power Meter con Arduino
- Board Linux Embedded a confronto
- Gestire i moduli GSM con Arduino
- myKIT: misura di temperatura con termistore
- Robot Hero alla RoboCup Junior Italia

Combinatore GSM con messaggi vocali



**Shield di espansione LCD
per Raspberry Pi**



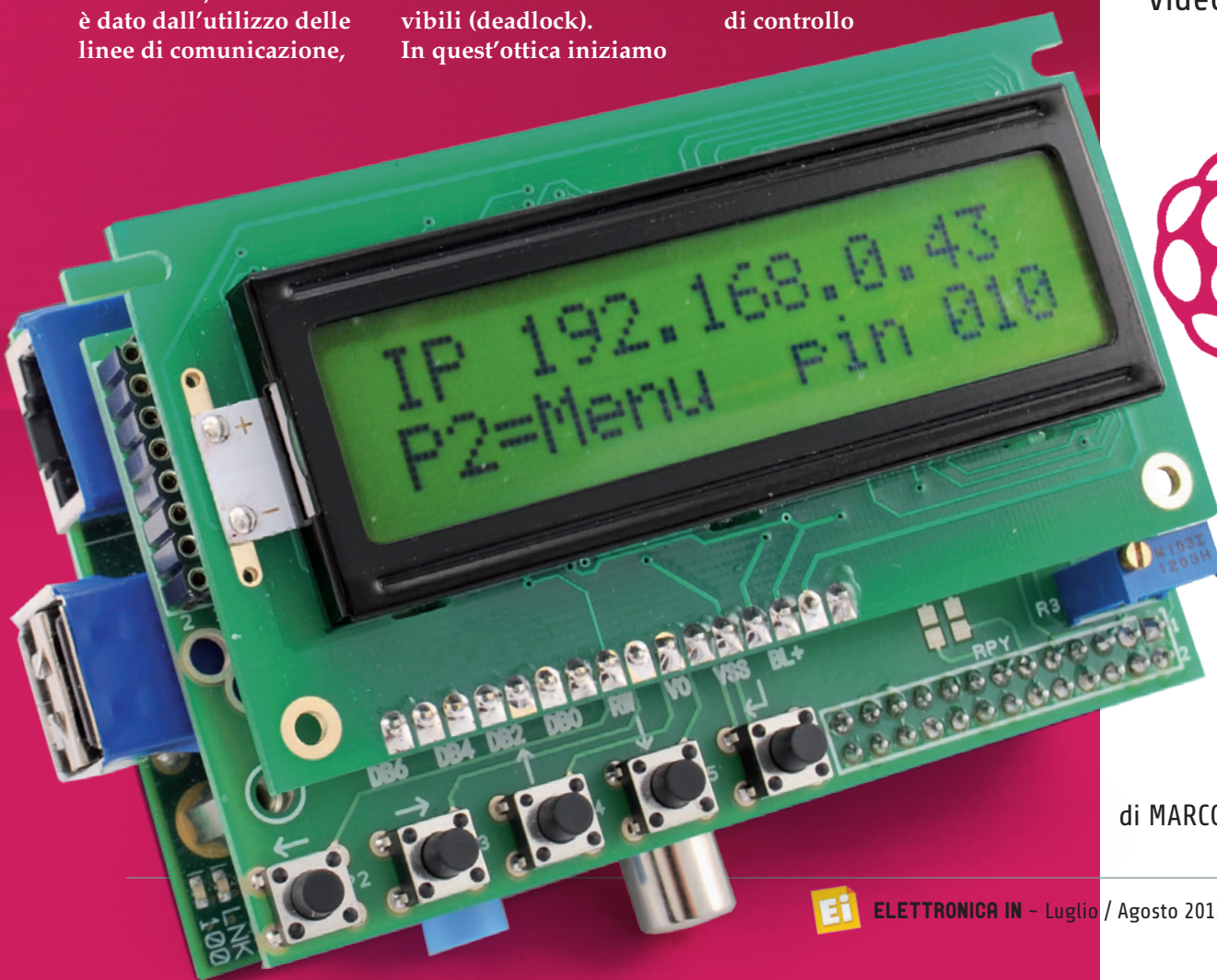
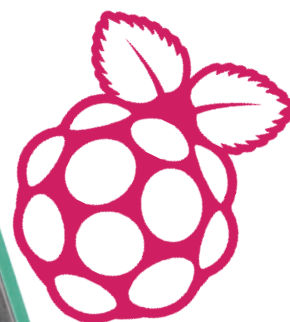
SHIELD DI ESPANSIONE LCD PER RASPBERRY PI

La specificità di un micro PC come Raspberry Pi rispetto a un microcontrollore è quella di poter svolgere diversi compiti in condivisione di risorse. Oltre alle periferiche già disponibili a bordo della scheda, è possibile aggiungerne altre in una catena pressoché infinita, il cui limite è dato dall'utilizzo delle linee di comunicazione,

dalle capacità elaborative e dalla memoria disponibile. Molto dipende anche da come sono progettate e realizzate le applicazioni, soprattutto per quanto riguarda la capacità di rispettarsi a vicenda e di condividere le risorse utilizzate senza intralciarsi l'un l'altra o creando "blocchi" irrisolvibili (deadlock). In quest'ottica iniziamo

a presentarvi una serie di shield di espansione appositamente realizzati per Raspberry Pi e progettati in modo da poter essere impilati uno sull'altro ed essere utilizzati in contemporanea. Il primo di questi è lo shield di espansione LCD, che permette di realizzare un'interfaccia di controllo

Ecco lo shield di espansione LCD per Raspberry Pi che permette di realizzare un'interfaccia di controllo esterna per le nostre applicazioni senza necessità di tenere collegati costantemente video, tastiera e mouse.



di MARCO MAGAGNIN

L'integrato MCP23017

L'integrato MCP23017 permette di realizzare una periferica di espansione con 16 canali di I/O digitali, pilotabili per mezzo di comunicazioni basate sul bus I2C. Disponibile in formato 28-PDIP, 28-SOIC e 28-SSOP, nel nostro circuito abbiamo adottato la "comoda" configurazione a 28 pin PDIP. La piedinatura è visibile in **Fig. A**.

Il sistema master, per pilotare l'integrato, deve impostare correttamente i registri di configurazione, ciascuno di 8 bit, mediante i quali è possibile selezionare la direzione di ciascun ingresso (in o out) e la polarità.

Nello schema di **Fig. B** è visibile l'architettura dell'integrato. I pin di I/O sono raggruppati in due porte che controllano 8 pin ciascuna: PORTA e PORTB. Mediante i registri di controllo IODIRA/B è possibile impostare la direzione di funzionamento di ciascun pin di I/O. La polarità di ciascuna porta può essere invertita intervenendo sul registro Polarity Inversion. Ovviamente il sistema di controllo master è in grado di leggere tutti i registri.

Registro d'indirizzamento

Il registro d'indirizzamento dell'integrato MCP23017 è lungo 7 bit (**Fig. C**). I quattro bit più significativi devono essere impostati al valore fisso "0100" e identificano il

produttore del chip. I valori dei tre bit rimanenti vengono impostati in base ai livelli di tensione assegnati ai pin A1, A2 e A3 dell'integrato. Questa configurazione permette di indirizzare sullo stesso bus I2C fino a otto integrati MCP23017 per un totale di 128 pin di I/O. L'ultimo bit del registro di indirizzo permette di impostare il tipo di operazione che si vuole eseguire, "0" per eseguire un'operazione di scrittura di registri e "1" per eseguire un'operazione di lettura.

Registri di gestione

Per poter controllare il funzionamento dell'integrato MCP23017 è necessario scrivere e leggere una serie di registri in sequenze ben definite. Una panoramica dei registri disponibili è

Fig. B

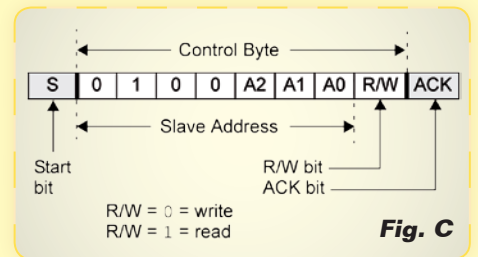
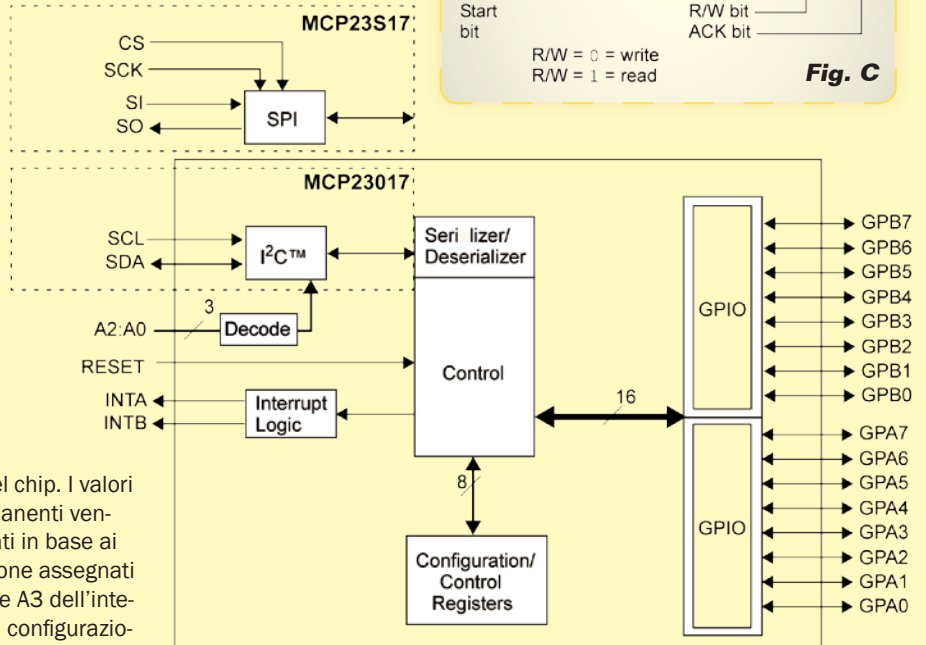


Fig. C

visibile in **Fig. D**: la prima colonna della tabella indica il nome del registro, la seconda colonna, l'indirizzo esadecimale con il quale è possibile leggere o scrivere il contenuto del registro. I registri IODIRA e IODIRB, lunghi ciascuno 8 bit, permettono di impostare ciascun pin di I/O sulle porte A o B come ingresso o come uscita. Per configurare il pin come ingresso, il corrispondente valore del bit nel registro deve essere impostato a "1", mentre per configurarlo come uscita deve essere impostato a "0". I registri GPIOA e GPIOB permettono di leggere lo stato dei pin di ingresso delle rispettive porte. Se il bit corrispondente alla posizione del pin è impostato a "0", il pin è a livello basso altrimenti si trova a livello alto. Parimenti, per impostare i livelli dei pin configurati come output è necessario

comporre i contenuti dei registri OLATA e OLATB in modo che il bit corrispondente a ciascun pin assuma il valore "0" nel caso si voglia impostare il livello basso o "1" per impostare il livello alto.

I registri IPOLA e IPOLB permettono di impostare ciascun pin di I/O in modo da lavorare a logica invertita. Se il bit corrispondente a un determinato pin è impostato a "1" i livelli saranno presentati in logica invertita.

I registri GPPUA e GPPUB permettono di attivare o meno per ciascun pin di I/O la resistenza interna di pull-up. Se il bit corrispondente a un determinato pin è impostato a "1" la resistenza di pull-up è attivata. Infine, un approfondimento sulla gestione degli interrupt messi a disposizione dall'integrato MCP23017: i piedini 19 e 20 dell'integrato sono collegati ai terminali del modulo di gestione degli

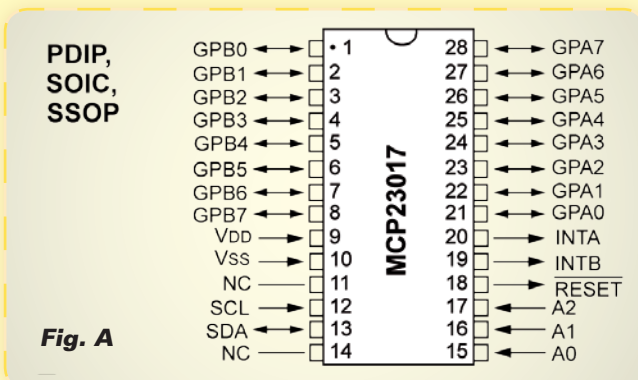


Fig. A

Fig. D - Mappa della memoria che contiene i valori dei registri di controllo dell'integrato MCP23017.

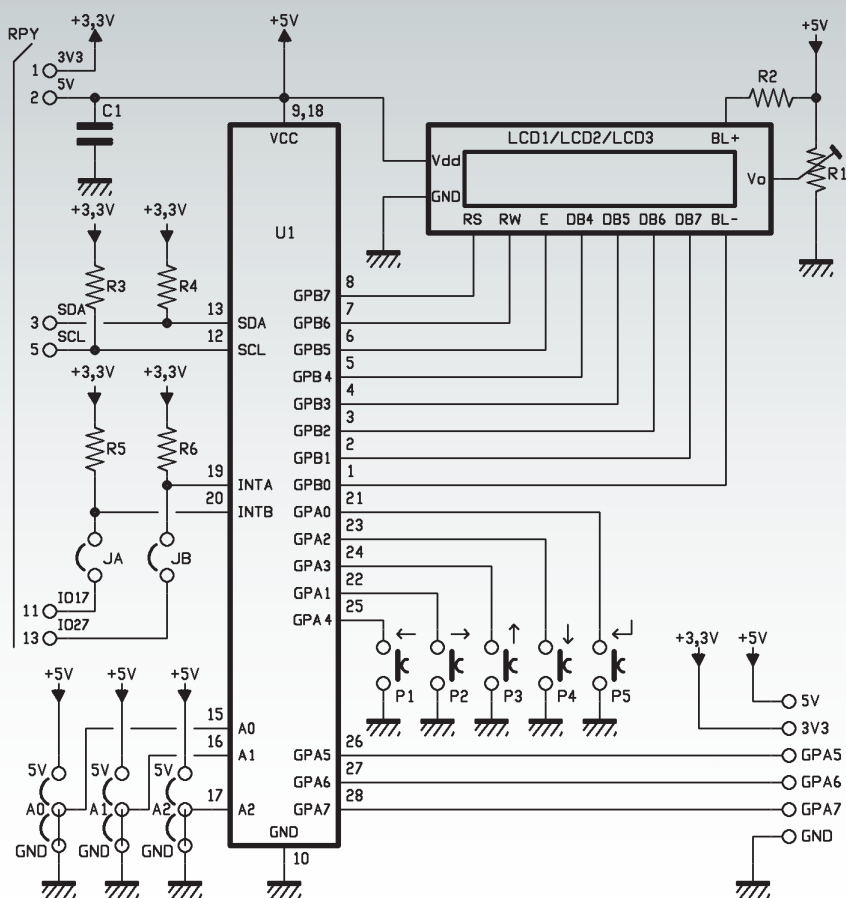
Register Name	Address (hex)	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	POR/RST value
IODIRA	00	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IODIRB	01	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IPOLA	02	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0	0000 0000
IPOLB	03	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0	0000 0000
GPINTENA	04	GPINT7	GPINT6	GPINT5	GPINT4	GPINT3	GPINT2	GPINT1	GPINT0	0000 0000
GPINTENB	05	GPINT7	GPINT6	GPINT5	GPINT4	GPINT3	GPINT2	GPINT1	GPINT0	0000 0000
DEFVALA	06	DEF7	DEF6	DEF5	DEF4	DEF3	DEF2	DEF1	DEF0	0000 0000
DEFVALB	07	DEF7	DEF6	DEF5	DEF4	DEF3	DEF2	DEF1	DEF0	0000 0000
INTCONA	08	IOC7	IOC6	IOC5	IOC4	IOC3	IOC2	IOC1	IOC0	0000 0000
INTCONB	09	IOC7	IOC6	IOC5	IOC4	IOC3	IOC2	IOC1	IOC0	0000 0000
IOCON	0A	BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	—	0000 0000
IOCON	0B	BANK	MIRROR	SEQOP	DISSLW	HAEN	ODR	INTPOL	—	0000 0000
GPPUA	0C	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0	0000 0000
GPPUB	0D	PU7	PU6	PU5	PU4	PU3	PU2	PU1	PU0	0000 0000
INTFA	0E	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	0000 0000
INTFB	0F	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	0000 0000
INTCAPA	10	ICP7	ICP6	ICP5	ICP4	ICP3	ICP2	ICP1	ICP0	0000 0000
INTCAPB	11	ICP7	ICP6	ICP5	ICP4	ICP3	ICP2	ICP1	ICP0	0000 0000
GPIOA	12	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
GPIOB	13	GP7	GP6	GP5	GP4	GP3	GP2	GP1	GP0	0000 0000
OLATA	14	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000
OLATB	15	OL7	OL6	OL5	OL4	OL3	OL2	OL1	OL0	0000 0000

interrupt, rispettivamente INTB per il banco B, INTA per il banco A. Sull'integrato MCP23017 è possibile configurare ciascun pin in modo che al variare del suo stato si attivi il pin di interrupt corrispondente al banco di appartenenza. In questo modo, da parte del sistema di gestione master, non è necessario leggere in continuazione tutti i pin per verificare se qualcuno cambia di stato, ma è sufficiente impostare una gestione di interrupt sui due pin INTA e/o INTB per essere avvisati quando avviene una variazione su uno dei pin monitorati. A questo punto si legge il registro di stato del banco corrispondente e si individua, confrontandoli con i valori precedenti, quale o quali hanno subito una variazione. Prima di attivare il monitoraggio dei pin con i registri GPINTENA e GPINTENB, è necessario configurare le condizioni in base alle quali deve essere attivato l'interrupt. Per fare questo si utilizzano le coppie di re-

gistri DEFVALA, DEFVALB e INTCONA, INTCONB. La prima coppia di registri permette di configurare, per ciascun pin dei due banchi, un livello di riferimento per comandare l'innesco dell'interrupt. In ogni caso questa configurazione dipende dalla configurazione dei registri INTCON. Se viene configurato a "0" il bit corrispondente ad un pin, l'interrupt verrà innescato quando il pin assumerà il livello "1", all'opposto nell'altro caso. La configurazione dei bit dei registri INTCONA e INTCONB, invece, permette di impostare se l'interrupt dev'essere comandato dalle condizioni impostate nei registri DEFVAL, oppure semplicemente ad ogni cambiamento di stato del livello dei pin. Se si imposta a "0" il bit corrispondente ad un pin nel registro INTCON, l'interrupt verrà innescato ad ogni cambiamento di stato del pin, se si imposta ad "1", l'interrupt verrà innescato secondo le condizioni

indicate dai registri DEFVAL. Le condizioni per le quali è possibile rilevare un interrupt dai pin INTA e INTB dipende dalla configurazione del bit INTCOL (bit 1) del registro IOCON, che permette di configurare diverse opzioni di funzionamento dell'integrato. Il valore predefinito è "0" che significa che i pin di interrupt sono normalmente ad un livello alto che diventa basso all'innescarsi dell'interrupt. Per capire quale dei pin monitorati ha innescato l'interrupt si legge uno dei registri della coppia INTFA, INTFB. Il o i bit impostati a "1" corrispondono al o ai bit che hanno generato l'interrupt. A questo punto il livello dei pin interessati può essere rilevato leggendo il registro corrispondente tra INTCAPA e INTCAPB. Questi registri fotografano lo stato dei pin al momento dell'interrupt e lo mantengono fin quando vengono letti. Per ulteriori e più approfondite informazioni potete far riferimento al datasheet dell'integrato.

esterna per le nostre applicazioni. In questo articolo descriviamo l'hardware dello shield di espansione, la libreria che permette una notevole semplificazione nell'utilizzo dello shield e un approfondimento dell'integrato MCP23017, usato come interfaccia tra Raspberry Pi e i moduli LCD. Prima di iniziare la descrizione, una nota a riguardo dello sviluppo di applicazioni. Abbiamo accennato alla necessità di progettare le applicazioni destinate ai sistemi embedded e, tra questi, quelli basati sul sistema operativo GNU/Linux, in modo che siano in grado di collaborare tra loro utilizzando le risorse disponibili in modo opportuno. Abbiamo trattato l'argomento in modo più approfondito nel libro "Raspberry Pi, il mio primo Linux embedded", quindi in questa sede ne faremo solo un breve cenno. Prendiamo come esempio lo shield descritto in questo articolo, che, come vedremo, richiede l'utilizzo del bus I²C, per la comunicazione con Raspberry Pi. Se costruiamo un'applicazione che gestisce il display LCD in esclusiva, il software da scrivere equivale a quello che avremmo scritto per un microcontrollore: un programma in un ciclo infinito che gestisce i pulsanti e che presenta sullo schermo le scritte in base alla logica predefinita. Se invece vogliamo che il display LCD sia di servizio rispetto ad una molteplicità di applicazioni dobbiamo gestire le cose in modo differente. Ipotizziamo ad esempio di voler sempre utilizzare lo shield come nel caso precedente, ma che sia in grado anche di visualizzare messaggi (provenienti da applicazioni diverse che lavorano in concomitanza) come l'arrivo di una chiamata telefonica, la variazione di un ingresso di I/O operata via web, la mancan-



rivolgersi alle architetture in grado di supportare applicazioni concorrenti e collaboranti, gestire comunicazioni tra i diversi attori, basate su server che orchestrano l'utilizzo di risorse da condividere e dove le diverse esigenze applicative devono essere realizzate come client che richiedono l'utilizzo delle risorse "centralizzate" mediante richieste basate su protocolli condivisi. Per esempio ipotizziamo che due risorse utilizzino in modo esclusivo le risorse a loro assegnate: lo shield LCD che utilizza il bus I²C e lo shield GSM/GPS (che vi presenteremo in seguito) che utilizza la porta seriale.

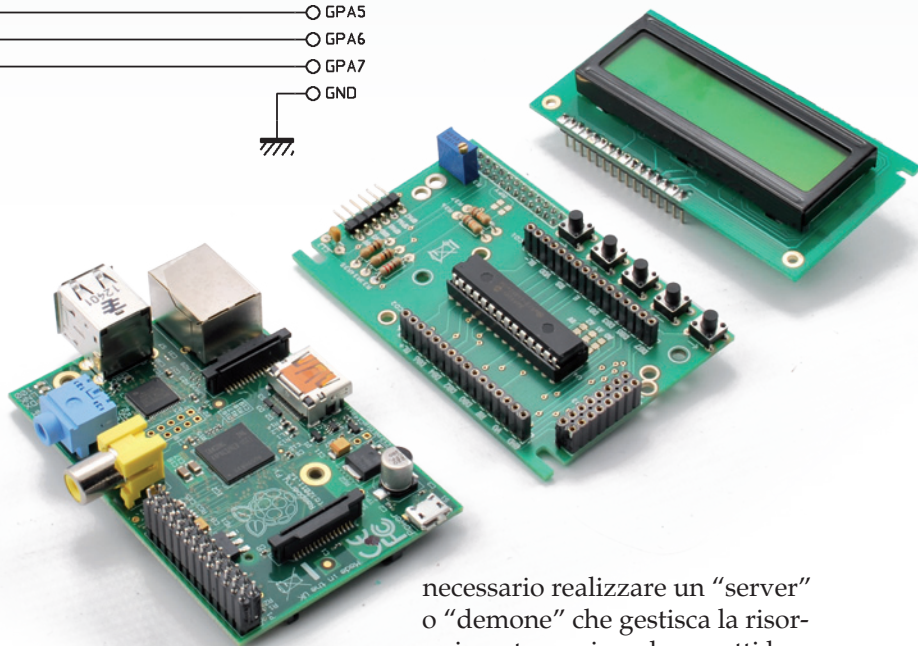
Per ciascuna di queste risorse è

za di corrente o di connettività di rete o qualsiasi altra cosa sia necessaria.

In un'architettura di questo tipo, diverse applicazioni funzionano in contemporanea sul sistema e, all'accadere di eventi prefissati, richiedono di impegnare il bus di comunicazione I²C verso l'LCD per comunicare il loro messaggio. Se un'applicazione tiene già impegnato il bus, per un motivo qualsiasi, le altre applicazioni che richiedono di impegnare lo stesso bus, trovandolo occupato (busy) non potranno fare altro che andare in errore. La prima soluzione che potrebbe venire in mente è di realizzare un'unica enorme applicazione che gestisca tutte le necessità in un modo monolitico, un po' come in un enorme microcontrollore. È intuitivo che questo modo di procedere comporti più aspetti

negativi che positivi: rigidità dell'applicazione, difficoltà di progettazione, realizzazione, manutenzione e aggiornamento, difficoltà di temporizzazione tra le diverse parti, fragilità (se una parte, anche insignificante, va in blocco, cade tutta l'applicazione). Per risolvere un'esigenza applicativa di questo tipo bisogna

necessario realizzare un "server" o "demone" che gestisca la risorsa in autonomia e che accetti le richieste dei diversi client su un canale di comunicazione predefinito. Vi sono molte alternative possibili: socket TCP/IP, funzionalità database, semplici file e così via. Per il momento non approfondiamo. L'importante è che i meccanismi di comunicazione siano in grado di gestire correttamente il fenomeno, per esempio generando code o



gestendo “semafori” per coordinare correttamente le richieste dei client. I client, infine, quando hanno bisogno di accedere a una delle risorse gestite centralmente, non vi devono accedere direttamente, ma devono utilizzare il canale di comunicazione predefinito per quella risorsa con il protocollo e le modalità previste. In sintesi, nel mondo embedded la progettazione non deve essere rivolta alla singola applicazione, ma deve tenere conto dell'architettura che si vuole realizzare nel suo insieme. Meglio o peggio rispetto ai microcontrollori? Questa domanda non ha una risposta definitiva. Dipende da quello che si vuole realizzare ed ai requisiti associati. Sicuramente ci troveremo invece sempre più

nelle condizioni di integrare e far collaborare questi due ambienti insieme ad altri come FPGA e PLC.

Fatta questa premessa, torniamo al nostro shield per il quale questa volta vi presentiamo un semplice programma di utilizzo, già adatto ad essere trasformato secondo l'architettura che vi abbiamo anticipato.

SCHEMA ELETTRICO

Per evitare di impegnare la maggior parte dei pin di ingresso/uscita presenti sul connettore

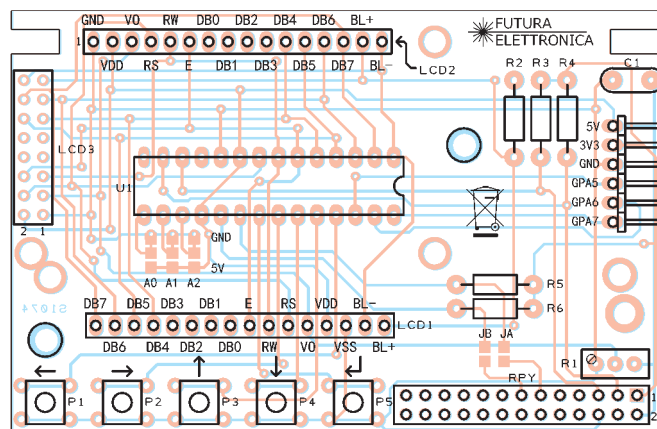
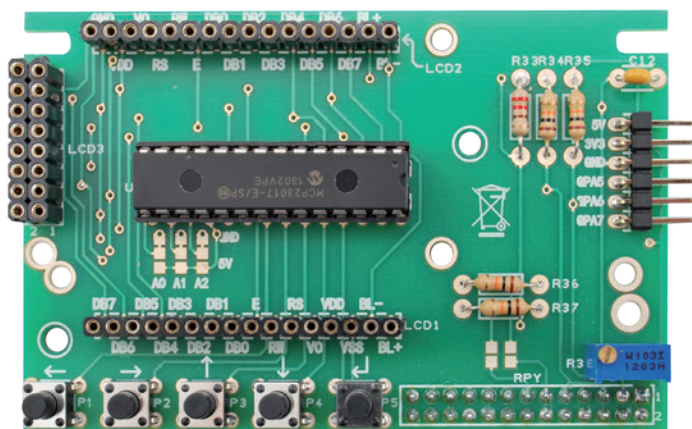
di Raspberry Pi abbiamo utilizzato l'integrato di Microchip MCP23017 che offre 16 ingressi/uscite digitali pilotabili mediante il bus I²C. Così facendo, impegnando in modo non esclusivo due soli pin del connettore di Raspberry Pi, possiamo pilotare uno schermo LCD, interfacciare cinque interruttori a pulsante e gestire tre ingressi ausiliari. La configurazione di base è pensata per realizzare un sistema di menu a cascata che permette di scegliere tra gerarchie di voci per poi configurare parametri o vi-

Rapiro, un robot umanoide con il cuore a forma di lampone

Tra le tante applicazioni più o meno scontate, dai web-server ai sistemi di riconoscimento targhe, dai media center alle stazioni meteo, ecco una delle prime applicazioni nel campo della robotica della economica board Linux Raspberry Pi: Rapiro, un piccolo robot umanoide completamente programmabile e come tale destinato principalmente al mondo “educational”. Rapiro dispone di 12 servocontrolli ed è in grado di camminare, muovere la testa e rispondere ai comandi



vocali; all'interno una scheda Raspberry Pi si occupa della gestione di tutti i sistemi e della connessione al controller remoto, tipicamente uno smartphone. Per la connessione viene utilizzato un dongle Bluetooth ed un apposito programma da caricare sullo smartphone. Il piccolo robot (alto circa 30 centimetri) dispone anche di una telecamera, occhi che si illuminano grazie a due LED, sensore di distanza e altoparlante. Per il controllo dei servo viene utilizzata una scheda Arduino-compatibile che garantisce la massima programmabilità. Secondo i fautori del progetto (che lo hanno proposto su Kickstarter ed hanno già raccolto i finanziamenti necessari), Rapiro verrà a costare $\frac{1}{4}$ rispetto a robot simili e un decimo rispetto ad altri robot umanoidi basati su schede Linux embedded. (www.kickstarter.com).



Elenco Componenti:

C1: 100 nF multistrato
R1: Trimmer multigiri 10 kohm
R2: 100 ohm
R3÷R6: 10 kohm
P1÷P5: Microswitch

U1: MCP23017-E/SP

Varie:

- Display LCD
- Torretta F/F 18 mm

- Vite 10mm 3 MA (2 pz.)

- Zoccolo 14+14

- Strip maschio 6 poli 90°

- Strip femmina 8 poli (2 pz.)

- Strip femmina 16 poli (2 pz.)

- Connettore Femmina 26 poli
per RaspberryPi

- Circuito stampato

sualizzare particolari set di informazioni. In realtà, da programma possiamo utilizzare i pulsanti e gli ingressi come meglio crediamo. Lo shield è provvisto di diversi connettori in grado di accogliere diversi tipi di LCD e in particolare i seguenti modelli che, pur avendo dimensioni diverse, condividono la stessa configurazione dei pin di comunicazione dati e comandi:

- LCD 16x2 retroilluminato (cod. Futura Elettronica 1446-ACM1602B-FL-YBW);
- LCD 16x2 retroilluminato bianco/nero (cod. Futura Elettronica 1446-LCD16x2WB);
- LCD 8x2 retroilluminato blu (cod. Futura Elettronica 1446-LCD8x2BN);

Tornando allo schema elettrico, l'integrato MCP23017, che descriviamo più approfonditamente nel riquadro dedicato, mette a disposizione 16 linee di I/O suddivise in due banchi da otto linee ciascuno, denominati GPA e GPB.

Il banco GPB è tutto dedicato alla gestione del display LCD come segue: i pin GPB1, GPB2, GPB3 e GPB4 sono collegati alle quattro linee dati dell'LCD, rispettivamente DB7, DB6, DB5 e DB4; il pin GPB5 comanda l'ingresso ENABLE dell'LCD, il pin GPB6 l'ingresso R/W ed il pin GPB7 l'ingresso RS che permette di selezionare i registri di istruzioni e scrittura.

I cinque interruttori a pulsante sono collegati ai pin del banco GPA: GPA0 al pulsante P5, GPA1 al pulsante P2, GPA2 al pulsante P4, GPA3 al pulsante P3 e GPA4 al pulsante P1.

L'alimentazione a 5V viene prelevata dal piedino 2 del connettore GPIO di Raspberry Pi e la massa è collegata al piedino 6. Il trimmer da 10 kohm, collegato al pin V0 dell'LCD permette di regolare il contrasto dello schermo.

Le uscite INTA e INTB sono collegate mediante i ponticelli JB e JA ai GPIO27 e GPIO17 di Raspberry Pi, in modo da renderli disponibili ad eventuali appli-

cazioni. Sullo shield sono resi disponibili su un connettore a pettine anche i pin GPA5, GPA6 e GPA7, del banco GPA dell'integrato MCP23017, "avanzati" dalla gestione dei pulsanti. Nel piano di montaggio è possibile vedere la disposizione dei componenti. Unico suggerimento, a parte le usuali attenzioni, è quello di posizionare il doppio connettore a pettine in modo da distanziare lo stampato dello shield da quello di Raspberry Pi per permetterne il montaggio "a pacchetto" nel caso lo si voglia utilizzare congiuntamente ad altri shield, come lo shield GSM che vi presenteremo nei prossimi numeri della rivista.

La possibilità di "impacchettamento" è data dal fatto che l'integrato MCP23017 è collegato al GPIO di Raspberry Pi unicamente con i pin necessari alla comunicazione I²C e all'alimentazione dello shield. Tutti gli altri pin di Raspberry Pi sono semplicemente passanti. Inoltre, anche se buona parte degli I/O dell'integrato

MCP23017 sono impiegati per la gestione del monitor LCD e dei pulsanti di comando presenti sullo shield, ne sono avanzati 3 che sono a disposizione per eventuali utilizzi personalizzati.

UTILIZZO PRATICO DELLO SHIELD

Come prima operazione montiamo uno dei display LCD sullo shield e poi montiamo lo shield sul connettore di Raspberry Pi, facendo attenzione che la parte inferiore dello shield non vada in contatto con i connettori USB o Ethernet. In caso di dubbio proteggiamo i connettori stessi con del nastro isolante. Colleghiamo periferiche, rete e alimentazione a Raspberry Pi nel solito modo e diamo tensione.

Per comunicare con l'integrato MCP23017 è necessario utilizzare il bus I²C e di conseguenza dobbiamo attivare il modulo di gestione del bus I²C che, come ormai saprete se ci seguite da un po', nella installazione predefinita di Raspbian è disabilitato. Per chi ci segue costantemente questa operazione non costituisce un problema, per chi è nuovo all'ambiente Raspbian - il sistema operativo GNU/Linux di Raspberry Pi - rimandiamo alle indicazioni riportate nell'articolo sulla gestione del termostato ADC con Raspberry Pi del numero 173 della rivista nonché

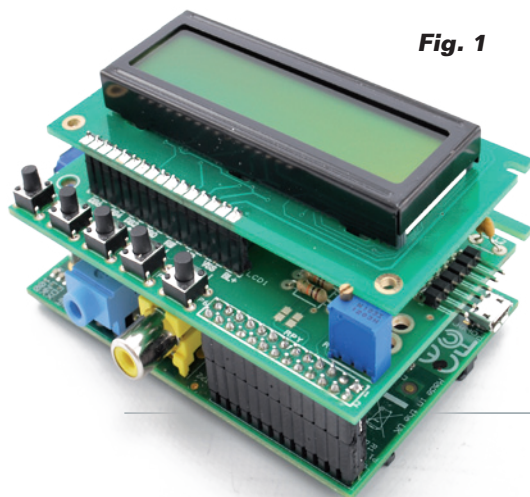


Fig. 1



Fig. 2

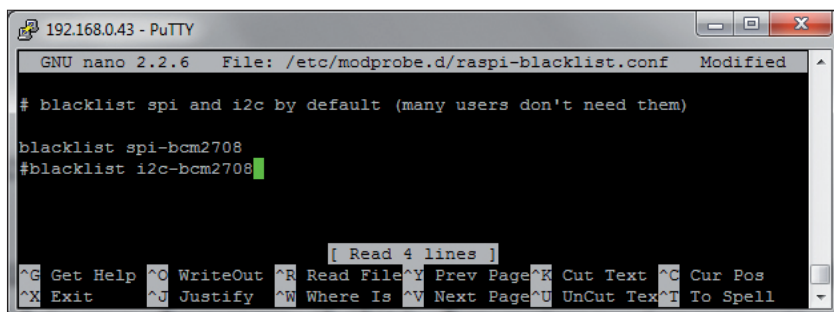


Fig. 3

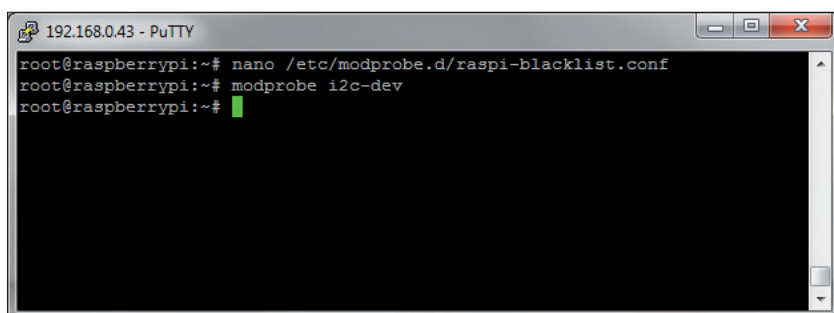


Fig. 4

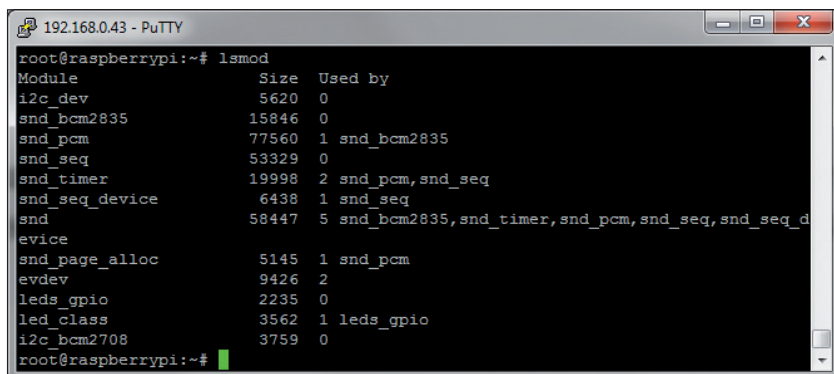


Fig. 5

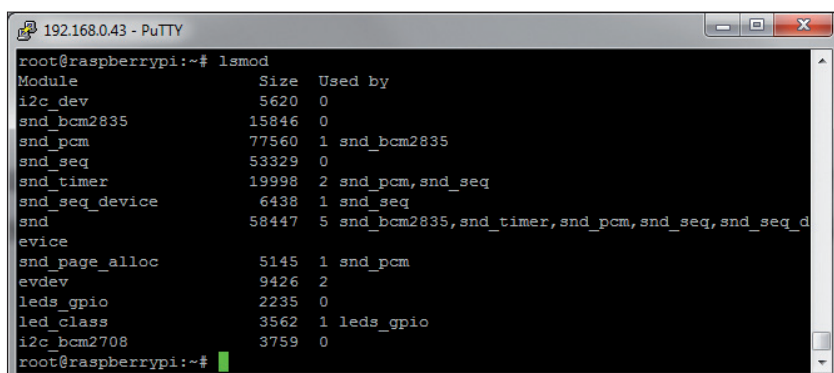


Fig. 6

Fig. 7

```

192.168.0.43 - PuTTY
root@raspberrypi:~# nano /etc/modules

```

Fig. 8

```

GNU nano 2.2.6      File: /etc/modules      Modified
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.
snd-bcm2835
i2c-dev

```

Fig. 9

```

192.168.0.43 - PuTTY
root@raspberrypi:~# apt-get install i2c-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  libi2c-dev python-smbus
The following NEW packages will be installed:
  i2c-tools
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 59.5 kB of archives.
After this operation, 223 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main i2c-tools armhf
3.1.0-2 [59.5 kB]
Fetched 59.5 kB in 1s (55.5 kB/s)

```

Fig. 10

```

After this operation, 223 kB of additional disk space will be used.
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main i2c-tools armhf
3.1.0-2 [59.5 kB]
Fetched 59.5 kB in 1s (55.5 kB/s)
Selecting previously unselected package i2c-tools.
(Reading database ... 61438 files and directories currently installed.)
Unpacking i2c-tools (from .../i2c-tools_3.1.0-2_armhf.deb) ...
Processing triggers for man-db ...
Setting up i2c-tools (3.1.0-2) ...
root@raspberrypi:~#
root@raspberrypi:~# adduser pi i2c
Adding user 'pi' to group 'i2c' ...
Adding user pi to group i2c
Done.
root@raspberrypi:~#

```

al libro “Raspberry Pi il mio primo Linux embedded” (codice 8330-RASPBOOK1).

Ricordiamo solo che per gli articoli sulla rivista dedicati a Raspberry Pi abbiamo adottato il sistema operativo Raspbian, in costante aggiornamento e miglioramento e gli strumenti di gestione remota tramite protocollo SSH, Putty (o Kitty) e WINScp.

Ora dobbiamo abilitare il driver per la gestione del bus I²C, installare la libreria in Python per la gestione dell’LCD e realizzare un primo programma di prova per vedere che tutto funzioni a dovere. Ricordiamo brevemente il processo per aggiornare il sistema operativo ed abilitare il driver per il bus I²C. Innanzitutto diamo i comandi (come utente “root”):

```

apt-get update
apt-get upgrade

```

Per rendere utilizzabile il modulo di gestione del bus I²C è necessario toglierlo dalla blacklist e poi “aggiungerlo” all’insieme dei moduli conosciuti dal kernel. Apriamo il file di configurazione che contiene l’elenco dei moduli blacklisted (oscurati), con il comando (Fig. 2):

```

nano /etc/modprobe.d/raspi-blacklist.conf

```

Nano è un editor di testo minimale che funziona in ambiente terminale.

Eliminiamo il modulo I²C dalla blacklist cancellando la riga o, come abbiamo fatto noi, commentandola con un “#” (Fig. 3). Premiamo Ctrl-X e poi Y alla richiesta di salvare il file dopo le modifiche.

Eseguiamo un reboot per rendere effettive le modifiche.

Ora dobbiamo fare in modo che il modulo “liberato” venga ca-

ricato e diventi parte integrante del kernel. Per questa operazione abbiamo due possibilità: la prima ci permette di caricare il modulo a comando, e ha validità per tutto il tempo nel quale Raspberry Pi rimane acceso. Al boot successivo il modulo dovrà essere ricaricato a comando. La seconda ci permette di caricare il modulo direttamente al boot del sistema operativo e renderlo disponibile alle applicazioni subito dopo il boot, condizione indispensabile in un sistema server unattended. La prima possibilità richiede l'utilizzo del comando *modprobe*. Scriviamo (Fig. 4):

modprobe i2c-dev

Possiamo vedere il buon esito dell'attivazione dei driver con il comando che mostra la lista di tutti i moduli installati (Fig. 5):

lsmod

Dato che in GNU/Linux tutto (o quasi) è un file, se andiamo nella cartella */dev* vediamo apparire i file di collegamento ai device *i2c-0* ed *i2c-1* (Fig. 6).

Il comando *modprobe* permette di caricare e scaricare i moduli a runtime e mantiene i suoi effetti fintanto che Raspberry Pi resta acceso. In caso di spegnimento, o anche solo di reboot, il nostro modulo dovrà essere ricaricato manualmente.

Questa condizione non è adatta a funzionare con un'applicazione stand alone, che deve funzionare in modo automatico. Il comando *modprobe*, con l'opzione *remove*, può essere utilizzato anche per disattivare un modulo caricato in precedenza.

modprobe -r i2c-dev

Se si desidera che il modulo

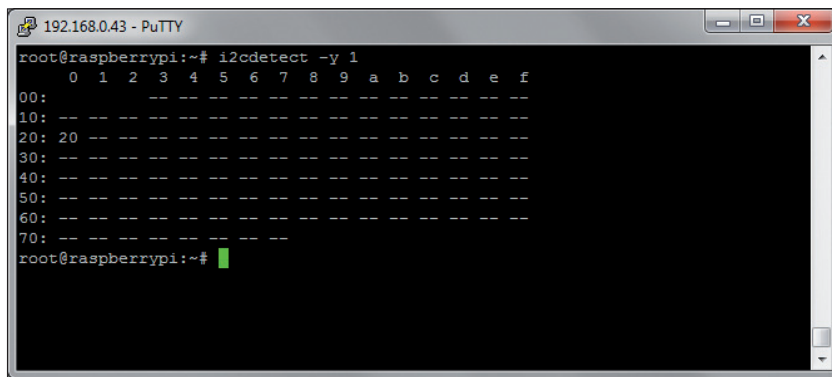


Fig. 11

venga caricato all'accensione di Raspberry Pi è necessario abilitare il caricamento permanente del driver I²C che si realizza modificando opportunamente il file di configurazione */etc/modules*, contenente la lista dei driver da caricare al momento del boot. In caso contrario dobbiamo ricordarci di caricare il modulo ad ogni accensione con *modprobe*. Per modificare il file possiamo usare il comando:

nano /etc/modules

ed aggiungere una nuova linea al file di configurazione che contiene (Fig. 7)

i2c-dev

Premete CTRL X e poi y per salvare le modifiche al file ed uscire (Fig. 8).

Ora installiamo il pacchetto *i2c-tools* che ci fornisce una serie di funzioni utilizzabili a linea di comando per verificare il funzionamento del bus I²C (Fig. 9):

apt-get install i2c-tools

Aggiungiamo il nostro utente *pi* al gruppo I²C (Fig. 10):

adduser pi i2c

Eseguiamo il reboot di Raspberry Pi per attivare le nuove configurazioni con il comando

reboot

Dopo che Raspberry Pi si è riavviato e vi siete riconnessi con Putty o Kitty, verificate se sul bus I²C è visibile il convertitore ADC con il comando:

i2cdetect -y 0 per Raspberry Pi rev. 1
oppure
i2cdetect -y 1 per Raspberry Pi rev. 2

Dovrete ottenere un risultato simile a quello visibile in Fig. 11 dove l'indirizzo 0x20 identifica l'integrato MCP23017.

Ora possiamo installare la libreria di supporto all'LCD. Utilizzeremo la libreria messa a disposizione con licenza BSD da Adafruit Industries nel repository GitHub. Il procedimento migliore per scaricarla è utilizzare lo

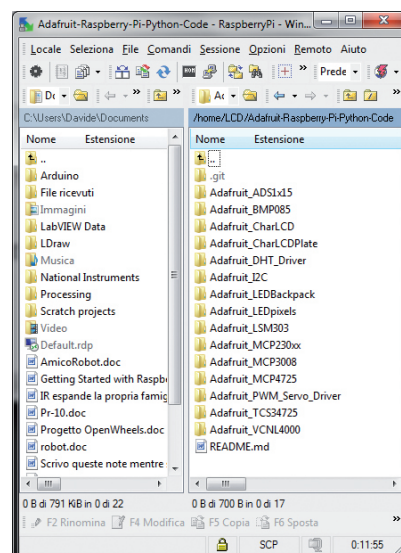
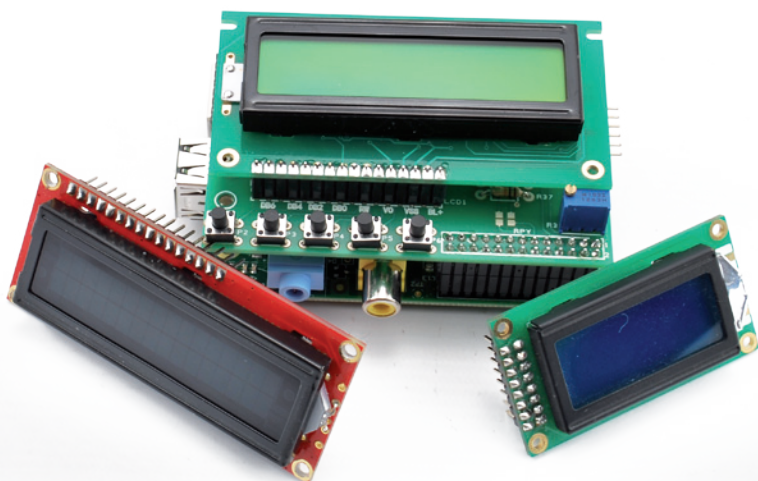


Fig. 12



strumento di gestione delle versioni di software "git".

Per installare "git" usiamo il comando:

```
apt-get install git
```

poi andiamo alla cartella home con il comando:

```
cd /home
```

creiamo una cartella per il nostro progetto, per esempio:

```
mkdir LCD
```

posizioniamoci nella cartella con il comando:

```
CD LCD
```

e scarichiamo le librerie Python realizzate da Adafruit con il comando:

```
git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
```

infine ci posizioniamo nella cartella che contiene la libreria per la gestione dell'LCD con i comandi:

```
cd Adafruit-Raspberry-Pi-Python-Code
```

Possiamo vedere il contenuto della cartella posizionandoci sulla stessa con WinSCP (**Fig. 12**).

Posizioniamoci nella cartella che contiene la libreria Adafruit_CharLCDPlate con il comando:

```
cd Adafruit_CharLCDPlate
```

Installiamo la libreria per la gestione del protocollo I²C con il linguaggio Python:

```
apt-get install python-smbus
```

A questo punto siamo quasi pronti a verificare il funzionamento dello shield LCD.

Nella libreria Adafruit_CharLCDPlate, i pin GPA6 e GPA7 sono utilizzati per la gestione dei colori di un LCD a colori RGB. Nel nostro caso, utilizzando LCD monocromatici, lasciamo disponibili i suddetti pin. Per poterli utilizzare è necessario modificare la libreria in modo da poter utilizzare i pin, per esempio, come ingressi.

Per comprendere le modifiche fate riferimento alle informazioni contenute nel riquadro dedicato all'integrato MCP23017. La libreria contiene il codice che esprime la classe Adafruit_CharLCDPlate per la gestione dei display LCD. Nel costruttore della classe abbiamo modificato l'impostazione iniziale dei registri per rendere i pin GPA5, GPA6 e GPA7 dei pin di ingresso, con resistenza di pull up interna e funzionanti a logica invertita (valore "1" quando il pin è a massa).

Potete scaricare la libreria modificata all'indirizzo www.elettronica.in.it. In particolare abbiamo modificato le seguenti linee:

```
riga 99 [ 0b11111111, # IODIRA  R+G LEDs=outputs, buttons=inputs orig. 0b00111111
riga 101 0b11111111, # IPOLA  Invert polarity on button inputs orig. 0b00111111
riga 112 0b11111111, # GPPUA  Enable pull-ups on buttons orig. 0b00111111
```

Alla riga 425 abbiamo modificato il metodo backlight in modo da eliminare l'utilizzo dei pin del banco GPA per la gestione del colore.

```
def backlight(self, color):
    c = ~color
    # self.porta = (self.porta & 0b00111111) | ((c & 0b011) << 6)
    self.portb = (self.portb & 0b11111110) | ((c & 0b100) >> 2)
    # Has to be done as two writes because sequential operation is off.
    # self.i2c.bus.write_byte_data(
    #     self.i2c.address, self.MCP23017_GPIOA, self.porta)
    self.i2c.bus.write_byte_data(
        self.i2c.address, self.MCP23017_GPIOB, self.portb)
```

Infine (riga 436) abbiamo aggiunto tre metodi per la lettura dei singoli pin "avanzati".

```
# Read and return bitmask of pin A5
def buttonpin5(self):
    return (self.i2c.readU8(self.MCP23017_GPIOA) >> 5) & 1

# Read and return bitmask of pin A6
def buttonpin6(self):
    return (self.i2c.readU8(self.MCP23017_GPIOA) >> 6) & 1

# Read and return bitmask of pin A7
```

```
def buttonpin7(self):
    return (self.i2c.readU8(self.MCP23017_GPIOA) >> 7) & 1
```

Abbiamo preparato un programma di prova modificando il programma di test incluso nella cartella della libreria, ottenendo il programma visibile nel **Listato 1**.

Il programma, premendo i pulsanti presenti sullo shield, permette di ottenere informazioni sulle performance e lo stato del sistema, sintetizzate sul display LCD.

Copiate il programma in un file di nome LCDProva.

py, oppure scaricatelo dal sito www.elettronica.in. Ricordatevi di lanciare il comando:

modprobe i2c-dev

Posizionatevi nella cartella Adafruit_CharLCDPlate e lanciate il programma con il comando:

python LCDProva.py

Dovreste vedere il messaggio di apertura sul display LCD e poi il menu dei pulsanti.

Listato 1

```
#!/usr/bin/python

from time import sleep
from Adafruit_CharLCDPlate import Adafruit_CharLCDPlate
from subprocess import *

cmd_ip = "ip addr show eth0 | grep inet | awk '{print $2}' | cut -d/ -f1"
cmd_cpu = "mpstat | awk '$11 ~ /[0-9.]+/ { print 100 - $11}'"
cmd_dfh = "df -h | grep /dev/root | awk '{ print $2 }'"
cmd_dfh_p = "df -h | grep /dev/root | awk '{ print $5 }'"
cmd_ps = "ps -ef | wc -l"

def run_cmd(cmd):
    p = Popen(cmd, shell=True, stdout=PIPE)
    output = p.communicate()[0]
    return output

class CPU(object):
    def __init__(self):
        """Init a CPU status object"""
        stat_fd = open('/proc/stat')
        stat_buf = stat_fd.readlines()[0].split()

        self.prev_total = float(stat_buf[1]) + float(stat_buf[2]) + float(stat_buf[3]) + float(stat_buf[4]) +
float(stat_buf[5]) + float(stat_buf[6]) + float(stat_buf[7])
        self.prev_idle = float(stat_buf[4])
        stat_fd.close()

    def usage(self):
        """return the actual usage of cpu (in %)"""
        stat_fd = open('/proc/stat')
        stat_buf = stat_fd.readlines()[0].split()
        total = float(stat_buf[1]) + float(stat_buf[2]) + float(stat_buf[3]) + float(stat_buf[4]) + float(stat_
buf[5]) + float(stat_buf[6]) + float(stat_buf[7])
        idle = float(stat_buf[4])
        stat_fd.close()
        diff_idle = idle - self.prev_idle
        diff_total = total - self.prev_total
        usage = 100.0 * (diff_total - diff_idle) / diff_total
        usage = usage / 10
        usage = round(usage, 1)
        self.prev_total = total
        self.prev_idle = idle
        return usage

    def memUsage():
        free_fd = os.popen('free -b')
        free_buf = free_fd.readlines()[1].split()
        usage = (float(free_buf[2]) / (float(free_buf[1]))) * 100
        usage = round(usage, 1)
        return usage
```

(Continua)

Listato 1 – SEQUE

```
cpu = CPU()

# Initialize the LCD plate. Should auto-detect correct I2C bus. If not,
# pass '0' for early 256 MB Model B boards or '1' for all later versions
lcd = Adafruit_CharLCDPlate()

# Clear display and show greeting, pause 1 sec
lcd.clear()
lcd.message("  RaspberryPi\n ElettronicaIn")
sleep(5)
lcd.clear()
lcd.message("P1=M P2=IP P3=CPU\nP4=DSK  P5=Proc.")

# Poll buttons, display message & set backlight accordingly
btn = ((lcd.LEFT , 'Pulsante 1' , lcd.BLUE),
       (lcd.UP , 'Pulsante 2' , lcd.BLUE),
       (lcd.DOWN , 'Pulsante 3' , lcd.BLUE),
       (lcd.RIGHT , 'Pulsante 4' , lcd.BLUE),
       (lcd.SELECT, ' ' , lcd.ON))
prev = -1

while True:
    pin5 = lcd.buttonpin5()
    pin6 = lcd.buttonpin6()
    pin7 = lcd.buttonpin7()
    pinT = pin5 * 100 + pin6 * 10 + pin7
    pinS = ("000" + str(pinT))[-3:]

    for b in btn:
        if lcd.buttonPressed(b[0]):
            if b is not prev:
                print b[0]
                if b[0] == 0:
                    lcd.clear()
                    lcd.backlight(lcd.ON)
                    ps = run_cmd(cmd_ps)
                    lcd.message('N.Processi %s\nP1=Menu  pin %s' % (ps , str(pinS) ))
                elif b[0] == 1:
                    lcd.clear()
                    lcd.backlight(lcd.ON)
                    ipaddr = run_cmd(cmd_ip)
                    lcd.message('IP %s \nP1=Menu  pin %s' % ( ipaddr , str(pinS) ))
                elif b[0] == 2:
                    lcd.clear()
                    lcd.backlight(lcd.ON)
                    dfh = run_cmd(cmd_dfh)
                    dfh_p = run_cmd(cmd_dfh_p)
                    lcd.message('DSK %s USED %s\nP1=Menu  pin %s' % (dfh.strip(), dfh_p.strip() , str(pinS)))
                elif b[0] == 3:
                    lcd.clear()
                    lcd.backlight(lcd.ON)
                    cpul = float(cpu.usage())
                    lcd.message('CPU %.2f%%\nP1=Menu  pin %s' % (cpul , str(pinS)))
                elif b[0] == 4:
                    lcd.clear()
                    lcd.backlight(lcd.ON)
                    lcd.message("P1=M P2=IP P3=CPU\nP4=DSK  P5=Proc.")
            prev = b
    break
```

Se non vedete nulla o lo schermo ha contrasto eccessivo, potete regolarlo mediante il potenziometro R1 fino ad ottenere un risultato soddisfacente. Come potete vedere, il listato è molto semplice: la maggior parte del lavoro viene svolto dai metodi forniti dalla libreria. All'apertura del programma

importiamo la classe `sleep`, per gestire i ritardi di esecuzione, la libreria `Adafruit_CharLCDPlate` e la libreria `subprocess`. La prima istruzione del programma richiama un metodo che permette di determinare a quale revisione appartiene il Raspberry Pi in nostro possesso e imposta di



Fig. 13



Fig. 14



Fig. 15



Fig. 16



Fig. 17

conseguenza il bus I2C, 0 o 1. Poi viene inviato il messaggio di apertura sul display, e, successivamente con un ritardo di un secondo, il messaggio di menu.

Infine, nel ciclo while vengono letti i valori dei tre pin di ingresso e poi viene intercettato quale pulsante viene premuto per eseguire la funzione corrispondente con la relativa presentazione dei risultati sul display.

I valori dei pin sono presentati in basso a destra nel display di ciascun comando.

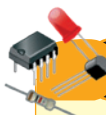
A scopo didattico, in questo programma di esempio, abbiamo voluto presentare il risultato di alcuni comandi che permettono di raccogliere informazioni sullo stato e/o sulle performance del sistema. Inutile ricordare che gli stessi comandi possono essere digitati nella finestra terminale come comandi normali, ottenendo gli stessi risultati.

I comandi sono eseguiti lanciando processi esterni al programma in esecuzione e recuperando il risultato alla fine dell'elaborazione (**Listato 1**).

In particolare la pressione del pulsante P2 richiama il menu; il pulsante P3 mostra l'indirizzo IP di Raspberry Pi e lo stato dei tre pin d'ingresso aggiuntivi, P4 l'utilizzo di CPU, P5 lo spazio occu-

pato su SD Card o su disco - sia in GB, sia in percentuale - e P6 il numero di processi in esecuzione su Raspberry Pi. Per modificare lo stato dei pin di ingresso potete utilizzare sperimentalmente e con molta attenzione un cavetto con i terminali femmina/femmina alle estremità. Collegate un'estremità del cavetto al terminale a massa e l'altro terminale a uno dei pin di ingresso. Premete uno dei pulsanti da P3 a P6 e dovrete vedere lo stato dei pin in basso a destra (**Fig 13, 14, 15, 16, 17**).

Nei prossimi numeri vi presenteremo ulteriori shield di espansione e approfondiremo le modalità di progettazione e realizzazione di programmi che rispettano lo schema architetturale descritto all'inizio dell'articolo.



per il MATERIALE

Tutti i componenti necessari a realizzare il progetto descritto in queste pagine sono facilmente reperibili in commercio; il master del circuito stampato nonché il firmware possono essere scaricati gratuitamente dal sito della rivista (www.elettronica.in.it). Lo shield LCD è anche disponibile in scatola di montaggio (cod. FT1074K) al prezzo di 17,00 Euro; il kit viene fornito col display LCD 16x2 retroilluminato (codice Futura ACM1602B-FL-YBW). Con questo kit possono essere utilizzati anche altri display quali il modello LCD 16x2 retroilluminato bianco/nero cod. LCD16x2WB (Euro 15,00) e il display LCD 8x2 retroilluminato blu cod. LCD8x2BN (Euro 7,50). 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

Continuiamo a illustrare le principali funzioni del GSM/GPRS/GPS shield e della relativa libreria attraverso semplici esempi da cui potete prendere spunto per i vostri progetti.

GESTIRE I MODULI GSM CON ARDUINO

di MARCO MARTINES

Siamo giunti alla quarta puntata per quanto riguarda gli esempi relativi allo shield GSM/GPRS/GPS. Nelle scorse puntate abbiamo visto come gestire gli SMS, effettuare chiamate e utilizzare la connessione GPRS per l'invio di email ed effettuare il download di pagine internet. Lo shield, ricordiamo, è stato sviluppato per alloggiare entrambi i moduli di SIMCom: SIM900 e SIM908. Quest'ultimo si caratterizza per avere al suo interno un ricevitore GPS a 42 canali, gestibile con comandi AT. Rimangono dunque da affrontare, in quest'ultima puntata, le funzioni della libreria sviluppate per la gestione del GPS. Il SIM908, in aggiunta alla presenza del GPS, offre la possibilità di gestire batterie agli ioni di litio. Oltre a

trarre alimentazione da esse, se collegato a un'alimentazione esterna, ne gestisce la ricarica.

Vedremo in questa serie di esempi come gestire le funzionalità aggiuntive fornite dal SIM908 relative alla geolocalizzazione.

Ovviamente, per chi utilizza il SIM900 questi esempi non funzioneranno.

ESEMPIO 1

Come primo sketch analizzeremo quello presente nella cartella della libreria che contiene diversi esempi (lo troviamo con il nome GSM_GPRS_Library_GPS).

Iniziamo con questo per prendere confidenza con le funzioni base per poi comprendere meglio il funzionamento degli esempi successivi.

Abbiamo parlato di GSM...

Le nostre attività di progettazione di applicazione con la rete GSM risalgono a parecchi anni fa, quando ancora non c'erano i moduli e si usavano i telefonini con degli appositi cavi dati per accedere alla rete cellulare. Con i moduli l'interfacciamento e il controllo si sono semplificati e già nel numero di Aprile 2011 abbiamo affrontato il tema dell'interfacciamento dei moduli GSM con Arduino. La soluzione pubblicata era basata su uno shield dedicato al SIM900 (TDGSM_900) mentre gli Arduino erano ancora alla versione Duemilanove. Nell'articolo, oltre a illu-



Fig. A

strare la nostra scheda su cui il modulo GSM era montato e reso più facilmente interfacciabile, venivano anche illustrate le funzioni accessibili tramite una ricca serie di comandi – implementati tramite libreria – da impartire con la comunicazione seriale alla scheda (Fig. A).

La nostra attività di progettazione e presentazione dei moduli GSM ha però radici più lontane: il modulo SIM900 è stato utilizzato anche nell'articolo che abbiamo pubblicato a Marzo 2011; era il localizzatore GSM basato proprio su questo modulo e per il quale sono state fornite ampie spiegazioni sull'interfacciamento e il funzionamento. Sempre di SIM900 abbiamo parlato nel numero Maggio/Giugno 2010 con il Telecontrollo GSM Low Cost che iniziava la serie dei nostri articoli basati su questo modulo applicandolo a un funzionale apricancello tanto innovativo quanto economica.

La tecnologia dei moduli si è evoluta con rapidità e quindi a Marzo 2012 abbiamo realizzato un nuovo localizzatore basato sul modulo SIM908 che,

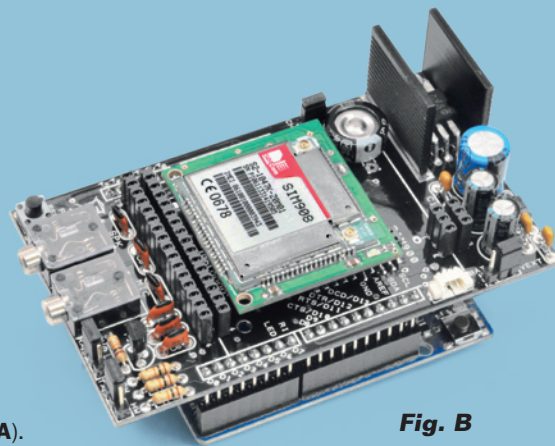


Fig. B

rispetto al precedente, dispone anche di un ricevitore GPS integrato. Nel progetto si spiega l'impiego di entrambe le funzioni del modulo per ottenere un posizionamento molto preciso e la trasmissione del dato con le varie modalità possibili.

Ad Aprile 2012, infine, abbiamo presentato uno shield universale (8190-GSMGPRSSHIELDV2) su cui montare le schede GSM basate su SIM900 e SIM908 di SIMcom per interfacciarle con schede Arduino UNO o compatibili (Fig. B).

Potete scaricare in versione elettronica tutti gli articoli citati utilizzando il nostro servizio per l'accesso e la consultazione degli arretrati disponibile su www.elettronica.in.it.

Si ricorda che per il corretto funzionamento del GPS è necessario effettuare un ponte sul jumper J1 presente sul modulo contenente il SIM908, che viene montato sullo shield.

Oltre al ponticello, indispensabile per portare l'alimentazione all'antenna, è necessario collegare quest'ultima e posizionarla all'esterno, altrimenti si rischia di pregiudicare il funzionamento del GPS.

Se posizionata all'interno di edifici o luoghi chiusi, difficilmente il ricevitore riuscirà ad effettuare il fix dei satelliti.

Come sempre, per ogni sketch relativo a questo shield, è necessario includere la libreria relativa (in questo caso `gps.h`) e in seguito istanziarne un oggetto. Tale procedimento può essere effettuato con i seguenti comandi:

```
#include "gps.h"
GPSGSM gps;
```

Ogni funzione appartenente a tale classe, per poter essere utilizzata, necessita di essere richiamata nel seguente modo.

```
nome_oggetto.nome_funzione(parametri)
```

Dopo aver definito diverse variabili necessarie per il funzionamento dello sketch, osserviamo la prima funzione destinata all'accensione del ricevitore GPS.

```
gps.attachGPS()
```

Con questa chiamata comunichiamo al modulo SIM908 di abilitare le funzioni GPS. L'utilizzo del GPS (così come in tutti gli smartphone) fa aumentare l'assorbimento di corrente elettrica da parte del modulo, con conseguente minore durata della batteria. Nel caso di alimentazione da rete elettrica è necessario che l'alimentatore utilizzato supporti picchi di corrente di circa 2A.

Dall'istante in cui viene chiamata questa funzione è necessario lasciare al modulo un po' di tempo per effettuare il fix dei satelliti, proprio come accade per i cellulari dotati di GPS. Troviamo dunque un ritardo per consentire al modulo di effettuare questa operazione (il tempo potrebbe anche essere inferiore, si consiglia al lettore di individuare il tempo ideale in base alla propria applicazione).

```
delay(20000);
```


Tabella 1

char getStat()	Controlla lo stato del GPS, i possibili output sono: 0: GPS non avviato. 1: Fix non eseguito (può essere causato da un segnale di scarsa qualità). 2: Fix 2D eseguito. 3: Fix 3D eseguito.	char stat; stat=gps.getStat(); if(stat==3) Serial.println("FIXED");
----------------	--	--

Allo stesso modo dei vari dispositivi dotati di GPS, il modulo presenta due modalità di avvio delle funzioni GPS a seconda che queste siano state utilizzate recentemente o meno. Infatti è prevista una modalità di "Hot Start" per compiere il fix in tempi brevi se questo è stato effettuato recentemente. Tale funzione risulta trasparente all'utente e viene gestita dal modulo autonomamente.

A questo punto, dopo aver attivato il modulo, verifichiamo il suo stato con il comando

```
gps.getStat();
```

che restituisce un valore rappresentante lo stato del ricevitore GPS.

Gli stati possibili sono 4 (GPS OFF, NOT FIXED, 2D FIXED, 3D FIXED).

Per quanto riguarda il significato si rimanda il lettore alla **Tabella 1** descrittiva della funzione.

Una volta che ci siamo assicurati dello stato del GPS relativamente al fix, possiamo richiedere le coordinate con il comando:

```
gps.getPar(lon,lat,alt,time,vel);
```

La funzione salverà le coordinate all'interno delle stringhe passate come parametro.

In tal modo sarà sufficiente stampare tali stringhe per conoscerne il contenuto.

Ovviamente se fosse necessario effettuare delle operazioni con i dati ottenuti, si dovrà convertire la stringa in intero o float con apposite funzioni.

Una volta ottenuti i dati, questi vengono stampati con la classica funzione dedicata (**Listato 1**).

```
Serial.print(char* stringa);
```

ESEMPIO 2

Il secondo esempio tratta l'interpretazione e la formattazione dei dati, relativi alla posizione geografica, forniti dal modulo SIM908.

In commercio esistono svariati dispositivi in grado di identificare la propria posizione fornendo le coordinate geografiche globali. Tali dispositivi vengono identificati genericamente come GPS (Global Position System).

Il problema è che, sebbene la posizione all'interno del globo sia univoca, esistono diversi formati per descri-

Listato 1

```
#include "SIM900.h"
#include <SoftwareSerial.h>
#include "gps.h"

GPSSGSM gps;

char lon[15];
char lat[15];
char alt[15];
char time[20];
char vel[15];
char msg1[5];
char msg2[5];

char stat;
char inSerial[20];
int i=0;
boolean started=false;

void setup()
{
  //Serial connection.
  Serial.begin(9600);
  Serial.println("GSM Shield testing.");
  //Start configuration of shield with baudrate.
  if (gsm.begin(9600)){
    Serial.println("\nstatus=READY");
    gsm.forceON(); //To ensure that SIM908 is not only _
                  in charge mode
    started=true;
  }
  else Serial.println("\nstatus=IDLE");

  if(started){
    //GPS attach
    if (gps.attachGPS())
      Serial.println("status=GPSREADY");
    else Serial.println("status=ERROR");

    delay(20000); //Time for fixing
    stat=gps.getStat();
    if(stat==1)
      Serial.println("NOT FIXED");
    else if(stat==0)
      Serial.println("GPS OFF");
    else if(stat==2)
      Serial.println("2D FIXED");
    else if(stat==3)
      Serial.println("3D FIXED");
    delay(5000);
    //Get data from GPS
    gps.getPar(lon,lat,alt,time,vel);
    Serial.println(lon);
    Serial.println(lat);
    Serial.println(alt);
    Serial.println(time);
    Serial.println(vel);
  }
};

void loop()
{
};
```

Listato 2

```
#include "SIM900.h"
#include <SoftwareSerial.h>
#include <stdlib.h>
#include "gps.h"

GPSSGSM gps;

char lon[15];
float lon_db;
char lat[15];
float lat_db;
char alt[15];
char time[20];
char vel[15];
char msg1[5];
char msg2[5];

char stat;
char inSerial[20];
int i=0;
boolean started=false;

void setup()
{
    //Serial connection.
    Serial.begin(9600);
    Serial.println("GSM Shield testing.");
    //Start configuration of shield with baudrate.
    if (gsm.begin(2400)){
        Serial.println("\nstatus=READY");
        gsm.forceON(); //To ensure that SIM908 is not _
only in charge mode
        started=true;
    }
    else Serial.println("\nstatus=IDLE");

    if(started){
        //GPS attach
        if (gps.attachGPS())
            Serial.println("status=GPSREADY");
        else Serial.println("status=ERROR");
    }
    delay(20000); //Time for fixing
};

void loop()
{
    //Read for new byte on serial hardware,
    //and write them on NewSoftSerial.
    serialhwread();
    //Read for new byte on NewSoftSerial.
    serialswread();
};

void serialhwread(){
    i=0;
    if (Serial.available() > 0){
        while (Serial.available() > 0) {
            inSerial[i]=(Serial.read());
            delay(10);
            i++;
        }

        inSerial[i]='\0';
        if(!strcmp(inSerial,"/END")){
            Serial.println("_");
            inSerial[0]=0x1a;
            inSerial[1]='\0';
            gsm.SimpleWriteIn(inSerial);
        }
        //Send a saved AT command using serial port.
        if(!strcmp(inSerial,"GPS")){
```

Continua

verla. Può dunque sorgere il problema di ritrovarsi ad utilizzare differenti formati con il rischio di confondersi e non venirne a capo. Per questo motivo risulta di fondamentale importanza analizzare sia il formato utilizzato dal SIM908 sia come convertirlo in altri formati, anch'essi largamente diffusi (come quello adottato da Google).

Il SIM908 utilizza il formato descritto dalla sigla *ddmm.mmmm*, mentre altri dispositivi e servizi (come ad esempio Google Maps) utilizzano il formato decimale *dd.ddddd*.

Il formato decimale è diffusamente impiegato, per la sua maneggevolezza, nel calcolo delle distanze e della velocità.

Risulta dunque necessario conoscere la relazione tra i due formati in modo da poter utilizzare quello che si preferisce in base all'esigenza.

La relazione che lega questi due formati è espressa dalla seguente formula:

$$dd.ddddd = dd + mm.mmmm / 60$$

Si ricorda che la libreria fornisce le coordinate sotto forma di stringa, dunque è necessario implementare o utilizzare un metodo per convertire tali coordinate in formato numerico se le si vuole utilizzare all'interno di una qualsiasi formula.

Una di queste funzioni è contenuta all'interno della libreria `stdlib.h`, includibile con il seguente comando:

```
#include <stdlib.h>
```

All'interno di tale libreria troviamo la funzione

```
atof (const char* str);
```

che restituisce un float contenente il valore espresso dalla stringa.

Per comprendere meglio quanto espresso in questo paragrafo analizziamo l'esempio relativo.

Nell'esempio proposto verrà mostrato come convertire il formato *ddmm.mmmm* del SIM908 nel formato *dd.dddd* largamente utilizzato.

Innanzitutto, per differenziarlo dall'esempio precedente, si è scelto di effettuare la geolocalizzazione solo in seguito alla richiesta dell'utente attraverso uno specifico comando inviato tramite la comunicazione seriale.

In particolare, quando l'utente vorrà conoscere la propria posizione, espressa in coordinate *dd.dddd*, dovrà inviare la stringa "GPS" attraverso la comunicazione seriale. Lo shield risponderà con i due valori di latitudine e longitudine nel formato compatibile con le applicazioni di Google (sarà sufficiente inserire queste

nella barra di ricerca di Google Maps per visualizzare la posizione sulla mappa).

Per l'interpretazione dei messaggi inviati dall'utente sulla porta seriale hardware si è scelto, ad ogni iterazione del loop, di salvare tutti i caratteri presenti su tale porta all'interno di un array di char.

Utilizzando successivamente la funzione:

```
strcmp(inSerial, "GPS")
```

si andrà a verificare se l'array contiene effettivamente il comando prestabilito, in questo caso "GPS".

Una volta verificata la condizione, se positiva, si entrerà nella porzione di codice destinata all'esecuzione di tale comando.

Dopo i comandi per richiedere la longitudine e latitudine nel formato nativo del SIM908 - che non affronteremo in quanto già analizzato nell'esempio precedente - troviamo la funzione *convert* utilizzata per la conversione delle coordinate.

La funzione è stata sviluppata appositamente ed è presente alla fine del codice.

Il suo funzionamento consiste, inizialmente, nel dividere la stringa nelle due parti *dd* e *mm.mmmmm*.

Il primo ostacolo che si incontra riguarda la prima parte relativa ai gradi, descritta con la sigla *dd*: questa non sempre risulta composta da due caratteri. Se il numero che rappresenta è compreso tra 0 e 9 questa parte verrà rappresentata con un solo carattere e l'output risulterà essere *dmm.mmmmm* invece che *0dmm.mmmmm*.

Per discriminare queste due situazioni andremo a ricercare il punto all'interno della stringa, verificando se questo sia presente in posizione 3 o 4 (ricordiamo che negli array il conteggio parte da 0).

Questa operazione può essere svolta con la funzione, sviluppata appositamente e presente alla fine del codice:-

```
int strpos(char *str, char *target)
```

che restituisce un intero rappresentante la posizione in cui si è trovato il carattere **target* all'interno della stringa **str*.

Una volta ottenuta quest'informazione estrapoleremo uno o due caratteri relativi alla prima parte della stringa.

Analogamente estrarremo la seconda parte relativa all'informazione *mm.mmmmm*, convertendola in un numero a virgola mobile per eseguire il calcolo finale descritto dalla formula vista in precedenza.

L'output sarà quindi composto da due coordinate relative alla latitudine e longitudine, espresse secondo il formato desiderato (**Listato 2**).

Listato 2 - SEQUE

```
stat=gps.getStat();
if(stat==1)
    Serial.println("NOT FIXED");
else if(stat==0)
    Serial.println("GPS OFF");
else if(stat==2)
    Serial.println("2D FIXED");
else if(stat==3)
    Serial.println("3D FIXED");
delay(5000);
//Get data from GPS
gps.getPar(lon,lat,alt,time,vel);
lon_db=convert(lon);
lat_db=convert(lat);
Serial.println("dd.dddddd data:");
Serial.print("LON: ");
Serial.println(lon_db,4);
Serial.print("LAT: ");
Serial.println(lat_db,4);

}
//Read last message saved.
if(!strcmp(inSerial, "MSG")){
    Serial.println(msg1);
}
else{
    Serial.println(inSerial);
    gsm.SimpleWriteln(inSerial);
}
inSerial[0]='\0';
}

void serialswread(){
    gsm.SimpleRead();
}

int strpos(char *str, char *target) {
    char *res=0;
    res = strstr(str, target);
    if (res == NULL) return false;
    else return res - str;
}

float convert(char* str){
    float mmmmmm;
    float dd;
    int pos;
    pos=strpos(str, ".");
    char dd_str[pos-1];
    dd_str[0]='\0';
    char mmmmmm_str[6];
    mmmmmm_str[0]='\0';

    for (int i=0; i<pos-2; i++){
        dd_str[i]=str[i];
    }
    dd_str[pos-2]='\0';
    dd=atof(dd_str);
    mmmmmm_str[0]=str[pos-2];
    mmmmmm_str[1]=str[pos-1];
    mmmmmm_str[2]=str[pos];
    mmmmmm_str[3]=str[pos+1];
    mmmmmm_str[4]=str[pos+2];

    mmmmmm_str[5]=str[pos+3];
    mmmmmm_str[6]='\0';
    mmmmmm=atof(mmmmmm_str);
    float result;
    result=dd+mmmmmm/1000/60;
    return result;
}
```

Listato 3

```
#include "SIM900.h"
#include <SoftwareSerial.h>

#include "sms.h"
#include "call.h"
#include "gps.h"

CallGSM call;
MSGSMS sms;
GPMSGSM gps;

char message[40];
char number[15];
byte state;
char lon[15];
float lon_db;
float lon_db_2;
char lat[15];
float lat_db;
float lat_db_2;
char alt[15];
char time[20];
char vel[15];
char msg1[5];
char msg2[5];

char stat;
char inSerial[20];
int i=0;
boolean started=false;

void setup()
{
    //Serial connection.
    Serial.begin(9600);
    Serial.println("GSM Shield testing.");
    //Start configuration of shield with baudrate.
    //For http uses is recommended to use 4800 or slower.
    if (gsm.begin(2400)){
        Serial.println("\nstatus=READY");
        gsm.forceON(); //To ensure that SIM908 is not only
in charge mode
        started=true;
    }
    else Serial.println("\nstatus=IDLE");

    if(started){
        //GPS attach
        if (gps.attachGPS())
            Serial.println("status=GPSREADY");
        else Serial.println("status=ERROR");
    }
    delay(20000); //Time for fixing
};

void loop()
{
    state=call.CallStatusWithAuth(number,8,10);
    if(state==CALL_INCOM_VOICE_AUTH)
    {
        Serial.println(value);
        delay(1000);
        call.HangUp();
        value=1;
        Serial.println("Chiamata in corso da numero autorizzato");
        stat=gps.getStat();
        delay(5000);
        gps.getPar(lon,lat,alt,time,vel);
        message[0]='\0';
        strcat(message,"lon: ");
        strcat(message,lon);
        strcat(message," lat: ");
        strcat(message,lat);
        sms.SendSMS(number,message);
    }
    delay(500);
};
```

ESEMPIO 3

Il terzo esempio vuole mostrare come con poche righe di codice sia possibile realizzare un localizzatore con funzioni base di invio della posizione in seguito alla chiamata da parte di un numero, se questo risulta essere autorizzato.

In questo modo per conoscere la posizione del nostro dispositivo sarà sufficiente chiamarlo, dopo aver salvato il numero del chiamante sulla rubrica, e lo shield risponderà con un SMS contenente latitudine e longitudine, inviato dopo aver rifiutato la telefonata. Per non complicare eccessivamente lo sketch non verrà effettuata alcuna conversione del formato delle coordinate.

Si ponga attenzione alla prima parte dello sketch, dove vengono istanziati gli oggetti relativi alle classi. Per la realizzazione di questo esempio è necessario utilizzare contemporaneamente le classi relative alla parte delle chiamate, degli SMS e del GPS. Sarà dunque necessario includere i tre file relativi alle classi e istanziarne i tre oggetti per richiamare le funzioni.

```
#include "sms.h"
#include "call.h"
#include "gps.h"
CallGSM call;
MSGSMS sms;
GPMSGSM gps;
```

Lo sketch controlla periodicamente lo stato delle chiamate e, una volta identificato lo stato di chiamata entrante da parte di un numero autorizzato, esegue la routine associata all'invio della posizione tramite SMS.

Rispetto agli esempi mostrati precedentemente, non troviamo niente di particolare, se non l'utilizzo della funzione `strcat(char *str1, char* str2)` per concatenare due stringhe.

Per la composizione del messaggio da inviare vengono concatenate diverse stringhe al fine di formare il testo da inviare:

```
message[0]='\0';
strcat(message,"lon: ");
strcat(message,lon);
strcat(message," lat: ");
strcat(message,lat);
```

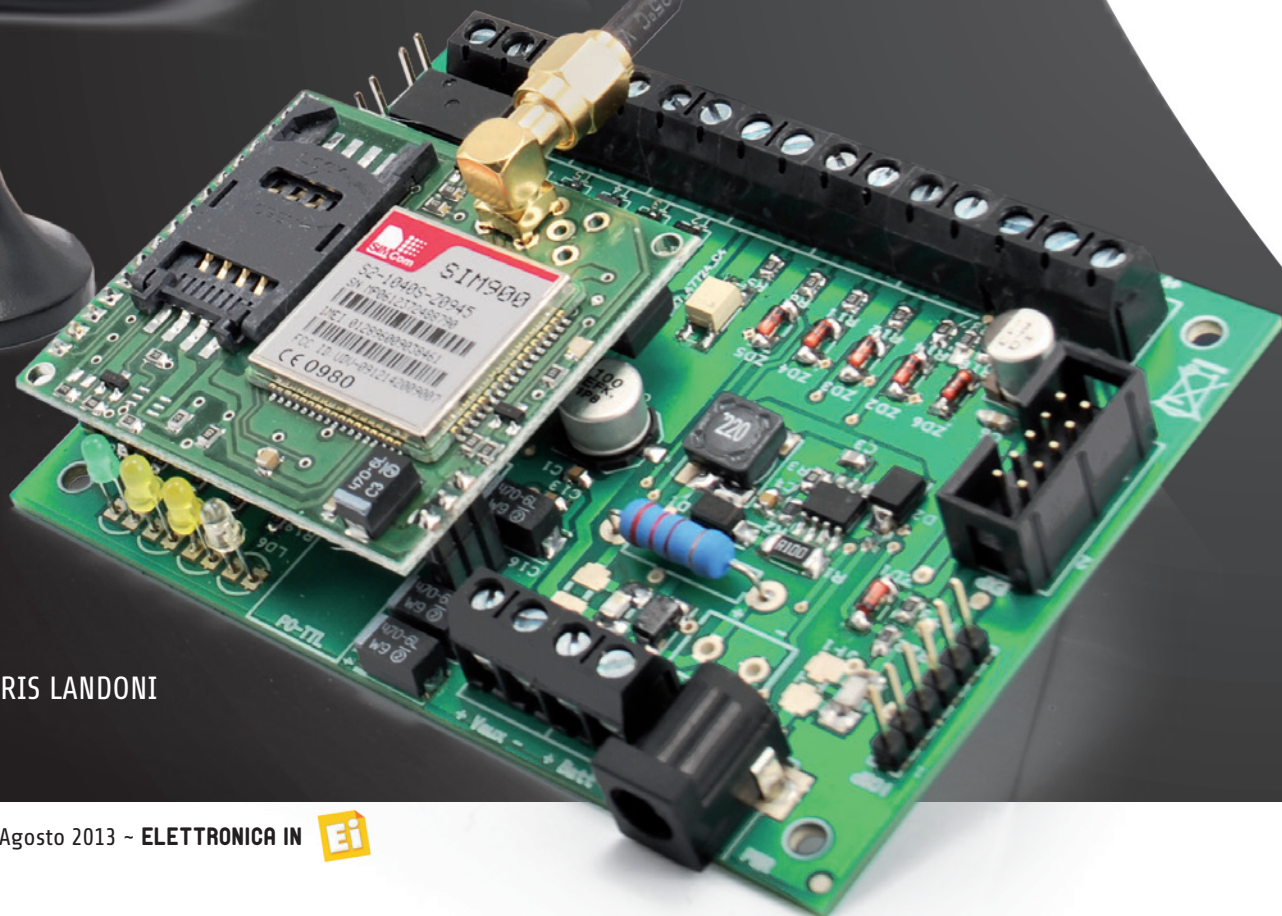
Si osservi come ogni volta sia necessario azzerare la stringa inserendo come primo carattere dell'array il terminatore `'\0'`. (**Listato 3**).

Per quanto riguarda ulteriori approfondimenti relativi alla gestione delle chiamate entranti e l'invio di messaggi, si rimanda il lettore alle tre puntate precedenti dove sono stati trattati questi argomenti. ■



COMBINATORE GSM CON FUNZIONE TTS

di BORIS LANDONI



Vero e proprio sistema d'allarme universale, comunica via SMS o a voce il verificarsi di eventi ai propri ingressi, grazie alla sintesi Text To Speech integrata nel modulo GSM. Si collega facilmente a qualsiasi impianto antifurto commerciale. Prima puntata.

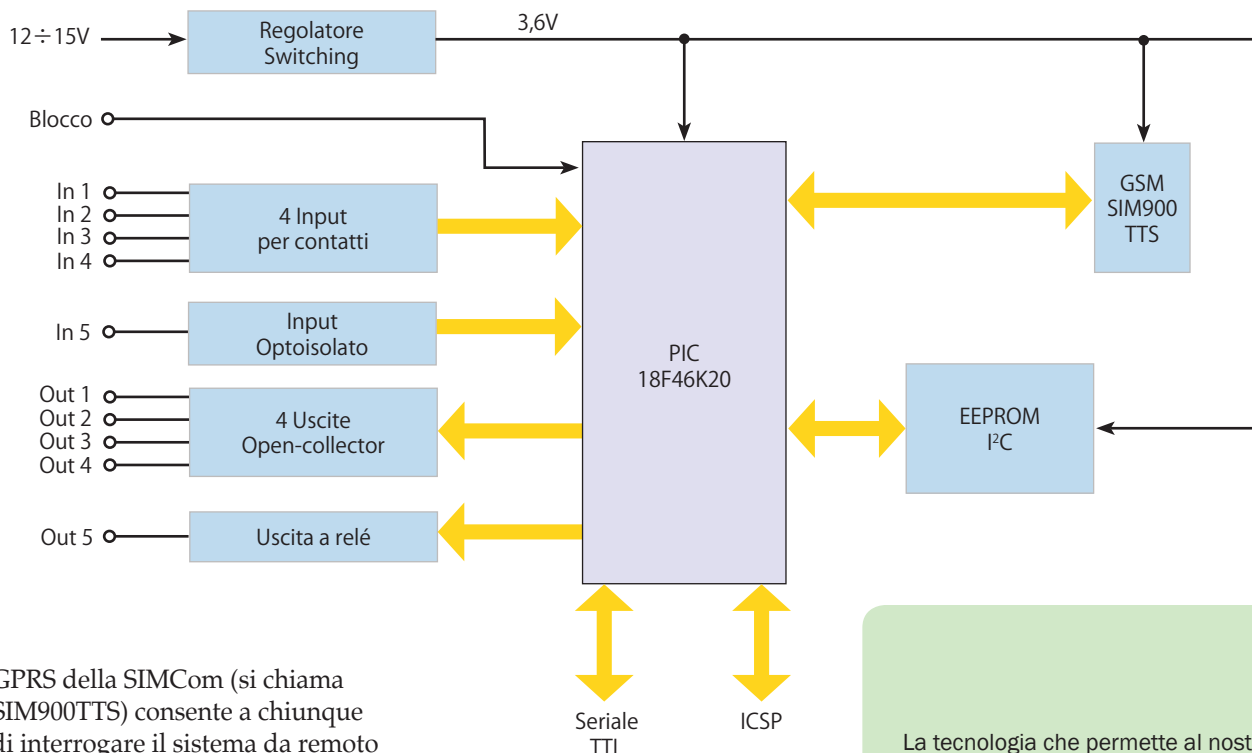
Un requisito dei servizi accessibili al telefono, ma anche delle moderne interfacce-uomo degli apparati elettronici, è la semplicità di accesso alle funzioni, che si traduce nell'intuitività dei comandi e dei messaggi; per quanto siamo tecnici e quindi predisposti -in un certo qual senso- ad interfacciarci con

menu, sintassi dei comandi e via di seguito, è indubbio che anche noi, come la gente comune, mal digeriamo l'interpretazione di simboli e la composizione di complessi comandi, sia pure composti da brevi frasi e simboli. Il successo presso il pubblico delle interfacce vocali usate nei numeri verdi, nei servizi clienti e simili, dove una voce guida ci assiste e attraverso i quali possiamo interagire efficacemente mediante i tasti del telefono, ci ha spinti a rivedere alcuni dei nostri progetti programmabili da remoto, pensati per rispondere alle nostre telefonate e, a seconda della condizione operativa, esporci la situazione o predisporci ad accettare comandi. Questa revisione inizia qui, dal combinatore telefonico che in queste pagine riproponiamo in una versione migliorata (perché dispone di svariati ingressi e uscite, quindi può essere non un semplice attuatore ma una centralina d'allarme autonoma) e dotata di interfaccia vocale. Proprio l'interfaccia vocale, realizzata grazie alla tecnologia TTS (Text To Speech) implementata in un nuovissimo modulo GSM/

ALLARME INCENDIO



Schema a blocchi



GPRS della SIMCom (si chiama SIM900TTS) consente a chiunque di interrogare il sistema da remoto ed ottenerne messaggi vocali sulla condizione operativa, ovvero navigare nelle sue funzioni mediante un menu vocale simile a quello che ascoltiamo quando chiamiamo -per esempio- il servizio clienti del nostro operatore telefonico. Meno comandi SMS, la cui sintassi può mettere in difficoltà chi è poco avvezzo a comporre messaggi di testo, ma semplici istruzioni da telefono e un menu vocale che ci assiste sempre.

Prima di spendere qualche paragrafo sulla tecnologia TTS, facciamo dunque una panoramica sulle funzioni del combinatore.

COME FUNZIONA

Il circuito è in realtà qualcosa di più di un combinatore, in quanto con questo termine viene normalmente chiamato un attuatore di un impianto di allarme, al pari di una sirena o altro avvisatore, solo che invece di generare azioni locali, il dispositivo è in grado, se sollecitato con una condizione elettrica (un livello logico o la chiusura di un interruttore, per esempio), di

effettuare una chiamata telefonica (impegnare la linea e comporre il numero) verso un utente di rete fissa o radiomobile, ed inviare segnalazioni acustiche o riprodurre messaggi vocali in base alla necessità del momento.

Il sistema può essere abbinato a centraline di allarme antifurto e antincendio, ma può lavorare da solo dotato di sensori di intrusione (contatti puliti, sensori P.I.R.) di fumo o di allagamento, così da poter effettuare telefonate di allarme in caso di tentativo di furto, di innesco di un incendio, ma anche di allagamento di uno scantinato, di una taverna o di altra zona della casa, provocato dalla rottura di un tubo o da una pioggia troppo abbondante.

Il rilevamento degli allagamenti è utile per evitare danni ai vicini di casa se si abita in condominio o per proteggere quadri elettrici e apparecchiature elettroniche collocati in basso vicino al pavimento.

Il circuito che proponiamo può

La tecnologia che permette al nostro combinatore telefonico di parlare è una sintesi vocale autentica, che nella sua forma generica permette di pronunciare delle parole che il dispositivo riceve in forma testuale: TTS significa infatti Text To Speech, ossia letteralmente "dal testo al parlato". Quindi un computer o altro dispositivo invia le parole da pronunciare e il dispositivo TTS le trasforma in parlato; l'invio può avvenire sotto forma di caratteri ASCII lungo un canale dati seriale di vario genere o in altra forma.

Nel nostro caso, la tecnologia è stata implementata in un modulo cellulare quad-band della SIMCom, siglato SIM900TTS, che è sostanzialmente l'arcinoto SIM900 da noi usato in svariati progetti di telecontrolli e combinatori telefonici GSM, però dotato di funzionalità TTS integrata.

Il modulo permette sia di pronunciare le parole che riceve sotto forma di testo contenuto in un SMS o in un comando seriale, sia di inviare in linea, durante una chiamata telefonica, l'audio corrispondente.

Nella nostra applicazione il TTS sostituisce quel blocco che sarebbe altrimenti stato costituito da un registratore digitale, come è stato, ad esempio, nel progetto descritto a pagina 39 del

essere triggerato da più segnali e dispone di uscite per produrre, oltre alle chiamate, azioni locali come il comando di sirene, avvisatori acustici, elettroserrature, relé ecc. Dispone infatti di quattro linee di attivazione a livello di tensione provviste di pull-up (quindi attivabili con semplici contatti) che possono essere configurate in modo da rilevare come allarme la chiusura a massa, la presenza di tensione o la transizione da aperta a chiusa e viceversa, più una linea optoisolata il cui funzionamento è identico. Tutti questi cinque

ingressi possono avviare l'esecuzione di chiamate telefoniche, con la riproduzione di messaggi a voce facilmente impostabili mediante comandi da remoto.

C'è poi un ulteriore ingresso che permette di inibire il combinatore durante la sequenza di allarme: serve ad esempio se rientriamo e ci accorgiamo che non c'è motivo di far partire le telefonate di allarme: chiudendo a massa questo ingresso (normalmente si trova a livello logico alto) la sequenza di allarme viene annullata e alla riapertura il dispositivo riparte in condizione

normale, ossia a riposo.

La dotazione di uscite locali consta di quattro transistor configurati in modo open-collector, ciascuno capace di commutare una corrente di 500 mA in circuiti alimentati a corrente continua con tensione non eccedente i 40 volt; ognuno ha sulla base un LED (provvisto dell'opportuna resistenza di limitazione della corrente) che ne segnala l'entrata in saturazione, ovvero indica che l'uscita corrispondente è stata attivata (portata a massa). C'è poi una quinta uscita, a relé, capace di commutare 1 A in circuiti

TTS: dal testo alla parola

fascicolo n° 166 (maggio 2012); in quel caso era necessario memorizzare delle frasi o parole parlando in prossimità di un apposito microfono e ciò andava fatto di presenza, ragion per cui, una volta assegnata una frase ad un evento, questa non poteva essere modificata né localmente da computer via porta seriale, né da remoto tramite un SMS o una telefonata. In questo nuovo combinatore, invece, disponiamo di un vero sintetizzatore vocale integrato nel modulo cellulare, il che ci permette di modificare le parole o frasi assegnate agli eventi, sia da remoto con degli SMS, sia localmente collegando un PC alla porta di comunicazione.

Rimangono l'ingresso audio (microfono o linea) e l'uscita per altoparlante, ma non sono gestiti per ragioni di limitata disponibilità della memoria del modulo, ossia perché il firmware occupa già tutto

lo spazio disponibile. Nel SIM900TTS la funzione Text To Speech può essere utilizzata in due direzioni: per far ascoltare all'interlocutore, durante una conversazione telefonica, l'audio corrispondente a un testo scritto, ovvero per convertire il testo di un SMS in parlato udibile attraverso l'altoparlante collegabile all'uscita del modulo. Nel nostro combinatore usiamo solo la prima modalità, tuttavia non va dimenticata l'importanza della seconda, che ci permette di realizzare dispositivi come i lettori di SMS (ne pubbli-

cheremo uno a breve...) molto utili sia per conoscere il contenuto dei messaggi di testo senza leggerli e operare sul cellulare (funzione comoda per chi deve operare a mani libere) sia per i non vedenti, che così possono ascoltare gli SMS o un testo che il proprio cellulare ha ricevuto o sta scaricando da Internet. Programmi Text To Speech, che riproducono il testo in varie lingue, italiano compreso, sono già disponibili per Personal Computer ed anche nei più sofisticati

smartphone e sono un validissimo ausilio per i non vedenti, per esempio; tra questi, Naturalreaders (www.naturalreaders.com), Ultra Hal TTS Reader (www.zabaware.com/reader/) ed il TTS integrato in Android e pensato per gli smartphone (per questi esiste anche SVOX, <http://svoxmobile-voices.wordpress.com/>). Molti software per TTS sono in grado di correggere alcune caratteristiche della voce per adattarla al proprio gusto: ad esempio timbro, accento, velocità.



La board **GSM**



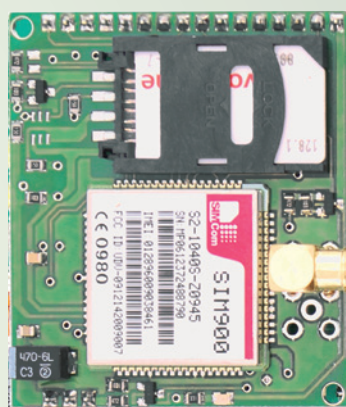
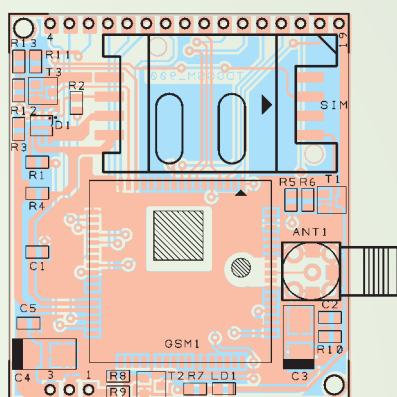
Le uscite si comandano mediante SMS oppure, con una telefonata, direttamente dal menu dell'interfaccia vocale, usando semplicemente la tastiera del telefono. L'intero combinatore si alimenta con un alimentatore da rete e prevede una batteria in tampone che interviene in caso di mancanza dell'alimentazione da rete o di sabotaggio (taglio dei cavi).

IL MODULO SIM900TTS

Vera novità del progetto, il mo-

dulo utilizzato che integra una sintesi vocale propria, implementata mediante l'incremento della memoria interna da 64 a 128 Mb e con la revisione del firmware del microcontrollore integrato che lo governa; infatti, il SIM-900TTS è in realtà un SIM900: dopo l'aggiornamento firmware reso disponibile per consentire al SIM900 di leggere direttamente i bitoni DTMF senza bisogno di utilizzare decoder esterni (leggi 8870...) la SIMCom ha ora realizzato e caricato nei recenti SIM-900TTS una release di firmware in grado di implementare il TTS, ossia di leggere i caratteri costituenti le parole ricevuti lungo

la linea seriale TX/RX, ovvero le parole contenute negli SMS e riprodurli a voce, per ora con un solo timbro e nelle lingue Italiano e Inglese. In questo progetto utilizziamo la sola funzione di riproduzione vocale dei testi dei messaggi di stato e di assistenza, necessari, questi ultimi, a realizzare l'interfaccia vocale che assiste l'utente nelle fasi di configurazione da remoto. Nella modalità di riproduzione delle parole inviate da seriale, il testo corrispondente viene inviato al modulo mediante comandi Hayes estesi, proprietari SIMCom; la modalità che permette di far pronunciare



Elenco Componenti:

C1: 220 nF multistrato (0805)
 C2: 100 nF multistrato (0805)
 C3: 470 μ F 6,3 VL tantalio (CASE-D)
 C4: 470 μ F 6,3 VL tantalio (CASE-D)
 C5: 100 nF multistrato (0805)
 LD1: LED verde (0805)
 R1: 15 ohm (0805)
 R2: 15 ohm (0805)
 R3: 15 ohm (0805)
 R4: 10 kohm (0805)
 R5: 4,7 kohm (0805)
 R6: 10 kohm (0805)
 R7: 330 ohm (0805)
 R8: 10 kohm (0805)

R9: 4,7 kohm (0805)
 R10: 10 kohm (0805)
 R11: 4,7 kohm (0805)
 T1, T2, T3: BC817
 GSM1: Modulo GSM
 SIM900TTS
 SIM: Slot per SIM-CARD

Varie:

- Connettore antenna MMCX 90°
- Strip maschio 16 poli passo 2,54 mm
- Circuito stampato

Il "pezzo forte" del combinatore telefonico è il modulo GSM, che per comodità di realizzazione viene fornito già montato e collaudato su una apposita scheda: schema e piano di montaggio sono riportati qui a lato. La scheda che ne risulta corrisponde, nello Schema elettrico generale, al blocco siglato GSM. Il cellulare è sostanzialmente il SIM900 della SIMCom, ma dotato di un nuovo firmware che alle funzionalità base aggiunge la lettura dei bitoni DTMF e soprattutto la sintesi vocale TTS. Questa modalità viene attivata mediante appositi comandi e permette la riproduzione dei testi introdotti da porta seriale o ricevuti dagli SMS in arrivo, in due modi: tramite l'altoparlante collegabile ai contatti + e - SPK del modulo SIM900 oppure durante una conversazione telefonica. La lettura degli SMS consente di ascoltare i testi dei messaggi, mentre l'invio in linea telefonica permette di comunicare con l'interlocutore inviando da un PC o microprocessore parole in formato binario.

al modulo, le parole giunte da seriale si chiama Play Audio Locally (riproduzione locale) e si attiva con il primo dei comandi del set previsto per il SIM900, ossia **AT+DTAM=<mode>**, dove il parametro *mode* vale 0 per la riproduzione locale o 1 per quella chiamata Play Audio Remotely, ossia la riproduzione in linea telefonica dei testi ricevuti da seriale. La voce usata per la sintesi può essere "aggiustata" impostandone il volume, la tonalità e la velocità di parlato, mediante il comando **AT+CTTSPARAM?**. Il SIM900TTS ha le stesse funzionalità del comune SIM900, ossia è un cellulare GSM/GPRS/EDGE

quad-band, in grado di gestire chiamate a voce ed SMS, nonché leggere una tastiera e riconoscere i bitoni DTMF.

SCHEMA ELETTRICO

Il combinatore è un sistema formato da due schede: una principale (base) e l'altra, più piccola, che si inserisce in essa, contenente il modulo GSM. La scheda base contiene il microcontrollore cui è affidata la gestione dell'intero sistema, oltre al relé per l'attivazione di azioni locali, al fotoaccoppiatore per rilevare le condizioni di allarme all'ingresso IN5, alle morsettiere per leggere gli ingressi per contatti IN1÷IN4, quello di inibizione (BLOCCO), e i

transistor per le uscite open-collector; su questa scheda si monta "a sandwich" il circuito contenente il modulo GSM/GPRS.

Analizziamo per prima la scheda base, la cui alimentazione è a tensione continua, anche non stabilizzata (applicata ai punti + e - PWR) di valore compreso tra 12 e 15 V; viene filtrata a valle del diodo di protezione dall'inversione di polarità (D1) mediante i condensatori C1 e C2. Il fusibile F1 ci permette di proteggere il circuito e la fonte di alimentazione in caso di cortocircuito nel regolatore integrato U1. Il microcontrollore, il modulo cellulare ed il resto del circuito vengono fatti funzionare

a 4 V, ricavati da uno switching basato su tale integrato (si tratta dell'MC34063) utilizzato come regolatore PWM in configurazione buck, la cui tensione di uscita dipende dall'energia immagazzinata in L1; i 4 volt vengono definiti e stabilizzati dalla retroazione operata dal partitore resistivo R2/R3, che retrocede al piedino 5 parte della componente d'uscita. C4 e C5 livellano gli impulsi prodotti dall'U1. La tensione a valle di questi condensatori viene poi ulteriormente filtrata da C7, C8, C13, C14, C15 e C16, situati sulle linee di alimentazione del microcontrollore e della scheda contenente il modulo GSM, che servono a sopperire ai picchi di assorbimento del GSM, evitando la propagazione alla logica degli impulsi di disturbo che ne conseguono.

Per garantire il funzionamento del combinatore anche in caso di mancanza dell'alimentazione principale, abbiamo previsto una batteria in tampone mantenuta in carica mediante la resistenza R4 e il fusibile F2, che la protegge nel caso si verificasse un cortocircuito sulla linea a valle del diodo D1, ovvero un forte sovraccarico del regolatore integrato. Per collegare la

batteria (che deve essere al piombo, da 12 volt) abbiamo previsto una morsettiera a passo 5 mm ed un connettore adatto a quelli montati nelle batterie al piombo-gel per sistemi d'allarme dotate di cavo. Il diodo D4 serve a bypassare la resistenza di limitazione della corrente di carica R4 durante la scarica, così da consentire l'erogazione della corrente che necessita al circuito, senza apprezzabile caduta di tensione (se non quella diretta sul diodo).

Il cuore della scheda base è il microcontrollore che governa l'attività del modulo GSM/GPRS, controlla la condizione logica degli ingressi, comanda le uscite open-collector e quella a relé. Il microcontrollore è un PIC18F46K20-I/PT il quale, una volta inizializzate le linee di I/O, verifica la condizione logica degli ingressi (RB1÷RB5) e quella delle linee RA4, RC5, RD0, RD3, RX, che servono a ricevere dal modulo cellulare le principali segnalazioni; precisamente, RD3 viene usata per rilevare l'arrivo delle chiamate entranti (si interfaccia con la RI del modulo cellulare) mentre RA4 controlla il "LED di campo" del GSM, la cui uscita LED pulsa alla frequenza di 1 Hz

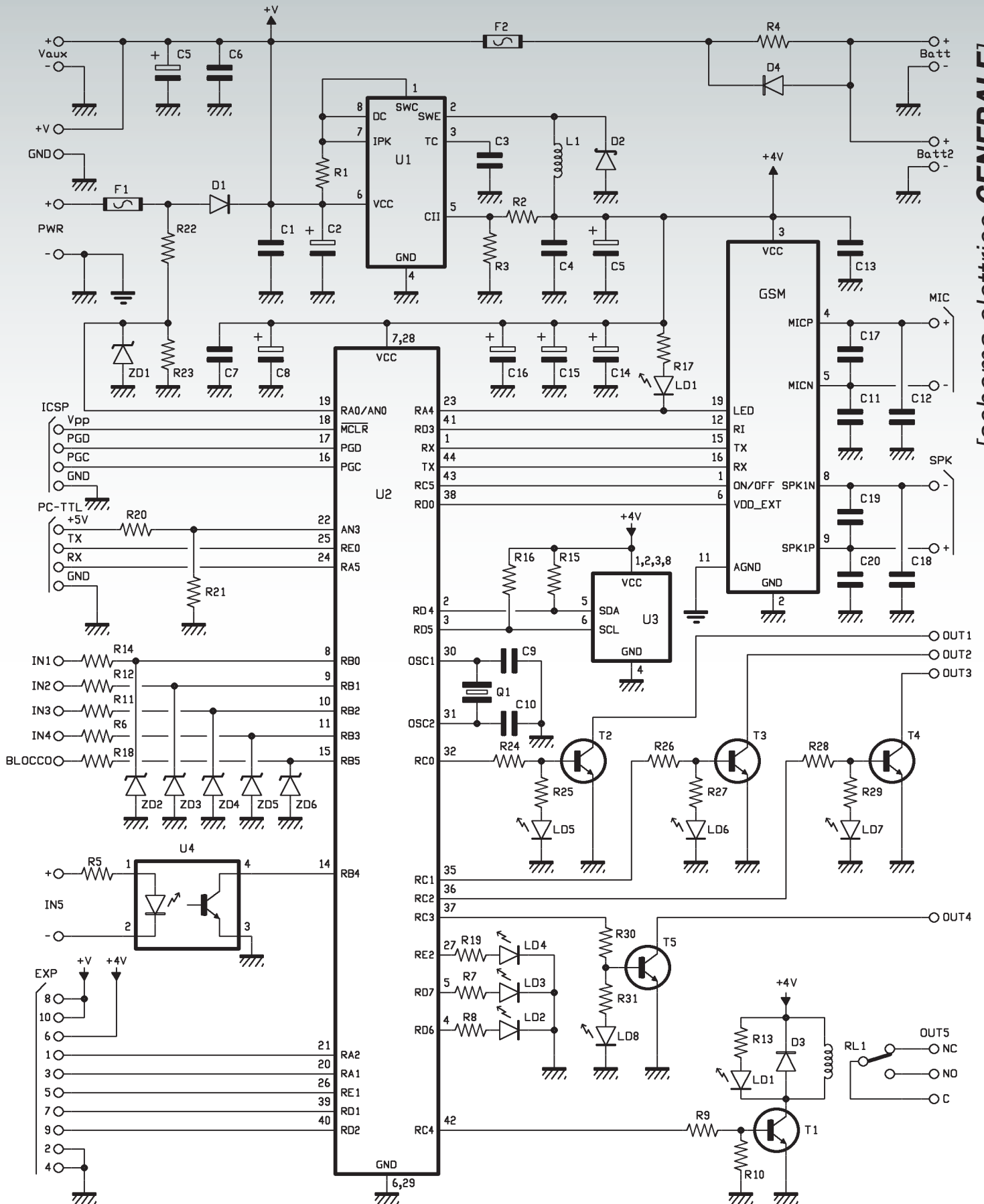
quando il modulo sta cercando la rete telefonica, mentre fornisce un impulso a zero logico della durata di 0,5 s seguito da una pausa di 2 s quando il modulo ha agganciato il segnale.

L'UART interno al PIC, accessibile dai piedini 44 (trasmissione dati) ed 1 (ricezione dati) serve ad impartire i comandi al modulo GSM: mediante il TX, il micro interroga ciclicamente il SIM900 per verificare l'arrivo di SMS, mentre TX ed RX insieme servono, durante le procedure di chiamata, ricezione e trasmissione messaggi, per la comunicazione tra il microcontrollore e il GSM. RD0 comanda l'accensione e lo spegnimento del GSM (tramite un transistor sistemato nella basetta del cellulare) mentre RC5 si occupa di accendere/spegnere e resettare il cellulare.

Vediamo ora gli ingressi, che sono cinque di allarme (ossia provocano l'invio delle telefonate con i messaggi vocali) ed uno di blocco, la cui funzione è già stata descritta: IN1, IN2, IN3 e IN4 vengono letti direttamente dal microcontrollore (mediante gli I/O, rispettivamente, RB0, RB1, RB2, RB3, provvisti di pull-up interno) e lo stesso vale per BLOCCO (il cui stato logico, normalmente alto e imposto dal resistore di pull-up interno al microcontrollore) viene letto dalla linea RB5. Tutti questi piedini sono protetti da un diodo Zener che impedisce l'inversione di polarità ed il superamento dei 5 volt, e da un resistore che protegge altresì il diodo Zener corrispondente quando la tensione eccessiva all'ingresso causerebbe lo scorrimento di una corrente di valore eccessivo.

L'ingresso IN5 è accoppiato mediante un optoisolatore e funziona a livello di tensione: applicando un potenziale da 3 a circa 30 volt in continua, l'input è attivo, mentre si trova a riposo con tensione nulla o inferiore a 3 volt. Il collettore del





Nuova release per il TTS IVONA

Android dispone di un proprio TTS, ma gli sviluppatori di questo magnifico sistema operativo non hanno perso l'occasione per creare nuove App in grado di aggiungere qualcosa alle funzioni esistenti, se non altro per offrire più voci personalizzabili in alternativa a quella un po' troppo "robotica" del TTS originale. Tra queste, una delle più valide è IVONA, basata sulla tecnologia BrightVoice e disponibile in 13 lingue, della quale a fine giugno è stata rilasciata l'ultima versione, scaricabile e utilizzabile gratuitamente per il periodo beta. Anche in

questa, per la versione italiana al momento è disponibile la sola voce maschile (chiamata Giorgio) e si attende quella femminile (Carla). IVONA si installa facilmente in Android al posto del text-to-speech (TTS) e permette di configurare i parametri caratteristici della voce sintetizzata, come timbro, velocità ecc. Le voci possono essere provate sul sito IVONA www.ivona.com. Il TTS può essere utilizzato nel navigatore di Google, nella lettura di testi o e-Book, nella lettura degli SMS e in tutte le applicazioni di terze parti che lo supportano.

transistor d'uscita del fotoaccoppiatore è collegato al pin RB4 del micro, che ne legge lo stato (alto quando all'IN5 non c'è tensione e basso in presenza di tensione). IN5 viene protetto da un resistore che limita la corrente nel LED interno al fotoaccoppiatore. Le resistenze sono da 4,7 kohm, ma si possono modificare scegliendone il valore con la formula:

$$R = (V_{in} - V_{led}) / 0,008$$

in cui V_{in} è la tensione in ingresso, V_{led} è la caduta di tensione del diodo (2 V circa) e 0,008 è la corrente diretta del LED. Passiamo adesso alle uscite, che, come accennato, sono una a relé e quattro open-collector: la prima fa capo alla linea RC4 del micro, che comanda la base del transistor NPN T1; lo stato logico alto manda in saturazione il transistor, determinando lo scorrimento di corrente nella bobina del relé. L'attivazione viene segnalata con un LED alimentato in parallelo alla bobina. Per proteggere la giunzione di collettore dei transistor nei passaggi dalla saturazione all'interdizione,

quando l'induttanza della bobina del relé genera picchi di tensione inversa, abbiamo collegato un diodo in parallelo alla bobina, che sopprime tali impulsi. Del relé viene reso accessibile tutto lo scambio, così da poter gestire circuiti che richiedono un contatto normalmente chiuso o uno normalmente aperto. Quanto alle quattro uscite open-collector, fanno capo ciascuna ad un transistor NPN: OUT1 viene gestita da T2, OUT2 da T3, OUT3 da T4 e OUT4 da T5; ciascuna delle uscite si trova normalmente in stato open (isolata) mentre si attiva, facendo scorrere corrente (max 500 mA) verso massa quando la linea che pilota la base del relativo transistor si porta a livello logico alto. Con le uscite open-collector si possono comandare vari dispositivi (segnalatori ottici e acustici quali cicalini) ma anche linee di ingresso a livello TTL di altri sistemi e centraline; si possono anche pilotare dei relé, con l'accortezza di collegare un diodo al silicio con l'anodo sul collettore e il catodo sul positivo +4V, allo scopo di proteggere la giunzione base-collettore dalle extratensioni inverse che la

bobina genera alla brusca interdizione del transistor stesso.

Il circuito prevede un'interfaccia seriale per la comunicazione con un PC, utile per effettuare la configurazione al banco mediante l'apposito software per ambiente Windows; a questa interfaccia seriale si può applicare un convertitore TTL/RS232 se si desidera collegare la scheda mediante una porta COM, ovvero un converter TTL/USB per interfacciare l'unità via USB. La comunicazione seriale avviene mediante le linee RE0 ed RA5, sfruttate rispettivamente come TX e RX (l'UART del microcontrollore è già impegnato nel dialogo con il SIM900); AN3, assegnata all'A/D converter interno, ci serve per rilevare la presenza dei 5 volt e quindi l'inserzione del connettore nel computer. Questa funzionalità si può sfruttare se si utilizza, come convertitore USB/TTL, l'FT782M della Futura Elettronica (www.futurashop.it). Notate che i dati di configurazione del sistema, indipendentemente dal fatto che vengano introdotti da seriale o da remoto (telefono) non vengono salvati nella EEPROM del

microcontrollore, ma in una memoria esterna di tipo 24FC256-SN; si tratta di una EEPROM CMOS da 256 kbit ad interfaccia I²C-Bus. Per dialogare con essa, il micro usa RD4 ed RD5, inizializzate per funzionare rispettivamente come SDA (linea dati) ed SCL (linea di clock). L'aver spostato i dati di configurazione del telecomando in una memoria esterna ci permette di riservare la EEPROM interna del microcontrollore ai dati di funzionamento, consentendoci di memorizzare più dati di configurazione o di aggiungere funzionalità, che inevitabilmente renderebbero il firmware più pesante.

Per caricare il firmware abbiamo previsto un'interfaccia ICSP, facente capo alle linee /MCLR (piedino 18) PGD (17) e PGC (16) dell'U6 e del connettore ICSP; a quest'ultimo giunge anche il collegamento di massa della scheda. La connessione ICSP da noi prevista è adatta all'uso del programmatore PicKit2. Concludiamo l'analisi dello schema elettrico con il connettore siglato EXP, che riporta le linee di I/O del microcontrollore non utilizzate e destinate a futuri sviluppi; se ve la cavate con il firmware e riuscite a liberare spazio nella Flash di programma del microcontrollore, potete implementare nuove funzionalità quali la visualizzazione su display LCD di parametri di funzionamento e configurazione.

SCHEDA GSM

Il circuito contenente il SIM900TTS (riferirsi al relativo schema elettrico) si monta e si collega a quello base mediante due pin-strip negli appositi connettori single-in-line della scheda base; i connettori sono uno da 3 pin ed uno da 16 pin: il primo trasporta positivo e negativo dell'alimentazione digitale, oltre alla linea di controllo dell'accensione (PWR), mentre il secondo contiene tutti i segnali e le

linee di comunicazione da e verso il modulo GSM, ma anche la massa della sezione analogica dei cellulari (AGND, contatto 11). La linea PWR serve al microcontrollore della scheda base per comandare l'accensione e lo spegnimento del modulo GSM, che comunque rimane costantemente sotto tensione (fornita dalla linea Vcc ai piedini 55, 56, 57) ma viene acceso o spento mediante il livello logico applicato alla sua linea PWR (piedino 1 del SIM900TTS). Quest'ultima linea è provvista di una resistenza di pull-up interna ed è attiva a zero logico, quindi il microcontrollore, per accendere il modulo cellulare, pone a livello logico alto PWR (contatto 1 del pin-strip) e manda in saturazione il transistor T2, il quale pone a livello basso la linea PWR del modulo SIMCom.

Il reset del cellulare avviene facendo inviare dal microcontrollore l'1 logico sulla linea RST, allorché il transistor T3 va in saturazione e pone a livello basso la linea NRESET del SIM900.

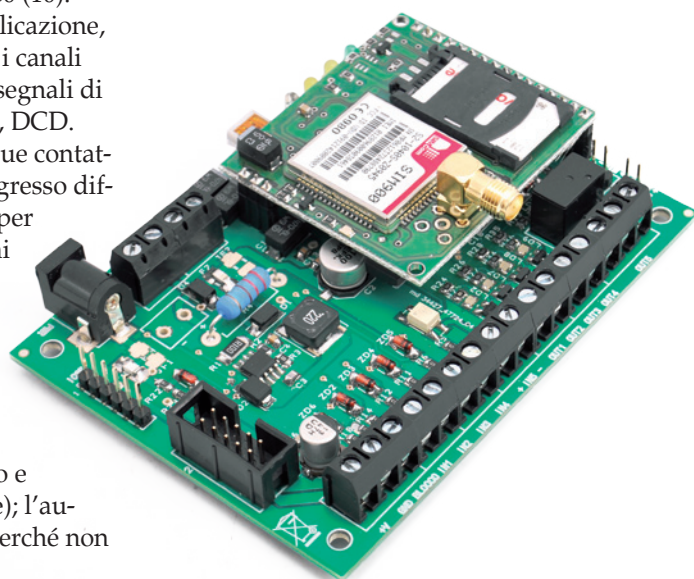
Procediamo con le linee di controllo dell'UART, ossia TXD, RXD, DTR, RTS, CTS, DCD, che vanno all'esterno mediante i contatti, rispettivamente, 15, 16, 17, 14 e 18 del pin-strip; lo stesso dicasi per VRTC (contatto 7) e ADC0 (10). Notate che in questa applicazione, della seriale usiamo solo i canali dati (TXD, RXD) e non i segnali di controllo RTS, CTS, DTR, DCD.

L'audio, facente capo a due contatti per il microfono (ad ingresso differenziale) ed altrettanti per l'altoparlante, transita dai contatti 4, 5, 8, 9, che corrispondono rispettivamente a MIC1P e MIC1N (positivo e negativo del microfono) ed SPK1N e SPK1P (rispettivamente negativo e positivo dell'altoparlante); l'audio in realtà non serve, perché non

dobbiamo registrare alcun messaggio (ci basta definirlo durante la configurazione da PC o da remoto via SMS) e nemmeno ascoltarlo: i contatti + e - MIC e quelli + e - SPK rimangono per futuri sviluppi e comunque per chi intendesse modificare il firmware al fine di gestire microfono e altoparlante. Il segnale di RI (arrivo chiamata) del SIM900TTS esce dal contatto 12 del pin-strip.

L'antenna del modulo GSM si collega direttamente sullo stampato del cellulare: ciò evita disagi dovuti a impedenza che possono manifestarsi portando le connessioni d'antenna dallo stampato del cellulare a quello base. La presa d'antenna è collegata al piedino 60. Il transistor T1 serve per comandare localmente il LED di campo del cellulare; la sua base viene polarizzata dal livello logico presente sul piedino 52 (NETLIGHT) del GSM1 e dal collettore parte la linea che porta al contatto 19 del pin-strip e che fa capo alla linea LED, con la quale il microcontrollore si informa sull'eventuale presenza della rete GSM e sullo stato di connessione del modulo (assenza segnale di rete, presenza rete ecc.).

Concludiamo l'analisi del circuito occupandoci delle linee di gestione



[piano di MONTAGGIO]

Elenco Componenti:

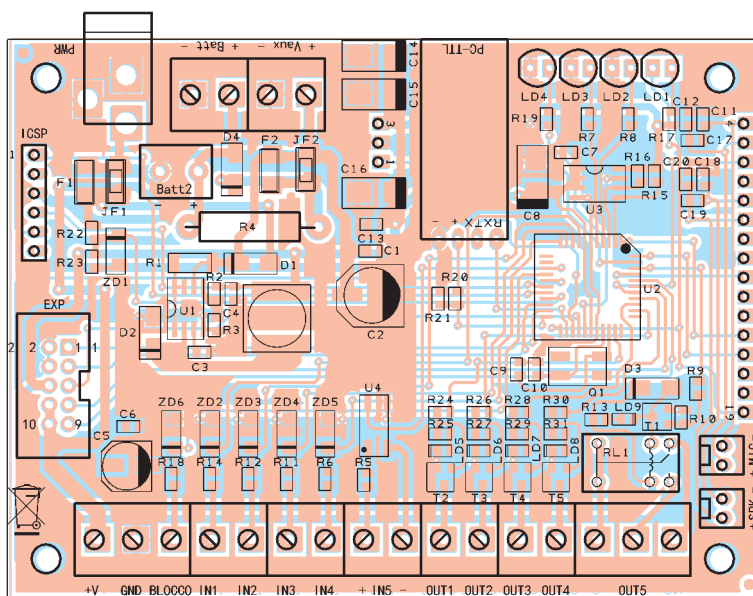
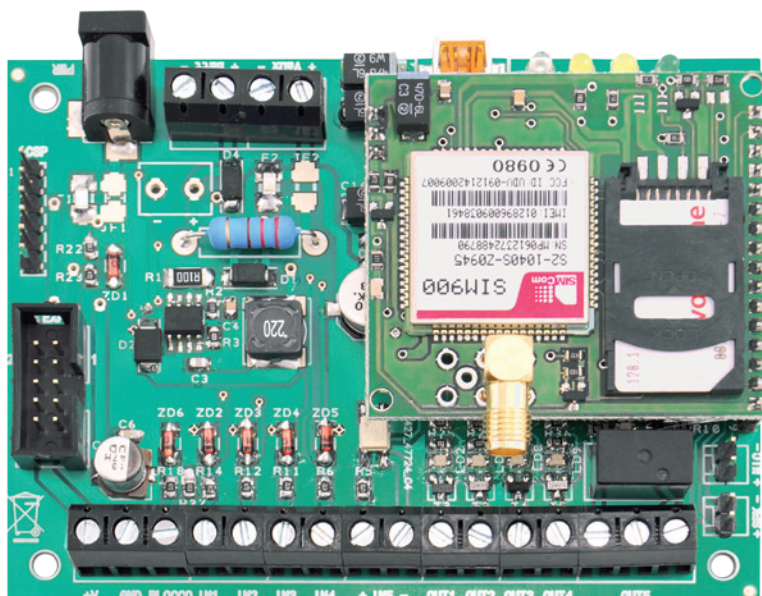
R1: 0,1 ohm 1W (2010)
 R2: 2,2 kohm (0805)
 R3: 1,2 kohm (0805)
 R4: 220 ohm 2W
 R5, R6, R9, R11, R12, R14, R18: 4,7 kohm (0805)
 R7, R8: 330 ohm (0805)
 R10: 10 kohm (0805)
 R13: 1 kohm (0805)
 R15, R16: 1,5 kohm (0805)
 R17, R19: 330 ohm (0805)
 R20, R25, R27, R29, R31: 1 kohm (0805)
 R21: 2,2 kohm (0805)
 R22: 15 kohm (0805)
 R23, R24, R26, R28, R30: 4,7 kohm (0805)
 C1, C4: 100 nF multistrato (0805)
 C2: 100 μ F 25 VL elettrolitico (F)
 C3, C6: 100 pF ceramico (0805)
 C5: 47 μ F 25 VL elettrolitico (D)
 C7: 100 nF multistrato (0805)
 C8: 470 μ F 6,3 VL tantalio (D)
 C9: non montato
 C10: non montato
 C11, C12: 47 pF ceramico (0805)
 C13: 100 nF multistrato (0805)
 C14÷C17: 470 μ F 6,3 VL tantalio (CASE-D)
 C17÷C20: 47 pF ceramico (0805)
 Q1: non montato
 U1: MC34063AD
 U2: PIC18F46K20-I/PT (MF1081)
 U3: 24FC256-SN
 U4: TLP181
 D1, D3, D4: GF1M-E3
 D2: MBRS140TRPBF
 ZD1÷ZD6: BZT55C3V6-GS08
 T1÷T5: BC817
 LD1: LED 3 mm verde
 LD2, LD3: LED 3 mm giallo
 LD4: LED 3 mm rosso
 LD5÷LD9: LED rosso (0805)
 L1: Bobina 22 μ H
 RL1: Relè 5V miniatura 1 scambio
 F1, F2: Fusibile ritardato 63V/2,5A (1206)
 GSM: Modulo GSM (TDG GSM_900TTS)

Varie:

- Morsettiera 2 poli (7 pz.)
- Morsettiera 3 poli (2 pz.)
- Plug alimentazione
- Strip maschio 6 poli

- Strip femmina 4 poli 90°
- Strip femmina 3 poli
- Strip femmina 16 poli
- Connettore POD 10 poli
- Connettore board-to-wire Maschio 2 poli (2 pz.)

- Connettore board-to-wire Femmina 2 poli (2 pz.)
- Microfono (opzionale)
- Altoparlante (opzionale)
- Circuito stampato



della SIM, che si interfaccia tramite il bus formato da SIM_CLK (clock) SIM_RST (reset) e SIM_DATA (canale dati); la linea SIM_VDD viene usata per accendere e spegnere la SIM e viene gestita dal modulo

GSM. Note la presenza del quadruplo Zener D1, i cui diodi proteggono la SIM-Card da eventuali picchi di tensione che possono arrivare dalla linea di alimentazione fino alle linee del bus di comunica-

zione tra SIM e cellulare; tuttavia D1 può non essere montato.

REALIZZAZIONE PRATICA

Per il combinatore occorre preparare i due circuiti stampati, che si

possono ottenere a partire dalle rispettive tracce lato rame (scaricabili da www.elettronica.in.it); sono entrambi a doppia ramatura. Visto che entrambi i circuiti sono formati in buona parte da componentistica SMD, la realizzazione richiede un minimo di esperienza e una certa attrezzatura base, la quale consiste semplicemente in un saldatore o stazione saldatrice a punta finissima, da 20 W di potenza, oltre a del flussante, a una pinzetta e del filo di lega saldatrice da 0,5 mm di diametro. Non guasta avere a disposizione anche una lente d'ingrandimento, per ispezionare le saldature dei componenti più piccoli.

Per facilitarvi il compito, la scheda contenente il modulo cellulare SIM900TTS è disponibile già montata presso Futura Elettronica (www.futurashop.it); il codice del prodotto è TDG GSM_900TTS): basta inserirlo nella scheda base ed è subito funzionante.

Per accogliere la scheda GSM, la piastra base necessita di due semplici connettori S.I.L. da 2,54 mm (si ottengono da strip femmina a passo 2,54 mm), mentre il circuito del GSM ha bisogno di semplici pin-strip.

L'antenna del modulo si collega direttamente su di esso, mediante un cavetto terminante con un apposito connettore SMA adatto a quello del SIM900TTS.

Per alimentare il telecomando occorre un alimentatore o batteria capace di erogare una tensione continua, anche stabilizzata, da 12 a 15 volt e una corrente dell'ordine dei 500 mA.

UTILIZZO DEL SISTEMA

Il combinatore svolge essenzialmente due funzioni:

- 1) avverte con messaggi vocali del verificarsi di una certa condizione elettrica agli ingressi;
- 2) comanda a distanza utilizzatori elettrici o gestibili elettricamente.

Per poter essere impiegato, va configurato in modo da identificare le condizioni degli ingressi, comandare le uscite ed impostare i numeri di telefono cui destinare le chiamate in caso di allarme; a tale riguardo va detto che questi numeri possono essere un massimo di 8 e che il primo va inserito mediante la procedura di Easy Setup, che si avvia accendendo il circuito dopo averlo montato e programmato con l'apposito firmware (da caricare via ICSP) e facendo una chiamata al numero della SIM montata nel modulo TDG GSM_900 entro tre minuti. Trascorso questo tempo invano, ovvero se entro tale intervallo giunge una chiamata, il sistema si porta nel normale funzionamento; nel secondo caso avviene la memorizzazione del numero nella prima posizione della lista. Notate che il circuito non risponde alla chiamata, ma la rifiuta, così da farvi risparmiare il costo dello scatto. Chiaramente l'Easy Setup si avvia solo se in memoria non c'è alcun numero telefonico, altrimenti il circuito si porta subito nel normale funzionamento, nel senso che se c'è già un numero nella lista, il circuito risponde con il menu vocale che elenca le possibili scelte.

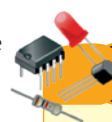
L'Easy Setup si può avviare anche dopo un reset totale, ordinato con l'apposito comando da SMS.

Per la configurazione e il controllo a distanza sono previsti due tipi di comando: uno tramite SMS, secondo la sintassi esposta nella prossima puntata, ed un secondo DTMF, usando la tastiera del telefono, dopo essere entrati nell'apposito menu; da SMS è possibile fare tutte le impostazioni e le configurazioni, mentre da DTMF si può solo controllare le uscite, leggere lo stato degli

ingressi, impostare l'attività di questi e fare alcune programmazioni (lista numeri abilitati al controllo, n° tentativi di chiamata ecc.).

Da SMS si inviano tutti i comandi che permettono di definire l'attività di ingressi e uscite, nonché quelli che permettono di ridefinire le parole o frasi che il sistema fa ascoltare durante la comunicazione telefonica nelle situazioni di allarme e nel comando delle uscite, ovvero nella lettura degli ingressi.

Nella prossima puntata spiegheremo la configurazione completa e tutti i comandi. Quanto alla configurazione da PC, può essere fatta scaricando l'apposito software dal link web www.futurashop.it/image/Download/Configuratore%20TDG.zip e installandolo, quindi connettendo il circuito alla USB del computer mediante il converter FT782M.



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Il combinatore è disponibile già montato e collaudato (cod. TDG145) al prezzo di 112,00 Euro; il dispositivo comprende le due board e l'antenna completa di cavo; non sono inclusi batteria, alimentatore, contenitore e SIM. La board GSM (codice TDG GSM_900TTS) è anche disponibile separatamente al prezzo di 56,00 Euro così come il modulo GSM codice SIM900TTS che costa 36,00 Euro.

Il converter USB/seriale FT782M che consente di utilizzare un PC per la programmazione del combinatore costa 15,00 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
<http://www.futurashop.it>

COME FARE UN CONTENITORE DA STAMPARE IN 3D

La stampante 3Drag si adatta molto bene alla realizzazione di oggetti utili e noi abbiamo realizzato il contenitore per il circuito di stampa autonoma.



di SIMONE MAJOCCHI

Il numero scorso vi abbiamo presentato un sistema da collegare al controller di 3Drag per gestire la stampa dei file in G-Code direttamente da scheda SD grazie a un display LCD e a un encoder rotativo con pulsante. La basetta è stata realizzata già con l'idea di racchiuderla in un

contenitore, ma evidentemente non esiste in commercio un contenitore che la possa ospitare senza dover effettuare una serie di forature e di lavorazioni meccaniche. Tutto questo era vero fino a qualche anno fa e molti hobbysti di elettronica sanno bene che "il contenitore" è spesso più laborioso di tutta l'intera parte elettronica, a meno che non si parta da qualche modello standard già in fase di layout della basetta. Oggi siamo invece liberi dai vincoli dei contenitori per i nostri progetti perché abbiamo l'asso nella

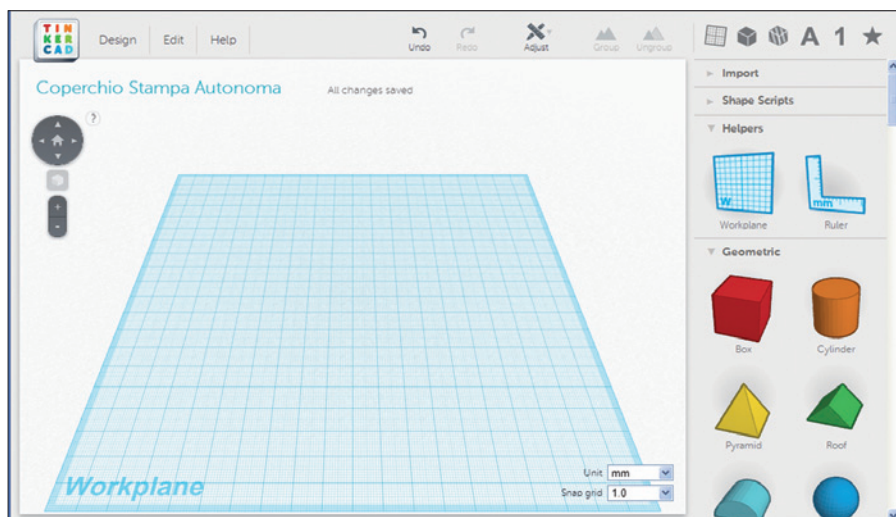


Fig. 1

manica, ovvero la stampante 3D. Come riesce a stampare vasi e fischietti, evidentemente è in grado di stampare anche contenitori e scatole, ma come si può facilmente immaginare, serve il modello fatto su misura per ciascun progetto. Siti come *Thingiverse* sono pieni di modelli, parti e pezzi, ma è quasi certo che non si trovi quello che serve esattamente al nostro scopo. La soluzione passa quindi per la progettazione fatta *ad hoc* e questo potrebbe sembrare un passo troppo complicato per una semplice scatola, ma con l'aiuto degli strumenti giusti anche la progettazione di oggetti relativamente complessi è alla portata di tutti, neofiti inclusi.

I FERRI DEL MESTIERE

Per affrontare il tema della progettazione sono necessari alcuni strumenti e nello specifico dovremmo poter utilizzare un calibro, un metro, carta e penna e, ovviamente, un'applicazione di modellazione solida.

Carta e penna vi permettono di annotare le misure, prese con metro e calibro, e di fare un primo schizzo di quello che poi realizzerete con l'applicazione CAD. Scarabocchiare i dettagli e le soluzioni tecniche è un passaggio molto utile per mettere a fuoco le strutture geometriche che poi dovrete costruire con l'applicazione su computer. Un elemento aggiuntivo da tener

presente nell'immaginare il pezzo in fase di progettazione riguarda le capacità di stampa della vostra 3D: sottosquadra e parti inclinate si possono realizzare ricorrendo ai supporti, ma questi rallentano molto la stampa e sono poi un po' noiosi da eliminare. L'ideale è riuscire a progettare senza creare elementi critici e più avanti vedremo di darvi qualche suggerimento in proposito.

QUALE APPLICAZIONE?

Con la diffusione di un numero crescente di stampanti 3D, il tema della progettazione e della modellazione solida è diventato di grande attualità: sempre più utenti non professionali stanno cercando soluzioni adatte alla realizzazione di file il cui unico scopo è la stampa sulla loro stampante. Non servono quindi librerie di parti o approcci adatti ad ambiti diversi quali il design, l'architettura e la meccanica: serve solo una soluzione che ragioni in unità di misura quali i millimetri e i pollici (per chi usa il sistema di misura imperiale), e permetta di avere delle primitive da assemblare con la logica della somma e della sottrazione, sia dotata di strumenti per allineamento e centrature, ma soprattutto non abbia una palette di strumenti di difficile interpretazione. Dopo aver provato numerose soluzioni, abbiamo constatato che –

pur con limiti e difetti – la web app Tinkercad poteva essere la soluzione migliore per chi non ha mai affrontato il tema della costruzione di modelli tridimensionali.

Essendo una web app, funziona all'interno di un browser che supporta la tecnologia WebGL, come Google Chrome o Mozilla Firefox, e non richiede il download di alcunché. Il suo funzionamento è quindi squisitamente online, con memorizzazione diretta sui server di Tinkercad di tutti i vostri lavori. Una volta creato il vostro account, avrete quindi sempre a disposizione sia i lavori, sia il programma, indipendentemente dal computer che utilizzerete, purché sia collegato a Internet e sia anche dotato di un browser compatibile.

La validità di questa soluzione è anche testimoniata dal fatto che di fronte a una possibile chiusura, si è addirittura attivata Autodesk Inc. – azienda storica nel mondo del CAD su computer – per salvare il servizio e dargli un nuovo impulso.

LA NOSTRA SCATOLA

Innanzitutto vi possiamo dire che l'intero contenitore con le staffe di aggancio a 3Drag, la manopola e il pulsante di reset sono stati interamente realizzati con Tinkercad, senza trucchi e senza scorciatoie. Poi, per questioni di spazio, ci dedicheremo solo alla realizzazione di una parte del contenitore quale tutorial per Tinkercad; infine vi diciamo che i file STL sono disponibili per il download sul nostro sito e che quindi – senza fatica – potrete comunque realizzare anche voi il contenitore dell'interfaccia di stampa autonoma per stampanti 3Drag e RepRap con controller Sanguinololu.

Fatta questa premessa, possiamo avventurarci nella realizzazione del contenitore, ricordando che deve avere una finestra per lo schermo LCD, un foro per la ma-

nopola dell'encoder e una soluzione che ci permetta di premere sul tastino di reset collocato sulla basetta.

Quest'ultima ha un ingombro di 160 x 62 millimetri e quando si monta il display, si arriva a circa 35 mm di altezza dalla parte inferiore della basetta. Con queste misure di base possiamo iniziare a ragionare su quello che potrebbe essere la scatola come ingombri: aggiungiamo 2 millimetri di aria attorno allo stampato e 2 millimetri di spessore per le pareti, quindi consideriamo circa 10 millimetri per l'altezza delle viti che fisseranno la basetta e due millimetri di spessore del fondo. Lo schermo LCD è da 45 x 90 millimetri, mentre il cavo di collegamento e la scheda SD hanno bisogno di una feritoia da almeno 25 millimetri. Il foro per la manopola dell'encoder può essere da 15 millimetri per ospitare una manopola con un perno da 10 millimetri. Il pulsante di reset è posizionato a circa 30 millimetri dal livello dello schermo LCD e questo andrà tenuto presente nel realizzare il suo sistema di attivazione.

Tutta questa serie di misure non vi deve spaventare, ma è indispensabile averle a portata di mano per capire come muoversi in fase di

progettazione. Mancano ancora dei valori che sono le distanze degli elementi dal bordo dello stampato, ma ce ne occuperemo al momento giusto.

L'INTERFACCIA DI TINKERCAD

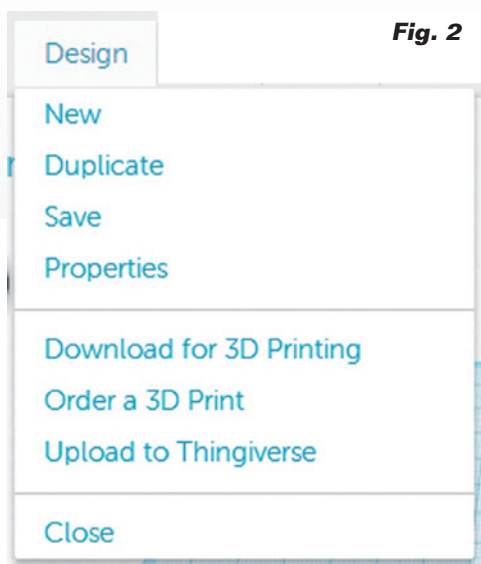
Prima di partire con il nostro progetto è necessario familiarizzare con l'interfaccia della web app Tinkercad: aprite il browser Google Chrome o Mozilla Firefox (se non li utilizzate, dovete sceglierne uno e scaricarlo prima di procedere) e inserite l'indirizzo www.tinkercad.com per aprire la pagina principale; create un account gratuito e accedete alla vostra pagina personale. Tinkercad vi offre una serie di lezioni per imparare ad usare le caratteristiche principali dell'applicazione e vi consigliamo caldamente di farlo per acquisire quella dimestichezza con i vari elementi di base che vi vengono messi a disposizione.

In Fig. 1 trovate l'interfaccia così come si presenta quando avviate un nuovo progetto premendo il pulsante "Create new design" dalla vostra pagina personale. Innanzitutto trovate in alto a sinistra il marchio Tinkercad che è anche la scorciatoia per tornare alla vostra pagina personale. I tre menu "Design", "Edit" ed "Help" sono invece delle tendine dalle quali accedere alle funzioni relative alla gestione del progetto o alla modifica dello stesso.

La prima cosa che vale la pena sottolineare è che l'applicazione si preoccupa di salvare sempre il vostro lavoro senza che voi dobbiate fare nulla. Periodicamente e quando fate alcune operazioni, si attiva il salvataggio e il vostro lavoro viene messo al sicuro. Un altro elemento importante riguarda il pulsante di "Undo" che vi permette di tornare indietro dopo una manovra errata o di cui non siete sicuri.



Fig. 2



Quando fate uno o più *Undo*, viene anche attivato il pulsante accanto *Redo* che ripete le operazioni in sequenza fino all'ultima cosa che avete fatto.

Ultimo elemento che ci preme sottolineare è la natura a oggetti del programma: via via che costruite qualcosa, Tinkercad mantiene sia la storia, sia la struttura di primitive utilizzata; è quindi sempre possibile smontare le cose attraverso l'ungroup degli elementi che avete assemblato con una o più operazioni di raggruppamento (*group*). Dal menu "Design", riprodotto in Fig. 2, si notano tre gruppi di funzioni; nel primo gruppo troviamo

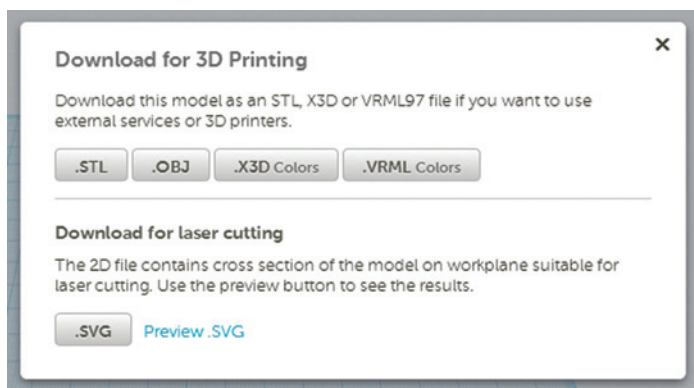


Fig. 3



Fig. 4

New che abbandona il progetto corrente e ne crea uno nuovo, *Duplicate* che crea una copia del progetto corrente e ci ripresenta la copia per l'editing, *Save* che forza il salvataggio del progetto e *Properties* che permette di accedere al nome del progetto assieme ai parametri di pubblicazione e diritti d'autore. Se in *Properties* impostate "Private", il lavoro è visibile solo a voi, mentre se scegliete "Public" il progetto diventa visibile a tutti gli altri utenti che possono a quel punto scaricarlo, copiarlo e modificarne la loro copia.

Il secondo gruppo contiene le funzioni per la gestione del modello, ovvero per scaricare il file in una serie di formati adatti alle successive elaborazioni e stampe, come visibile in Fig. 3, oppure per ordinare una stampa fatta con le macchine e i materiali professionali offerti dai partner di stampa online del sito (Fig. 4), o per caricare il progetto su Thingiverse.

Il terzo gruppo semplicemente permette di chiudere l'editor dell'oggetto e di tornare alla vostra pagina che offre le miniature di tutti i vostri lavori (Fig. 5); dalle miniature è possibile sia avere un'anteprima statica o tridimensionale interattiva, sia tornare alla modifica dell'oggetto.

Se state consultando gli oggetti pubblicati da altri, avete a disposizione le stesse funzioni, ad eccezione della modifica che è possibile solo su una copia dell'originale. Osservando il menu a icone, di cui

abbiamo già spiegato *Undo* e *Redo*, troviamo *Adjust*, *Group* e *Ungroup*. Anche in questo caso le icone diventano nere (accessibili) solo quando siete in una situazione che lo prevede. Da *Adjust* avete accesso ad *Align* e a *Mirror*: due funzioni che portano a delle visualizzazioni di elementi aggiuntivi attorno all'oggetto selezionato proprio per svolgere le funzioni scelte (Fig. 5). Proseguendo nell'esplorazione dell'interfaccia trovate una serie di icone posizionate sopra alla barra laterale della palette di oggetti e funzioni e queste non sono che delle scorciatoie per scorrere verticalmente la palette fino all'area rilevante.

PRIMI PASSI DI EDITING

Quando trovate la forma che vi interessa nella palette a destra, clic-

cateci sopra e trascinatela sul piano di lavoro. Questa sarà automaticamente appoggiata al piano, avrà dimensioni di 20 x 20 millimetri e un'altezza dipendente dal tipo di forma. Le lettere e i numeri sono invece più piccoli.

In Fig. 6 vedete il cubo, così come viene creato di default e su cui abbiamo fatto un clic di selezione per fare apparire le maniglie (quadrantini neri, bianchi con bordo nero o freccia nera); con le maniglie visibili ci siamo quindi portati con il mouse su quella in basso a destra che, per confermare la funzione, è diventata rossa e ha fatto apparire le quote. Ogni volta che siete in prossimità di una maniglia, questa diventa rossa e fa apparire la misura su cui poi andrete ad agire se cliccate con il sinistro e trascinate. Ispezionate tutte le maniglie

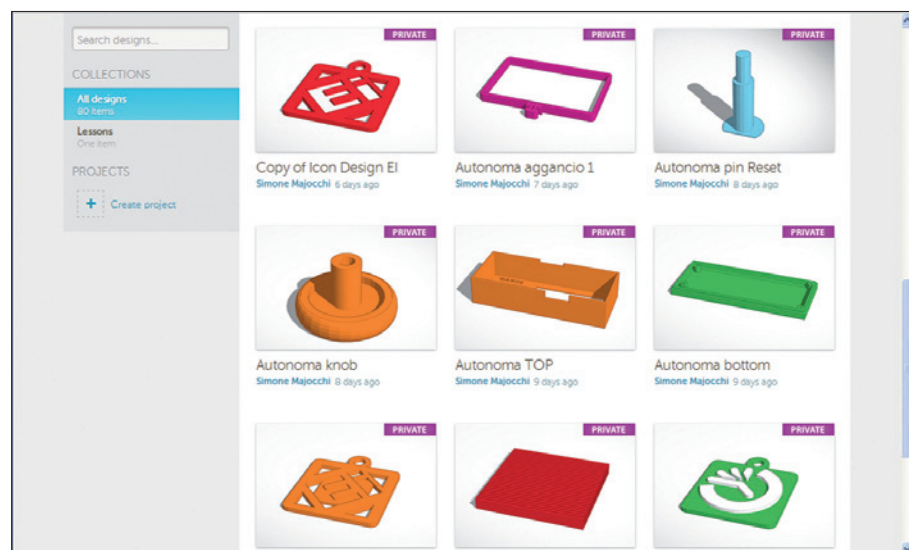


Fig. 5

e scoprirete che quelle più piccole, nere, in centro ai lati, agiscono solo su una dimensione, quelle bianche alla base su due dimensioni e quelle in alto su una. Le frecce invece attivano le rotazioni sui vari assi, come da goniometro visualizzato. Se non cliccate su nessuna maniglia, ma solo sull'oggetto, lo spostate sul piano; anche in questo caso lo spostamento viene esplicitato con la visualizzazione delle quote sui due assi ed il valore dello spostamento rispetto al punto di partenza.

È importante notare come la carta millimetrata del piano di lavoro non sia puramente di bellezza, ma rappresenti sia una risorsa per avere sott'occhio le misure, sia la griglia a cui si agganciano gli oggetti e i lati durante le manovre di spostamento e modifica. In Fig. 7 abbiamo evidenziato i due controlli grazie ai quali è possibile cambiare l'unità di misura e la dimensione della griglia a cui si agganciano spostamenti e modifiche. Imparare a gestire l'unità di misura permette di ottenere nei lavori la precisione richiesta.

Nella sua semplicità, Tinkercad offre delle funzioni "nascoste" che permettono di semplificare alcune

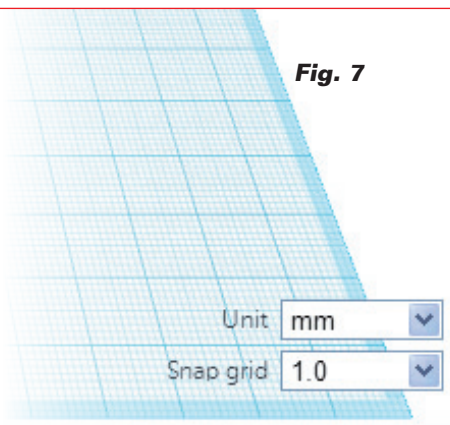


Fig. 7

operazioni: sul lato sinistro sotto al nome dell'oggetto, si trova un'icona tonda con quattro frecce e una casetta, quindi un quadrato in prospettiva e infine due pulsanti con + e -. Tutti questi elementi servono per manovrare la visualizzazione spostando il punto di vista e l'inclinazione del piano di lavoro, ma nella pratica si tratta di movimenti che si possono eseguire anche dal mouse: fate clic con il tasto destro sulla carta millimetrata e vedrete che spostando il mouse fate esattamente quello che vi permettono di fare le quattro frecce; la rotella attiva lo zoom in entrata o in uscita così come fanno i pulsanti + e -. Quello che vi manca è il pulsante con il cubetto che porta a pieno schermo l'oggetto correntemente selezionato, mentre alle icone manca una funzione che si esegue solo tramite mouse. In Fig. 8 avete il riassunto dell'operatività con il mouse, ricordando che dovete cliccare non su un oggetto, ma sulla carta millimetrata.

Quello che compare in questo frangente è l'utilizzo con il mouse del tasto "shift" (maiuscolo, quello sopra al tasto CTRL per intenderci). L'aggiunta della pressione di shift durante varie operazioni ne cambia il risultato e per chiarezza facciamo l'elenco di quello che succede.

Aggiungendo SHIFT durante:

- Spostamento oggetto sul piano -> si limita lo spostamento a uno dei due assi;
- Spostamento di una maniglia

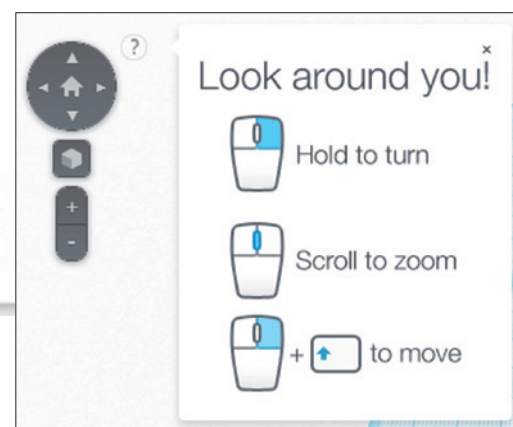


Fig. 8

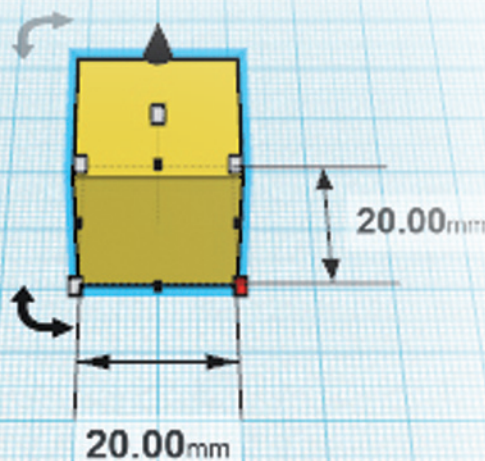
-> la deformazione avviene su tutte e tre le dimensioni, in proporzione;

- Spostamento maniglia per alzare/abbassare rispetto al piano -> nulla.

Se durante una qualsiasi operazione si lascia il tasto shift, l'azione in corso torna ad avere il suo effetto previsto.

Quando usate il ridimensionamento contemporaneo sui tre assi, avrete lo snap sulla direzione del vostro movimento e per il lato selezionato, mentre sugli altri assi avrete un ridimensionamento che porta i lati o le dimensioni a valori proporzionali, ma potenzialmente decimali. Ciò comporta degli effetti pratici che vanno dall'impossibilità di avere lo snap sulla carta millimetrata senza intervenire su ciascuna dimensione per "riagganciarla" alla griglia. Il ridimensionamento proporzionale è però alla base di una serie di operazioni e quindi può essere inevitabile perdere l'aggancio dell'oggetto su x e y, su uno dei due lati. La griglia aggancia comunque dal lato su cui state agendo sulla maniglia e quindi se avete un lato che ha assunto un valore con decimali e volete riportarlo ad uno che si aggancia alla griglia, dovete agire sulle maniglie da un lato e dall'altro perché solo così porterete entrambi gli estremi al passo della griglia.

Fig. 6



ALLINEAMENTI E CENTRATURE

Lavorare con una griglia permette di avere quella precisione richiesta dalla modellazione solida basata su somma e sottrazione di parti, ma quando per questioni varie le misure non sono più adatte allo snap, si può usare lo strumento di allineamento che permette di far combaciare spigoli e superfici, oppure di centrare esattamente solidi su uno o più assi.

Perché la funzione sia attiva è necessario avere selezionato più di un solido: si seleziona il primo solido cliccandoci sopra, poi si preme e si tiene premuto shift e si selezionano gli altri solidi cliccandoci sopra. L'allineamento si attiva dall'icona *Adjust*, accanto a *Redo*, scegliendo poi la voce *Align*. Attorno ai due (o più) solidi

selezionati appaiono una serie di maniglie come in Fig. 9.

Le maniglie a sinistra e davanti sono per gli allineamenti sul piano di lavoro, mentre le tre a destra visivamente sviluppate in verticale servono ad allineare gli oggetti sulla base, al centro verticalmente o verso l'alto. È anche molto utile passare semplicemente su ciascuna maniglia per avere l'anteprima dell'allineamento. Quando gli oggetti sono già allineati su una specifica situazione (ad esempio sono già centrati sulla X), la maniglia corrispondente diventa grigia. Quando due oggetti sono allineati su un lato o sull'altro potete star certi che le facce su quel lato sono perfettamente allineate e questo diventa vitale per fare delle operazioni di sottrazione efficaci.

Fig. 9

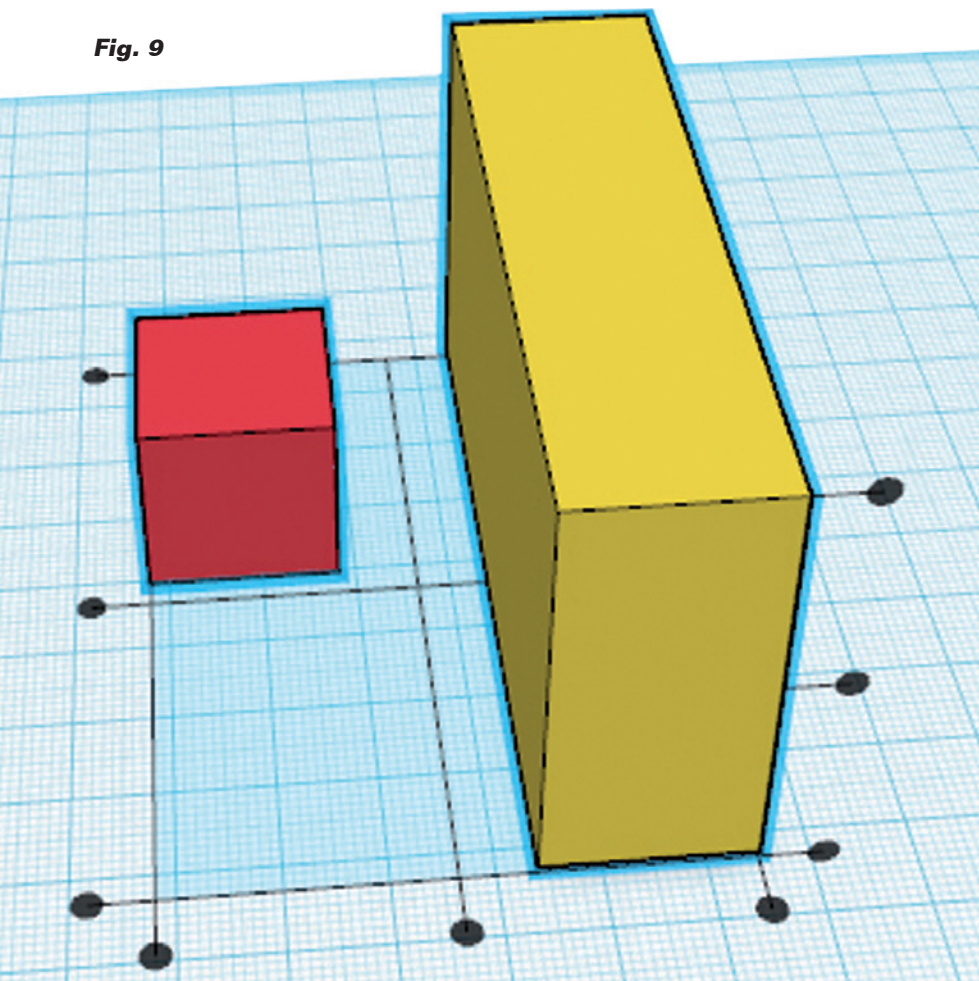


Fig. 10

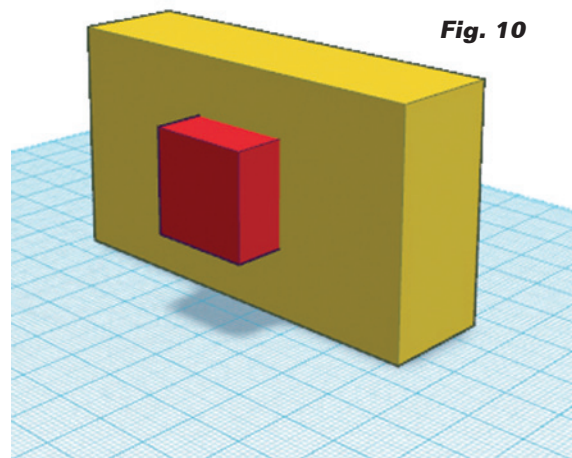
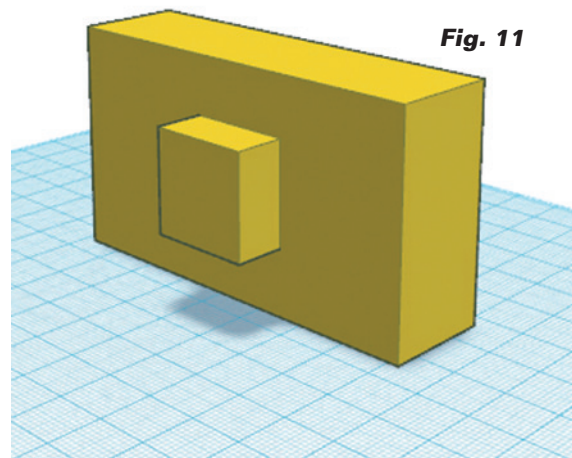


Fig. 11



SOMME E SOTTRAZIONI

L'ultimo pezzo del puzzle delle funzioni di Tinkercad che manca per avere un quadro abbastanza completo di quello che si può fare riguarda da un lato la funzione "group / ungroup" e dall'altra il "colore" degli oggetti.

Quando entrambi gli oggetti da raggruppare hanno un colore (capirete il perché di questa precisazione), questi vengono fusi in un unico oggetto più complesso (Fig. 10 e Fig. 11) che non è solo la

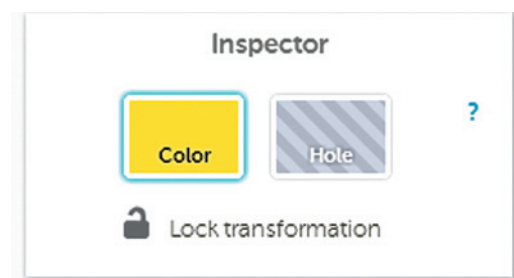


Fig. 12

superficie esterna, ma anche tutta la parte interna.

Per intenderci, se una sfera all'interno è cava e sommiamo un cilindro, il cilindro riempirà parzialmente la sfera cava una volta che questo viene fuso con essa tramite *Group*.

Il colore unico assunto dagli oggetti raggruppati ne conferma la trasformazione in un unico solido. Il pannello "*inspector*" che appare ogni volta che selezionate un oggetto – **Fig. 12** – vi permette di definire il colore partendo da una palette di colori pastello oppure tramite scelta dei valori fondamentali per i colori personalizzati, mentre sul lato destro si trova il colore "*Hole*" che con il suo tratteggio rappresenta il "vuoto".

Se un oggetto è "*hole*", anche se entra in un oggetto solido, non accade nulla, mentre se lo si raggruppa, allora viene sottratto dal pieno, creando appunto un vuoto per l'intero suo volume, dopodiché la parte che non si interseca "sparisce" e non ha più alcun effetto sulle successive operazioni. Questo appare chiaro nelle **Fig. 13** e **Fig. 14**. Con questo sistema di somma e sottrazione di forme, fatto anche in modo progressivo sul medesimo solido, si riescono a costruire con precisione forme anche molto complesse.

Dal punto di vista progettuale, dovete quindi cercare di immaginare come ottenere una specifica forma sommando e sottraendo fra loro solidi, senza dimenticare che un solido ottenuto per sottrazione può a sua volta essere trasformato in "*hole*" per essere sottratto da un altro solido. Il ragionamento può sembrare contorto, ma diventa invece molto chiaro dopo le prime sperimentazioni.

IMPORTAZIONE 2D E 3D

Concludiamo con una funzione aggiunta di recente a Tinkercad: la

possibilità di importare file bi- e tri- dimensionali, rispettivamente di tipo SVG e STL. I file SVG vettoriali bidimensionali sono importati come strutture estruse di qualche millimetro, ma di altezza uniforme e sono utili per creare figure complesse come scritte, marchi e loghi da inserire su altri oggetti; il loro spessore può essere variato con le solite maniglie al pari di tutti gli altri oggetti. In fase di importazione potrebbero esserci dei problemi di compatibilità se non si usa il formato SVG puro. Per prova abbiamo preso un SVG da Internet (http://upload.wikimedia.org/wikipedia/commons/e/e3/Google_Nexus.svg) e lo abbiamo importato, ottenendo il risultato in **Fig. 15**. L'importazione di STL è invece interessante per ritoccare e modificare, con il sistema dell'aggiunta/sottrazione, oggetti realizzati con altre applicazioni o scaricati dal web. Il modello STL importato viene gestito come un oggetto complesso non suddivisibile nelle sue primitive, ma è possibile aggiungere e sottrarre parti come se fosse un solido nativo Tinkercad. Un esempio di importazione STL è in **Fig. 16**.

E ADESSO SI COSTRUISCE!

Ora che avete acquisito tutti i

fondamenti di Tinkercad, potete avventurarvi assieme a noi nella realizzazione del coperchio della nostra scatola per proteggere il circuito della stampa autonoma. Per non occupare troppe pagine spiegheremo solo la costruzione

Fig. 15

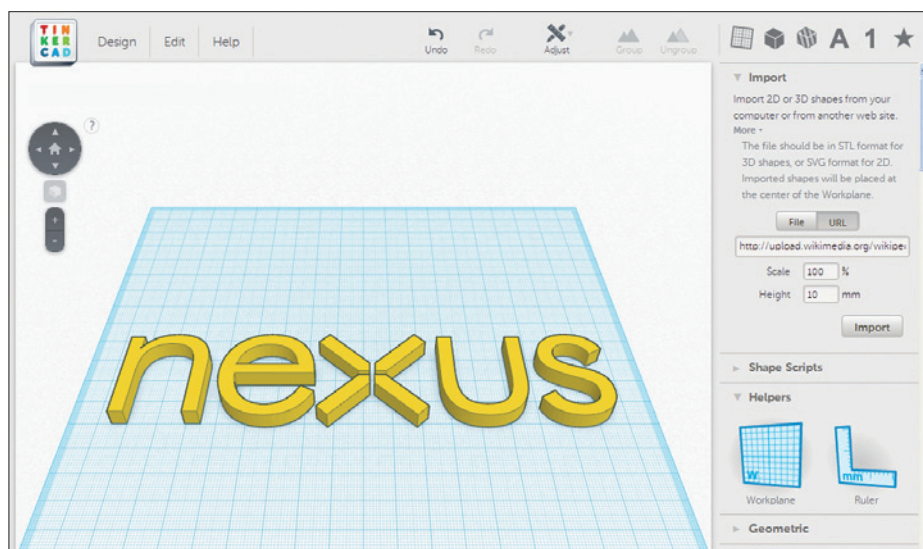


Fig. 13

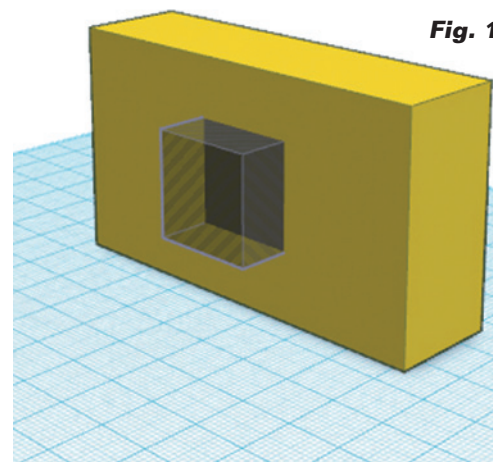


Fig. 14

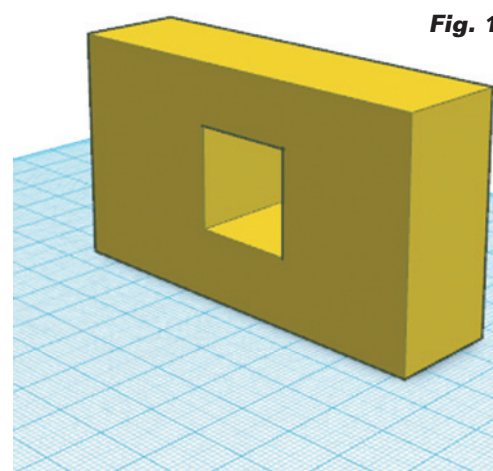
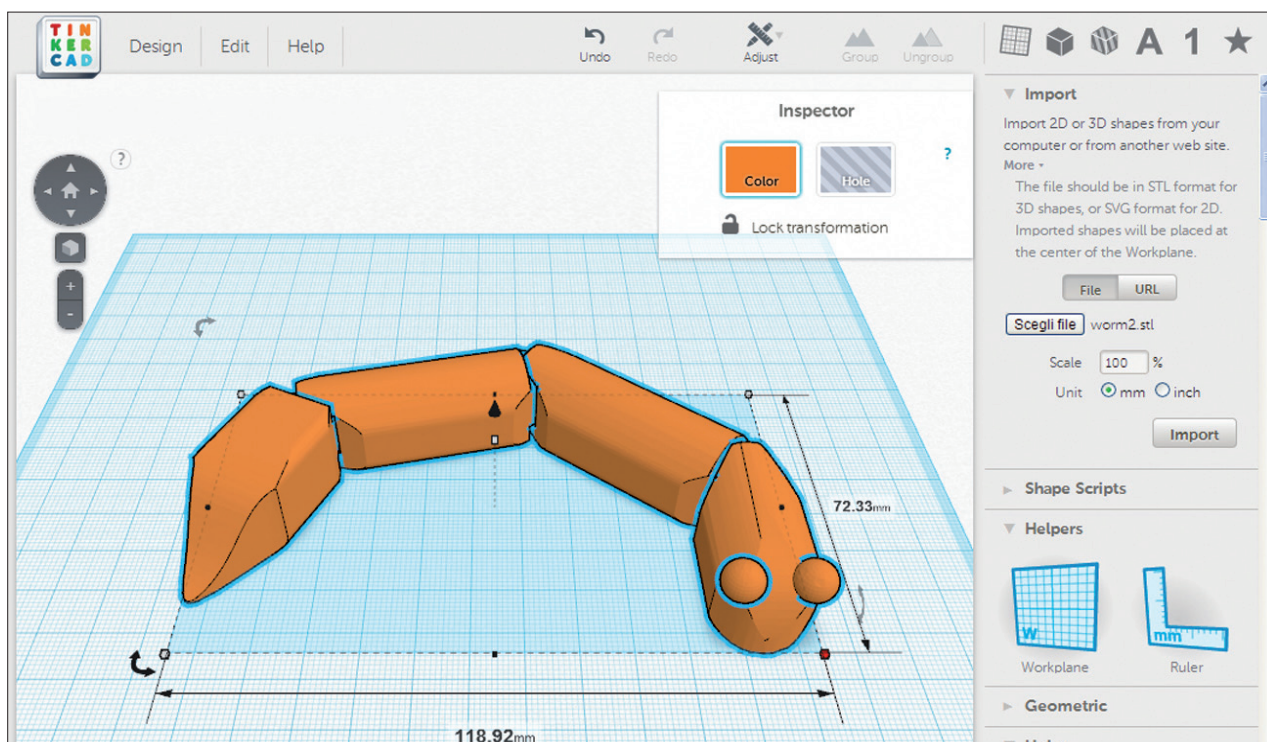


Fig. 16



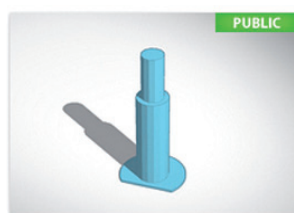
della parte superiore, ma vale comunque la pena notare che per fissare le due parti abbiamo scelto di creare un incastro fra coperchio e parte inferiore. Le viti di fissaggio dello stampato entrano in quattro torrette da 6 mm di diametro, con fori da 3mm, mentre un'ulteriore cornice interna rende la scatola un po' più rigida. Due torrette hanno anche una parte rimossa per consentire ai bulloni delle torrette che fissano lo schermo LCD di non interferire con le torrette stesse. Le pareti del coperchio sono spesse 2 mm e – conseguentemente – la cornice da ritagliare nelle pareti della base per creare l'incastro, alto 3 mm, è spessa 2 mm; allo spessore della base si aggiungono 8 mm in cui internamente trovano posto le torrette delle viti, per un totale di 13mm di spessore della base. In questo modo lo stampato appoggia sulle torrette ed è allineato con la parte superiore della base; questo semplifica la realizzazione del foro rettangolare tramite cui

potremo inserire e togliere le schede SD. Manopola e pulsante di reset sono relativamente semplici, mentre per le cornici di supporto ci siamo inventati un sistema di aggancio con i profili in alluminio. Quando li avrete stampati potrete apprezzarne la

praticità. Per chi desidera capire ancora meglio come abbiamo costruito i vari pezzi, stiamo rendendo disponibili i modelli in Tinkercad e così potrete usare la funzione *ungroup* per smontarli nei vari elementi e capire i passaggi. In Fig. 17 vedete cosa dovete cercare in Tinkercad. ■



3Drag Stampa Autonoma
Supporto
Simone Majocchi less than a minute ago



3Drag Stampa Autonoma
pin Reset
Simone Majocchi less than a minute ago



3Drag Stampa Autonoma
Manopola
Simone Majocchi about a minute ago

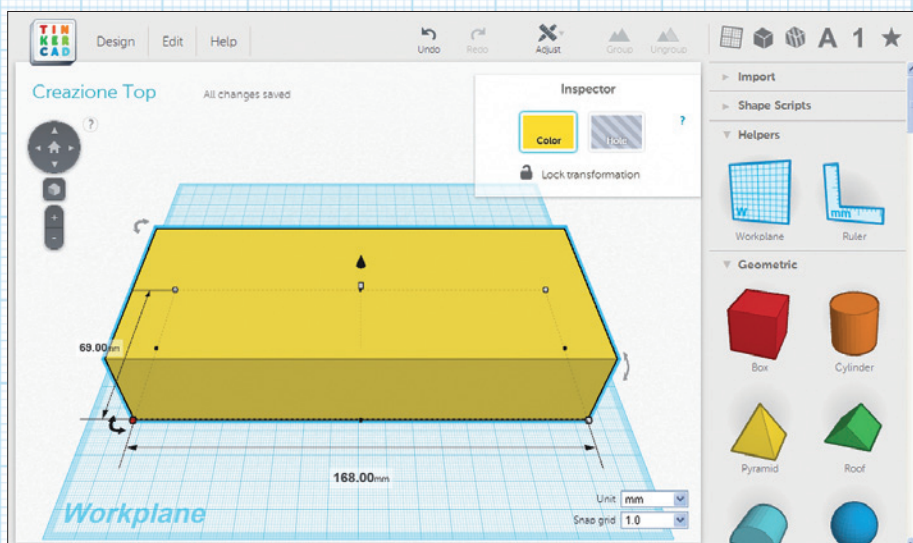


3Drag Stampa Autonoma
Top
Simone Majocchi about a minute ago



3Drag Stampa Autonoma
Bottom
Simone Majocchi 3 minutes ago

Fig. 17

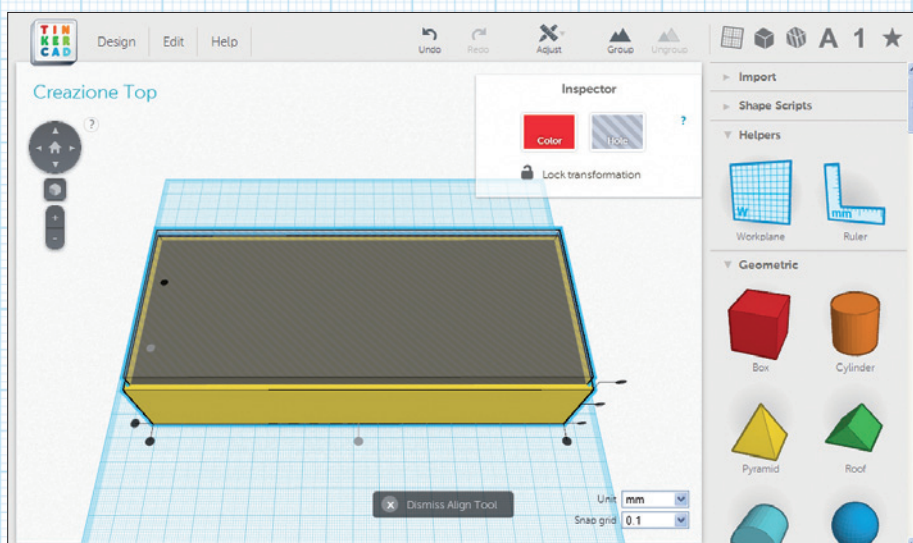
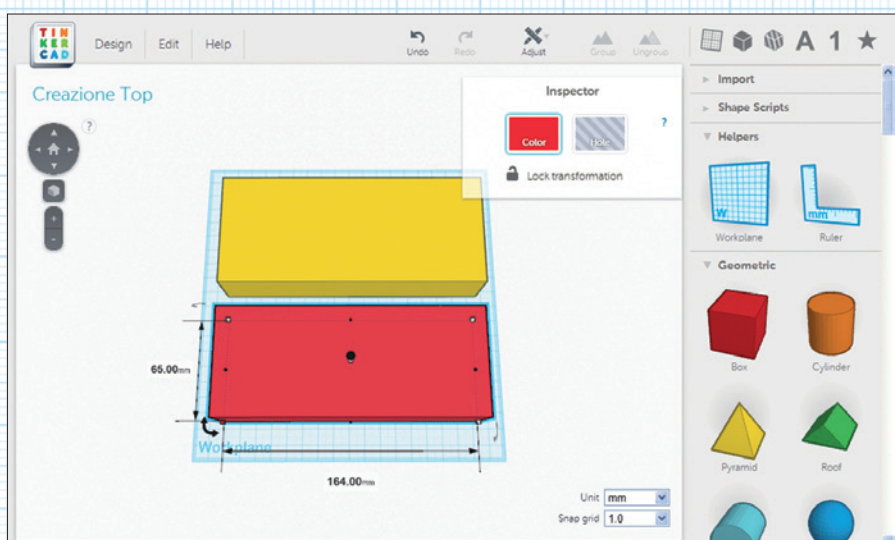


Step 1

Avete cliccato su *New Design* dalla vostra pagina personale e vi trovate con l'area di lavoro vuota. Vogliamo che il contenitore abbia circa 2 millimetri di aria per lato e che le pareti siano spesse 2 mm. I conti ci portano quindi a 168 mm di larghezza, 69 mm di profondità, mentre 36 mm di altezza ci permettono di avere lo schermo LCD arretrato di circa 1 mm rispetto alla superficie. Cliccate sul cubo nella palette laterale e tenendo premuto il tasto sinistro del mouse trascinatelo in centro all'area di lavoro, quindi tirate le maniglie centrate (i quadratini neri) sui due lati della base e portateli a misura; il quadratino centrato superiormente va invece usato per modificare l'altezza del nostro parallelepipedo fino a 36 mm. Posizionatelo in centro al piano e cambiate colore – giusto per far pratica – cliccando sulla casella *Color* del pannello *Inspector*. Posizionandovi su una delle maniglie con l'interno bianco, sugli angoli alla base, dovrete leggere le misure corrette. Con solo questo oggetto sul piano, fate anche pratica per lo spostamento del punto di vista e lo zoom usando il mouse e la rotella.

Step 2

Per creare il vuoto all'interno della scatola prepariamo un secondo parallelepipedo le cui misure sono calcolate sottraendo lo spessore delle pareti laterali, ovvero 2+2 mm per lato. Ecco quindi che le dimensioni da imporre sono 164x65 mm. L'altezza del parallelepipedo non è invece un problema, purché sia almeno pari o superiore a 36 mm. Al momento della fusione lo solleveremo facendolo sbucare dall'alto e quindi ogni abbondanza non crea problemi. Da notare che Tinkercad ci mette a disposizione un piano di lavoro da stampante 3D e quindi siamo limitati a 20 x 20 centimetri. Spostate i due parallelepipedi per farli stare sul piano di lavoro e controllate ancora una volta che le misure siano quelle esatte per la base. Non vi preoccupate di allineamenti in questa fase, mentre potete cliccare su *Design* -> *Properties* per dare un nome più consono al progetto su cui state lavorando.

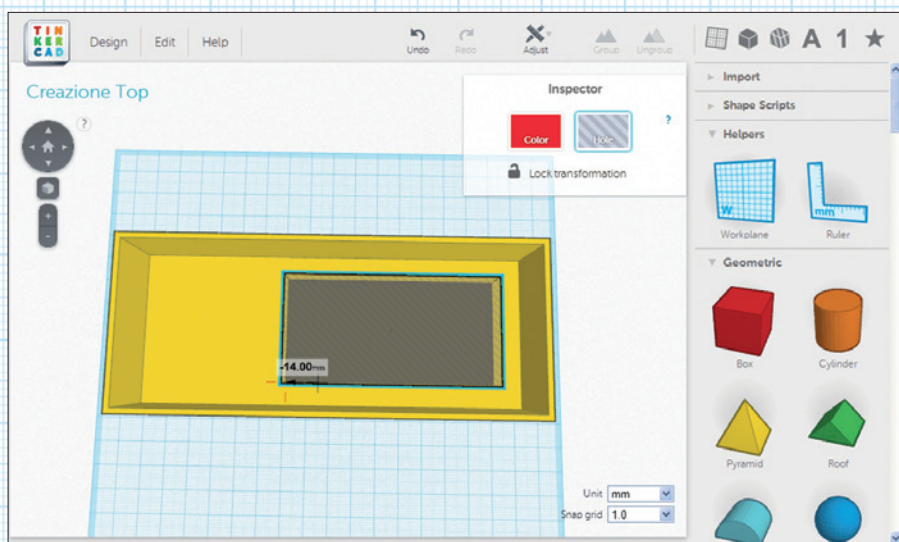
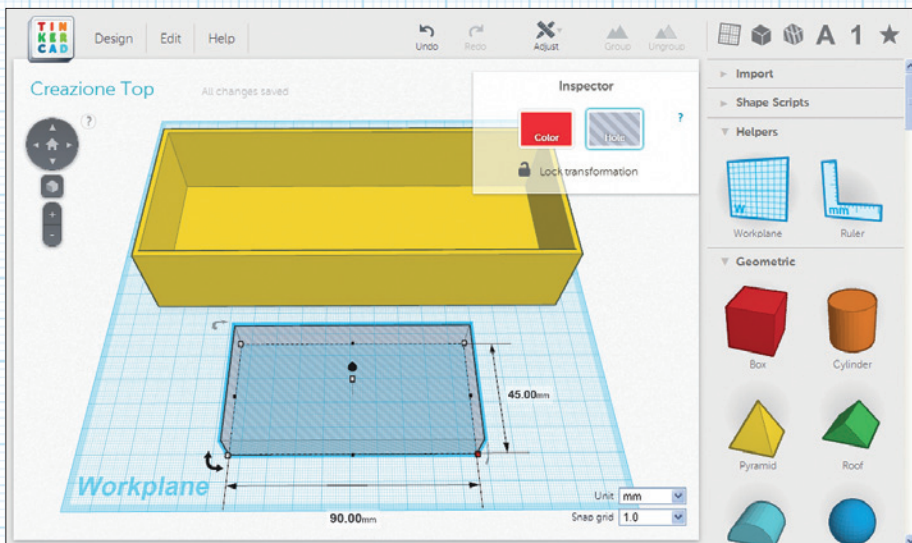


Step 3

Per sistemare i due parallelepipedi nel modo corretto utilizziamo lo strumento di allineamento. Innanzitutto dovete selezionare il parallelepipedo più piccolo e scegliere *Hole* nell'*Inspector* (diventa grigio semitrasparente). Selezionate entrambi gli oggetti, quindi scegliete *Adjust* -> *Align* per far comparire le maniglie di allineamento. Scegliete quelli per la centratura – in grigio nella nostra schermata – solo per gli assi x e y. Lo spessore del frontale del contenitore lo gestiamo invece alzando il parallelepipedo più piccolo di 1,6 mm rispetto al piano. Per fare questo dovete innanzitutto selezionare 0.1 in basso a destra nel menu a tendina *Snap grid*; quindi dopo aver selezionato il parallelepipedo interno, usate la maniglia a forma di cono che appare circa in centro alla faccia superiore. Trascinate con il mouse fino a leggere sulla quota che appare il valore 1.6 mm. Riportate a 1.0 mm lo *Snap grid*, selezionate entrambi gli oggetti – ora allineati come richiesto – e cliccate su *Group*.

Step 4

Spostate il coperchio creato nel passaggio precedente nella parte alta dell'area di lavoro e procedete alla creazione del parallelepipedo "negativo" con cui produrremo il foro per lo schermo LCD. Lo schermo è da 42 x 86mm e quindi se predisponiamo una finestra da 45 x 90mm e la posizioniamo opportunamente, potremo vedere lo schermo senza avere il contatto fisico fra coperchio e LCD. I pochi millimetri che abbiamo scelto di dare di agio diventeranno poi una piacevole cornice attorno allo schermo. La procedura è quella già vista in precedenza: trascinate il cubo sul piano, poi con le maniglie imponete le misure e infine lo fate diventare un solido da sottrarre assegnando l'attributo "Hole" dall'Inspector. Anche in questo caso l'altezza non importa più di un tanto in quanto in caso di fusione con un solido, viene eliminata la parte sovrapposta di pieno e vuoto, mentre l'eccedenza di vuoto semplicemente sparisce.

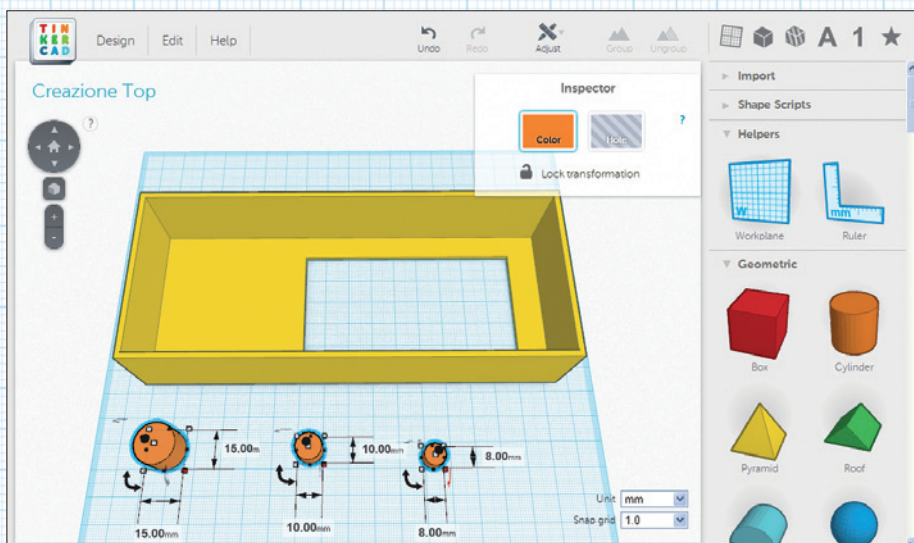


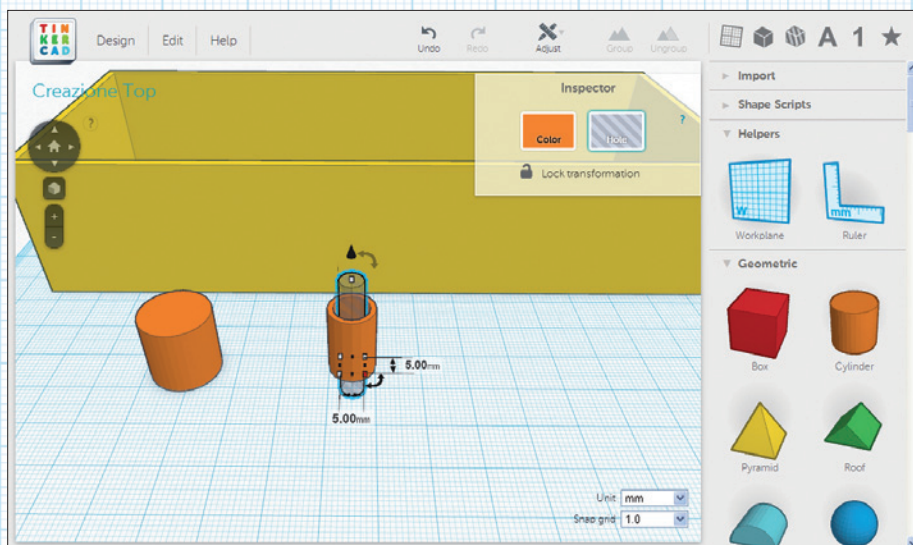
Step 5

In questo passaggio impariamo a sfruttare il sistema di quotatura e il blocco dello spostamento. L'obiettivo è quello di posizionare esattamente il foro sopra al display e le misure prese in precedenza ci dicono che verticalmente è centrato, mentre si trova a circa 12 mm dal bordo della basetta. Dalla basetta all'esterno del contenitore ci sono altri 3,5 mm e quindi il posizionamento della nostra cornice a 14 mm dall'esterno del contenitore ci permette di avere quel paio di millimetri di aria previsti, con circa mezzo millimetro di tolleranza. Per posizionare il foro correttamente usiamo lo strumento per allineare e centriamo foro e coperchio sull'asse y, quindi usiamo la maniglia che allinea a destra entrambi i solidi. Avrete il foro centrato verticalmente e tutto a destra. Ora selezionatelo e iniziate a spostarlo a sinistra, quindi premete il tasto shift per limitare lo spostamento a un solo asse e fermatevi quando leggerete -14 mm esatti di spostamento. Selezionate i due solidi e cliccate Group.

Step 6

Ora dobbiamo realizzare i due fori, per la manopola e per il pulsante di reset. Il foro della manopola è da 15 mm, mentre per il reset creiamo un cilindro vuoto che servirà da guida a un pistoncino con due diametri diversi. In questa immagine vedete i tre cilindri che dovete creare in questo passaggio, sapendo che l'altezza di quello da 15 mm non importa dato che sarà un foro, mentre quello da 10 dovrà essere alto 18 mm. Il suo foro da 8 mm, ovviamente, potrà essere alto almeno 18 mm per riuscire a bucarlo una volta che li combineremo. Per comprendere il meccanismo che stiamo costruendo, considerate che il foro che creeremo sul frontale sarà allineato sopra al pulsante di reset che, però, è alto solo 5 mm; ci mancano 30 mm al frontale e poi dovremo avere qualcosa che sporge da poter premere. Creando un cilindro da 35 mm potremmo arrivare al pulsante, ma tenere il tutto allineato sarebbe difficile ed è per questo che predisponiamo quel cilindro cavo in cui far scorrere il pistone. Per evitare che il nostro pistone fuoriesca, lo faremo nella parte inferiore da 7 mm di diametro e nella metà superiore da 4 mm.



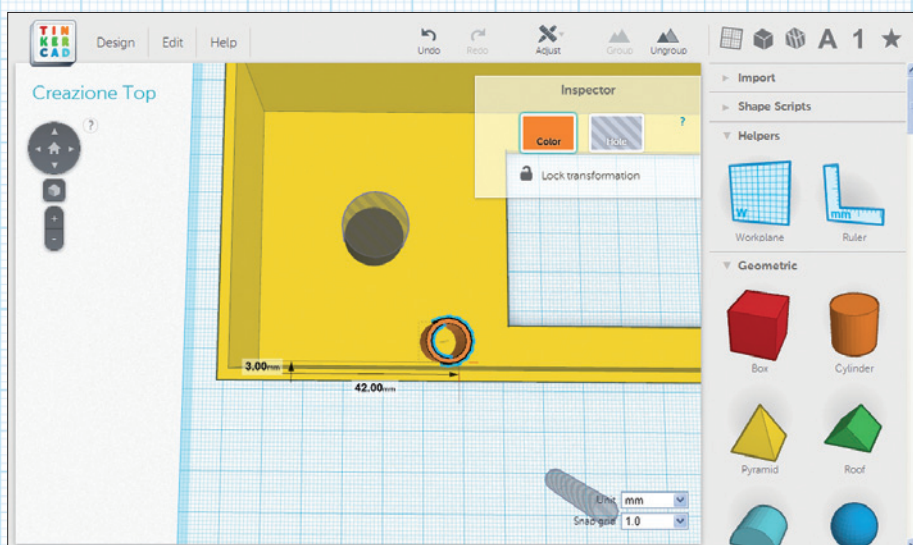
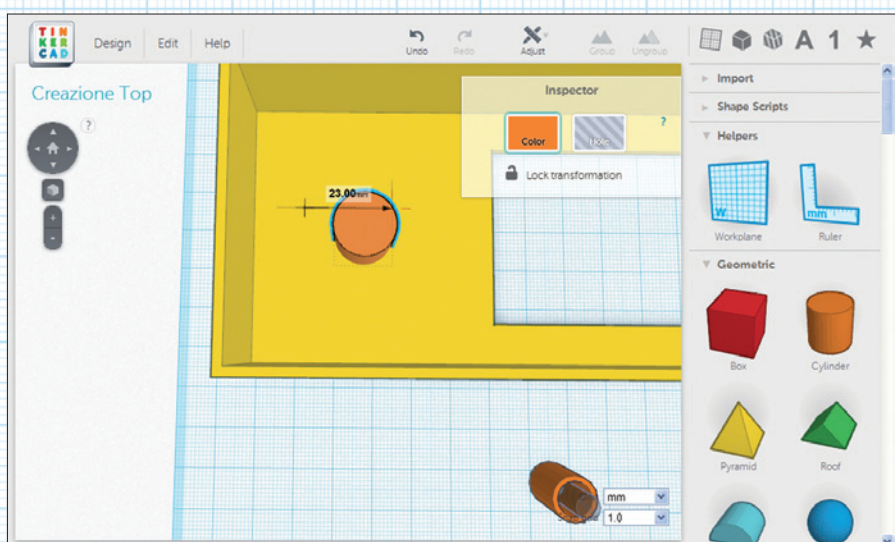


Step 7

Per arrivare al risultato che vedete qui riportato, dovete assegnare *Hole* al cilindro da 8 mm, quindi allinearlo al centro su X e Y con il cilindro da 10 mm ed infine raggrupparli. Avrete come risultato il cilindro vuoto con le pareti da 1 mm da posizionare nella parte interna del coperchio del contenitore, ma ci manca ancora il cilindro con cui fare il foro nel pannello frontale. Questo deve essere da 5 mm così che il pistone da 4 mm abbia buone possibilità di passarci già al primo colpo, altrimenti a stampa finita dovreste dare un ritocco con una lima tonda piccola per togliere il materiale in eccesso tipico dello strato iniziale di stampa. Purtroppo, non è possibile raggruppare un solido con un altro vuoto (*Hole*) mantenendo il volume negativo: resta sempre solo la parte "piena". Per creare il cilindro di guida e il foro dovremo quindi tenerli separati, posizionare il primo e poi allineare il secondo. Quello che però possiamo fare in questa fase è far sprofondare un po' il "foro" per creare poi il foro vedendo da sotto dove si posiziona.

Step 8

Il foro per la manopola dell'encoder va anch'esso posizionata con cura, anche se la manopola coprirà il foro sottostante nascondendo eventuali piccoli disallineamenti. Anche in questo caso abbiamo preso le nostre misure e sappiamo che il perno dell'encoder è centrato verticalmente e a circa 26 mm abbondanti dal bordo dello stampato. Aggiungiamo 2 mm per il bordo, 2 mm di spessore della parete e sottraiamo 7,5 mm per ragionare sul bordo del cilindro e non sul centro. Otteniamo 22,5 mm che arrotondiamo a 23 mm; questa è la distanza che dovrà avere il bordo del cilindro dalla parete sinistra. Come abbiamo fatto per il foro dello schermo LCD, iniziamo selezionando coperchio e cilindro, quindi con *Align* portiamo il cilindro a sinistra e lo allineiamo verticalmente al centro. Selezioniamo solo il cilindro e usando lo shift blocchiamo lo spostamento solo sull'asse X, trascinate verso destra fino a leggere 23 mm sulla quota che appare durante lo spostamento. Ora potete assegnare il materiale *Hole* al cilindro e, se volete, effettuare il *Group* con il coperchio.

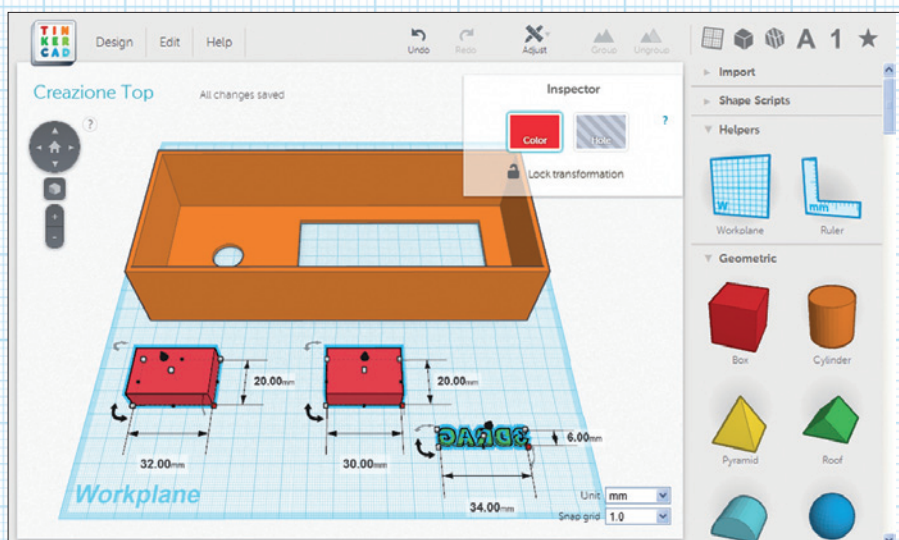
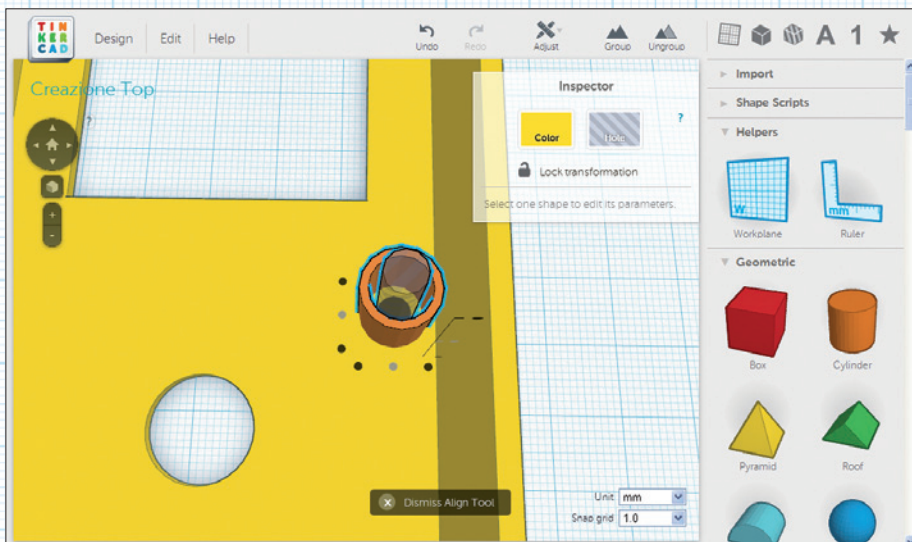


Step 9

Il posizionamento del cilindro che ospita il pistoncino del tasto di reset richiede un po' di attenzione, ma ci permette di sperimentare come Tinkercad sia in grado di facilitare anche le operazioni di precisione. Il sistema si basa sull'allineamento e poi il successivo spostamento in base alle quote che indicano di quanto l'oggetto è spostato rispetto al punto di partenza. Se durante lo spostamento, malauguratamente, lasciate il tasto del mouse, il punto di partenza diventa quello in cui avete lasciato inavvertitamente cadere l'oggetto e quindi è necessario ricominciare. Il tasto *Undo*, in questo caso, è più che mai utile. Nella schermata qui a fianco vedete come il cilindro vuoto vada posizionato a 42 mm dal bordo a sinistra e a 3 mm da quello in basso. In pratica resta 1 mm di spazio fra la parete e il cilindro. Da notare come abbiamo lasciato il foro nella sua posizione precedente. Il sistema è quello appena utilizzato: usate *Align* per portare il cilindro nell'angolo in basso a sinistra del coperchio e poi muovetelo fino a leggere le quote indicate.

Step 10

Dobbiamo praticare il forellino da 5 mm nel frontale e abbiamo posizionato il cilindro guida già nella sua posizione corretta. Quello che resta da fare è quindi l'allineamento del foro al centro del cilindro guida. Se avete provato a cliccare sulle maniglie della funzione *Align* avrete anche notato che non c'è un oggetto che resta sempre fermo fra i due da allineare e quindi se non si ragiona sulla sequenza degli allineamenti da fare, si rischia di spostare il solido sbagliato. Il rettangolo che appare fra i due oggetti selezionati per l'allineamento mostra dove gli oggetti si appoggiano in caso di allineamento sui lati esterni, mentre per la centratura entrambi gli oggetti vengono spostati, a meno che uno non sia all'interno dell'area dell'altro. Nel nostro caso, quindi, spostiamo prima il foro all'interno del cilindro e solo dopo attiviamo *Align*: in questo modo il cilindro guida non si sposta e il foro può essere centrato senza problemi. A centratura avvenuta, potete selezionare foro, cilindro e coperchio per farne un *Group*.

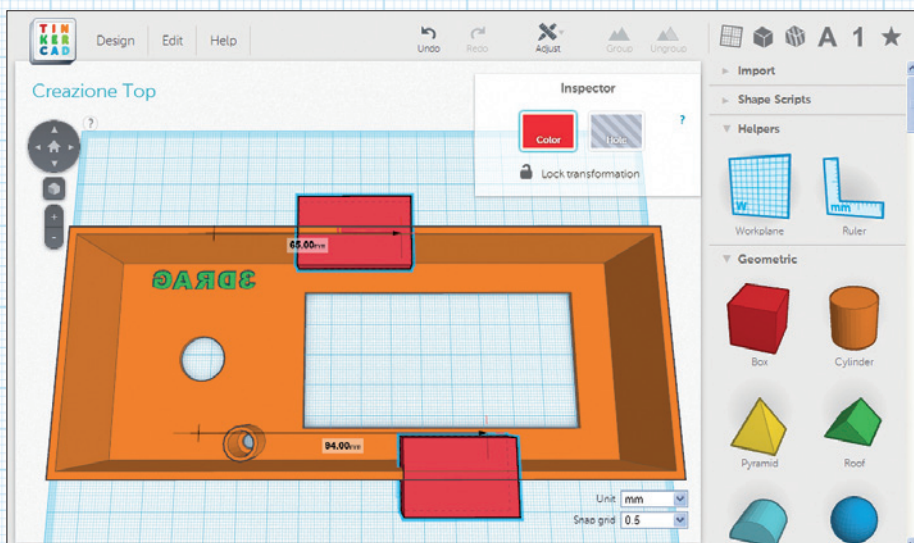


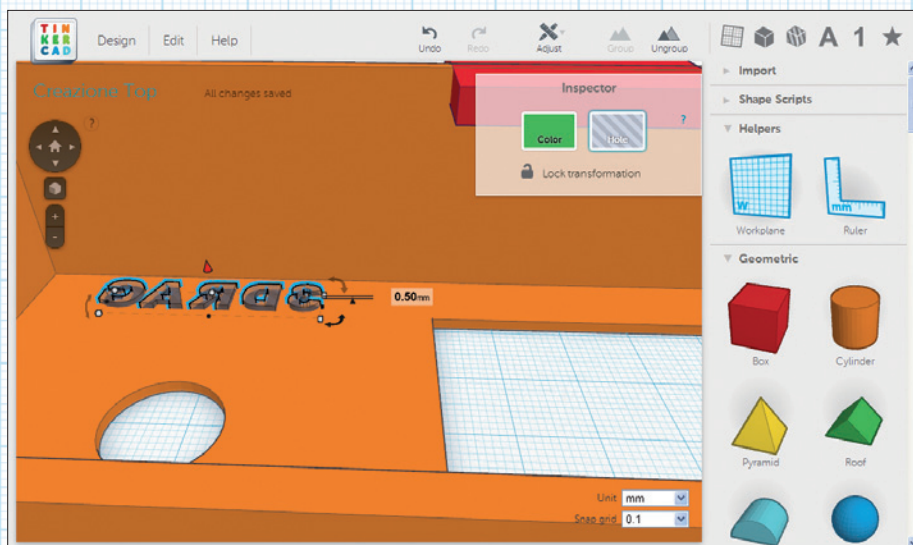
Step 11

I fori sul frontale sono ora tutti a posto e possiamo passare alla predisposizione delle due feritoie con cui potremo inserire la scheda SD e far uscire il cavo piatto verso il controller. Per semplificare il montaggio e avere un buon risultato, creeremo dei fori che arrivano fino al bordo della parete. La scheda di memoria ha bisogno di almeno 25 millimetri, mentre il cavo piatto da 20 conduttori è largo 26 mm. Aggiungiamo un po' di agio da entrambe le parti e otterremo 32 mm per il cavo flat e 30 mm per la scheda SD. Lo spessore del box largo 32 deve essere 9 mm, mentre quello da 30 deve essere spesso 4 mm. Visto che ci siamo, creiamo anche una scritta 3DRAG da posizionare sul pannello, sopra la manopola dell'encoder. Prendete dalla palette a destra le lettere e componete la scritta, lasciando le lettere un po' spaziate, quindi raggruppatele e portatele alla dimensione che vedete in figura. Con la maniglia per la rotazione rispetto all'asse Y ribaltate la scritta così che sia allo specchio.

Step 12

Per posizionare i due fori al loro posto, utilizziamo ancora una volta le misure dallo stampato. Il cavo è quasi allineato con il lato sinistro dello schermo LCD, mentre lo slot per la scheda di memoria SD è posizionato al suo centro. Trasformando il tutto in misure, il foro da 30 mm deve essere a 65 mm dal lato sinistro, mentre quello per la scheda di memoria deve essere a 94 mm, sempre dal bordo. Utilizzate l'allineamento al bordo sinistro e visto che ci siete allineate anche al bordo superiore, così che i fori creino una feritoia fino al bordo superiore, interrompendolo. L'allineamento e la dimensione sull'asse y di questi due parallelepipedi non è significativa in quanto verranno sottratti alle pareti. In questa fase posizioniamo anche la scritta 3DRAG sopra la manopola, verso il bordo superiore. Lasciate 3 mm circa fra la scritta e l'interno del coperchio, ricordando che la manopola è da 38 mm di diametro e quindi dal centro del foro sale di 19 mm. Se la scritta è troppo bassa, viene coperta dalla manopola.



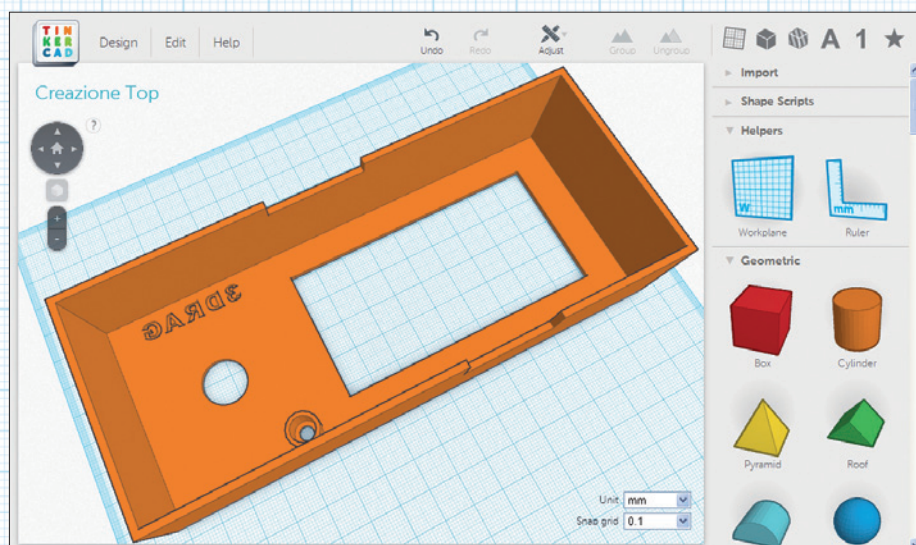
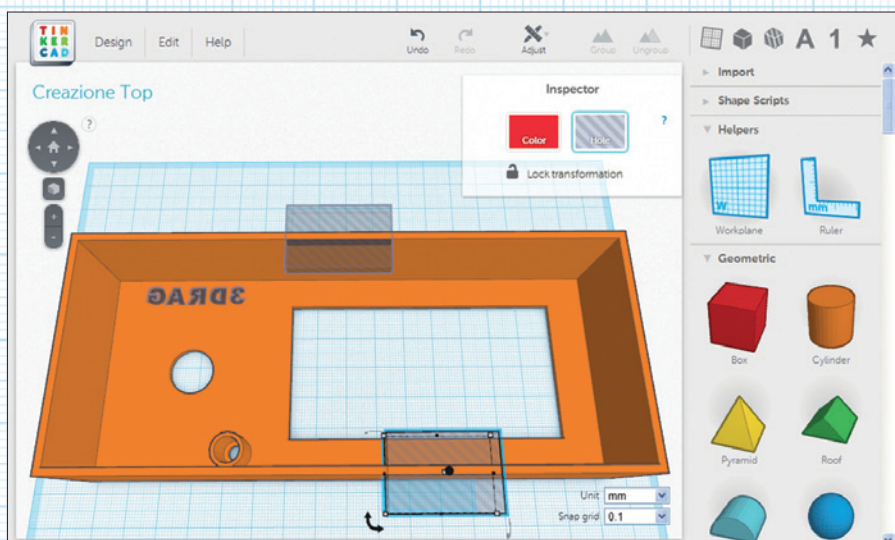


Step 13

È arrivato il momento di occuparci della scritta 3DRAG, che avete creato e ribaltato un paio di passaggi fa. L'idea che abbiamo avuto per creare qualcosa di un po' diverso e comunque facile da stampare è quella di creare un incavo con la scritta sul pannello frontale senza però forarlo (altrimenti alcuni pezzi di lettera non potrebbero esistere, come l'interno della D e della A). Se lo avessimo scavato dall'esterno verso l'interno del coperchio, la stampante avrebbe dovuto "chiudere" i caratteri da sopra visto che la parte esterna e frontale è a contatto con il piatto di stampa. Noi abbiamo invece sollevato di un paio di strati di stampa la scritta così che la trasparenza degli strati avrebbe permesso di vedere la scritta, appunto, in trasparenza. Facendo due conti, se solleviamo di 0,5 mm la scritta avremo la stampa di due strati completi prima di avere lo scavo della scritta. Per alzare la scritta, cambiate *Snap grid* e poi sollevate con la maniglia a cono, evidenziata in rosso nell'immagine. Controllate anche che la scritta sia sostanzialmente centrata orizzontalmente rispetto al foro della manopola.

Step 14

Siamo quasi arrivati in fondo e dobbiamo incorporare gli ultimi pezzi "vuoti" nella struttura del coperchio. Cliccate su ciascuno dei pezzi e assegnate il colore "Hole" tramite *Inspector*, quindi selezionate coperchio e pezzi per dare il comando di *Group* che integrerà i vuoti nel modello. Con il mouse potete ispezionare il risultato finale e osservare come l'allineamento dei lati di due oggetti con la funzione *Align* permetta di avere sempre dei "tagli" perfetti fra pieni e vuoti, cosa che non sempre accade con applicazioni di CAD. Per vedere la parte frontale del contenitore, basta spostare il punto di vista sotto al piano di lavoro che ha una sua parziale trasparenza.



Step 15

Il lavoro è concluso e appoggia sul piano con il frontale del coperchio; questo ci permette di fare una stampa di difficoltà molto ridotta dato che non ci sono inclinazioni particolari o sottosquadra. L'unica difficoltà può essere la superficie abbastanza estesa e quindi potenzialmente soggetta al distacco dal piano di stampa. Cliccate su *Design* e scegliete *Download for 3D print* per far apparire la finestra che vi propone i vari formati. Scegliete ovviamente STL e poi preparate la stampante per questo nuovo lavoro. Scaricate anche dal nostro sito – come indicato nell'articolo – gli altri elementi del nostro contenitore per stampa autonoma. Per una stampa senza problemi, vi suggeriamo di stampare contemporaneamente due o tre pulsanti di reset: la parte terminale da 4 mm è troppo piccola per poter essere stampata da sola senza deformazioni, mentre se la stampa viene fatta su due o tre pulsanti ad ogni strato, la plastica riesce a raffreddarsi e a mantenere la forma.

Assembliamo scheda e contenitore



Tra progettazione, modellazione e stampa delle varie parti, abbiamo tutto il necessario per inscatolare la basetta della stampa autonoma per 3Drag. Per completare l'assemblaggio, ci servono solo quattro viti autofilettanti da 3,5 mm e con filetto lungo fra 6 e 8 mm. Nella **Foto 1**, in giallo, potete vedere la base del contenitore, il pistoncino per il reset e le due cornici di fissaggio, mentre in rosso abbiamo stampato la manopola e in verde la parte superiore – o coperchio con tutti i vari fori di servizio.

Il primo passaggio è quello del fissaggio della basetta stampata alla base, utilizzando le quattro viti autofilettanti - indicate prima - che sono sufficientemente lunghe per bloccare la basetta, ma non tanto da bucare il fondo.

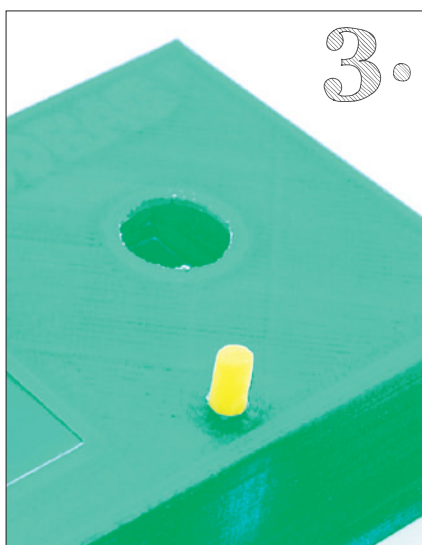
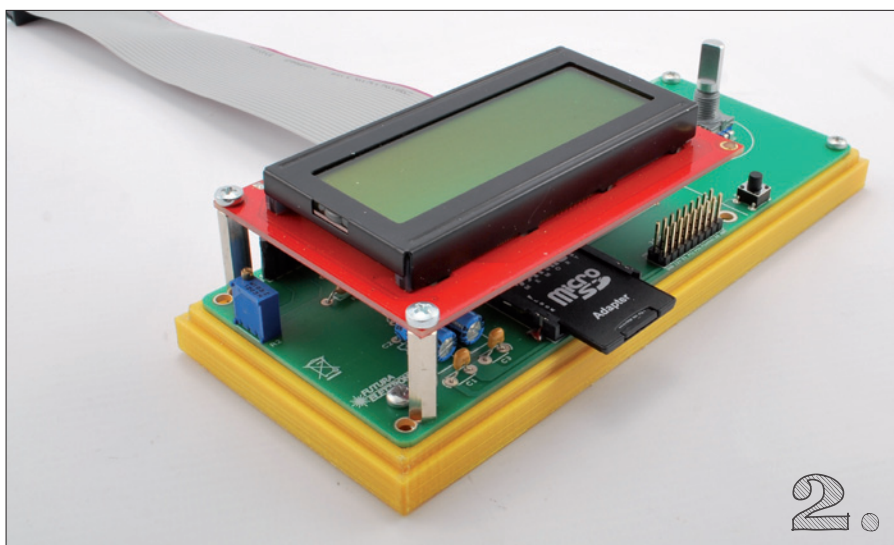
Dal lato dell'encoder i fori sono in corrispondenza degli angoli della basetta, mentre dal lato opposto i fori sono posizionati all'interno rispetto a quelli delle due torrette di supporto dello schermo LCD. Le torrette sono state sagomate per non interferire con il dado di fissaggio della torretta, ma se ci fossero problemi basta ruotare il dado.

Come si può notare nella **Foto 2**, ci sono un paio di millimetri di spazio fra la basetta e il primo bordo del fondo, mentre lo scalino che si vede è quello che permetterà di incastrare il coperchio.

Nell'immagine abbiamo già collegato il flat cable, mentre nella realtà sarà l'ultima cosa da collegare prima di unire fondo e coperchio.

Il passaggio successivo coinvolge il tastino di reset e il coperchio: sapendo che la stampa genera sul primo strato uno schiacciamento, potrebbe essere necessario ripassare con una piccola lima tonda il foro; il pistone, invece, potrebbe avere delle bavette e delle imprecisioni da eliminare con una limetta diamantata per le unghie. Prendete quindi coperchio e pistoncino e lavorate entrambi finché non avete uno scorrimento senza attriti nella sede prevista. In **Foto 3** vedete il pistoncino fuoriuscire dal proprio foro.

Sistemato il pistoncino del Reset possiamo passare all'adattamento della manopola al perno dell'encoder. Si tratta di un'operazione potenzialmente delicata, ma inevitabile: in base alle impostazioni di slicing e al feedrate, potreste avere un foro abbondante o scarso. Nel modello siamo stati più larghi di un paio di decimi di millimetro rispetto alla misura nominale, ma ci sono ottime possibilità che il perno dell'encoder non entri ancora nel foro previsto. La soluzione che vi suggeriamo – se la manopola non entra sul perno – è quella di scaldare il perno con un accendino (scaldare, non arroventare) per poi spingerci sopra la manopola con un movimento oscillatorio e rotatorio atto ad allargare il foro. State attenti a non spingere troppo incastrando perno e manopola perché potrebbe diventare difficile staccare la manopola. Avanzate di qualche millimetro alla volta, sfilate e riscaldare nuovamente il perno dell'encoder per ripetere la manovra fino ad ottenere il risultato della **Foto 4**. Sistemato il pistoncino del Reset possiamo passare all'adattamento della manopola al perno dell'encoder. Si tratta di un'operazione potenzialmente delicata, ma inevitabile: in base alle impostazioni di slicing e al feedrate, potreste avere un foro abbondante o scarso. Nel modello siamo stati più larghi di un paio di decimi di millimetro rispetto alla misura nominale, ma ci sono ottime possibilità che il perno



dell'encoder non entri ancora nel foro previsto. La soluzione che vi suggeriamo – se la manopola non entra sul perno – è quella di scaldare il perno con un accendino per poi spingerci sopra la manopola con un movimento oscillatorio e rotatorio atto ad allargare il foro. State attenti a non spingere troppo incastrando perno e manopola perché potrebbe diventare difficile staccare la manopola. Avanzate di qualche millimetro alla volta, sfilate e riscaldiate nuovamente il perno dell'encoder per ripetere la manovra fino ad ottenere il risultato della **Foto 4**.

La manopola non deve avere gioco sul perno in quanto dovremo poterla premere per il clic e non deve neanche essere troppo infilata sul perno, altrimenti toccherebbe sul coperchio senza poter essere premuta efficacemente.

Il passaggio successivo richiede il fissaggio del cavo flat sulla struttura di 3Drag fino a portare un capo alla scheda controller e l'altro nel punto in cui volete fissare il contenitore. Il punto che noi abbiamo individuato è quello in alto a sinistra, dal lato opposto della bobina di filo, come visibile in **Foto 5**. Con meno di un metro di cavo piatto è possibile eseguire il collegamento pur celando il flat dietro al profilato in alluminio. Aiutatevi con le fascette in nylon per tenere il tutto ben aderente ai profilati.

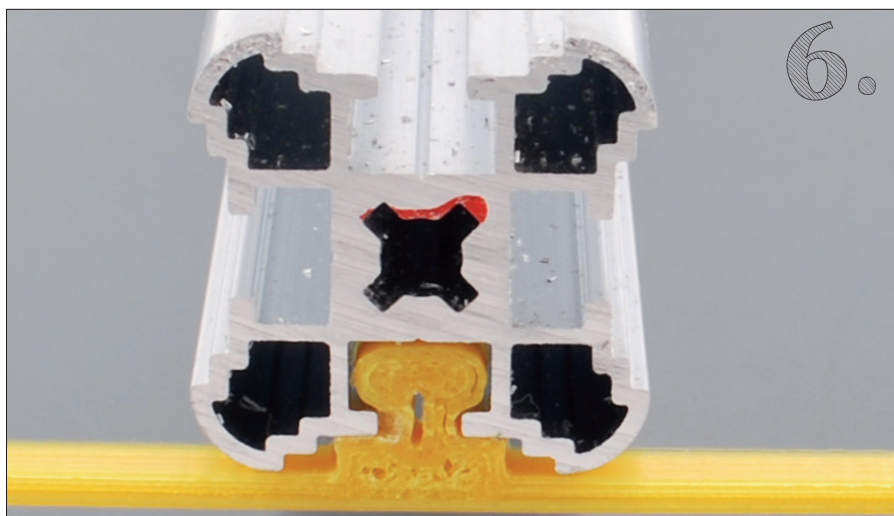
Decisa la posizione, possiamo inserire nel profilato le due cornici di supporto avendo l'accortezza di dare un'eventuale colpetto di lima dove il profilato strozza l'inserito in PLA. Per montare ciascuna cornice, dovete inserirla orizzontalmente e poi ruotarla di 90 gradi, ma non forzate e dopo una prima leggera rotazione, sfilate la cornice e date appunto un colpetto di lima sottile nei due punti dove vedete che la plastica è stata forzata.

Nella **Foto 6** potete vedere nel dettaglio come funziona il sistema di aggancio.

Con le due cornici inserite nei profilati, possiamo finalmente chiudere il contenitore, quindi dopo aver inserito il cavo flat nel modo corretto per il tipo di controller della vostra stampante, dovete inserire il pistoncino del tasto del reset e quindi fissare il coperchio alla base, partendo dall'aggancio sul lato corto e poi accompagnando la chiusura a pressione sull'altro lato. Se le due parti fanno fatica ad incastrarsi, anche provando due angoli di un medesimo lato per volta, è il caso di dare una passata con una lima piatta agli spigoli del fondo e all'interno degli angoli del coperchio (i punti dove fondo e coperchio si incastrano).

Ricordatevi che c'è il cavo flat che deve uscire dalla propria feritoia e piegarlo in modo opportuno sul suo connettore così che sfilì verso la direzione corretta, ovvero dal lato opposto al display.

Una volta chiuso il contenitore, potete agganciarlo alle due cornici di sostegno e questo dovrebbe essere possibile senza esercitare troppa pressione. Se non riuscite nella manovra, controllate che non ci siano bavette all'interno della cornice e che il coperchio o il fondo non abbiano il primo strato a contatto con il piatto di stampa che si è "spiazzicato" eccessivamente creando una specie di scalino. Anche in questo caso un po' di sapiente lavoro di lima e tutto dovrebbe andare a posto, come in **Foto 7**.



POWER METER CON INTERFACCIA WEB BASATA SU ARDUINO

L'energia elettrica costa, e soprattutto, se esageriamo, il nostro contatore interrompe l'erogazione proprio nel momento del maggior bisogno. Presentiamo un progetto, con shield specifico per Arduino, che ci permette di controllare i consumi di elettricità, anche da remoto.

di BORIS LANDONI



Il mondo dei sistemi di automazione e controllo si sta sempre più orientando verso la ricerca di soluzioni basate sull'integrazione di componenti e moduli già disponibili sul mercato, sia hardware che software, e "confezionati" in modo da soddisfare i requisiti di progetto. In questo scenario, diventano importanti, oltre alle conoscenze di elettronica tradizionale, anche la capacità di individuare e scegliere componenti esistenti e con questi progettare architetture in grado di sfruttarli al meglio ai fini della soluzione da realizzare. Non a caso sul mercato tra le ... poche richieste di collabo-

ratori, si possono trovare le figure di integratori di sistemi e di "architetti". Per "architettura" si intende un sistema più o meno complesso che prevede dei dispositivi di misurazione, attuazione e controllo integrati con sistemi di raccolta, presentazione dati e gestione dei sistemi, sia in locale che in remoto. Queste soluzioni si classificano come SCADA/HMI. L'acronimo SCADA (dall'inglese "Supervisory Control And Data Acquisition", cioè "controllo di supervisione e acquisizione dati") indica un sistema informatico distribuito per il monitoraggio elettronico di sistemi

fisici mentre HMI (in inglese Human-Machine Interface) interfaccia uomo/macchina, si riferisce alle soluzioni software che permettono agli "esseri umani" di utilizzare macchine e sistemi di supervisione e controllo nei modi richiesti.

Dopo questa premessa sui "massimi sistemi" torniamo con i piedi per terra, ovvero al nostro progetto. Nello schema di **Fig. 1** vediamo l'architettura del progetto; un sistema di misurazione e monitoraggio dei consumi di corrente elettrica con interfaccia web per il controllo remoto.

L'acquisizione dei consumi di corrente avviene per mezzo di una piccola pinza amperometrica posta a cavallo del conduttore di fase della sezione di linea elettrica che si vuole controllare. Il sistema di acquisizione è basato sul dispositivo wireless FR491 di Futura Elettronica che comprende una o più pinze amperometriche col relativo trasmettitore e una

unità di ricezione e controllo; la trasmissione avviene a 433,92 MHz ed il sistema è in grado di coprire una distanza di circa trenta metri. L'unità centrale raccoglie le letture inviate via radio dai dispositivi periferici, li memorizza e presenta una serie di informazioni circa consumi, picchi di utilizzo e costi, direttamente sul proprio display ma anche mediante PC.

L'unità centrale è infatti dotata di un'uscita RJ45 che permette di collegarla ad un PC mediante un convertitore TTL-USB.

Lo scopo di questo progetto è di collegare l'uscita seriale dell'unità centrale ad un sistema basato su Arduino e realizzare un sistema di monitoraggio remoto tramite web. Per ottenere tutto ciò abbiamo utilizzato l'uscita seriale della centralina che è stata collegata direttamente ai pin RX (D6) e TX (D7) della board Ardui-

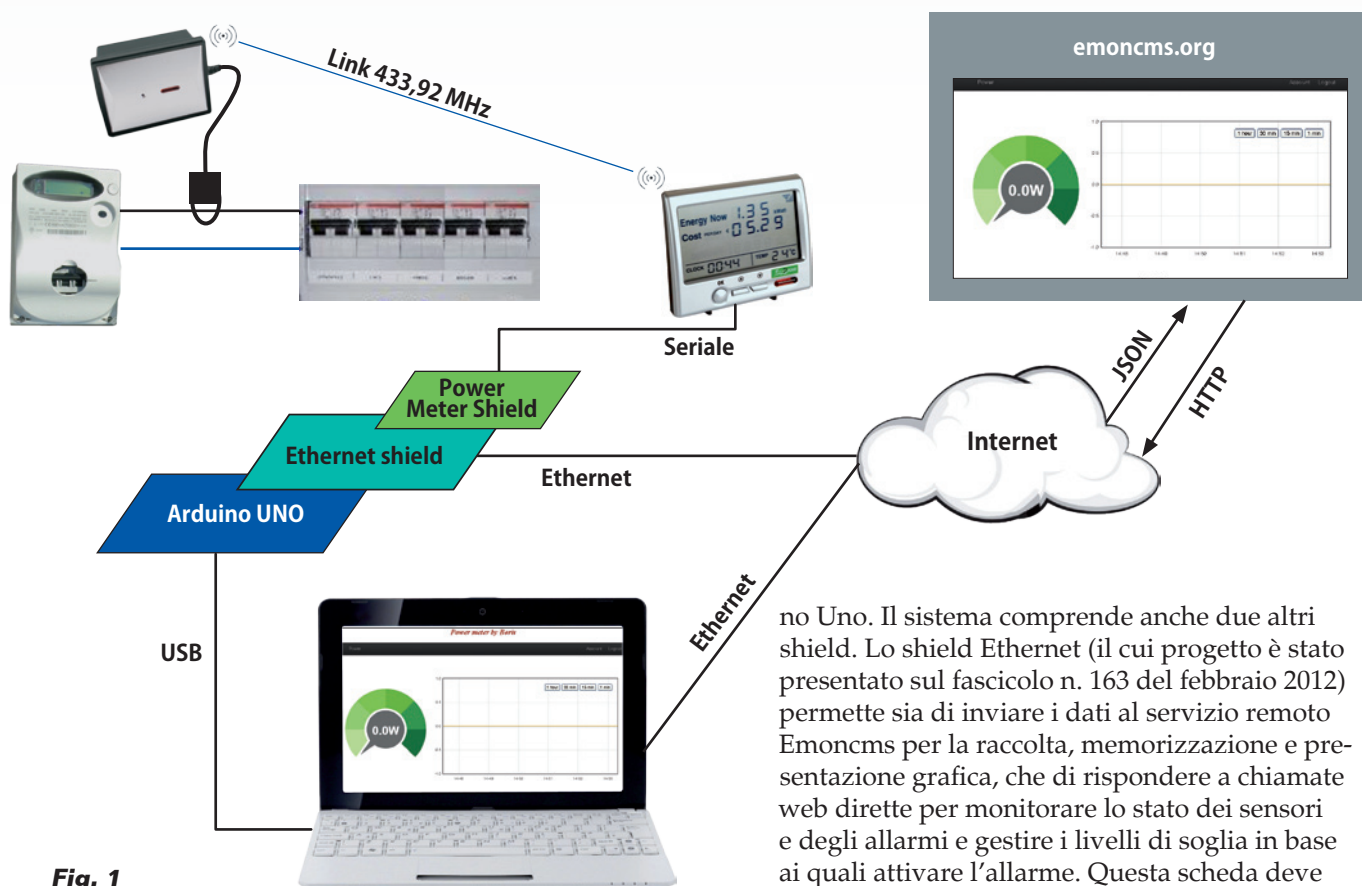


Fig. 1

no Uno. Il sistema comprende anche due altri shield. Lo shield Ethernet (il cui progetto è stato presentato sul fascicolo n. 163 del febbraio 2012) permette sia di inviare i dati al servizio remoto Emoncms per la raccolta, memorizzazione e presentazione grafica, che di rispondere a chiamate web dirette per monitorare lo stato dei sensori e degli allarmi e gestire i livelli di soglia in base ai quali attivare l'allarme. Questa scheda deve essere ovviamente collegata, tramite la porta

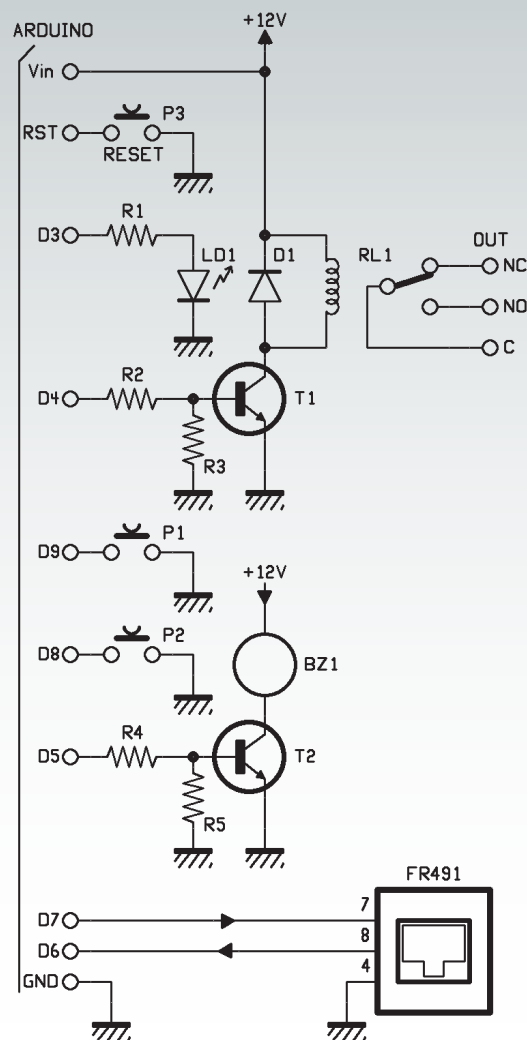
Ethernet, ad una rete che permetta l'accesso ad Internet. Il secondo shield (contraddistinto dal codice FT1046 e descritto in queste pagine), costituisce l'unità locale di controllo e gestione degli allarmi: lo schema elettrico è visibile in questa pagina qui a sinistra.

Lo shield dispone di un'uscita a relè ed un buzzer che vengono attivati se il consumo supera la soglia impostata tramite la pagina web, e di due pulsanti (oltre al pulsante di reset) che permettono di attivare e, una volta scattato, di mettere in pausa la funzione d'allarme. Lo sketch che gira su Arduino è realizzato in modo che il sistema d'allarme sia comunque pronto a riattivarsi appena il consumo torna al di sotto della soglia impostata (naturalmente gli avvisatori entreranno in funzione al superamento della soglia). Un LED segnala lo stato generale nel quale si trova il sistema. Dalla pagina web integrata in Arduino è possibile attivare o disattivare l'allarme, metterlo in pausa e impostare la soglia di allarme. In caso di attivazione dell'allarme il buzzer riproduce una musicchetta. Per quanto riguarda Emoncms, di cui ci siamo già occupati nel fascicolo 175 quando abbiamo presentato l'articolo del termostato web con Raspberry Pi, non potendo ovviamente installare il server su Arduino, utilizzeremo le funzionalità disponibili dal sito pubblico di Emoncms stesso, integrandole con la pagina disponibile dal web server interno di Arduino.

SCHEMA ELETTRICO DELLO SHIELD

Lo shield del Power Meter utilizza l'alimentazione a 12 V di Arduino, per fornire tensione alla bobina del relè ed al buzzer. La massa dello shield è ovviamente collegata alla massa di Arduino.

Analizziamo ora la linea seriale proveniente dalla centralina dell'FR491. Il segnale in uscita dalla centralina, prelevato dalla linea 8 del connettore RJ45 è collegato al pin D6 (RX) di Arduino mentre il segnale in ingresso sulla linea 7 del connettore RJ45 è collegato al pin D7 (TX) di Arduino. Per utilizzare i pin D6 e D7 per la comunicazione seriale è necessario, nello sketch, richiamare la libreria `SoftSerial`. Questa scelta è stata fatta per lasciare libera la seriale "hardware" di Arduino, condivisa con il connettore USB, in modo da permettere il debug dello sketch utilizzando il monitor seriale fornito con l'IDE di programmazione. Le uscite di attivazione dell'allarme di superamento della soglia di consumi si attestano ai pin D4 per il relè pilotato dal transistor di commutazione T1 ed al pin D5 per il buzzer, anch'esso pilotato dal transistor di commutazione T2. Il pin D3 pilota il LED LD1.



Sullo shield sono presenti due ulteriori pulsanti, P1 collegato al pin D9, per attivare e disattivare l'allarme localmente, e P2 collegato al pin D8, per mettere in pausa la funzione di allarme.

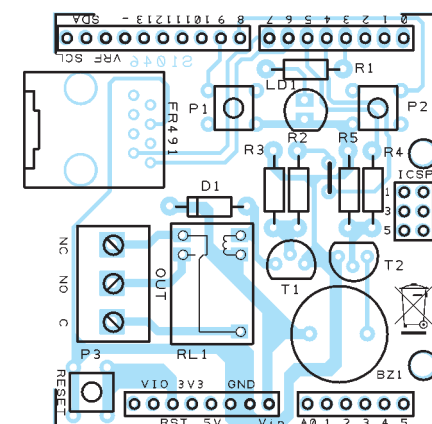
Il montaggio della scheda non presenta particolari problemi, richiedendo unicamente un minimo di esperienza nell'utilizzo del saldatore. Il posizionamento dei componenti lo si può desumere dal piano di montaggio; per quanto riguarda i collegamenti tra i diversi "moduli" del sistema, iniziamo col collegare la pinza amperometrica al cavo di fase della linea che vogliamo tenere sotto controllo. Posizioniamo lo shield Ethernet sulla scheda Arduino Uno e sopra ad entrambi montiamo lo shield del Power Meter. Utilizziamo poi un cavo RJ45 per collegare tra loro la centralina FR491 e lo shield del Power Meter, un cavo di rete tra la LAN domestica o aziendale e la presa Ethernet dell'Ethernet Shield ed infine colleghiamo un cavo USB tra il PC e Arduino. Ora possiamo dare alimentazione a tutti i circuiti.

Elenco Componenti:

R1: 470 ohm
 R2: 4,7 kohm
 R3: 10 kohm
 R4: 4,7 kohm
 R5: 10 kohm
 T1: BC547
 T2: BC547
 D1: 1N4148
 RL1: Relé 12V 1 scambio
 P1: Microswitch
 P2: Microswitch
 P3: Microswitch
 BZ1: Buzzer 12V
 LD1: LED 5 mm rosso

Varie:

- Morsettiera 3 poli
 - Connettore RJ45
 - Strip Maschio/Femmina 3 poli (2 pz.)



- Strip Maschio/Femmina 6 poli (1 pz.)
 - Strip Maschio/Femmina 8 poli (2 pz.)
 - Strip Maschio/Femmina 10 poli (1 pz.)
 - Circuito stampato

Per la configurazione iniziale ed il funzionamento del kit FR491 vi rimandiamo al manuale in italiano incluso nella confezione. Carichiamo ora lo sketch messo a punto per questo progetto, che possiamo prelevare dal sito www.elettronica.in.it.

LO SKETCH PER ARDUINO

In Fig. 2 possiamo osservare il diagramma a blocchi dello sketch di Arduino. In ciascun blocco abbiamo riportato la label che caratterizza la routine corrispondente nello sketch. Con dei riferimenti numerici abbiamo evidenziato le funzioni che vengono attivate a seconda dei comandi impartiti dalla pagina web. All'inizio dello sketch possiamo individuare la definizione delle variabili necessarie a colloquiare con il server Emoncms, e quelle relative alla configurazione dell'interfaccia Ethernet: in questo caso dovete impostare il mac address del vostro shield Ethernet (in genere specificato su una apposita etichetta apposta sullo shield stesso) e un indirizzo statico compatibile con lo schema di indirizzamento della vostra rete domestica.

```
// ethernet interface mac address
static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };
// ethernet interface ip address
static byte myip[] = { 192,168,0,99 };
// gateway ip address
static byte gwip[] = { 192,168,0,1 };
static const byte dnsip[] = { 192,168,0,1 };
```

Il blocco Setup configura i pin di Arduino per la gestione dell'allarme ed inizializza la porta seriale di debug. Se è la prima accensione in assoluto del sistema, viene impostato in EEPROM il valore di

3000 come valore di soglia per l'attivazione dell'allarme. Viene configurata anche la porta Ethernet ed inizializzata la porta seriale di comunicazione con la centralina FR491.

Il blocco loop è quello che viene eseguito in ciclo continuo. La prima azione è di porsi in ascolto di eventuali richieste HTTP esterne, intercettarle e gestirle in base alle "action" ricevute.

Le istruzioni che verificano la presenza o meno di richieste http esterne sono:

```
word len = ether.packetReceive();
word pos = ether.packetLoop(len);
```

Richiamando da un browser esterno il solo indirizzo IP di Arduino si riceve in risposta la pagina base, con integrato il frame proveniente da Emoncms. Nella

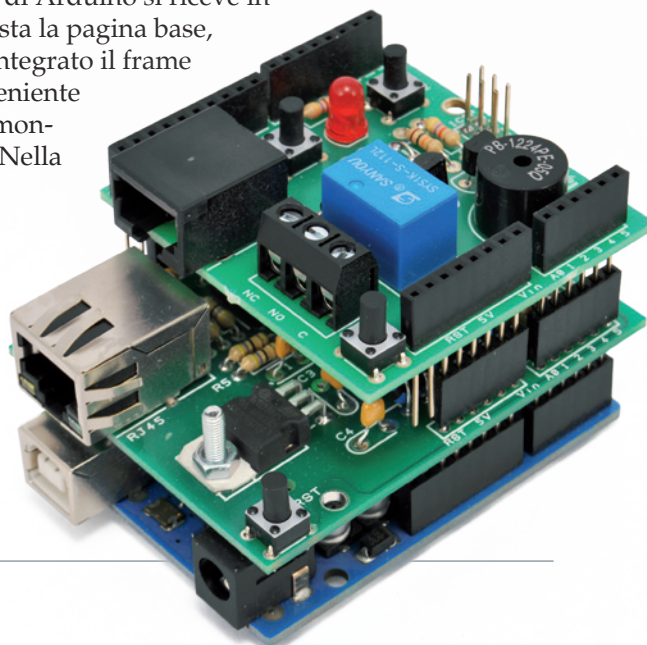
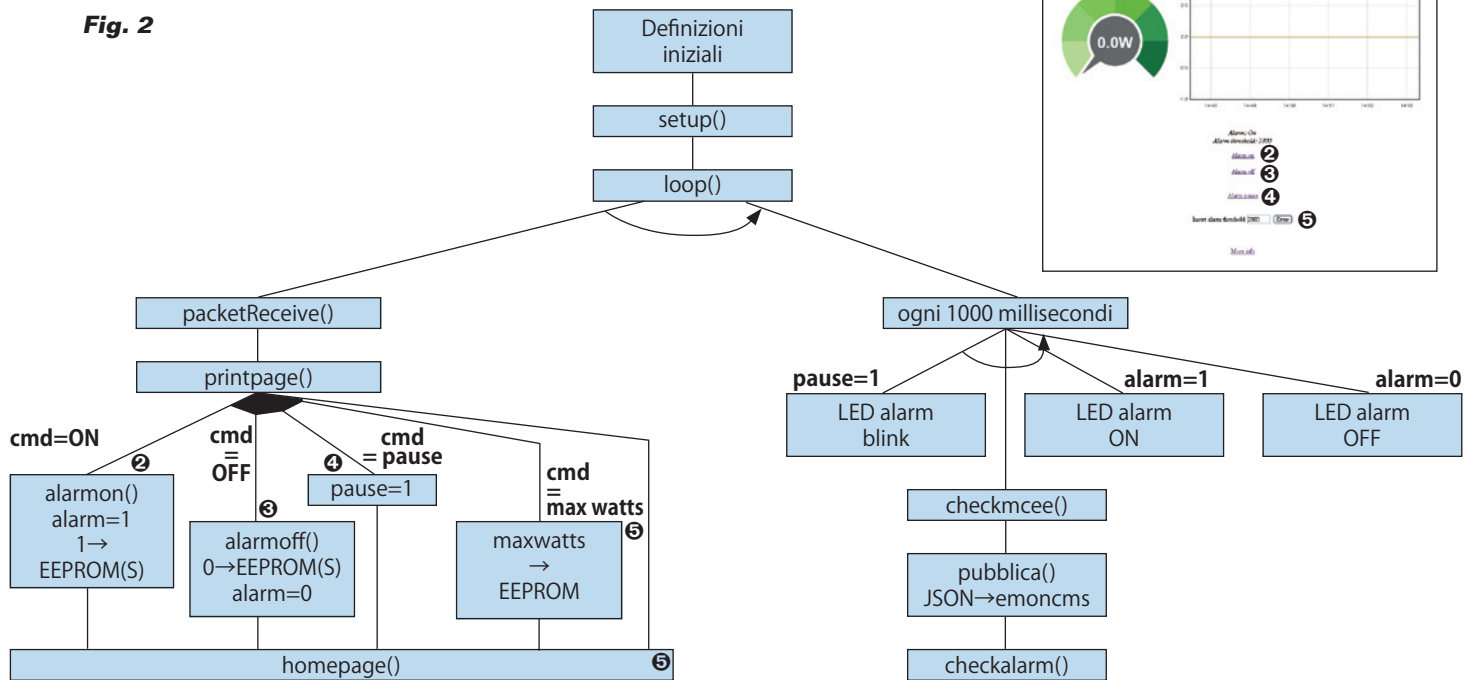


Fig. 2



pagina sono poi disponibili i link ed il campo di testo per impostare la soglia di allarme, attivare, disattivare e mettere in pausa l'allarme. Se non vi sono richieste, ad ogni intervallo impostato nella variabile `power_postrate` (tre secondi) viene eseguita la lettura delle misure memorizzate nella centralina FR491, mediante la funzione `checkmcee()`. Per interrogare l'FR491 è necessario mandare i comandi:

```
mySerial.write((byte)0xAA);
mySerial.write((byte)0x02);
mySerial.write((byte)0x00);
mySerial.write((byte)0xAD);
```

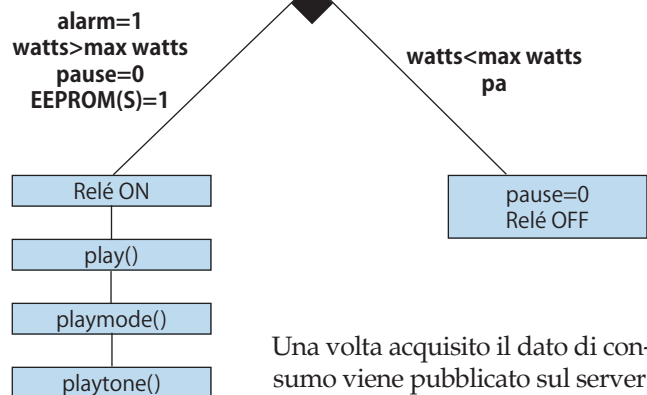
In questo modo l'FR491 risponderà con il consumo totale e di ogni singola spina supplementare (FR491/PR).

La centralina FR491 risponde con l'invio di una stringa composta come nell'esempio seguente:

```
S01\0xff\0xff\0x00\0x002
S02\0xb73\0x00\0x002
.....
S16\0xff\0xff\0x00\0x002\0xf3
```

La lettura del consumo del sensore di nostro interesse si trova nelle posizioni 5 e 6 (dopo la "S") della stringa S02, con il byte meno significativo (5) per primo e il più significativo (6) in seconda posizione. Viene "raddrizzato" con l'istruzione:

```
watts=(inData2[6]<<8) + inData2[5];
```

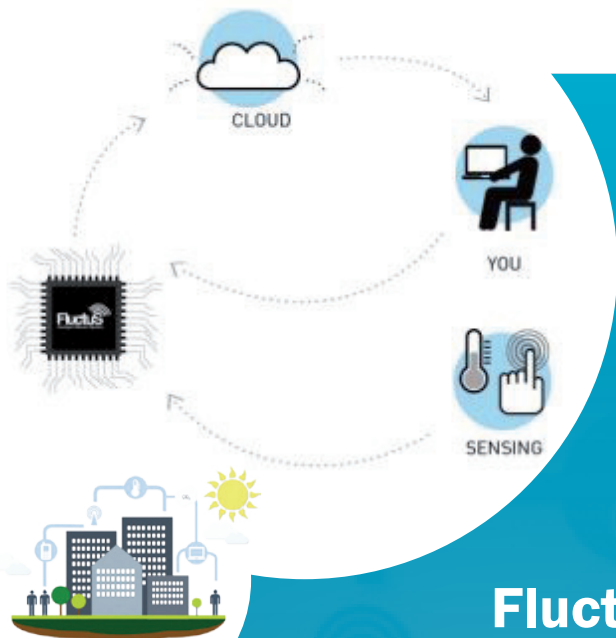


Una volta acquisito il dato di consumo viene pubblicato sul server Emoncms, con la funzione `pubblica()`, utilizzando una chiamata

HTTP ed il protocollo JSON.

Infine viene verificato, con la funzione `checkalarm()`, se il valore appena letto supera la soglia di allarme oppure no. In caso affermativo viene attivato il relè collegato al pin D4 di Arduino e, a seconda delle condizioni (allarme attivato o meno e pausa attivata o meno, come visibile nella struttura a blocchi dello sketch), viene attivato il buzzer collegato al pin D5.

Sempre nel ciclo `loop()` vengono analizzati gli stati dei pulsanti P1 e P2 per verificare se localmente sono stati richiesti l'attivazione o la disattivazione dell'allarme (P1) o la pausa dell'allarme stesso (P2). In caso di pressione del pulsante P1 vengono attivate le stesse funzioni che vengono attivate dai comandi della pagina web: `alarmon()`, `alarmoff()`. In caso di pressione del pulsante P2 viene portato a "1" il valore della variabile `pause` per pilotare il funzionamento della funzione `checkalarm()` (blink



FluctuS, una piattaforma professionale per l'Internet of Things (IoT) e le Smart Cities

FluctuS - Intelligent Sensor System (www.e-fluctus.com) è una piattaforma open software ed open hardware completamente ingegnerizzata per le applicazioni dell'Internet of Things (IoT) e delle Smart Cities, sviluppata completamente in Italia da Cubit ed altri partner tecnologici (Witech e IsTech).

L'iniziativa si rivolge ad un target professionale, System Integrators, società di progettazione e consulenza, e tra le sue caratteristiche ha quella di potersi appoggiare ad una piattaforma cloud appositamente predisposta.

FluctuS è il risultato di oltre 10 anni di ricerca e sviluppo sui temi delle tecnologie wireless ed è frutto di un progetto sperimentale che ha come obiettivo quello di trasferire tecnologie innovative al mercato. La soluzione FluctuS è certificata CE, modulare, interoperabile, innovativa, completamente ingegnerizzata e di immediata applicazione. FluctuS è composto da moduli hardware in grado di raccogliere informazioni provenienti da una varietà di sensori (es. Gas metano, CO, NOx, SOx-Air quality, Temperatura, Umidità, Pressione atmosferica, Telecamere, ecc.) ed inviarli tramite differenti tecnologie (UMTS/GPRS/WiFi/Zigbee/Bluetooth/Ethernet) ad una piattaforma Cloud dedicata. FluctuS intende essere un luogo "virtuale" di confronto tra i soggetti del settore, una community in rete che

condivida le proprie applicazioni con FluctuS e gli asset tecnologici innovativi, in modo collaborativo, aperto ed interoperante.



di LD1 in caso di allarme attivato). Dalla pagina web oltre ai comandi "Alarm On", "Alarm Off" e "Alarm Pause", che attivano le stesse funzioni dei pulsanti, è anche presente un campo di testo per impostare la soglia, in Watt, di attivazione dell'allarme ed il relativo pulsante di Invio (Enter). Quest'ultimo caso viene gestito nella funzione `printpage()`.

Il corpo della pagina web viene composto ed inviato dalla funzione `homePage()` richiamata dall'istruzione :

```
ether.httpServerReply(homePage());
```

Nella composizione della pagina viene richiamata, all'interno di un "iframe", la pagina grafica di visualizzazione dati di consumo prodotta dal server `emoncms`.

Prima di utilizzare tutto il sistema è necessario configurare il server `emoncms`. Per la configurazione del server, per l'utilizzo del protocollo JSON e per il richiamo della pagina grafica con l'iframe, vi rimandiamo anche ai numeri 175 e 176 della rivista (articolo Termostato web con RaspberryPi) oppure al relativo capitolo del libro "Raspberry Pi, il mio primo Linux embedded".

CONFIGURAZIONE DEL SERVER EMONCMS

Per prima cosa dobbiamo creare un account sul server `emoncms` all'indirizzo <http://emoncms.org/user/login>.

Come utente utilizziamo la nostra e-mail e come password quella che preferiamo (**Fig. 3**).

Ci viene proposta la pagina di amministrazione dove possiamo cambiare la password e altre im-

Fig. 3

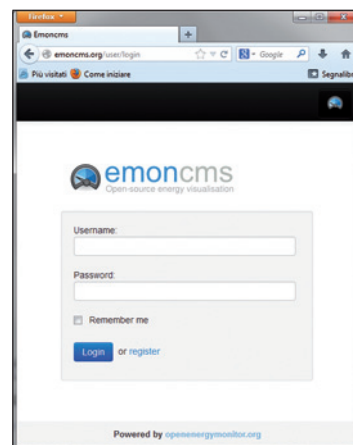
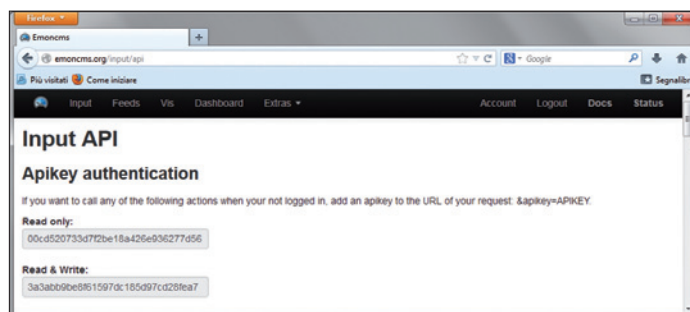


Fig. 4



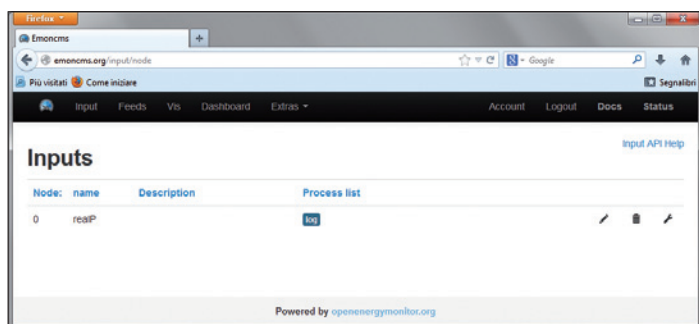


Fig. 5

stazioni, la saltiamo per andare al pannello Input e clicchiamo sul link in alto a destra "Input API Help" che ci apre una pagina dove troviamo due lunghe stringhe di dati, le chiavi API per utilizzare le funzionalità emoncms in lettura e scrittura (Fig. 4). Useremo l'Apikey in scrittura come chiave di autenticazione all'interno del messaggio JSON inviato dal programma che gira su ARDUINO.

Per produrre l'Input per il nostro misuratore di consumi è sufficiente fare eseguire il programma di Arduino almeno una volta. Prima però dobbiamo configurare opportunamente nello sketch le righe di codice:

```
#define APIKEY "3a3abb9be8f61597dc185d97cd28fea7"
```

Per ora copiamo l'APIKEY della nostra configurazione sostituendo quella nello sketch.

Compiliamo il codice e lo carichiamo su Arduino. Con tutto collegato lo mandiamo in esecuzione ed aspettiamo qualche minuto. Torniamo ad aprire il pannello Input, che ci apparirà vuoto.

Clicchiamo sul carattere di espansione "+" e, come per incanto, vedremo il nome del nostro Input, "realP" nella pagina (Fig. 5).

Gli ingressi a emoncms di tipo Input creano solo il punto di ingresso per i valori, una specie di porta assegnata a ciascun canale di alimentazione. Per creare i contenitori dove i dati vengono accumulati e conservati è necessario creare dei Feed. Lo facciamo dalla pagina Input. Clicchiamo due volte sul nome blu dell'input "realP" per aprire la pagina

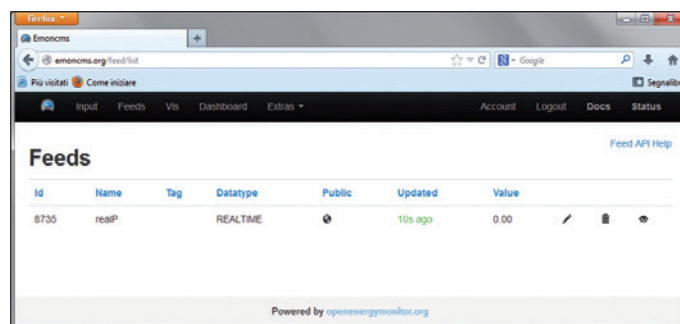


Fig. 6

di creazione del Feed, diamo un nome al Feed, per esempio lo stesso nome "realP" e premiamo il pulsante ADD.

Vediamo creato un Feed per il nostro Input, potremmo crearne altri per lo stesso Input, ma ce ne basta uno. Andiamo alla Pagina Feed e clicchiamo sull'icona "+" di espansione. Troviamo la situazione di Fig. 6.

Ora possiamo passare alla composizione dell'interfaccia grafica (dashboard), lo facciamo selezionando il pannello Dashboard, che ci presenta una pagina dove saranno contenuti i riferimenti alle pagine grafiche, con la scritta "No dashboards created". Per creare la nostra dashboard clicchiamo sul piccolo pulsante rotondo in alto a destra con il segno "+" al centro. Ci appare una riga che indica il riferimento ad una dashboard ancora vuota e vedremo una serie di piccoli pulsanti sotto la voce Action. Se avviciniamo il mouse a ciascuno, vedremo apparire il relativo significato. Clicchiamo subito sulle icone "Published" e "Public" in modo da rendere disponibile la nostra pagina grafica all'esterno, poi clicchiamo sul pulsante "Draw" e otteniamo un pannello vuoto dove possiamo cominciare a comporre la nostra pagina grafica.

Per inserire gli elementi nella pagina abbiamo a disposizione i combo box Widgets, Text, Containers e Visualizations.

Il metodo per comporre la nostra pagina grafica consiste nello scegliere un componente dai combo box nel menu, spostare il cursore nel punto dove si vuole posizionare il componente e cliccare col tasto

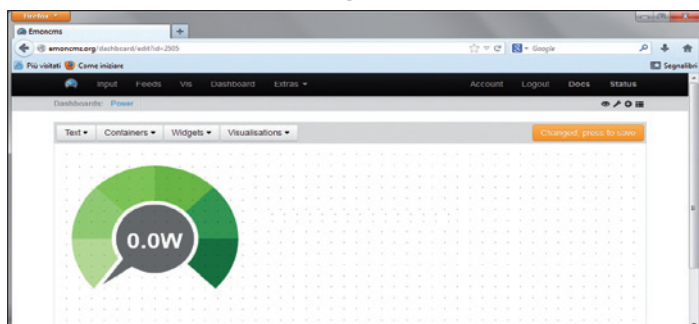


Fig. 7

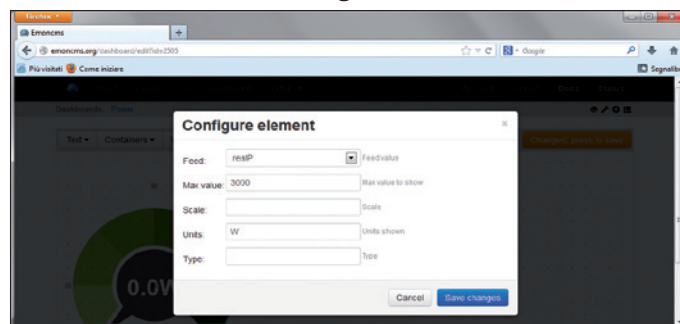


Fig. 8



Fig. 9

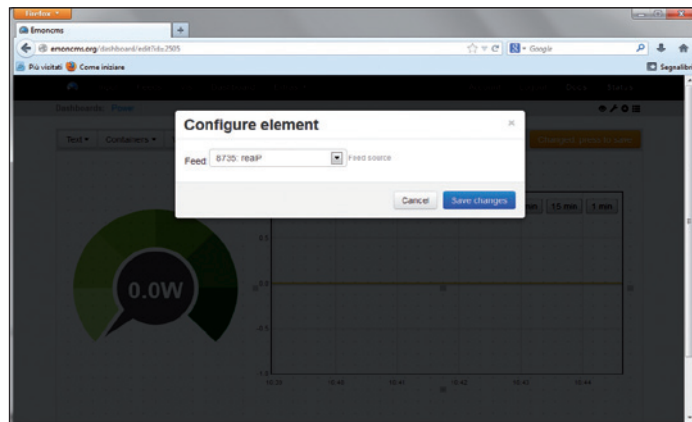


Fig. 10

sinistro del mouse per fissarlo. Poi dobbiamo associare il Feed al componente. Lo possiamo fare subito oppure successivamente selezionando il componente e premendo il pulsante Option. Se sbagliamo o vogliamo cambiare qualcosa possiamo cancellare il componente selezionandolo con il mouse e premendo Delete. Ricordatevi di salvare spesso la configurazione della pagina con il pulsante Save, per evitare di perdere il lavoro fatto. Possiamo posizionare i componenti dove vogliamo, selezionandoli e spostandoli tenendo premuto il tasto sinistro del mouse; al rilascio rimangono fissati nell'ultima posizione. Salvate. In una certa misura è anche possibile ridimensionare i componenti, ma la funzionalità non lascia grandi margini di manovra. Iniziamo ad inserire un indicatore a lancetta per visualizzare il consumo in Watt. Scegliamo dal combo box Widgets il componente Dial e lo posizioniamo a piacere (Fig. 7). Ora selezioniamo Option e configuriamo alimentazione e apparenza.

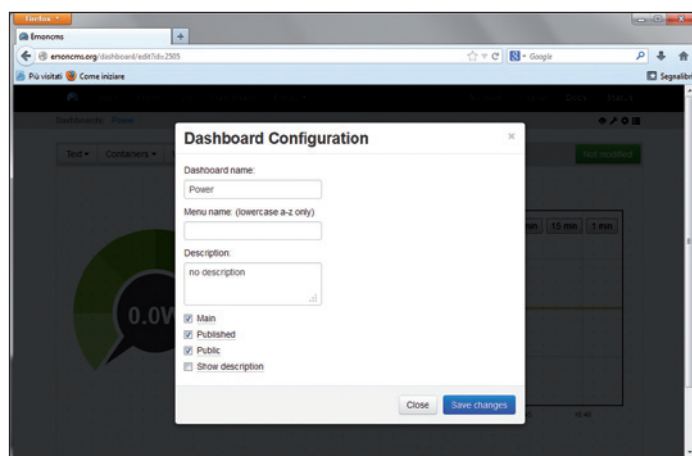


Fig. 11

Selezioniamo il Feed "realP", Il valore massimo Max, per esempio 3.000 ed impostiamo il campo units a "W".

Ora nel campo type possiamo scegliere l'aspetto che vogliamo dare al Dial.

Ecco i valori possibili con i relativi significati, sperimentate pure, se lo desiderate:

0. Dial verde
1. A settori con colori dal Rosso al Verde
2. Dal Verde a Rosso con il valore 0 a sinistra
3. Dal Verde a Rosso con il valore 0 al centro
4. Dal Rosso al Verde con il valore 0 a sinistra
5. Dal Rosso al Verde con il valore 0 al centro
6. Centro Verde che diventa Arancione verso entrambe le estremità
7. Dial Blu elettrico
8. Da Blu a Rosso chiaro a Rosso intenso

Scegliamo il valore 0 oppure lasciamo vuoto il campo (Fig. 8). Ricordatevi di salvare, e provate a seguire nuovamente il programma che legge la centralina di controllo del consumo di corrente. Dovrebbe apparire il valore in Watt al centro



Il ricevitore FR491 è collegato con cavo seriale allo Power Meter Shield.

dell'indicatore e la lancetta dovrebbe portarsi in una posizione che rappresenta graficamente il valore letto. Infine possiamo aggiungere un grafico dell'andamento dei consumi nel tempo. Dalla combo box "Visualizations" selezioniamo "realtime", posizionare il grafico dove desiderate (Fig. 9) e personalizzate le opzioni, selezionate sempre il Feed "realP" com in Fig. 10. Salvate le opzioni del grafico e salvate la dashboard. Possiamo impostare un nome per la dashboard cliccando sull'icona a forma di piccola chiave da meccanico. Inserite un nome nella pagina di configurazione, per esempio "power" e salvate (Fig. 11). Salvate ancora l'intera Dashboard, selezionate il pannello "Dashboard", e poi cliccate sull'icona "list", quella a nove puntini in forma di quadrato. Da qui cliccate sull'icona a forma di occhio ("view"). Ora dovete comporre il link da inserire nello sketch di Arduino per alimentare l'iframe. Copiate l'URL della Dashboard, nel nostro caso:

<http://emoncms.org/dashboard/view?id=2505>

Con questo componiamo il link da inserire nella "#define emonlink" con il criterio:

<http://emoncms.org/> + <nostro-user> + &id= + <il-numero-id-dell'URL> ,

nel nostro caso:

```
#define emonlink "http://emoncms.org/boris&id=2505"
```

Ricompilete e ricaricate lo sketch su Arduino. Da questo momento possiamo utilizzare il nostro sistema in modalità normale. Da una finestra del browser inseriamo l'indirizzo che abbiamo assegnato ad Arduino. Ci apparirà la videata di Fig. 12 dove possiamo vedere lo stato dell'allarme ed il livello di soglia impostato. Al di sotto troviamo i link per attivare e disattivare l'allarme. Ancora più sotto il link per "mettere in pausa" la segnalazione di allarme dopo che si è attivato ed infine il campo dove poter inserire un nuovo livello di soglia ed il pulsante per inviare la richiesta di modifica. Abbiamo già anticipato che le funzioni di attivazione, disattivazione e messa in pausa dell'allarme possono essere eseguite anche in locale utilizzando i pulsanti P1 e P2. Vediamo ora il comportamento del LED LD1, del relè e del buzzer. In situazione normale con "Alarm Off", il LED è spento; se il consumo supera la soglia impostata non accade nulla. Con "Alarm On" e "Alarm Pause" disattivato, il

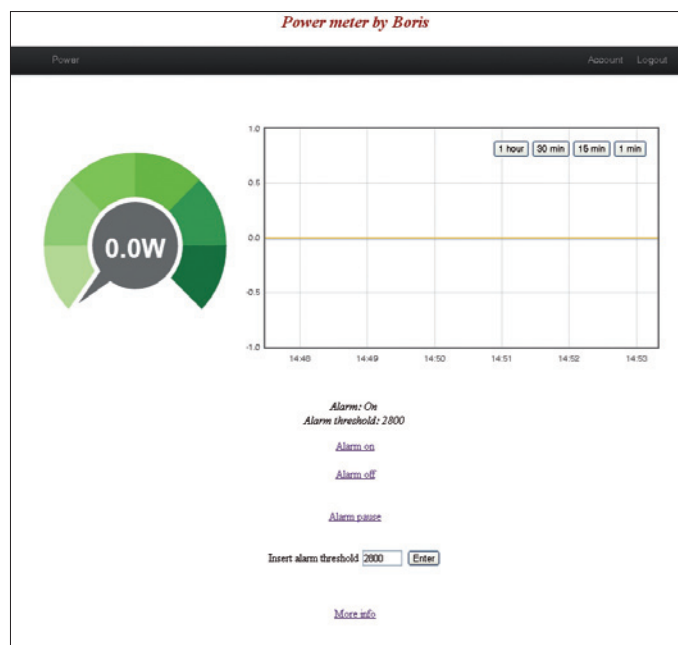
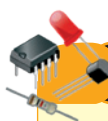


Fig. 12

LED è acceso a segnalare la condizione di sistema "armato"; in caso di superamento della soglia di allarme, il relè si attiva ed il buzzer emette la serie di note impostate nello sketch. Attivando "Alarm Pause", il LED lampeggia, il buzzer smette di suonare ed il relè resta attivato. Con "Alarm On" e "Alarm Pause attivato" il LED lampeggia; in caso di superamento della soglia di allarme il relè si attiva ma il buzzer non suona; al ridiscendere dei consumi sotto la soglia di intervento dell'allarme l'"alarm Pause" si disattiva e rimane attivato "Alarm On", il LED è acceso. ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Il Power Meter Shield è disponibile anche in scatola di montaggio al prezzo di 9,50 Euro (cod. FT1046K); il kit comprende tutti i componenti, le minuterie e la basetta forata e serigrafata. Il monitor conta energia è disponibile presso Futura Elettronica (cod. FR491, Euro 99,00) così come la board Arduino Uno rev. 3 (Euro 24,50) ed il kit dell'Ethernet Shield (cod. ETHSHIELDKIT, Euro 24,00). Tutti i prezzi si intendono IVA compresa.

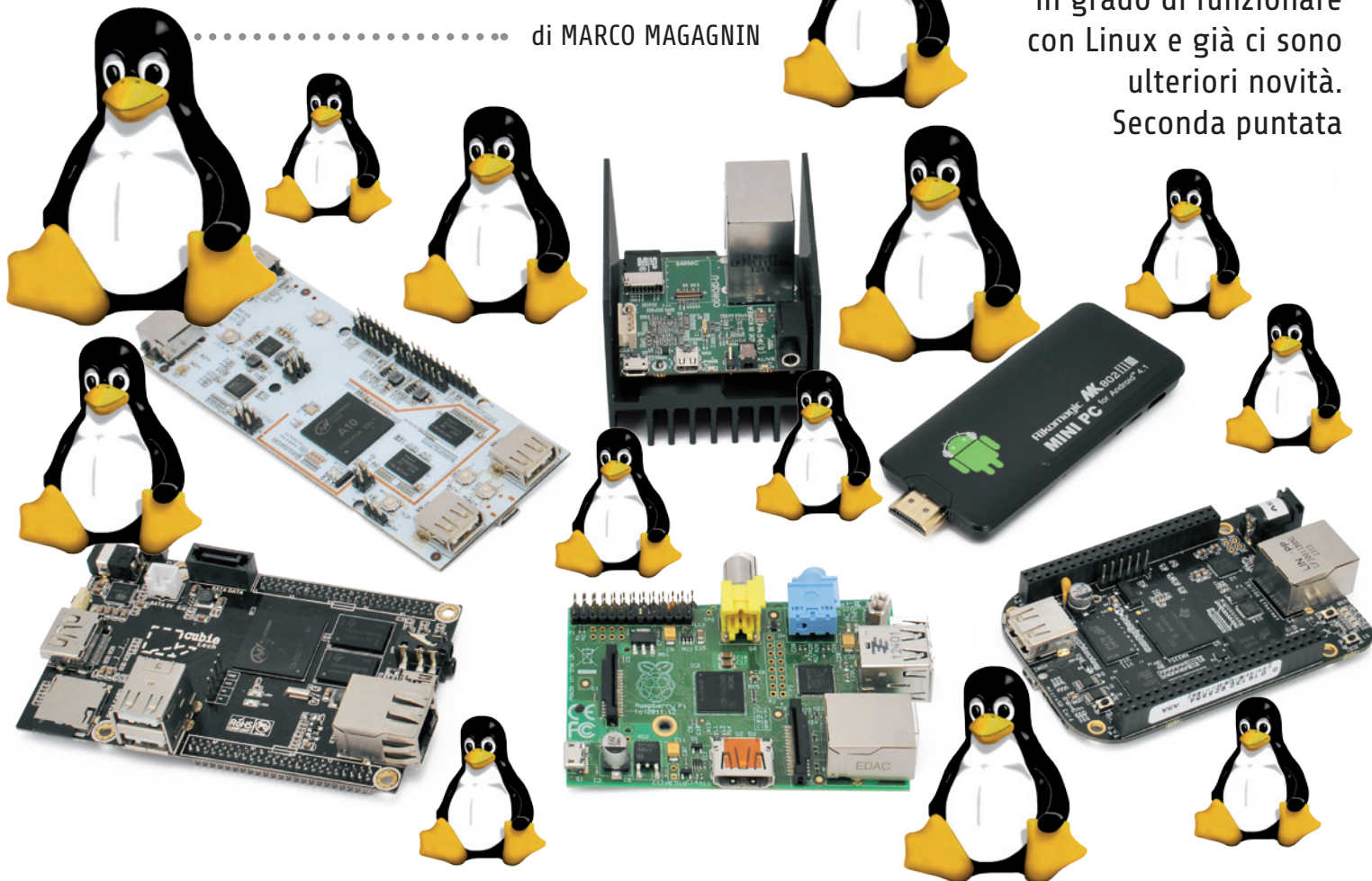
Il materiale va richiesto a:

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

LA CARICA DEI PINGUINI

di MARCO MAGAGNIN

Prosegue la panoramica sulle schede embedd in grado di funzionare con Linux e già ci sono ulteriori novità. Seconda puntata



Come avevamo previsto, nel tempo trascorso fra i due numeri della rivista è comparso, tra gli altri, un nuovo attore sul mercato dei micro PC compatibili con GNU/Linux. Si tratta di Cubieboard2: la nuova versione di Cubieboard "motorizzata" dal SoC Allwinner A20, il primo dual core della categoria con GPIO. La nuova scheda è compatibile con la precedente e quindi dovrebbe valere tutto ciò che abbiamo scritto riguardo a Cubieboard. Il SoC

Allwinner A20 è composto dai chip CPU Cortex A7 dual core e GPU MALI 400 MP2. Il SoC è pin compatibile con il predecessore A10. In attesa di provarla e applicarla in qualche progetto da presentarvi continuiamo con la panoramica sui dispositivi rimanenti.

BeagleBone Black

BeagleBone Black ci è stata consegnata poco prima di scrivere questo articolo in quanto le consegne sono contingentate e prevedono

lunghi tempi d'attesa: a significare che il mercato dei computer su scheda dal costo contenuto è in forte espansione con dinamiche difficilmente prevedibili anche da parte dei produttori stessi. In questo mercato, inoltre, mancano i segnali di una standardizzazione, soprattutto dal punto di vista dei SoC. BeagleBone Black, infatti, adotta sempre una CPU ARM Cortex A8 da 1 GHz di clock, ma all'interno di un SoC Sitara AM335x di Texas Instruments. Anche l'architettura

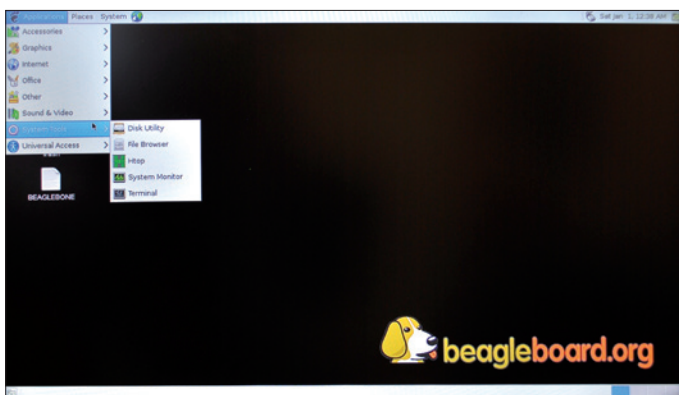


Fig. 1

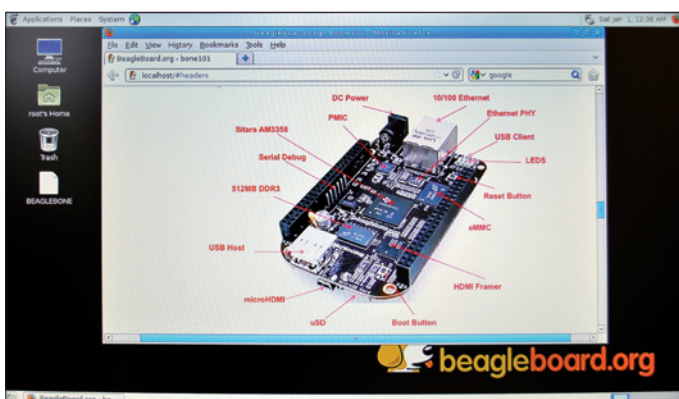


Fig. 2

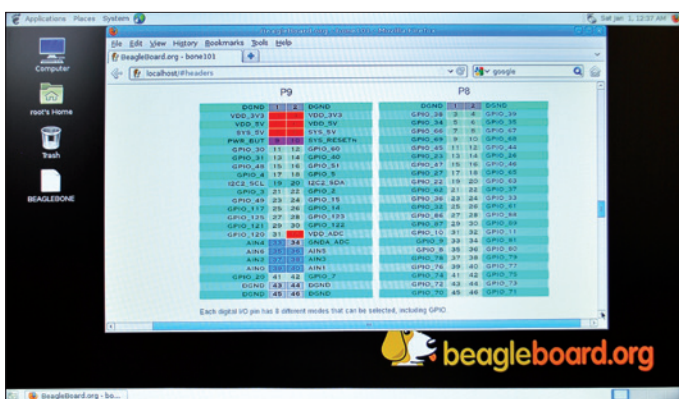
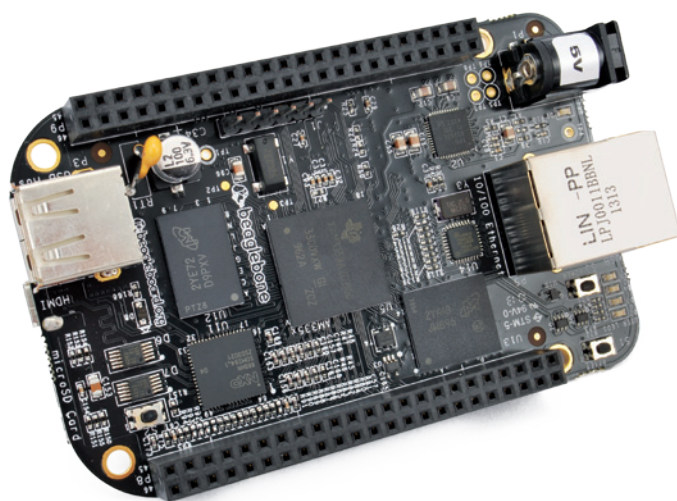


Fig. 3



Fig. 4



software di supporto si discosta dalle “tendenze”: la scheda BeagleBone Black, dotata di 2GB di memoria di massa on board, viene fornita con preinstallata la distribuzione di GNU/Linux Ångström. Lo slot per la scheda microSD può essere utilizzato per aumentare la memoria di massa disponibile o per installare un sistema operativo alternativo. La RAM è di 512 MB DDR3L. Sono presenti due prese USB delle quali una host, un connettore per il debug seriale, la presa Ethernet, l’uscita video HDMI, la presa di alimentazione e il pulsante di accensione/spengimento. In aggiunta a queste interfacce troviamo i due connettori femmina, ciascuno da 44 pin a passo 2,54 mm che mettono a disposizione:

- 65 ingressi/uscite digitali;
- 7 ingressi analogici;
- 4 porte seriali;
- 2 bus SPI;
- 2 bus I2C;
- 8 uscite PWM;
- 4 Timer;
- ed altro che scopriremo man mano.

All’accensione il sistema operativo esegue il boot in

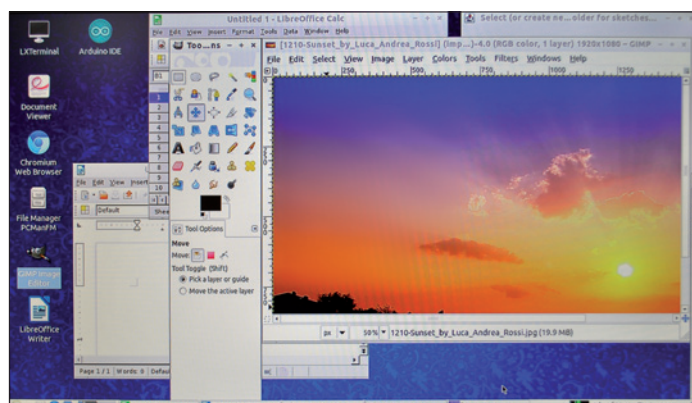


Fig. 5

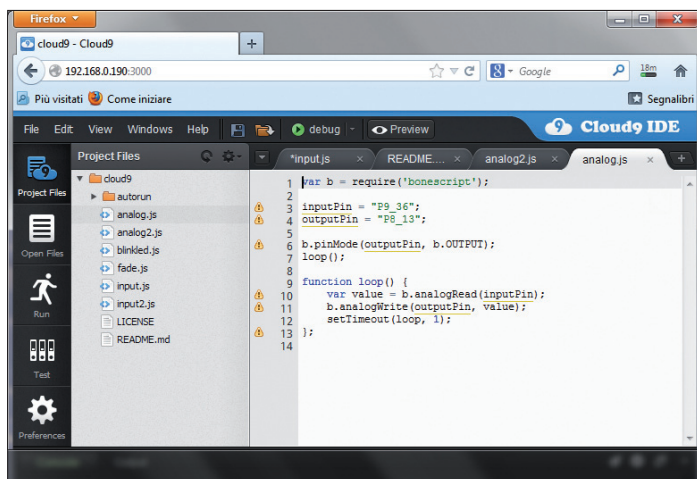


Fig. 6

pochi secondi mettendo a disposizione la schermata d'apertura (Fig. 1). Sono già disponibili una serie di applicazioni come gli strumenti di Libreoffice, Gimp, e gli strumenti di amministrazione (Fig. 2, 3, 4 e 5). L'approccio adottato dai produttori di BeagleBone è di farne uno strumento didattico, sulla scia dell'esperienza di Arduino. Alla prima accensione è infatti già disponibile un sito web locale che contiene la documentazione della scheda e l'ambiente di sviluppo Cloud9, anch'esso basato su web (Fig. 6); questo mette a disposizione il linguaggio di sviluppo, Bonescript, che rispecchia la logica di Arduino, la cui documentazione, come visibile in Fig. 7 è disponibile nel web interno. Essendo arrivata da poco in Redazione ci fermiamo qui, con la promessa che approfondiremo la disponibilità di sistemi operativi e modalità di accesso alla interessante struttura del GPIO e delle periferiche attestate sui connettori.

MK802 E MK802II

A questo punto descriviamo dispositivi che stanno al di là della linea di demarcazione ideale che abbiamo fissato tra dispositivi che mettono a disposizione i terminali GPIO - mediante appositi connettori - e quelli che non dispongono di GPIO. I dispositivi della serie MK802 di Rikomagic appartengono alla categoria senza GPIO. In particolare i PC stick MK802 e MK802II sono già stati descritti approfonditamente nel numero 175 della rivista. Le versioni di GNU/Linux disponibili per questi dispositivi sono stabili e permettono di realizzare applicazioni anche impegnative come server web, server database, streaming di immagini da webcam, access point WiFi, router e firewall, tanto per rimanere nel campo dell'embedded. Ovviamente possono reggere un'applicazione come quella descritta nel numero 176 della rivista per gestire la stampante 3Drag in remoto. In fondo anche queste "chiavette" hanno a bordo un SoC Allwinner A10 che mette a disposizione prestazioni uguali a pcDuino e



Fig. 7

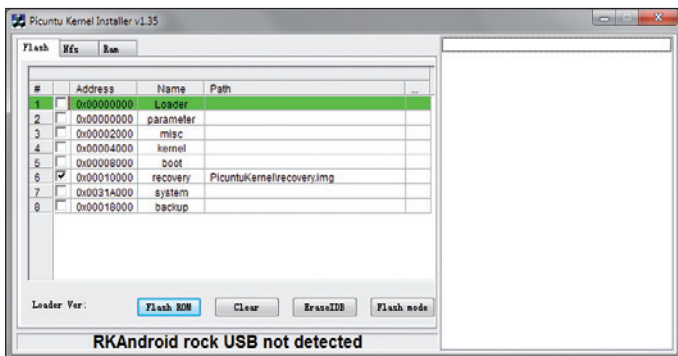


Fig. 8

Cubieboard. Abbiamo provato a inserire sull'MK802 una SD Card con montato BerryBoot e Raspian - creata con la CubieBoard - ed ha funzionato tranquillamente, mentre così non è stato per la MK802II che richiede un'immagine di boot dedicata. Come già detto, disponendo di una rete WiFi con protezione WAP, potete provare a utilizzare BerryBoot direttamente dalle MK802, con il procedimento descritto per CubieBoard. Entrambi i dispositivi funzionano correttamente sia utilizzando il WiFi interno, sia collegandosi alla rete LAN Ethernet con un convertitore da Ethernet a USB come quello disponibile da Futura Elettronica con il codice 8220-PCUSBLAN.

MK802III E MK802IIIS

Le nuove versioni di MK802, la III e la IIIS, hanno

sostituito le precedenti nella disponibilità commerciale. L'incremento di prestazioni di queste "chiavette" rispetto alla serie precedente è più che notevole. Stiamo infatti superando un'altra linea di demarcazione: quella che divide i dispositivi mono processore da quelli multi processore. Il SoC Rockchip RK3066 ospita al suo interno un dual core ARM Cortex A9 in grado di funzionare fino ad una frequenza di clock di 1,6 GHz. Nascono con Android 4.1 mentre per quanto riguarda GNU/Linux è disponibile una distribuzione di PicUntu (sempre della famiglia Ubuntu, derivata da Debian) ormai stabile, anche se ci costringe a fare i conti con quanto abbiamo affermato all'inizio, ovvero la diversità di composizione dei SoC impiegati nei dispositivi e le modalità di gestione del processo di boot. Il SoC utilizzato nell'MK802III si differenzia da quello a bordo di MK802IIIS per i chip dedicati alla connettività wireless. MK802III offre la sola connettività WiFi utilizzando il chip Realtek RTL8188EUS, mentre MK802IIIS dispone di connettività sia WiFi, sia Bluetooth, impiegando i chip Mediatek MT5931 per il WiFi e MT6622 per Bluetooth. La versione di PicUntu disponibile al momento, la RC3, dispone dei driver solo per il chip RTL8188EUS di MK802III che funziona benissimo in WiFi, mentre per quanto riguarda MK802IIIS bisogna accontentarsi di utilizzare



Fig. 9

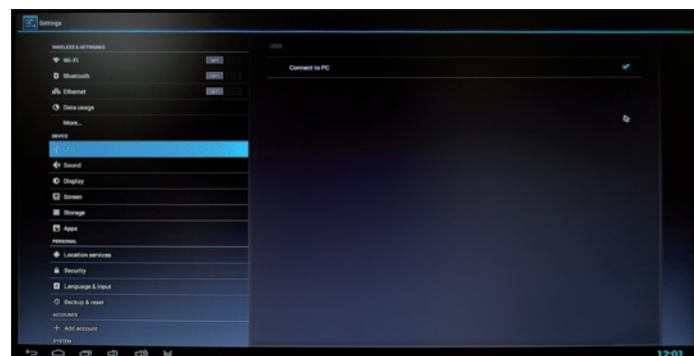


Fig. 10

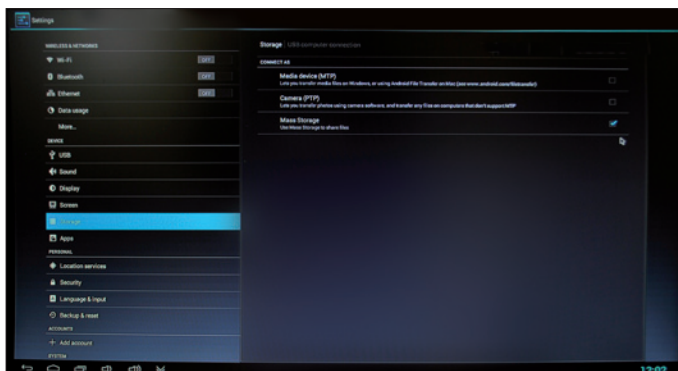


Fig. 11

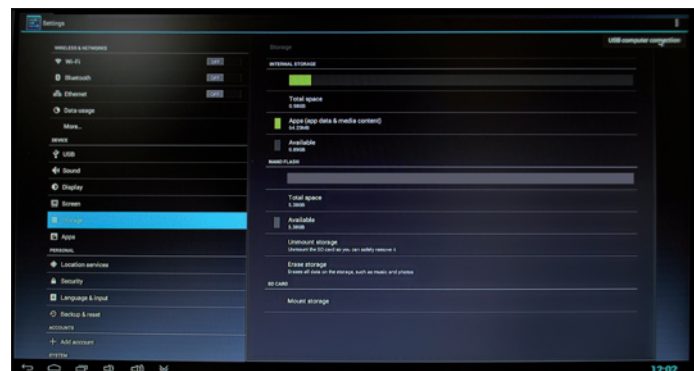


Fig. 12

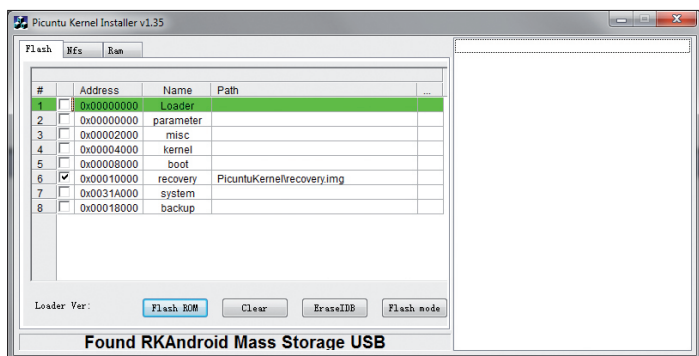


Fig. 13

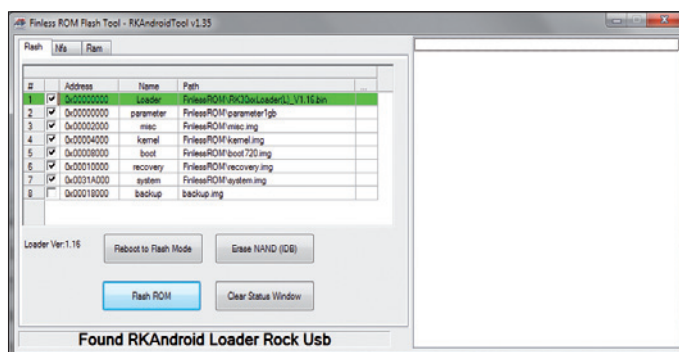


Fig. 14

al momento un dongle esterno. Il collegamento alla rete LAN Ethernet funziona invece perfettamente se si usa un convertitore Ethernet USB come nel caso dei dispositivi MK802(II).

Vediamo velocemente il processo di installazione di Picuntu, valido per entrambi i dispositivi. In questo caso non è sufficiente creare un'immagine su SD Card e inserirla nello slot di MK803III in quanto il dispositivo non è predisposto per il boot diretto da SD Card. MK802III utilizza la memoria interna flash, suddivisa opportunamente in modo da contenere le partizioni di boot, il kernel ed una partizione di recovery. La partizione di recovery serve per poter accedere, con un'apposita procedura, al dispositivo nel caso il sistema operativo principale sia rovinato. Per utilizzare GNU/Linux si utilizza proprio la partizione di recovery, dove si installa il kernel di Picuntu. Dopodiché si accende il dispositivo, si lascia partire Android e da questo si esegue un reboot forzando ad utilizzare la partizione di recovery. È necessario attivare l'utente SuperUser (root), che a sua volta richiede un'opportuna procedura. Procediamo con ordine.

Innanzitutto procuriamoci gli "attrezzi" necessari all'operazione. Si tratta di scaricare alcuni software da installare su PC Windows assieme alla partizione di root del sistema operativo Picuntu.

Cominciamo dall'immagine della partizione di root dall'indirizzo: <http://download.g8.net/picuntu/picuntu-linuxroot-0.9-RC2.2.tgz> Questa contiene solo la parti-

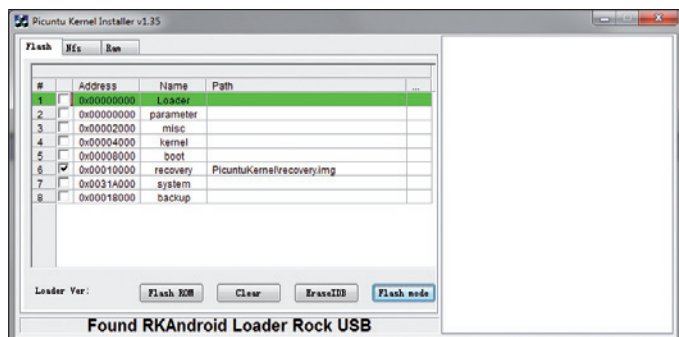


Fig. 15

zione di root del sistema operativo perché il kernel e la partizione di boot dovremo caricarli nella partizione di recovery di MK802III con un apposito software. Una

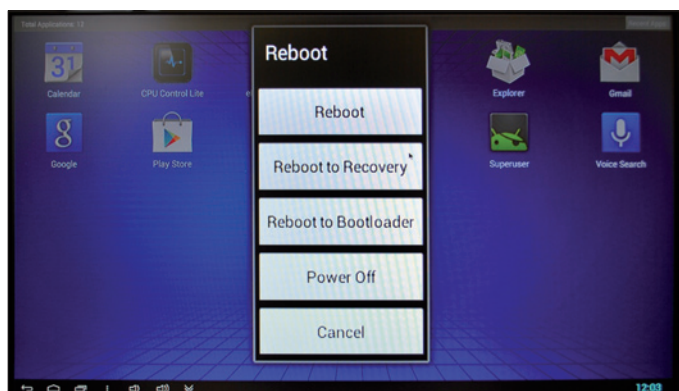


Fig. 16



Fig. 17

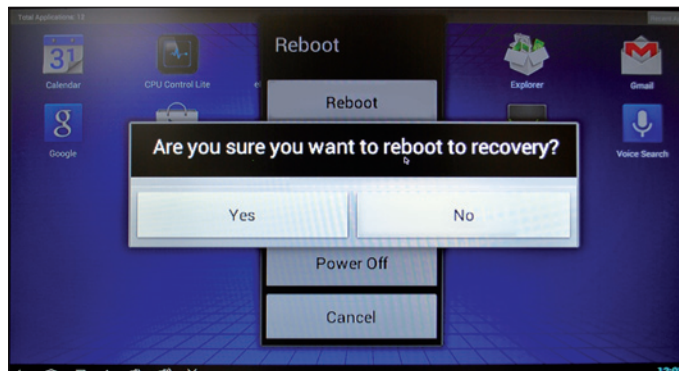


Fig. 18

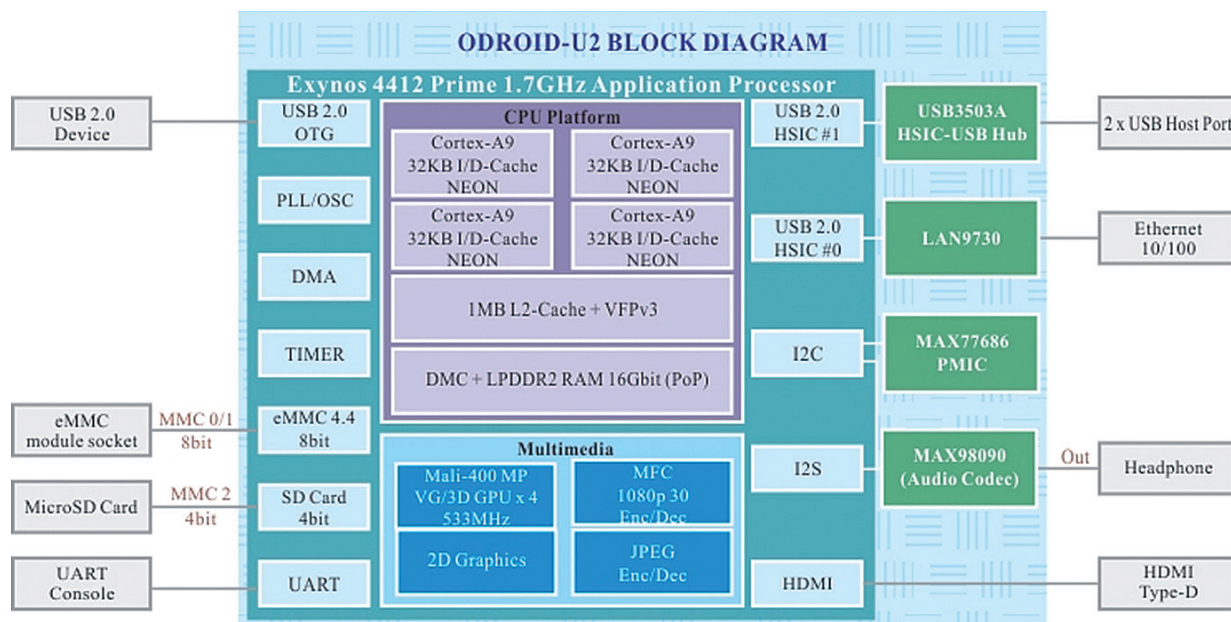


Fig. 23

cercare il driver all'interno del computer. Premete il pulsante "Browse" e nella finestra di dialogo, andate all'interno della cartella dove avete scompattato il file PicuntuKernelInstaller4, aprite la cartella "Driver", selezionate ulteriormente la cartella che corrisponde all'architettura del vostro PC (32 o 64) bit e infine la cartella che corrisponde al vostro sistema operativo. Se vi viene segnalato che il driver non è certificato per Windows, scegliete l'opzione che vi permette di proseguire ugualmente. Al termine un reboot del PC non può fare che bene. Scollegate MK802III, aspettate che il PC si riavvii completamente e poi riconnettetevi MK802III. Ripetete i passi precedenti fino al punto dove avete installato il driver e questa volta dovrete vedere il vostro dispositivo connettersi con il messaggio "RKAndroid Loader Rock USB" (Fig. 14). Ora potete premere il pulsante "Flash ROM". Nel riquadro a destra vi verrà segnalato l'avanzamento del caricamento dell'immagine di boot nella partizione di recovery e poi MK802II eseguirà il reboot ... in Android. Per fare partire GNU/Linux è necessario fare eseguire il boot dalla partizione di recovery e per fare questo il dispositivo deve avere l'utente root abilitato. Se il vostro dispositivo dispone dell'App Reboot o non ha la possibilità di aprire una finestra di terminale, allora dovete necessariamente aggiornare il firmware dell'MK802III. Vi consigliamo di utilizzare la ROM Finless 1,7 che dispone di tutte le feature necessarie. Noi l'abbiamo scaricata dall'indirizzo <https://docs.google.com/file/d/0B9BFwCzfMWZ8SUIHbVdKY2UzZmc/edit>. Le istruzioni per installarla sono in un file di testo all'interno del file compresso. Scompattate il file in una cartella e seguite le istruzioni. All'interno vi sono diverse immagini, sceglietene una senza overclock per

il dispositivo UG802, che è identico a MK802III (Fig. 15). Al termine, andate nelle App di Android, cliccate su Reboot e scegliete Reboot to recovery (Fig. 16, Fig. 17 e Fig. 18). La scheda ripartirà con il sistema operativo GNU/Linux. Le password sono "picuntu" per l'utente "picuntu" e "root" per l'utente "root". Una volta entrati aprite una finestra terminale e andate alla cartella /usr/local/picuntu, da lì lanciate il comando ./picuntu-da-server.sh, vi apparirà uno strumento di configurazione che vi permetterà di configurare diverse opzioni di Picuntu. Come dispositivo selezionate "UG802", poi il nostro consiglio è di installare GNOME come interfaccia grafica e Webmin come strumento di amministrazione remota. Se volete è disponibile un aggiornamento RC3 a Picuntu. Il software e le istruzioni sono disponibili all'indirizzo <https://www.miniand.com/wiki/Appendix+C--Using+PicuntuRC3KernelInstaller>.

Noi, per provare le prestazioni di MK802III, abbiamo installato il software Repetier-Server per gestire in remoto la nostra stampante 3Drag (Fig. 19), che funziona in contemporanea con lo strumento di amministrazione Webmin (Fig. 20, Fig. 21 e Fig. 22) che mette a disposizione via web le funzionalità di gestione e amministrazione per GNU/Linux, come la shell a linea di comando, il file manager con tanto di editor, il gestore dei servizi e molto altro. Ormai è in funzione da settimane pur essendo bistrattata senza pietà: spegnimento alla sera togliendo la corrente a tutto il laboratorio e 10 ore di stampa al giorno. Malgrado tutto funziona perfettamente e con prestazioni di tutto rispetto.

ODROID-U2

Qui siamo veramente su un altro pianeta. Quello che

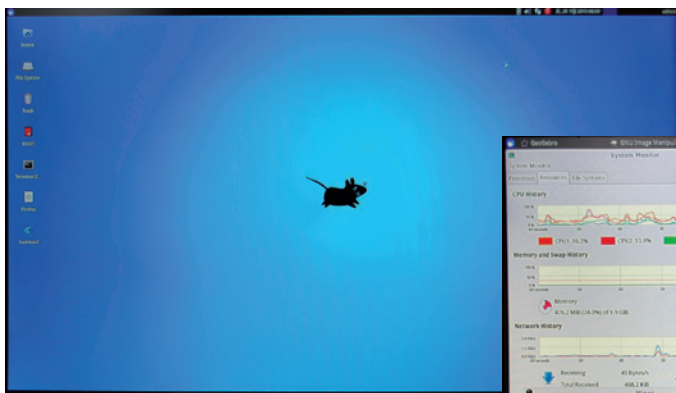


Fig. 24

abbiamo di fronte è un “cubetto” di poco più di 5 cm per lato, per lo più occupati dal radiatore. All’interno del radiatore troviamo un circuito stampato quadrato con a bordo un SoC Samsung Exynos4412 con una CPU con 4

processori a 1.7 GHz, 2 GB di RAM e un processore video Mali400 (**Fig. 23**). Il SoC originariamente è stato progettato per i telefoni cellulari di alta gamma e quindi si è prestata particolare attenzione alle dimensioni e ai consumi. Con il consumo inferiore ai 5 W questo dispositivo offre un rapporto consumo/potenza decisamente impressionante. Siamo di fronte ad un PC vero e proprio, anche se in miniatura: abbiamo una presa Ethernet, due USB, una micro HDMI e il connettore per l’alimentazione. Purtroppo non abbiamo una presa SATA per un disco fisso e nemmeno un USB 3. Il SoC Exynos4 non li prevede, il SoC Exynos5 prevede un connettore SATA, però la CPU è provvista di due soli processori. Questo riconferma ancora una volta il nostro discorso iniziale sulla mancanza di standardizzazione che caratterizza attualmente il mercato.

Speriamo che il successo di questi dispositivi porti i costruttori a convergere verso configurazioni un po’ più “coerenti” rispetto alla situazione attuale, in modo da facilitare anche la vita agli sviluppatori dei sistemi operativi che, al momento, stanno letteralmente impazzendo per seguire tutte le specificità dei singoli prodotti. Per il resto la sintesi di questo dispositivo è presto fatta. Non è un dispositivo strettamente per elettronici però è un mostro di potenza e, da un certo punto di vista, anche di facilità d’uso. Rispetto ad un PC tradizionale, che richiede l’installazione e la configurazione di sistema operativo e applicazioni, nel nostro caso scaricate la distribuzione di GNU/Linux dall’indirizzo http://dn.odroid.com/Ubuntu_U2/20130503/xubuntu-13.04-desktop-armhf-odroidu2_20130503.img.xz, scompattate il file e trasferite l’immagine in una micro SD Card, meglio se da

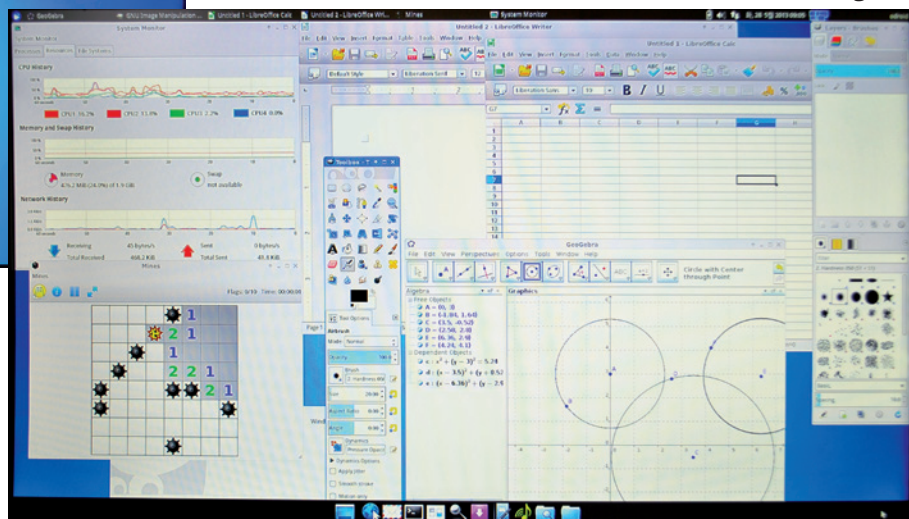
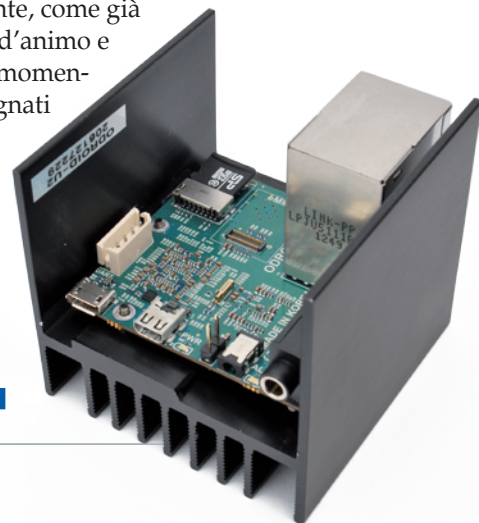


Fig. 25

almeno 8GB e in Classe 10. Al termine inserite la SD Card nell’apposito slot di Odroid U2, date tensione e... caricate e utilizzate quello che volete. Noi abbiamo installato Libre Office, Firefox, Geogebra (il software di simulazione di Geometria e Analisi Matematica). Abbiamo aperto un’infinità di finestre come stress test e il tutto funziona senza alcun problema (**Fig. 24** e **Fig. 25**).

CONCLUSIONI

Per quanto riguarda le considerazioni finali su questa panoramica confermiamo ciò che abbiamo scritto al termine della prima puntata. A riprova della continua evoluzione citiamo la comparsa della Cubieboard2, che alza l’asticella delle prestazioni per le schede dotate di connettori GPIO mentre la disponibilità di mostri come l’ODROID U2 ci permette di pensare ad applicazioni impegnative dal punto di vista delle prestazioni e delle risorse come l’elaborazione di immagini, il calcolo scientifico e la simulazione in campo elettronico ed ingegneristico. Intanto, nei prossimi mesi, sicuramente qualche novità tenterà di sorprenderci ancora. L’importante, come già detto, è di non perdersi d’animo e continuare a studiare. I momenti di crisi sono accompagnati anche dalle opportunità derivanti dalle nuove tecnologie e innovazioni. Piuttosto che aspettare il futuro è meglio anticiparlo e prepararsi a viverlo da protagonisti e non da semplici spettatori. ■

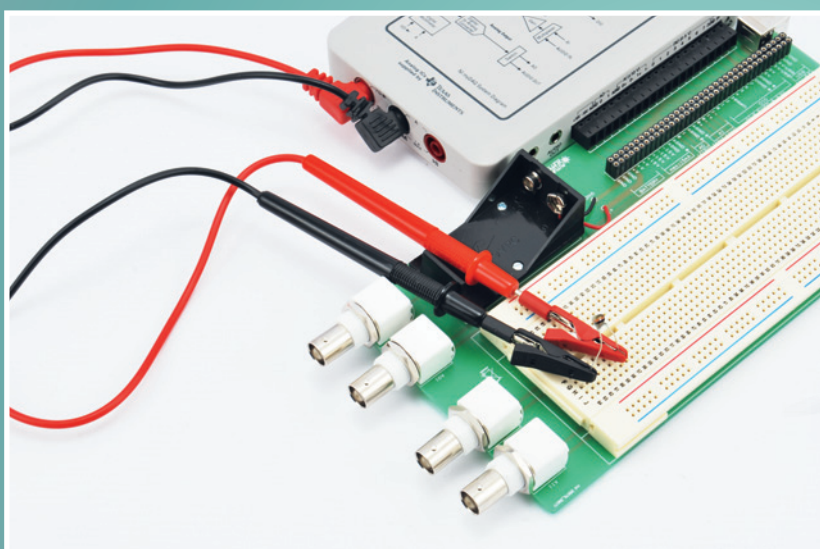


myKIT

MISURA DI TEMPERATURA CON TERMISTORE

..... di DAVIDE SCULLINO

Finalizzato a verificare in pratica i concetti teorici appresi, questo esperimento - mediante NI myDAQ e LabVIEW - si propone di misurare e tenere sotto controllo la temperatura ambiente utilizzando come sensore un termistore di tipo NTC.



Lo strumento NI myDAQ è un dispositivo di acquisizione da collegare al Personal Computer, prezioso e particolarmente utile per leggere segnali analogici o per generarli e che si presta all'esecuzione di svariate misure elettroniche.

Per fare un po' di pratica con NI myDAQ e testare sul campo le possibilità offerte, può essere utile impiegare myKIT, il set per esperimenti descritto nel fascicolo n. 171, composto dalla protoboard (una scheda elettronica pensata per essere abbinata a NI myDAQ contenente una breadboard - ovvero una basetta sperimentale con la quale risulta molto semplice e veloce realizzare montaggi di circuiti elettronici senza dover ricorrere alla saldatura - e alcuni connettori BNC più una presa

USB) nonché i componenti con cui realizzare una serie completa di esercitazioni che fanno uso di NI myDAQ e del software LabVIEW.

Per compiere gli esperimenti basta un PC sul quale caricare LabVIEW e i relativi VI, NI myDAQ connesso al PC mediante un cavo USB e la protoboard. Per l'alimentazione del circuito in prova possono essere utilizzate le tensioni disponibili sul connettore di NI myDAQ (5V e $\pm 15V$), ma anche altre sorgenti (batteria e +5V proveniente da una seconda presa USB), da utilizzare nel caso servano correnti di maggior

Fig. 1 - Il termistore Vishay utilizzato in questa esercitazione.



intensità. Usando myKIT e la scheda myDAQ, sotto la guida delle dettagliate dispense didattiche, potete testare in pratica programmi realizzati con LabVIEW per acquisire dati da svariati tipi di sensore, gestire trasduttori, analizzare il funzionamento di circuiti come i filtri elettronici e tanto altro ancora. Nel fascicolo n° 172 abbiamo descritto il primo esperimento della serie, spiegando come utilizzare la protoboard insieme ad NI myDAQ per compiere vari tipi di misure e simulazioni su circuiti analogici e digitali. Mantenendo l'impegno di proporre in ogni fascicolo un esperimento di quelli contemplati dal programma didattico di myKIT, presentiamo in queste pagine un'altra esercitazione, finalizzata a misurare la temperatura mediante un termistore di tipo NTC (*Negative Temperature Coefficient*) e visualizzarne il valore in un Pannello Frontale in ambiente LabVIEW, sfruttando un VI che verrà appositamente creato con le indicazioni fornite in queste pagine. In breve, questa esercitazione descrive come effettuare una misura di temperatura (ambiente, per esempio) utilizzando un termistore collegato ad NI myDAQ. I dati forniti da quest'ultimo saranno acquisiti tramite *DAQ Assistant* (che viene installato in LabVIEW con il driver NI DAQmx) e convertito in una misura di temperatura utilizzando la programmazione in LabVIEW. Il termistore che verrà utilizzato è un NTC di Vishay da 10 kΩ e per effettuare la misurazione della temperatura si dovrà collegare ai terminali DMM (Digital Multimeter) della scheda myDAQ.

REALIZZARE IL CIRCUITO DI MISURA

Il termistore è cablo nel circuito come una resistenza;

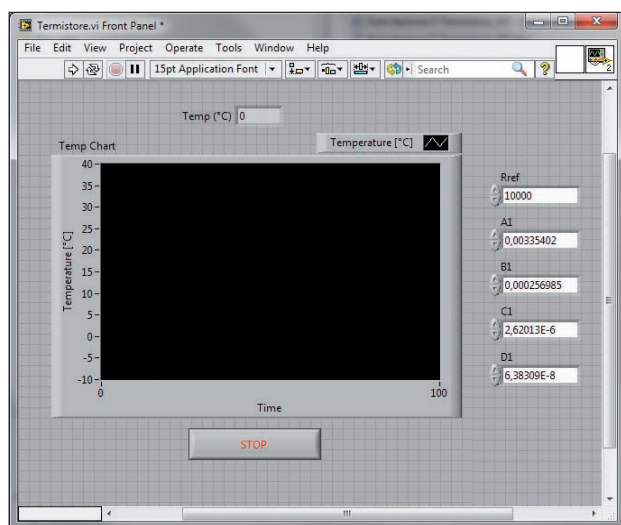
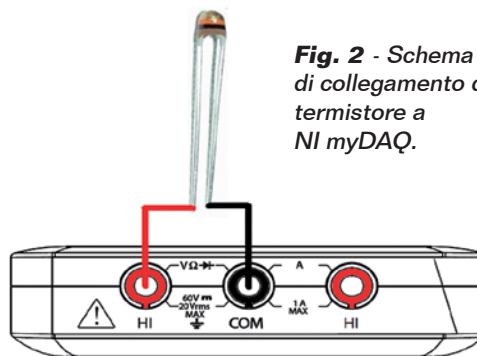


Fig. 3 - Pannello Frontale in LabVIEW.

Fig. 2 - Schema di collegamento del termistore a NI myDAQ.



per collegarlo, teoricamente non sarebbe necessario utilizzare la protoboard, ma in realtà, siccome tutti gli esperimenti prevedono che NI myDAQ sia inserito nel connettore laterale della Protoboard, è opportuno adoperarla, se non altro perché realizza la connessione dati con NI myDAQ sfruttando tale connessione laterale e l'apposito connettore USB di cui dispone. Su NI myDAQ bisogna disporre di un ingresso positivo (HI) su un lato e un ingresso negativo (COM) sull'altro lato; l'orientamento non è importante, dato che il termistore, come tutti i resistori, non ha polarità. Il cablaggio deve essere effettuato come mostrato nella Fig. 2, dove vedete che l'NTC è stato connesso tra un input HI ed il comune (COM) che poi rappresenta la massa di riferimento della misura analogica. Notate che l'HI utilizzato è quello relativo alle misure VW del Digital Multimeter; non bisogna utilizzare l'HI relativo alle misure amperometriche. Con questa connessione

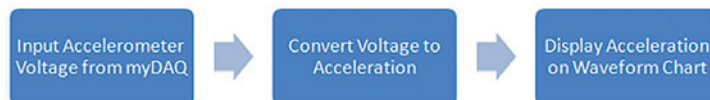


Fig. 4 - Schema a blocchi di codifica.

si sfruttano myDAQ ed il Personal Computer per misurare una generica resistenza, che in questo caso è quella dell'NTC.

COS'È UN TERMISTORE?

Il termistore è un sensore di temperatura basato su un resistore la cui resistenza varia (in un certo intervallo) linearmente al variare della temperatura; è tipicamente costituito da un ossido metallico semiconduttore depositato su un supporto variamente sagomato ed incapsulato in una resina epossidica che ha lo scopo di passarlo, ovvero impedire che gli agenti atmosferici ne alterino le caratteristiche fisiche ed elettriche. In questa esercitazione verrà utilizzato il termistore NTCLE-100E-3103 Vishay da 10 kΩ, che è un elemento a basso costo ma caratterizzato da un'elevata precisione, utilizzabile in una vasta gamma di applicazioni in ambienti sensibili alla temperatura. Il termistore di cui ci avvaliamo è un NTC, cioè presenta un coefficiente

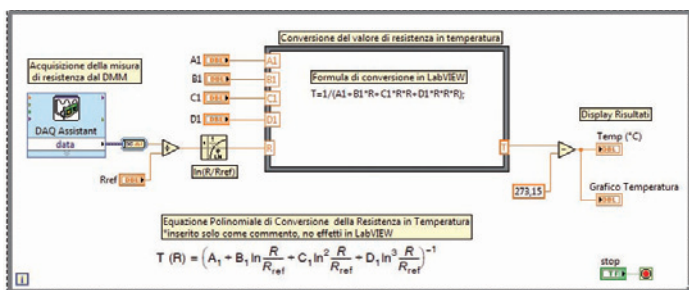


Fig. 5 - Diagramma a blocchi LabVIEW.

di temperatura negativo; questo significa che la resistenza elettrica misurabile fra i suoi estremi decresce al salire della temperatura e aumenta al decrescere della temperatura. Va comunque precisato che la variazione segue una relazione di proporzionalità diretta, sebbene essa non sia lineare, almeno non in tutto il range di temperatura misurabile.

In linea generale, per rilevare la temperatura si fa scorrere nel termistore una corrente costante di valore noto e si legge la caduta di tensione ai suoi capi. Utilizzando la Legge di Ohm, la tensione e la corrente permettono di determinare la resistenza assunta, che varia con la temperatura. Il valore di resistenza viene quindi utilizzato per calcolare la temperatura in Kelvin [K] utilizzando la seguente equazione detta Equazione di Steinhart-Hart.

$$T(R) = \left(A_1 + B_1 \ln \frac{R}{R_{ref}} + C_1 \ln^2 \frac{R}{R_{ref}} + D_1 \ln^3 \frac{R}{R_{ref}} \right)^{-1}$$

PANNELLO FRONTALE LABVIEW

L'interfaccia utente che bisogna creare in questa esercitazione mostra i valori della temperatura rilevata attraverso l'NTC e NI myDAQ, riportati su un grafico e su un indicatore numerico. Il grafico delle temperature memorizza i valori precedenti in modo che si possano vedere le variazioni di temperatura nel tempo. Inoltre sono presenti dei controlli numerici per inserire la resistenza di riferimento Rref (nel nostro caso 10 kΩ) e i parametri A1, B1, C1, D1 del termistore - forniti dal costruttore - necessari per risolvere l'equazione di Steinhart-Hart che vi permetterà di ricavare il valore di temperatura espresso in Kelvin [K]. Dovrete impostare questi valori mediante dei controlli invece che tramite delle costanti, perché così facendo il software si potrà utilizzare anche per termistori con specifiche differenti.

CODIFICA DELLA STRATEGIA

In LabVIEW dobbiamo misurare il segnale proveniente dal termistore, la cui resistenza spazia da 100 Ω a 100 kΩ. Questo valore è convertito in una temperatura Kelvin [K] utilizzando l'equazione polinomiale di Steinhart-Hart. Infine, il codice LabVIEW fornirà in uscita il risultato a un indicatore numerico e ad un

grafico della temperatura espressa in gradi Celsius [°C] sul Pannello Frontale.

Come mostrato nella Fig. 5, il diagramma a blocchi di LabVIEW è molto simile allo schema a blocchi di codifica illustrato nella Fig. 4.

COME FUNZIONA

Con riferimento al diagramma a blocchi LabVIEW, all'interno del ciclo While, sulla sinistra è situato DAQ Assistant configurato per acquisire un singolo valore dai terminali DMM di NI myDAQ ogni volta che viene eseguito. Il valore acquisito, prima di essere passato nel Formula Node viene diviso per la resistenza di riferimento Rref; successivamente viene calcolato il logaritmo naturale del quoziente. Una volta passato il valore di resistenza nel Formula Node, viene convertito in temperatura espressa in Kelvin usando l'equazione polinomiale di Steinhart-Hart. Successivamente la temperatura è convertita in gradi Celsius per essere visualizzata sull'indicatore Temp (Celsius) e sul grafico del pannello frontale. Tutto il codice presente nel ciclo While continua a funzionare fino a che non viene premuto il pulsante Stop sul Pannello Frontale. Tenete presente che le misure effettuate tramite gli ingressi DMM sono temporizzate dal software, quindi la velocità di aggiornamento è influenzata dal carico sul computer e dall'attività sull'USB.

COSTRUZIONE DEL PANNELLO FRONTALE

Sul Pannello Frontale, cliccate con il pulsante destro del mouse e comparirà la palette Controls, dove sono elencati tutti i tipi di controllo e di indicatore, ma anche decorazioni, tabelle, ecc. Se cliccate sulla puntina in alto a sinistra della palette potete bloccarla sullo

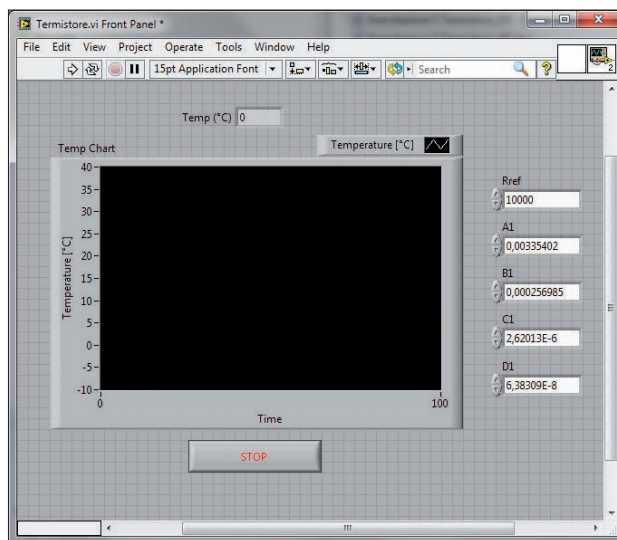


Fig. 6 - Pannello Frontale LabVIEW completato.

schermo; ciò vi risparmierà di dover fare clic con il pulsante destro del mouse sul Pannello Frontale ogni volta che dovete farla riaprire.

Ora estendete la voce *Modern* (la prima) della palette *Controls*, selezionate *Boolean* e inserite un pulsante *Stop* sul pannello frontale. Sempre alla voce *Modern*, selezionate *Numeric* e inserite un indicatore numerico e cinque controlli numerici. Infine, sempre alla medesima voce, selezionate *Graph* e inserite un *Waveform Chart*. Usando lo strumento *Positioning*, disponete i vari oggetti come in **Fig. 6** e con lo strumento *Labeling* rinominateli, incluso l'asse y del grafico (**Fig. 6**). In ultimo cliccate con il pulsante destro del mouse in corrispondenza dell'etichetta dell'asse y del grafico e deselectionate la voce *Autoscale y*, quindi con lo strumento *Labeling* inserite agli estremi dell'asse il valore massimo e minimo di temperatura che potreste misurare (nel vostro caso l'intervallo -10, +40 è più che sufficiente). Adesso potete già inserire i valori nei controlli numerici utilizzando lo strumento *Labeling*. Questi valori sono quelli utilizzati nell'equazione di Steinhart-Hart per calcolare la temperatura e sono forniti dal costruttore del termistore nel datasheet. Voi inserirete quelli per il termistore Vishay NTCLE-100E-3103 10k Ω che sono:

Ref=10000

A1=0,00335402

B1=0,000256985

C1=2,62013E-6

D1=6,38309E-8

Ora andate su edit nella barra menu in alto e selezionate *Make Current Values Default* per non dover inserire nuovamente questi valori ogni qualvolta aprirete e utilizzerete questo VI (ovviamente se utilizzerete termistori diversi con parametri diversi dovrete inserire i valori corretti e ripetete questa operazione).

COSTRUZIONE DEL DIAGRAMMA A BLOCCHI

Ora implementerete il diagramma a blocchi.

Se non lo visualizzate, potete farlo con il comando *Window>Show Block Diagram* o digitando <Ctrl+E>. Gli oggetti del pannello frontale appaiono come terminali sul diagramma a blocchi. I terminali sono porte d'ingresso e di uscita che scambiano informazioni tra il pannello frontale e il diagramma a blocchi.

Qui dovrete già visualizzare i controlli numerici e il pulsante *Stop* con le stesse etichette che avevate messo sul Pannello Frontale.

Inoltre dovrete visualizzare i terminali, ad esempio quello del pulsante *Stop*, in questo modo: , ma è preferibile poterlo visualizzare con questa icona  per

Tabella 1

Tipo di collegamento	Scalare	Array 1D	Array 2D	Colore
Numerico	 	 	 	Arancione (virgola mobile), Blu (intero)
Booleano				Verde
Stringa				Rosa

ridurre lo spazio che occupa sul diagramma a blocchi. Per fare ciò, impartite il comando *Tools»Options..* e sulla finestra che si aprirà selezionate sulla sinistra *Block Diagram*, quindi alla voce *General* disabilitate *Place front panel terminals as icons*.

Ora, prima di iniziare a implementare il diagramma a blocchi, è utile osservare che vi sono due caratteristiche che distinguono un controllo da un indicatore sul diagramma a blocchi. La prima è una freccia sul terminale che indica la direzione del flusso dei dati (i controlli hanno frecce che mostrano il dato partire dal terminale, mentre gli indicatori hanno una freccia che mostra il dato entrarvi) e la seconda è il bordo intorno al terminale (i controlli hanno un bordo più spesso e gli indicatori più sottile). Se volete trasformare un indicatore in un controllo dovete cliccare con il pulsante destro del mouse sopra il rettangolo che rappresenta l'indicatore e selezionare *Change to Control*.

Questa operazione può essere effettuata anche dal Pannello Frontale.

Notate che il pulsante *Stop* è già impostato, nella nostra applicazione, come controllo.

Prestate attenzione ai collegamenti che effettuerete sul diagramma a blocchi: essi permettono di trasferire dati tra oggetti del diagramma a blocchi; ogni collegamento parte da una sorgente di dati e la collega ad uno o più VI e funzioni che leggono i dati stessi. I collegamenti hanno differenti colori, stili e spessori, in funzione del tipo di dati.

Un collegamento spezzato  appare come una linea nera tratteggiata con una X rossa in mezzo. I collegamenti possono rompersi per svariati motivi: ad esempio se si tenta di collegare due oggetti con tipi di dati tra loro incompatibili. La **Tabella 1** illustra i tipi di collegamento più comuni.

I collegamenti possono essere realizzati solamente tra ingressi e uscite che veicolino dati compatibili tra loro. Nel diagramma a blocchi, facendo clic con il pulsante destro del mouse compare la palette *Functions*, che contiene i VI, le funzioni e le costanti da usare per creare il diagramma a blocchi. Se volete bloccare la palette *Functions* sullo schermo, fate clic sulla puntina che si trova in alto a sinistra.

Ora potete iniziare la costruzione del diagramma a blocchi. In questo VI il blocco DAQ Assistant è configurato per l'acquisizione di valori di resistenza dagli ingressi DMM del dispositivo NI myDAQ. Le seguenti

istruzioni vi permettono di effettuare passo-passo la configurazione di *DAQ Assistant*:

1. Assicurarsi che NI myDAQ sia collegato al PC;
2. Premere Ctrl-Spazio per aprire la finestra *Quick Drop* (al primo utilizzo, questa operazione richiederà circa un minuto);
3. Cercare la voce *DAQ Assistant* e fare doppio clic su di essa quando appare nella lista;
4. Trascinare sul diagramma a blocchi (finestra bianca);
5. Quando appare un nuovo riquadro per le attività di configurazione rapida, selezionate:
 - a. *Acquire Signals*;
 - b. *Analog Input*;
 - c. *Resistance*;
 - d. Dev 1 (NI myDAQ) * Nota: Se si dispone di altri hardware NI installati, myDAQ non sarà Dev1;
 - e. dmm;
 - f. *Finish*;
6. Inserire in *Resistance Setup* il corretto *Signal Input Range*:
 - a. Max: 100k;
 - b. Min: 100;
7. Cambiare il *Timing Settings*:
 - a. 1 *Sample (On Demand)*;
8. Fare clic su OK.

La finestra di dialogo per la configurazione di *DAQ Assistant* è visibile nella Fig. 7.

Una volta inserito il blocco *DAQ Assistant*, andate sulla palette *Functions* e da *Programming»Numeric* inserite un blocco *Divide* e uno *Subtract*.

Ora, da *Mathematics»Elementary & Special Functions»Exponential Functions* inserite un blocco *Natural Logarithm* e inserite anche il blocco *Convert from Dynamic Data* situato sulla palette *Functions* alla voce *Express»Signal Manipulation*. Appena inserirete il blocco *Convert from Dynamic Data* si aprirà una finestra per la configurazione; accedendo da *Conversion* selezionate *Single Scalar* e fate clic su OK. Adesso, con lo strumento *Positioning* disponete meglio i blocchi

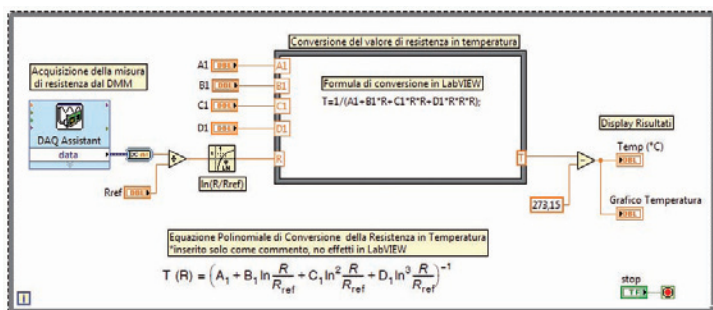


Fig. 8 - Diagramma a blocchi LabVIEW completo e riportante sia la formula di Steinhart-Hart, sia il blocco DAQ Assistant.

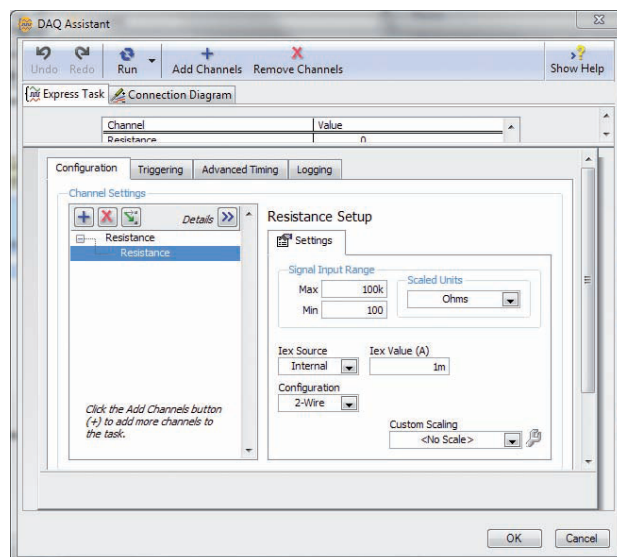


Fig. 7 - Configurazione DAQ Assistant.

appena inseriti e utilizzando lo strumento *Wiring* collegateli tra loro aiutandovi con la Fig. 8. Ora vi occuperete del *Formula Node* dove inserirete l'equazione di Steinhart-Hart. Andate su *Programming»Structures*, selezionate *Formula Node* e disponetelo nel diagramma a blocchi.

Adesso, utilizzando lo strumento *Positioning*, fate clic con il pulsante destro del mouse sul bordo in alto a sinistra del *Formula Node* e selezionate *Add Input*. Comparirà un riquadrino sul bordo del *Formula Node*; con lo strumento *Labeling* assegnategli il nome A1. Ora, con lo strumento *Wiring* collegate il controllo numerico A1 all'ingresso del *Formula Node* appena creato. Ripetete questa operazione altre quattro volte per poter collegare al *Formula Node* i restanti ingressi B1, C1, D1 e R; quest'ultimo andrà collegato all'uscita del blocco *Natural Logarithm* (se avete dubbi, aiutatevi sempre con la Fig. 8).

Questo permette all'espressione che inserirete nel *Formula Node* di avere un riferimento a ogni ingresso (o coefficiente) che utilizzerà. Adesso fate clic con il pulsante destro del mouse sul bordo destro del *Formula Node*, selezionate *Add Output* e nominate questa uscita con "T". Per inserire l'espressione di Steinhart-Hart all'interno del *Formula Node* utilizzate lo strumento *Labeling* e digitate:

$$T=1/(A1+B1*R+C1*R*R+D1*R*R*R);$$

Notate che questa non è esattamente l'equazione di Steinhart-Hart, perché all'esterno del *Formula Node* è già stato calcolato il logaritmo naturale di R/Rref. Se volete visualizzare l'espressione nel centro del *Formula Node* premete più volte Invio per andare a capo di una nuova riga. Come vedete nell'espressione, sono presenti diversi termini che avete già inserito come

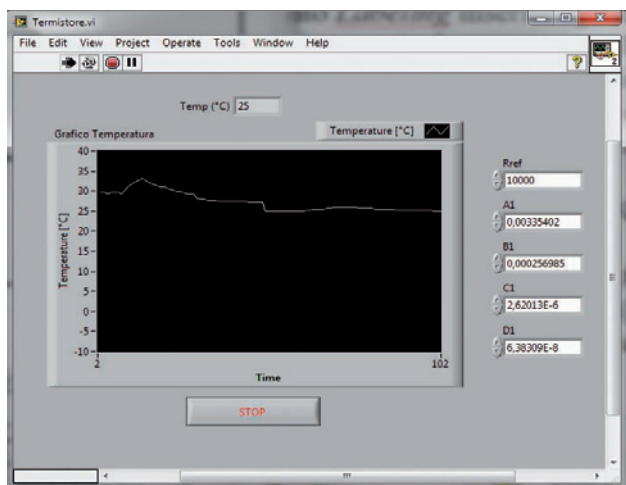


Fig. 9 - Il VI dell'esercitazione in esecuzione.

ingressi sul bordo sinistro del *Formula Node*. Inoltre il risultato, corrispondente alla temperatura rilevata, viene passato come valore in uscita all'output "T" presente sulla destra del *Formula Node*. Ora, con lo strumento *Wiring* collegate l'uscita "T" del *Formula Node* al primo ingresso del blocco *Subtract*, poi in corrispondenza del secondo ingresso del medesimo blocco cliccate con il pulsante destro del mouse, selezionate *Create»Constant* e, nel rettangolino creato scrivete il valore "273,15". Questo permette di trasformare il valore di temperatura da Kelvin a gradi Celsius.

Adesso potete collegare con lo strumento *Wiring* l'uscita del blocco *Subtract* all'indicatore numerico "Temp (°C)" e al grafico "Temp Chart". Per finire, inglobate tutto il diagramma a blocchi fatto finora dentro un *While Loop* che troverete, nella palette *Functions*, alla voce *Programming»Structures*, e collegate con lo strumento *Wiring* il terminale del pulsante *Stop* al *Conditional Terminal* nell'angolo in basso a destra del ciclo *While*.

Potete inserire i commenti nei riquadri a vostro piacimento, utilizzando lo strumento *Labeling*; i commenti non interferiscono con il codice grafico ma ne aumentano la leggibilità. Se preferite cambiare le impostazioni di scrittura (font, dimensione, colore, ecc.), sia nel Pannello Frontale sia nel diagramma a blocchi, nella barra del menu in alto cliccate su *Text Settings* (è il riquadro che riporta scritta la dimensione e il font che state utilizzando: **15pt Application Font**).

Notate che nella **Fig. 5** la formula di Steinhart-Hart non è stata inserita come commento ma come immagine, perché nei commenti non esiste il formato formula; per ottenere la schermata è bastato aprire la prima pagina del file di questa esercitazione in formato pdf, ritagliare la formula e fare un semplice Copia+Incolla incollando il ritaglio nel diagramma a blocchi.

Quando avrete finito di costruire il vostro VI andate su

File»Save as e scegliete un nome e una destinazione per salvare il vostro VI.

Ora cliccate sul pulsante *Run* nella barra degli strumenti, per mettere il VI in esecuzione; la **Fig. 9** mostra la finestra del VI in esecuzione.

Notate che mentre il VI è in esecuzione, l'icona del pulsante *Run* si trasforma nella seguente:

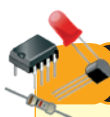
Una volta terminata l'esecuzione, il pulsante *Run* torna al suo stato originale e gli indicatori del Pannello Frontale mantengono i dati visualizzati. Mentre il VI è in esecuzione, appare il pulsante *Abort Execution* (●), facendovi clic potete fermare il VI immediatamente.

Il pulsante *Abort Execution* va usato soltanto se non riuscite ad arrestare il VI in altra maniera.

Notate che se un VI non va in esecuzione, vuol dire che degli errori non lo rendono eseguibile. Se il VI che state creando o modificando contiene errori, il pulsante *Run* appare spezzato (■); in questo caso, anche dopo aver finito i collegamenti nel diagramma a blocchi, il VI non è eseguibile. Generalmente l'errore può consistere in un ingresso richiesto e non collegato o in un collegamento spezzato.

Per risolvere il problema, fate clic sul pulsante *Run* spezzato in modo da accedere alla finestra di dialogo *Error List*, dove viene elencato e descritto ogni errore riscontrato; qui, facendo doppio clic su un errore andate direttamente alla posizione in cui esso si trova nel diagramma a blocchi.

Finisce qui un altro esperimento di quelli facenti parte del programma didattico di myKIT; l'appuntamento è con la prossima esercitazione, nel fascicolo n° 179 di Elettronica In.



per il MATERIALE

Questo progetto fa parte del set di esperimenti didattici realizzato da IRS-ElettronicaIn per la board NI myDAQ con il supporto del software LabVIEW. Il set (cod. SETmyKIT, Euro 232,00) comprende tutti i componenti per realizzare i 22 esperimenti, i VI relativi ed una protoboard. È possibile richiedere protoboard aggiuntive (cod. PROTOBOARDmyDAQ) al prezzo di 52,00 Euro. Tutti i prezzi sono comprensivi di IVA e non comprendono la board NI myDAQ né il software LabVIEW. Futura Elettronica distribuisce questi prodotti presso gli utenti privati mentre la distribuzione presso gli Istituti Scolastici è a cura di IRS srl (Via Vigonovese 81 - 35127 PADOVA - tel: +39 049 8705156 - info@irsweb.it).

Il materiale va richiesto a:

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

RobotHero

Anche per l'ITIS Vito Volterra di San Donà di Piave è arrivato il momento di bilanci; tra le tante iniziative portate avanti quest'anno, una più delle altre è destinata a lasciare il segno.

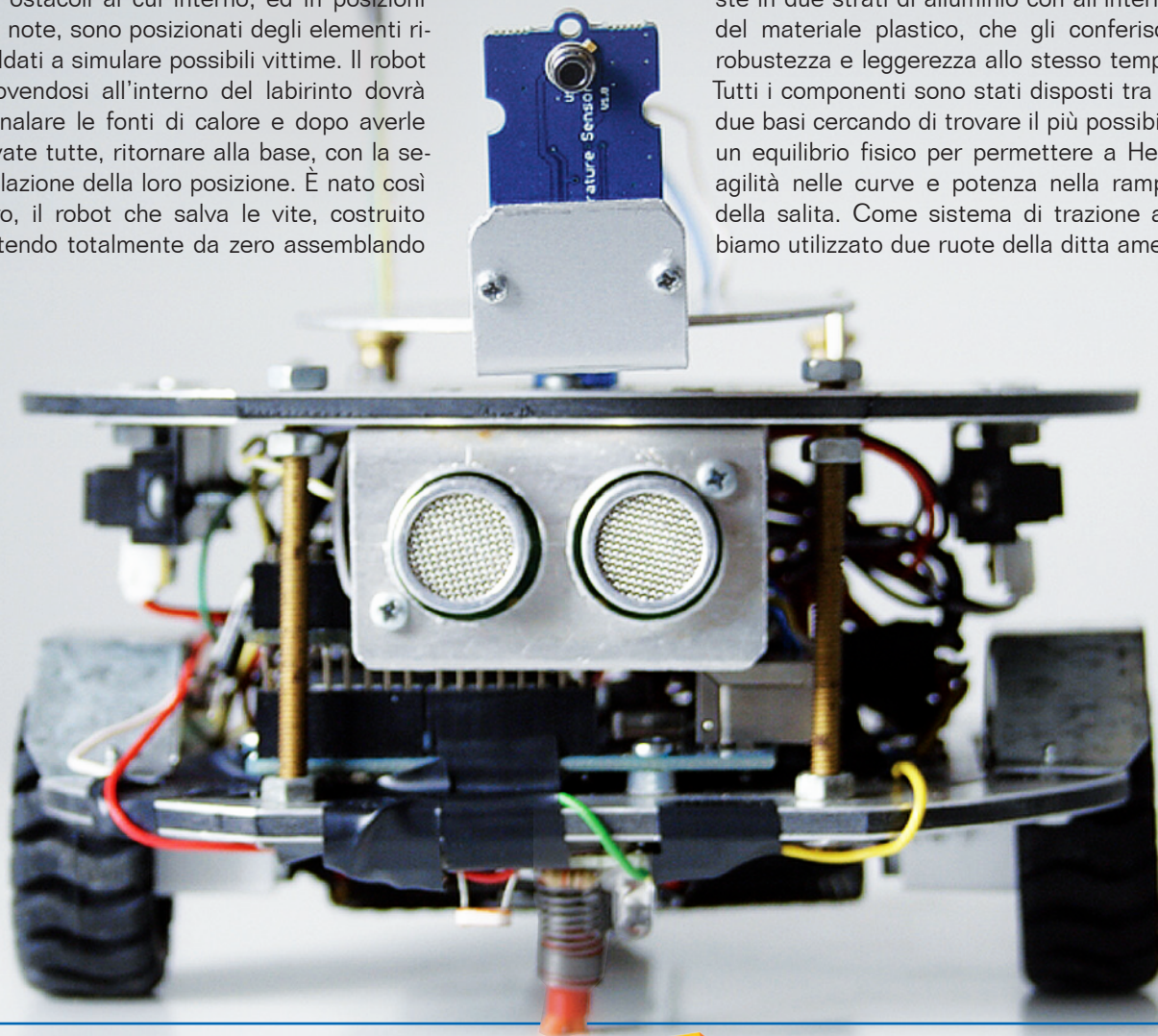
Per la prima volta un team composto da alunni delle classi quarte si è avventurato nella difficile competizione della Robocup Junior categoria Rescue B; il duro lavoro è valso al team il primo posto nei campionati regionali Veneti ed il buon piazzamento ai campionati nazionali. Questa competizione simula uno scenario di post-combattimento in cui un robot deve cercare possibili superstiti e, ritornando alla base, segnalarne la loro posizione. Lo scenario di guerra è rappresentato da un labirinto con ostacoli al cui interno, ed in posizioni non note, sono posizionati degli elementi riscaldati a simulare possibili vittime. Il robot muovendosi all'interno del labirinto dovrà segnalare le fonti di calore e dopo averle trovate tutte, ritornare alla base, con la segnalazione della loro posizione. È nato così Hero, il robot che salva le vite, costruito partendo totalmente da zero assemblando

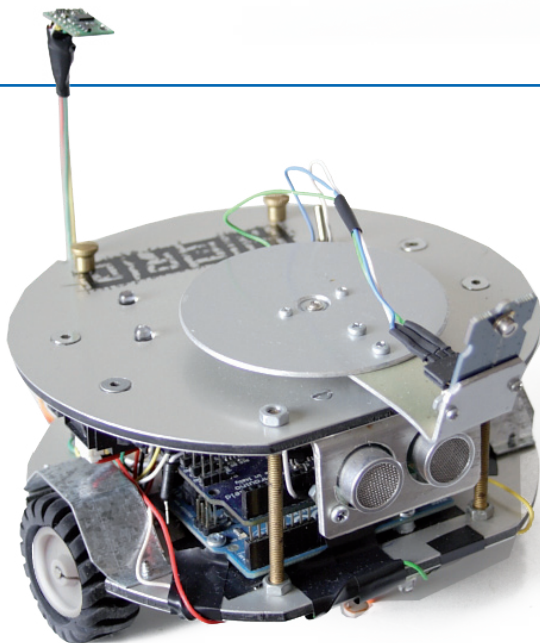
di
Vigani Filippo,
Andrici Silviu,
De Zotti Nicolò,
Franceschetto
Riccardo, Burato
Stefano.

autonomamente sia la parte meccanica, sia la parte elettronica (robot custom). Il nome, "eroe" in italiano, è stato scelto appunto perché il piccolo robot è stato programmato per "salvare vite" e munito di un "coraggio" da vendere.

Struttura meccanica e componentistica

La struttura principale del robot è composta da due speciali lamierini in Alucobond; questo materiale, di ultima generazione, consiste in due strati di alluminio con all'interno del materiale plastico, che gli conferisce robustezza e leggerezza allo stesso tempo. Tutti i componenti sono stati disposti tra le due basi cercando di trovare il più possibile un equilibrio fisico per permettere a Hero agilità nelle curve e potenza nella rampa della salita. Come sistema di trazione abbiamo utilizzato due ruote della ditta ameri-





cana Pololu in gomma morbida ad alto grip di codice RP4219, collegate ciascuna ad un micro-motoriduttore MMG298 sempre di Pololu. Sono state utilizzate solamente due ruote trainanti in modo da permettere una più facile rotazione all'interno del labirinto, a differenza di un sistema di trazione a cingoli, generalmente più idonei a percorsi rettilinei. Un ball-caster è stato usato come terzo appoggio per il robot e, nella parte anteriore anche una molla, con il compito di riportare il robot in assetto durante le frenate.

Per quanto riguarda i sensori, ci siamo serviti di un sensore di distanza ad ultrasuoni SRF05 che permette al robot di rilevare ostacoli frontalmente e di valutarne la distanza; lateralmente, sempre per rilevare la distanza dai muri del labirinto, abbiamo utilizzato due sensori a infrarossi GP2D120. Entrambe le tecnologie usate nei sensori, gli ultrasuoni e i raggi infrarossi, sono risultate valide, con qualche punto in più per il sensore SRF05 che fornisce misure accurate sino a distanze di un solo centimetro. Per simulare la presenza di crateri, durante la gara sono posti in posizione del tutto casuale dei cartoncini di colore nero, rilevati i quali, il robot si deve subito allontanare. È necessario quindi utilizzare un sensore, posto verso il basso, in grado di rilevare il differente colore della superficie di avanzamento. Il primo approccio è stato quello di utilizzare un sensore di linea del tipo a riflessione tipo QTR-1A che però si è rilevato non ottimale a causa del tipo di cartoncino uti-

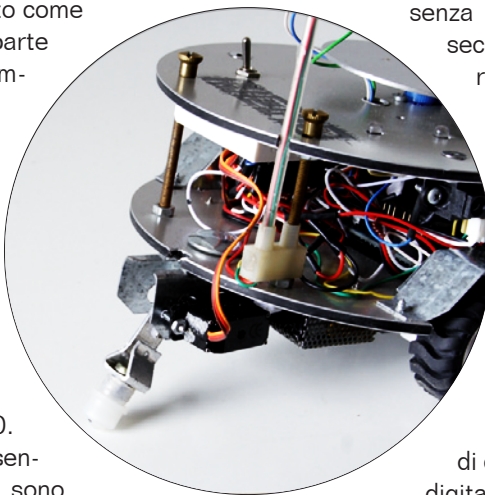
lizzato, troppo riflettente nei confronti dei raggi infrarossi. Ci siamo quindi ingegnati e abbiamo trovato una soluzione che, seppur semplice, è risultata molto efficace. Si è trattato di realizzare un sensore a riflessione utilizzando un LED bianco come sorgente ed una semplice fotoresistenza per captare i raggi riflessi. Per rilevare le fonti di calore è stato utilizzato un sensore in grado di rilevare la temperatura di una superficie a distanza senza contatto, si tratta del sensore OTP-538U disponibile in breakout con il sistema Grove. Hero è stato dotato inoltre di una bussola con accelerometro integrato LSM303DLHC in maniera tale da correggere i movimenti all'interno del labirinto e individuare la salita.

Un servo da modellismo è stato usato per gestire la rotazione della torretta sulla quale era posizionato il sensore di temperatura; in questo modo, ad ogni casella del labirinto è stato possibile rilevare eventuali fonti di calore tutto attorno al robot senza necessariamente farlo girare. Un

secondo servo è stato utilizzato per risolvere i problemi di assetto durante la salita; nel momento in cui Hero affronta la salita questo servo abbassa un ball-caster in modo da sollevare il robot e farlo avanzare in assetto orizzontale anche in salita.

Il controllo di Hero è stato affidato alla piattaforma di sviluppo Arduino UNO: un sistema basato su microcontrollore che permette di realizzare diversi tipi di circuiti elettronici. Possiede 14 pin digitali programmabili come ingressi o uscite (i quali hanno anche la capacità di essere utilizzati per funzioni dedicate come la generazione di segnale PWM o la comunicazione UART) e 6 ingressi per l'acquisizione ed elaborazione di segnali analogici. Dispone di una memoria programma di 32kB, decisamente pochi per la complessità del programma.

Per gestire i motori abbiamo utilizzato uno shield denominato "Dual DC Motor Shield per Arduino" che permette di gestire due motori DC tramite quattro linee di comando due di tipo digitale, per la direzione e due segnali PWM per la velocità. Infine



tutta l'alimentazione è stata fornita da una batteria LiPo da 7,4 volt; queste batterie sono molto leggere, forniscono correnti più elevate e la loro ricarica richiede solo un'ora.

Il software

Il programma è stato strutturato banalmente in modo procedurale, ed è costituito da funzioni ciascuna con il proprio compito, alcune delle quali demandate alle librerie di sistema. Abbiamo lavorato per mantenere il programma il più semplice e lineare possibile, in modo da risultare facilmente modificabile e adattabile alle varie esigenze della gara.

Per poter rilevare più efficacemente le fonti di calore all'interno del labirinto è stato implementato l'algoritmo denominato "Flood Filling Algorithm", ossia "algoritmo di riempimento di un liquido" che consiste nell'uso di vettori di byte contenenti il numero di volte (per ogni cella) sulla quale il robot è passato; questo permette di ottimizzare il percorso affinché ciascuna cella non venga visitata troppe volte. Purtroppo non ci è stato possibile utilizzare un algoritmo più complesso (che avrebbe memorizzato e fatto uso di più informazioni) a causa della scarsa memoria di Arduino.

Il programma è formato da una regione di *setup*, in cui si inizializza il necessario e si calibrano i sensori, e da una regione di *loop* in cui sono implementate le seguenti funzioni:

- controllo vittime e muri
- decisione della via da prendere
- settaggio variabili per il movimento
- movimento

L'algoritmo termina quando, dopo aver visitato tutte le stanze, il robot si trova nella cella di partenza. Come il regolamento chiedeva, siamo stati in grado di far risolvere al robot un labirinto attraversando ogni singola cella e, verificando attraverso un "check", se vi fossero vittime presenti.

Particolare attenzione è stata posta nella gestione dei sensori affinché la lettura risultasse affidabile, anche attraverso una post elaborazione dei dati acquisiti tramite medie matematiche. Il programma seppur strutturato in maniera apparentemente semplice ci ha dato parecchio filo da torce-

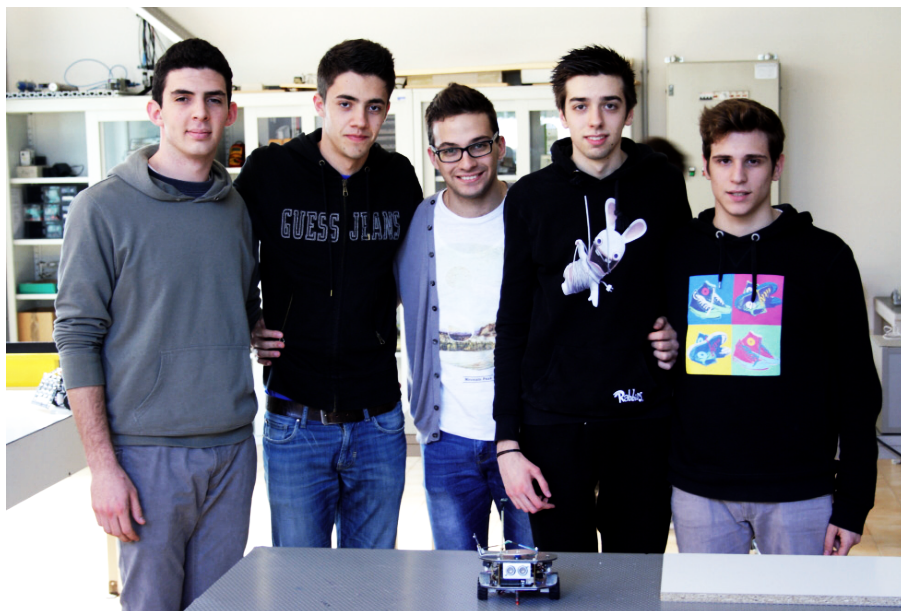
re durante la competizione: per motivi vari è capitato che il piccolo Hero non si comportasse come avrebbe dovuto costringendoci a continui ripensamenti e miglioramenti. Durante la competizione, il problema più grosso è stata la rotazione del robot che doveva risultare esattamente di 90° per poter essere sempre parallelo alle pareti; una piccola deviazione nella traiettoria portava inevitabilmente il robot a non riuscire a raggiungere le celle seguenti; l'ausilio della bussola è stato infatti inficiato dai diversi disturbi presenti sul campo di gara. Nel complesso siamo comunque rimasti soddisfatti della parte hardware e software realizzata.

Osservazioni finali

Hero è motivo d'orgoglio per tutto il nostro team, oltre ad aver sbaragliato i diretti concorrenti nella gara regionale, si è piazzato quinto nella gara nazionale di Pescara. Ma oltre ai risultati ottenuti è motivo d'orgoglio perché rappresenta tutto ciò che abbiamo imparato finora in tutta la nostra carriera scolastica. Siamo riusciti a progettare e a realizzare da zero un intero robot e programmarlo in modo più che appropriato, nonostante le mille difficoltà incontrate lungo il percorso.

La strada da fare è ancora tanta ma Hero continuerà ad evolvere e si cercheranno per il prossimo anno nuove alternative e soluzioni per migliorare ulteriormente le prestazioni del nostro piccolo eroe. □

*Nella foto
gli allievi Viganì
Filippo, Andrici
Silviu, De
Zotti Nicolò,
Franceschetto
Riccardo, Burato
Stefano dell'ITTS
Vito Volterra di San
Donà di Piave che
hanno partecipato
alla RococupJr 2013
RescueB con
il robot Hero.*



DAL 1° AGOSTO DISPONIBILE IL NUOVISSIMO LEGO MINDSTORMS EV3



Presentato oltre un anno fa, dal primo di agosto sarà realmente disponibile la piattaforma Lego Mindstorms EV3 in versione Education, che rappresenta la terza generazione di tecnologie robotiche Lego destinate alle aule scolastiche.

Si tratta di una soluzione per l'insegnamento - sviluppata con educatori qualificati - volta a coinvolgere attivamente gli studenti in diverse aree fondamentali, quali informatica, scienze, tecnologia e matematica, in linea con i curricula nazionali.

La nuova piattaforma consente ora di costruire e programmare un robot completamente funzionante in soli 45 minuti.

La soluzione EV3 comprende:

- attività didattiche personalizzabili in formato digitale;
- una piattaforma hardware basata su tecnologie robotiche del mondo reale;
- una piattaforma software intuitiva, che consiste sia di un ambiente di programmazione che di un sistema di acquisizione, elaborazione e interpretazione dei dati (data-logging), con 48 tutorial "step-by-step";
- materiale di approfondimento per lo sviluppo professionale.

Lego Mindstorms Education EV3 è compatibile con i sistemi operativi iOS e Android, oltre che con le versioni precedenti delle piattaforme Education Min-

Inizia la missione di ATLAS il robot umanoide del DARPA

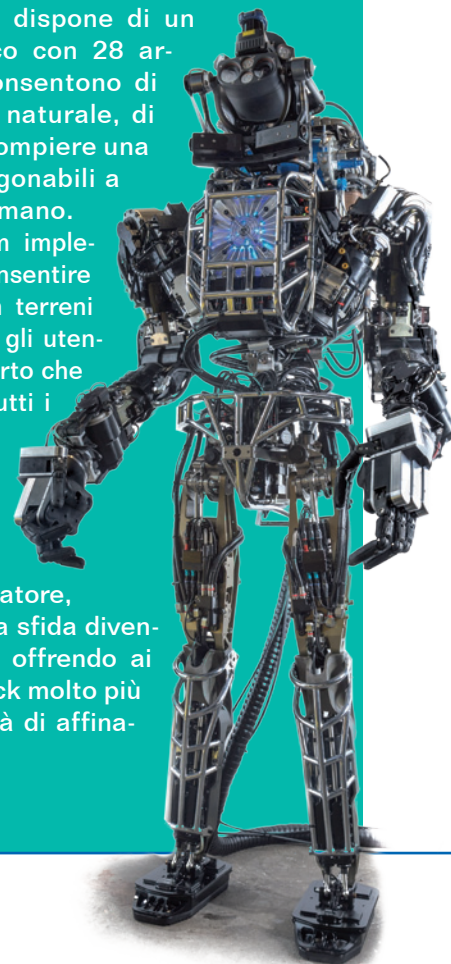
È stato presentato pochi giorni fa ATLAS, il robot antropomorfo progettato da Boston Dynamics per conto del DARPA (Defense Advanced Research Projects Agency), l'Agenzia statunitense che finanzia i progetti di ricerca avanzata; in questo caso (almeno così è stato detto) il progetto ha risvolti prevalentemente civili in quanto lo scopo di questo robot è quello di intervenire in caso di disastri causati dall'attività umana o di catastrofi naturali dove l'intervento di squadre di soccorso sarebbe troppo pericoloso per la vita degli stessi soccorritori. La memoria va subito al disastro nucleare di Chernobyl ma anche alle operazioni di soccorso in occasione dei recenti terremoti dell'Aquila e di Reggio Emilia.

ATLAS, che è stato sviluppato partendo dalla piattaforma umanoide PETMAN, verrà utilizzato nel corso della fase finale del DARPA Robotics Challenge durante la quale i team finalisti si sfideranno programmando questo ed altri robot per renderli in grado di affrontare al meglio i compiti assegnati. Alla fine di quest'anno, durante i "Trials", le squadre metteranno in campo i propri robot ed il DARPA svelerà le prove che gli stessi dovranno affrontare nella finalissima del dicembre 2014 che si svolgerà presso il DARPA con in palio un premio di 2 milioni di dollari.

ATLAS ha le stesse dimensioni di un essere umano: alto 1 metro e 88 centimetri per 150 kg di peso, dispone di un azionamento idraulico con 28 articolazioni che gli consentono di muoversi in maniera naturale, di fare ginnastica e di compiere una serie di attività paragonabili a quelle di un essere umano. Il software che i team implementeranno dovrà consentire al robot di lavorare in terreni accidentati utilizzando gli utensili ed i mezzi di trasporto che gli umani adoperano tutti i giorni.

Le squadre finaliste sono state selezionate sulla base dei risultati ottenuti lavorando su un simulatore, mentre da ora in poi la sfida diventerà molto più reale, offrendo ai sette team un feedback molto più preciso che consentirà di affinare i propri algoritmi.

www.darpa.mil



dstorms NXT: ciò garantisce che i sensori, i motori e gli elementi di costruzione relativi ai precedenti set possano essere riutilizzati.

È anche possibile programmare il mattoncino NXT usando il nuovo software EV3.

La piattaforma comprende un percorso didattico in formato digitale personalizzabile, incluso nel software di programmazione Mindstorms di Lego Education.

La soluzione EV3, inoltre, prevede la possibilità di installare anche l'activity pack aggiuntivo Attività di Progettazione Ingegneristica (DEP), un pacchetto di programmi con 30 ore di istruzioni per il lavoro in aula. Infine, segnaliamo che l'intuitiva piattaforma software per EV3 si basa sul software di programmazione grafica National Instruments LabVIEW.

www.campustore.it



Azzurra e Milano alla ricerca di finanziatori

Sono questi i nomi della mano e del braccio robotico al centro di un interessante articolo scritto da Christian Cipriani per IEEE Spectrum dove vengono descritte le caratteristiche di questi innovativi dispositivi e dove viene raccontata la storia di Prensilia, la spin-off della Scuola Superiore Sant'Anna di Pisa fondata nel 2009 dallo stesso Christian Cipriani e da Marco Controzzi, entrambi ricercatori presso il Sant'Anna. Un articolo che ha anche lo scopo di dare un impulso all'aspetto imprenditoriale della loro attività. Scrive infatti Cipriani *"Siamo attualmente alla ricerca di partner e investitori che desiderino unirsi a noi e che ci aiutino a sviluppare e costruire un prodotto da offrire non solo ai mercati accademici e della ricerca". "Abbiamo fondato Prensilia perché crediamo che l'obiettivo della ricerca sia quello di migliorare la vita delle persone. In particolare la nostra*



speranza è che la robotica possa offrire un reale aiuto a quanti hanno perso un arto, mettendo a disposizione protesi molto più performanti rispetto a quelle attuali". La storia di Azzurra e Milano arriva da molto lontano da quando, fin dai primi anni 1990, sotto la guida del Prof. Paolo Dario, un gruppo di ricercatori della Scuola Superiore Sant'Anna di Pisa iniziò a lavorare alla tecnologia della mano robotica; anche Christian Cipriani e Marco Controzzi hanno preso parte a progetti incentrati sulle protesi degli arti superiori e della robotica umanoide, sotto la supervisione della Prof.ssa Maria Chiara Carrozza, attuale Ministro dell'Istruzione. Il punto di partenza è stato, nel 2008, il progetto SmartHand, finanziato con fondi comunitari e che ha consentito di realizzare un prototipo di mano robotica con cinque attuatori, leggero e affidabile. Da questo prototipo è nato IH2 Azzurra, una mano antropomorfa programmabile che si presta ad essere usata non solo nella robotica ma anche nella ricerca in neuroscienze, nella protesica e nella riabilitazione. Grazie ad una serie di attuatori e sensori di forza, Azzurra riesce ad afferrare una grande varietà di oggetti in maniera effi-

ciente calibrando la forza, a contare e a premere pulsanti. Un altro aspetto importante è il peso, di soli 500 grammi, e la disposizione dei componenti che si trovano tutti all'interno del palmo della mano. Anche Milano, la mano robotica di Prensilia, è figlia di un altro progetto europeo, CyberHand. Essendo una mano intrinseca, Azzurra è l'ideale per le protesi e la ricerca di riabilitazione, mentre Milano, è in grado di adattarsi meglio alla



robotica umanoide (dove i requisiti dimensionali non sono così rigide ed è richiesta una maggiore flessibilità). Detto questo, sia Milano che Azzurra sono mani antropomorfe dotate di sistemi sensoriali e di controllo avanzato, e come tali possono essere utilizzate in tutti i settori di ricerca in cui è importante avere una mano artificiale che si comporta come una mano naturale. www.prensilia.com

Continuiamo il nostro viaggio alla scoperta di Flowcode, l'innovativo sistema di sviluppo grafico per microcontrollori proposto da Matrix Multimedia. Flowcode permette di sviluppare con facilità il software per applicazioni embedded in quanto il codice viene scritto facendo uso di oggetti grafici, in luogo dei classici linguaggi di programmazione come il C e l'Assembler. Seconda puntata.

corso di programmazione in **FLOWCODE** di Francesco Ficili e Daniele Defilippi

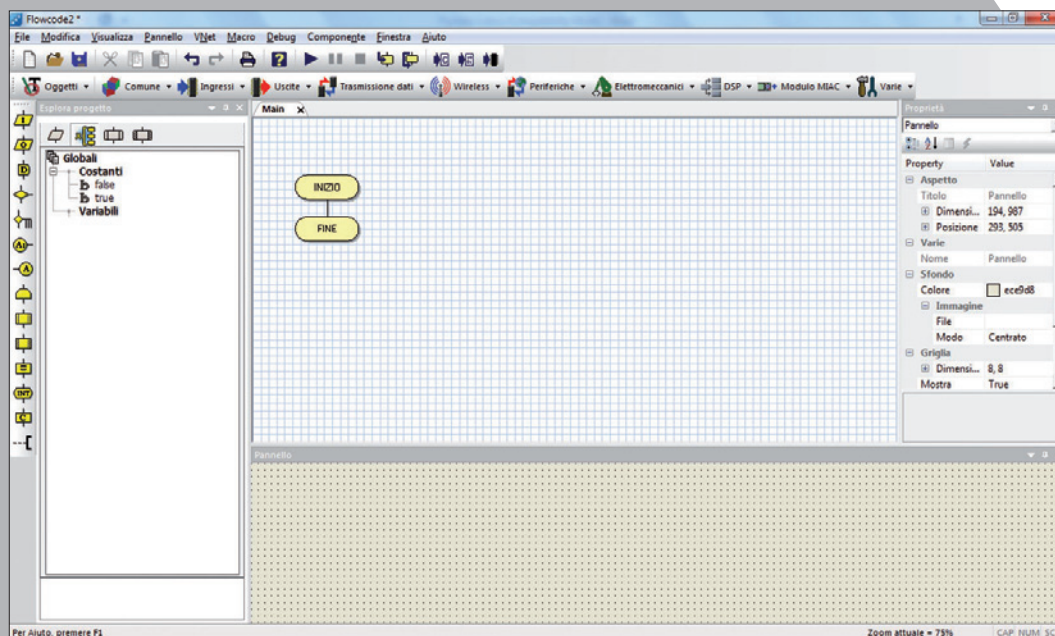
Nella scorsa puntata del corso abbiamo iniziato a prendere confidenza con l'ambiente di sviluppo grafico FlowCode, sviluppato da Matrix Multimedia. È stato presentato l'ambiente grafico, descritta dettagliatamente l'interfaccia, presentate le strutture di controllo ed è stato illustrato un primo esempio di programmazione. In questa puntata spiegheremo l'uso delle variabili, delle costanti e descriveremo i blocchi ingresso e calcolo. Successivamente verranno descritte due funzionalità molto potenti di flowcode: le macro e i componenti. Con le conoscenze acquisite realizzeremo il secondo progetto pratico: la gestione del modulo E-block

LED tramite il modulo KeyPad, usandone i rispettivi componenti software.

Le variabili in Flowcode

Nella puntata precedente abbiamo utilizzato unicamente periferiche di uscita e non abbiamo mai fatto uso di variabili, che sono invece una parte molto importante della programmazione in quanto ci permettono di memorizzare dati ed effettuare calcoli. Attraverso il menu principale di Flowcode "Visualizza->Esplora progetto", apriamo il pannello aggiuntivo nel quale possiamo esaminare tutte le costanti e le variabili globali disponibili all'interno del nostro progetto, come illustrato in Fig. 1.

Fig. 1
Progetto
Flowcode
con pannello
esplora
progetto
visibile sulla
destra della
schermata.



Aggiungere variabili

Per aggiungere una nuova variabile dobbiamo posizionare il mouse sull'etichetta "variabili" e cliccare con il tasto destro in modo da poter scegliere dal menu a tendina la casella "aggiungi nuovo", come mostrato in Fig. 2.

Premendo sul pulsante "Aggiungi nuovo" viene aperta la finestra di creazione di una nuova variabile, come evidenziato in Fig. 3.

I tipi di dati finora supportati spaziano da logici a numerici, con o senza segno, numeri a virgola mobile e stringhe. Per configurare la variabile dobbiamo inserire il nome, il valore iniziale, una breve descrizione e il tipo. Premendo il pulsante "OK" la variabile viene creata e aggiunta alla lista nella finestra di esplorazione del progetto.

Aggiungere costanti

Alla creazione di un nuovo progetto sono già presenti due costanti pre-configurate: "false" e "true", rispettivamente ai valori 0 e 1. Per ag-



Fig. 2 - Aggiunta variabile.

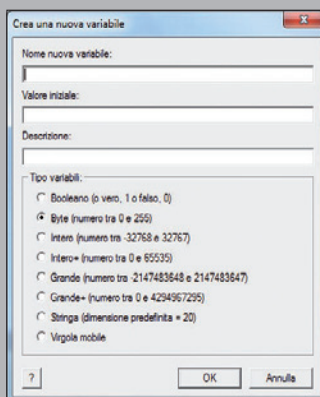


Fig. 3 - Finestra di creazione/configurazione variabile.

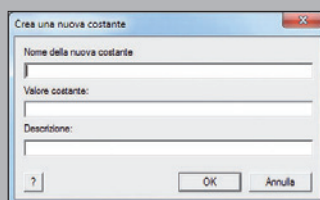


Fig. 4 - Finestra di creazione/configurazione costante.

giungere una nuova costante posizioniamo il mouse sull'etichetta "costanti", clicchiamo con il tasto destro e scegliamo dal menu a tendina "Aggiungi nuovo". Nella finestra di configurazione, mostrata in Fig. 4, inseriamo il nome della costante, il suo valore e una breve descrizione. Le costanti non necessitano dell'indicazione di tipo perché vengono trattate come delle "#define".

Rimozione di variabili/costanti

Qualora volessimo cancellare una variabile o una costante è sufficiente selezionare con il mouse l'elemento desiderato dal pannello "esplora progetto", all'interno della sezione "Globali", e premere il pulsante "Canc". Al termine del progetto potrebbe risultare molto utile rimuovere tutte quelle variabili/costanti che risultano incluse nella lista, ma non sono utilizzate all'interno del nostro progetto. Per far ciò basta cliccare con il tasto destro su "Variabili" o "Costanti" all'interno del pannello "Esplora progetto" e scegliere

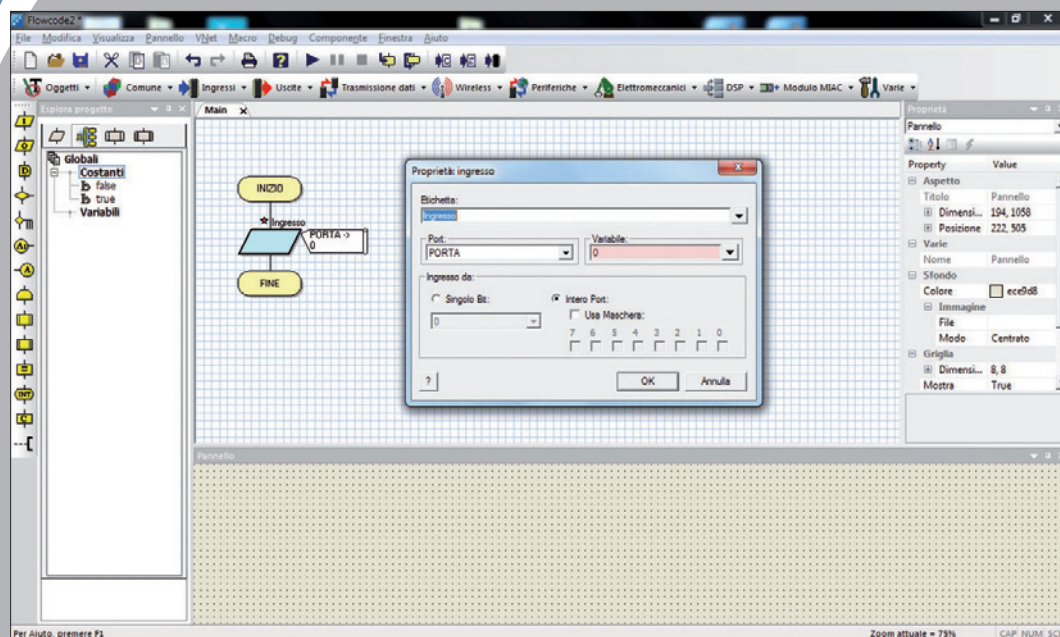


Fig. 5
Blocco di
ingresso e
finestra di
configurazione

dal menu a tendina l'opzione "Cancella inutilizzati".

Blocco ingresso e calcolo

La scorsa puntata avevamo esaminato il blocco di output e quello di ritardo. Siamo quindi capaci di imporre il valore di una porta/pin di uscita e di attendere un tempo prefissato tra due operazioni. Impariamo ora a utilizzare il blocco di ingresso e di calcolo.

Blocco ingresso

Il blocco ingresso è il blocco duale del blocco uscita che abbiamo utilizzato nella puntata precedente per controllare i pin di uscita del microcontrollore. Per inserire il blocco, trasciniamo dalla "Barra delle icone" l'icona corrispondente nel diagramma di flusso. Come per gli altri blocchi, eseguendo un doppio clic sull'icona, è possibile aprire la finestra di configurazione, mostrata in Fig. 5. Selezionando dalla lista a scorrimento "Port:", impostiamo la porta dalla quale vogliamo effettuare la lettura e, nella casella di testo "Variabile:", inseriamo il nome della variabile nella quale vogliamo caricare il valore letto. Premendo sulla freccia a destra della casella di testo apriamo la finestra di

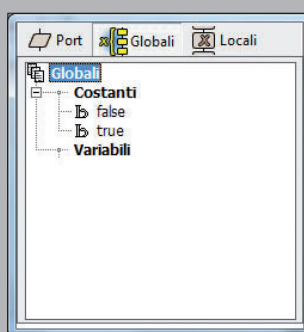


Fig. 6 - Finestra di selezione variabile.

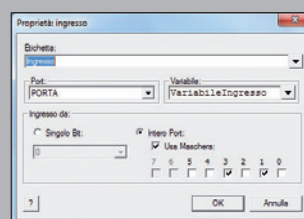


Fig. 7 - Esempio di mascheratura porta di ingresso.

selezione che può aiutarci a scegliere quale variabile utilizzare, come mostrato in Fig. 6. Possiamo scegliere tra le variabili locali o tra quelle globali e, qualora non fossero ancora state create, possiamo crearle direttamente dalla finestra come abbiamo fatto in precedenza dal pannello "esplora progetto". Facendo doppio clic su una variabile presente nella lista, possiamo inserirla direttamente nella casella di testo di configurazione del pannello.

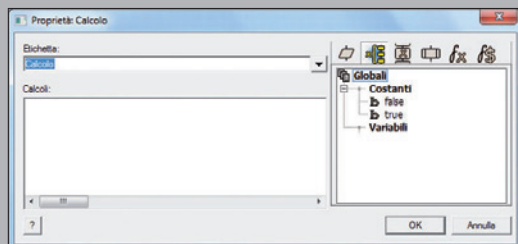
Esattamente come per il pannello di configurazione delle uscite, sono disponibili le opzioni "Singolo Bit" e la mascheratura. Nell'esempio in Fig. 7 la variabile assumerà il valore di PORTA, ma unicamente per il 2° e 4° bit

perché solo il pin 1 e 3 hanno la spunta. Facciamo un esempio. Supponiamo che PORTA abbia un valore pari a 9, che in binario equivale a 0b00001001. Nella variabile "VariabileIngresso" avremo il valore 8 mascherato con 0b00001010 che ha come risultato 0b00001001 AND 0b00001010 = 0b00001000, cioè 8.

Blocco calcolo

Altro blocco di fondamentale importanza è

Fig. 8
Finestra
proprietà del
blocco calcolo.



quello che ci permette di svolgere i calcoli all'interno del nostro diagramma di flusso. Per inserirlo basta scegliere l'icona calcolo dalla "Barra delle icone" e trascinarlo all'interno dell'area di progetto. Facendo doppio clic sul blocco entriamo nella finestra di configurazione, come mostrato in Fig. 8.

Nella casella di testo "Calcoli" possono essere inserite le righe nelle quali specifichiamo i calcoli da effettuare, mentre nella parte di destra è presente un pannello di selezione dal quale possiamo attingere i valori delle variabili, delle costanti e delle funzioni che Flowcode ci mette a disposizione. Gli operatori logico/matematici che possiamo utilizzare sono rappresentati in **Tabella 1**.

Le funzioni valide sono invece riportate in **Tabella 2**.

Per capirne meglio il funzionamento facciamo un semplice esempio. Supponiamo di avere la variabile "LunghezzaInch" che contiene il valore di una lunghezza rappresentato in pollici e di volerla

Tabella 1 - Operatori logico/matematici.

(.)	Parentesi
=, <, >	Uguale a, diverso da
+, -, *, /, MOD	Addizione, sottrazione, moltiplicazione, divisione e modulo
<, <=, >, >=	Minore di, minore o uguale a, maggiore di, maggiore o uguale a
>>, <<	Spostamento a destra, spostamento a sinistra.
~, &, , ^	NOT, AND, OR, OR Esclusivo, bit per bit
NOT, AND, OR, XOR	NOT, AND, OR, OR Esclusivo, bit per bit
&&, , !	AND, OR, NOT, logici

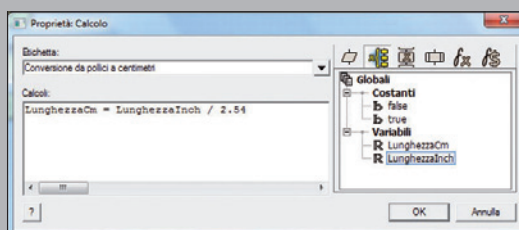


Fig. 9
Esempio di
calcolo per
conversione di
unità di misura
da pollici in
centimetri.

Tabella 2 - Funzioni utilizzabili in Flowcode.

float = fadd(float, float)	Somma due numeri in notazione a virgola mobile
float = fsub(float, float)	Differenza tra due numeri in notazione a virgola mobile
float = fmul(float, float)	Moltiplica due numeri in notazione a virgola mobile
float = fdiv(float, float)	Divide due numeri in notazione a virgola mobile
float = fmod(float, float)	Esegue il MODULO di due numeri in notazione a virgola mobile
byte = isinf(float)	Verifica se il numero in virgola mobile è infinito
byte = isnan(float)	Verifica se il numero in virgola mobile è un numero
byte = float_eq(float, float)	Verifica l'uguaglianza di due numeri in virgola mobile
byte = float_ge(float, float)	Controlla se un numero in virgola mobile è maggiore o uguale all'altro
byte = float_gt(float, float)	Controlla se un numero in virgola mobile è maggiore dell'altro
byte = float_le(float, float)	Controlla se un numero in virgola mobile è minore o uguale all'altro
byte = float_lt(float, float)	Controlla se un numero in virgola mobile è minore dell'altro
int = random()	Genera un numero casuale tra -32768 e 32767

convertire in centimetri all'interno della variabile "LunghezzaCm". Il frammento di codice necessario per svolgere questa semplice operazione è indicato in Fig. 9.

Le macro e i componenti software

Gli esempi che abbiamo visto fino ad ora utilizzavano un numero limitato di blocchi per implementare la funzionalità, ma normalmente non è possibile implementare un intero flusso su un unico diagramma. Il blocco macro ci è di ausilio in questo compito. In pratica ci permette di creare delle sottoparti di diagramma su aree di lavoro differenti e di poterle richiamare agevolmente.

Creare una macro

Per creare una macro entriamo nel menu "Macro" e premiamo su "Nuova...". La finestra di configurazione ci permette di gestire e configurare tutte le caratteristiche salienti della macro. All'interno della casella "Nome della nuova macro" inseriamo il nome con il quale richiameremo la macro all'interno

Fig. 10
Barra dei componenti.



del nostro diagramma di flusso e, anche se non è obbligatorio, è vivamente consigliato inserire dei commenti all'interno della casella "descrizione nuova macro" per facilitare in futuro la comprensione della funzionalità che svilupperemo all'interno.

Una macro può comunicare con il diagramma che la istanzia facendo uso di parametri, di valori di ritorno e di variabili globali, esattamente come avviene per una classica funzione in linguaggio C. I parametri sono utilizzati quando è necessario passare dei valori dal diagramma chiamante alla macro. Il valore di ritorno è utilizzato quando si vuole restituire al diagramma chiamante un valore mentre le variabili globali possono essere utilizzate in tutti gli altri casi. Quando una macro utilizza delle variabili globali sarebbe buona norma indicare, nella descrizione della macro, tipo e nome delle variabili globali in modo che, qualora si dovesse riutilizzare la macro in altri diagrammi, si possano ricreare tutte le variabili in modo corretto. Una volta creata, la macro può essere istanziata nel diagramma di flusso usando l'icona "macro" e valorizzando opportunamente gli eventuali parametri. Durante la spiegazione del progetto pratico di questa puntata, vedremo un esempio che ci aiuterà a comprendere come configurare e utilizzare una macro all'interno di un progetto Flowcode.

Esportare e importare una macro

Mediante il menu "macro" nella barra principale di Flowcode è possibile esportare una macro in modo da renderla disponibile per altri progetti. La macro scelta verrà quindi salvata in formato "*.fcm" e potrà essere importata utilizzando lo stesso menu. Questa potenzialità di Flowcode ci permette in pratica di creare una libreria di macro che possiamo incrementare ogni qualvolta creiamo una funzione che pensiamo possa essere utile e riutilizzabile in altri progetti.

I componenti

La possibilità di creare delle macro è molto importante, mentre fondamentale è la possibilità di utilizzare i componenti. I

componenti sono stati realizzati da Matrix Multimedia per esserci di aiuto nello sviluppo di progetti complessi. La libreria dei componenti è suddivisa in base alla tipologia. Possiamo trovare i seguenti principali raggruppamenti: Comuni, Ingressi, Uscite, Trasmissioni Dati, Wireless, Periferiche, Elettromeccanici, Moduli Miac e Varie. Per il momento i gruppi e i componenti disponibili sono quelli che vi mostreremo durante il corso, ma nelle versioni successive questa libreria sarà sicuramente più ampia e ricca di nuove funzionalità.

La barra dei componenti

La libreria dei componenti di Flowcode è accessibile mediante la "barra dei componenti" mostrata in Fig. 10. I blocchi disponibili sono organizzati per area tematica in modo da facilitarci nella ricerca. I componenti possono essere utilizzati per aiutarci nella realizzazione di sistemi, anche molto complessi, permettendoci di trascurare l'implementazione effettiva del driver del componente, lasciandoci così concentrare sulla

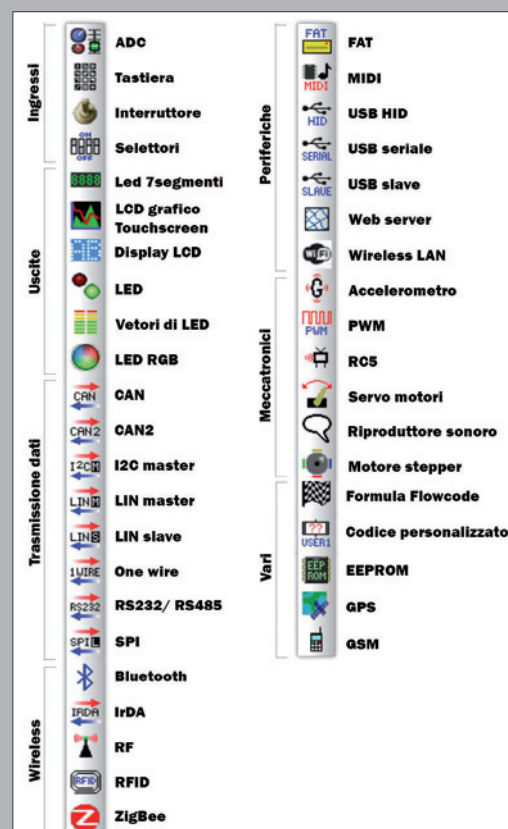
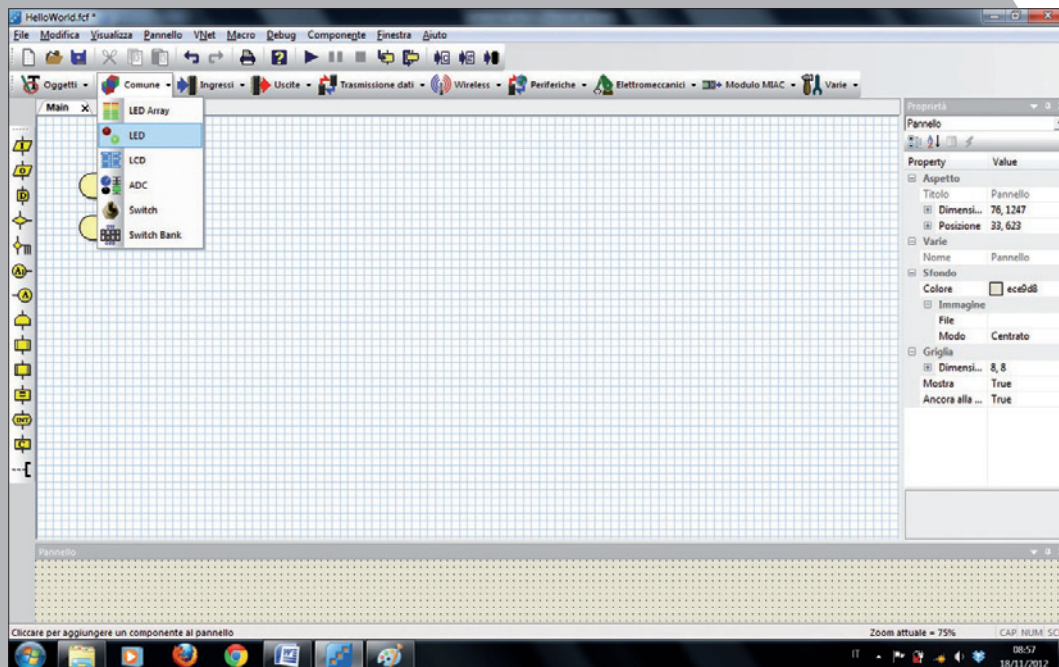


Fig. 11
Principali componenti disponibili in Flowcode.

Fig. 12
Inserire il
componente
"LED"
utilizzando
la "Barra dei
componenti".



funzionalità che vogliamo implementare. Per essere correttamente utilizzato, un componente - dopo essere stato inserito - deve essere configurato utilizzando la propria interfaccia specifica. Nelle sezioni successive ne vedremo alcuni esempi.

Aggiungere un componente software

Per aggiungere un componente software basta cliccare sulla barra di libreria (nel nostro caso inseriamo un LED, come rappresentato in Fig. 13) e vedremo apparire

nel pannello, in basso, il componente che abbiamo scelto.

Ora il componente è attivo all'interno del nostro progetto ma, per funzionare correttamente, deve avere le connessioni configurate opportunamente. Clicchiamo sul componente con il tasto destro del mouse, come in Fig. 13. A questo punto si apre un menu a tendina: premiamo su "Connessioni" in modo che si apra la finestra di settaggio come mostrato in Fig. 14. Configuriamo le connessioni per collegare

Fig. 13
Menu
connessioni
relativo al
componente.

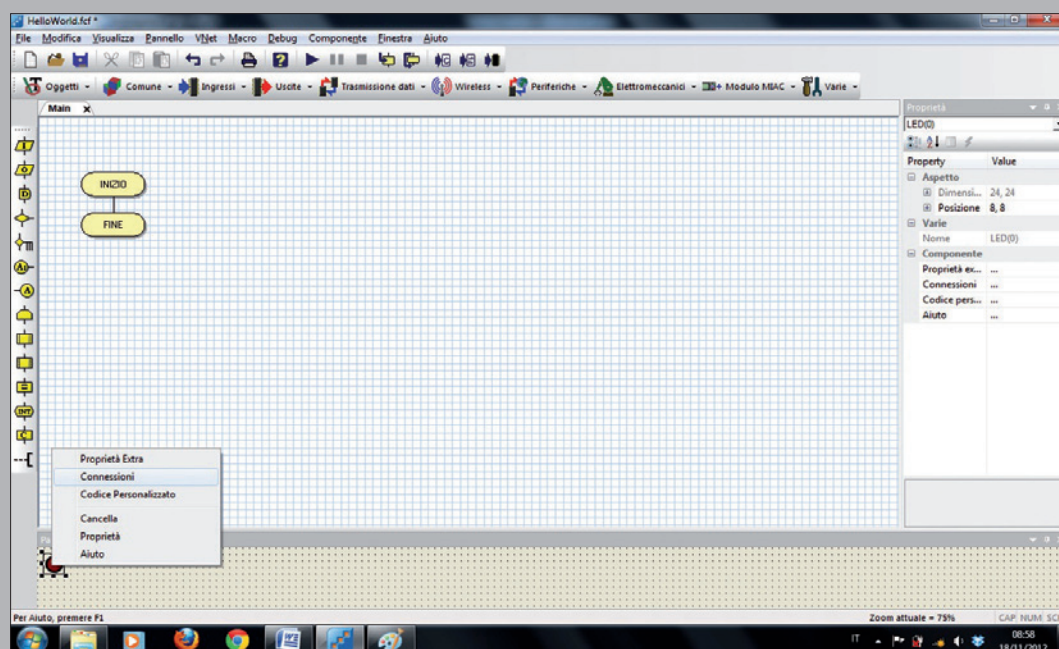
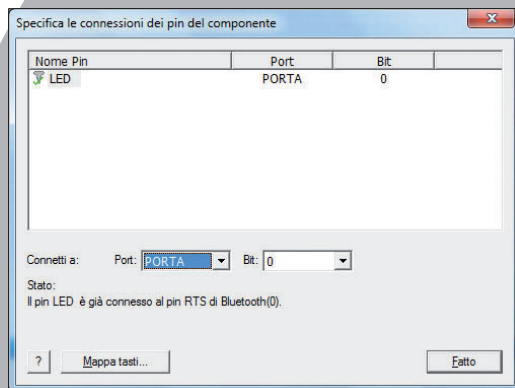


Fig. 14
Esempio
di connes-
sione per il
componente
LED.



il componente LED al pin 0 di PORTA. Ora il componente software è configurato ed è possibile inserirlo, dove necessario, all'interno del nostro flusso di programma. Per far ciò basta inserire l'icona "macro componente", selezionandola dalla "Barra delle icone" e trascinarla in posizione corretta all'interno del nostro diagramma. Più blocchi possono utilizzare lo stesso componente in modo da svolgere una o più attività all'interno del diagramma di flusso. Per poter funzionare correttamente, il blocco deve essere configurato. Per far ciò basta che con doppio clic entriamo nella finestra "Proprietà: Componente macro" e selezioniamo nella parte di sinistra il componente al quale deve far riferimento il blocco, nel nostro caso LED(0). Vedremo così apparire nella parte di destra le macro che possono essere chiamate, come mostrato in Fig. 15. In questo caso, avendo scelto il LED, abbiamo a disposizione due macro: quella per accendere il led "LEDon" e quella per spegnerlo "LEDOff". Selezioniamo la funzionalità voluta e premiamo il pulsante "OK" per terminare la configurazione. Tutti i componenti hanno inoltre delle proprietà extra, accessibili mediante tasto destro sul componente, selezionando la voce "Proprietà extra". Per ogni componente verrà aperta una finestra specifica con differenti caratteristiche configurabili. Nel pannello relativo al componente LED è possibile impostarne la forma, il colore e la polarità, come mostrato in Fig. 17.

Progetto Pratico: gestione di 3 LED tramite Keypad

Passiamo adesso alla realizzazione di un progetto che ci consenta di mettere in pratica alcuni dei concetti esposti in questa puntata e che ci aiuti a prendere maggior

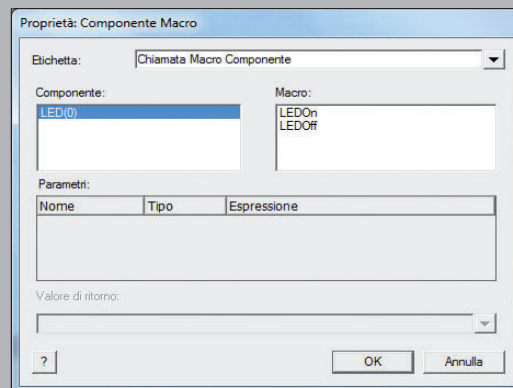
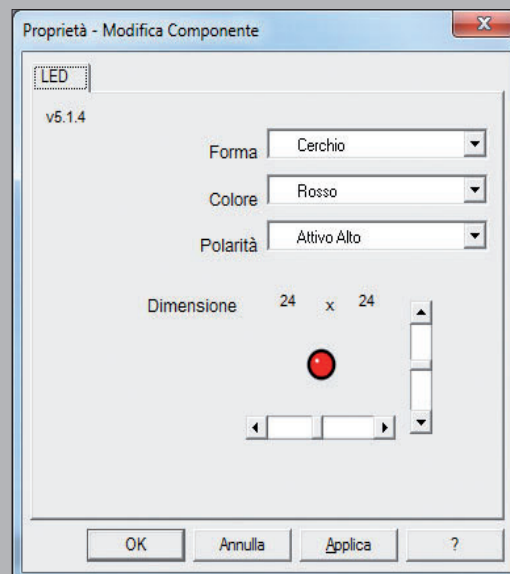


Fig. 15
Macro
disponibili
per il
componente
LED.

confidenza con i primi componenti di Flowcode. Quello che ci proponiamo di realizzare è un sistema di controllo che gestisca i dati in ingresso da un Keypad a 12 tasti (il classico Keypad telefonico) e che utilizzi le informazioni lette per gestire un gruppo di tre LED. Le specifiche del nostro sistema sono le seguenti:

- alla pressione del tasto 1 del Keypad accendere il LED1 e spegnere gli altri LED;
- alla pressione del tasto 2 del Keypad accendere il LED2 e spegnere gli altri LED;
- alla pressione del tasto 3 del Keypad accendere il LED3 e spegnere gli altri LED;
- alla pressione del tasto cancelletto del Keypad spegnere tutti i LED.

Abbiamo utilizzato i LED per semplicità didattica ma avremmo potuto, con lo stesso sistema, pilotare dei relè, delle elettrovalvole o altri carichi esterni.



**Fig. 16 - Proprietà extra relative
al componente LED.**

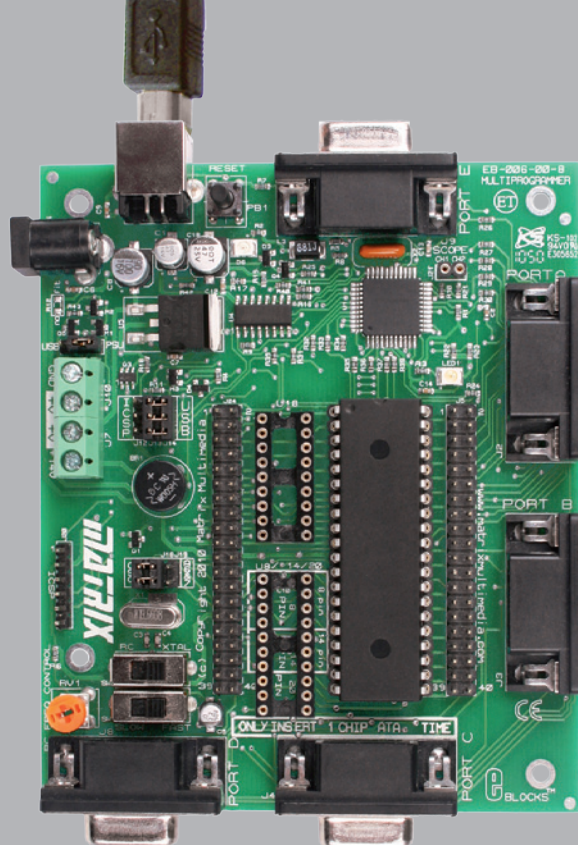
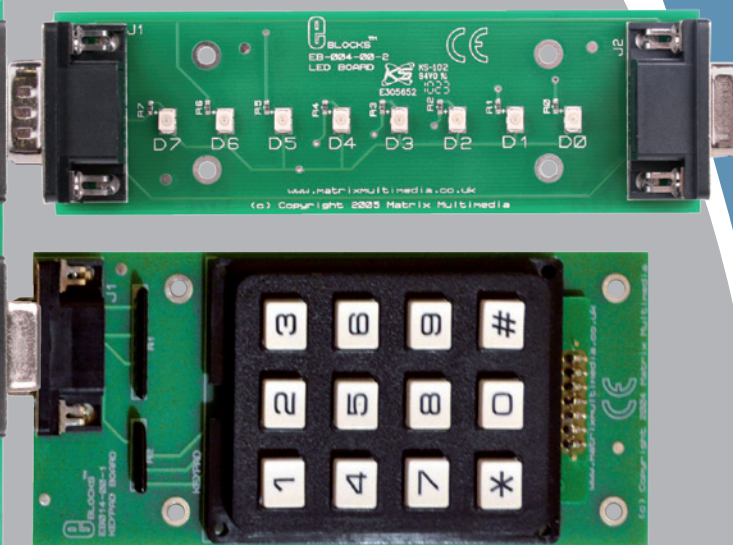


Fig. 17 - Collegamento E-Blocks per la realizzazione del progetto pratico.



Per realizzare questo progetto abbiamo bisogno dei seguenti componenti:

- EB006 "USB PICmicromultiprogrammer-board" (PIC18F4550 e Quarzo 12MHz)
- EB004 "LED Board"
- EB014 "Keypadboard"

Colleghiamo ora le schede in modo da avere i LED su PORTA ed il tastierino sulla PORTB, come mostrato in Fig. 17. Apriamo ora il progetto vuoto che avevamo creato con il nome "ConfigurazioneBase.fc" e salviamolo con il nome "SwitchLed.fc". In questo modo il microcontrollore risulta già configurato per funzionare con l'hardware a nostra disposizione: possiamo così dedicarci alla realizzazione del diagramma di controllo del sistema.

In questo nuovo progetto faremo uso dei primi componenti, quindi, dalla barra dei componenti inseriamo nel nostro progetto un componente "Keypad" e tre componenti "LED", che sono le periferiche che dovremo gestire. Come spiegato in precedenza, per inserire un componente bisogna selezionarlo dalla barra dei componenti. Dopo averlo selezionato, il componente comparirà nel pannello sotto l'area di progetto. A questo punto configuriamo le connessioni dei componenti, cominciando dal KeyPad. Eseguiamo un clic con il tasto destro sul componente keypad comparso sul pannello e selezioniamo l'opzione "connessioni". Comparirà la schermata di Fig. 18. Come evidenziato in figura, colleghiamo le tre colonne e le quattro righe alla porta B (le tre colonne ai pin 0, 1 e 2 e le quattro

Fig. 18 Finestra connessioni del componente keypad.

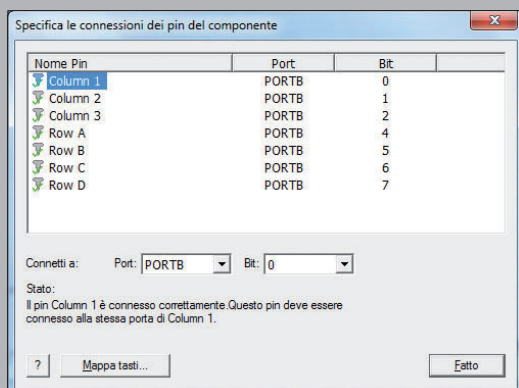


Fig. 19 Finestra connessioni del componente LED.

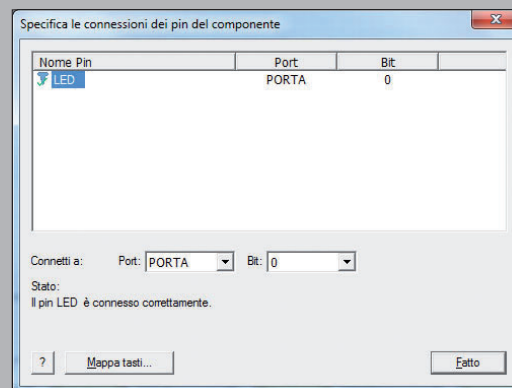
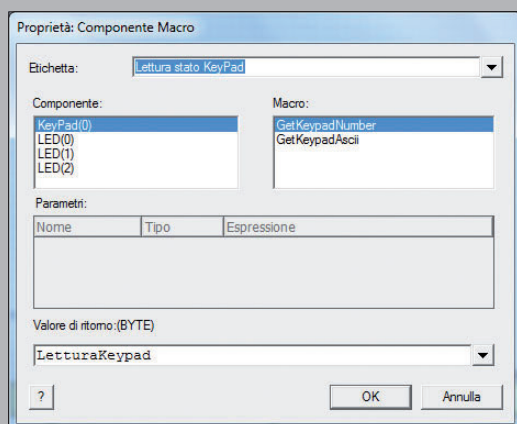


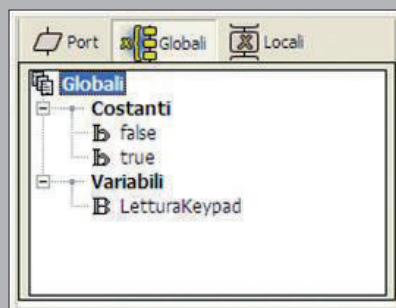
Fig. 20
Finestra di proprietà
del componente.



righe ai pin 4, 5, 6 e 7). Una volta eseguite le connessioni confermiamo premendo su "Fatto...". Allo stesso modo configuriamo i tre LED, collegandoli rispettivamente ai pin 0, 1 e 2 della porta A.

In Fig. 19 è possibile vedere la configurazione del primo LED. A questo punto possiamo iniziare la stesura del blockdiagram, inserendo il classico ciclo principale, all'interno del quale creeremo la nostra applicazione. Una volta realizzato il ciclo, inseriremo anche un blocco "Macro Componente" ed una struttura "Selezione",

Fig. 21
Creazione della
variabile di
LetturaKeypad.



dotandola di quattro casi. A questo punto eseguiamo un doppio clic sul blocco "Macro Componente", ed esaminiamo la finestra che si apre, riportata per comodità in Fig. 21. Selezioniamo il componente Keypad e la macro *GetKeypadNumber* che ci consentirà di rilevare il tasto premuto. A questo punto ci serve una variabile per immagazzinare il valore, che poi utilizzeremo nella successiva struttura di selezione.

Creiamo una variabile sfruttando il pulsante con il simbolo della freccia in basso presente sulla finestra attiva, e chiamiamola *LetturaKeypad* (come tipo scegliamo un *unsigned byte*). Se avete eseguito tutto correttamente dovreste avere un risultato simile a

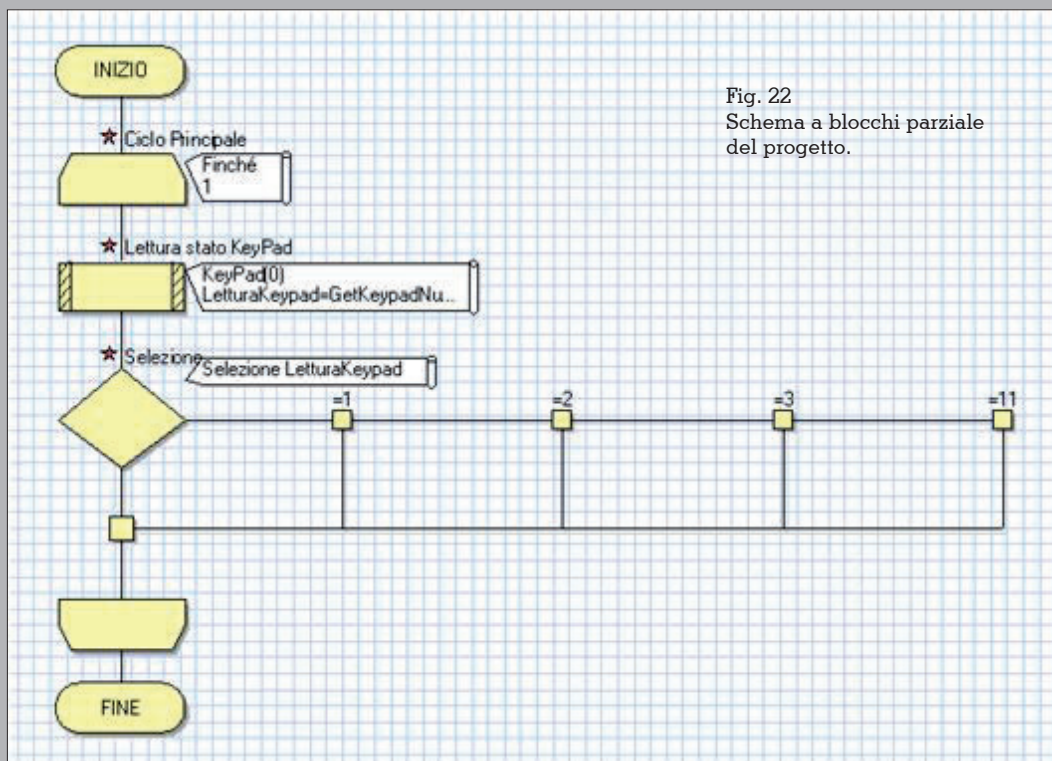


Fig. 22
Schema a blocchi parziale
del progetto.

Fig. 23
Macro LED0.

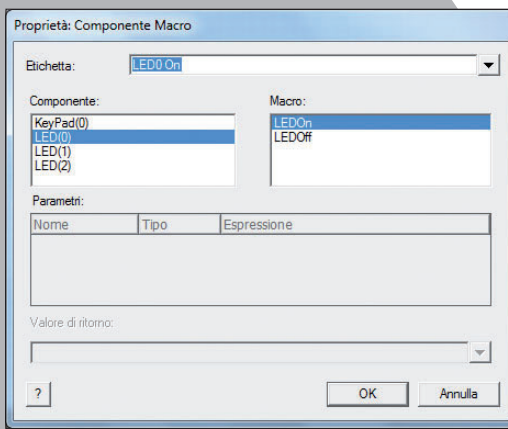
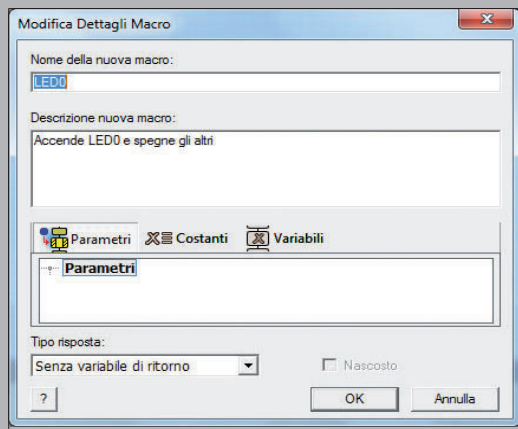


Fig. 24
Proprietà del componente LED.

quello riportato in Fig. 21. Passiamo ora alla gestione della struttura di selezione. Associamo alla selezione la variabile precedentemente creata e valorizziamo i primi tre casi con i valori numerici 1, 2 e 3 ed il quarto con il valore 11 (cancellito). A questo punto il diagramma di flusso dovrebbe apparire come quello riportato in Fig. 22. Abbiamo quasi finito: non ci resta che gestire correttamente il pilotaggio delle uscite nei vari casi. Per evitare di rendere il blockdiagram troppo complesso, utilizziamo delle macro specifiche per il pilotaggio delle uscite. Cominciamo dal caso corrispondente al tasto 1 che, secondo le nostre specifiche, dovrebbe accendere il LED0 e spegnere gli altri. Creiamo una nuova macro, che chiameremo appunto LED0, come abbiamo visto in precedenza nel paragrafo dedicato alle macro ed inseriamo

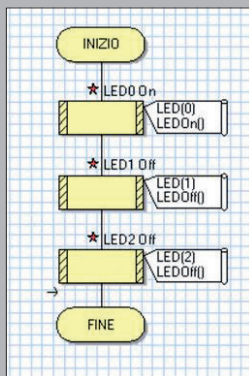


Fig. 25
Schema a blocchi della macro LED0.

la descrizione, come riportato in Fig. 23.

Nella macro inseriamo tre blocchi "Macro componente" con i quali gestiremo i componenti LED precedentemente creati. Chiamiamo la funzione LEDOn per il LED0 e la funzione LEDOff per i LED 1 e 2. Se abbiamo eseguito tutto correttamente otterremo uno schema come quello riportato in Fig. 25. Creiamo altre due macro analoghe per l'accensione dei LED 1 e 2 ed una macro di spegnimento che chiameremo LEDOff.

Ora posizioniamo quattro blocchi "Chiamata Macro" nei quattro rami della struttura di selezione e chiamiamo opportunamente le macro appena create. Il nostro progetto è concluso; il blockdiagram complessivo è riportato in Fig. 26, a questo punto non ci resta che collegare opportunamente gli E-block e testare quanto realizzato.

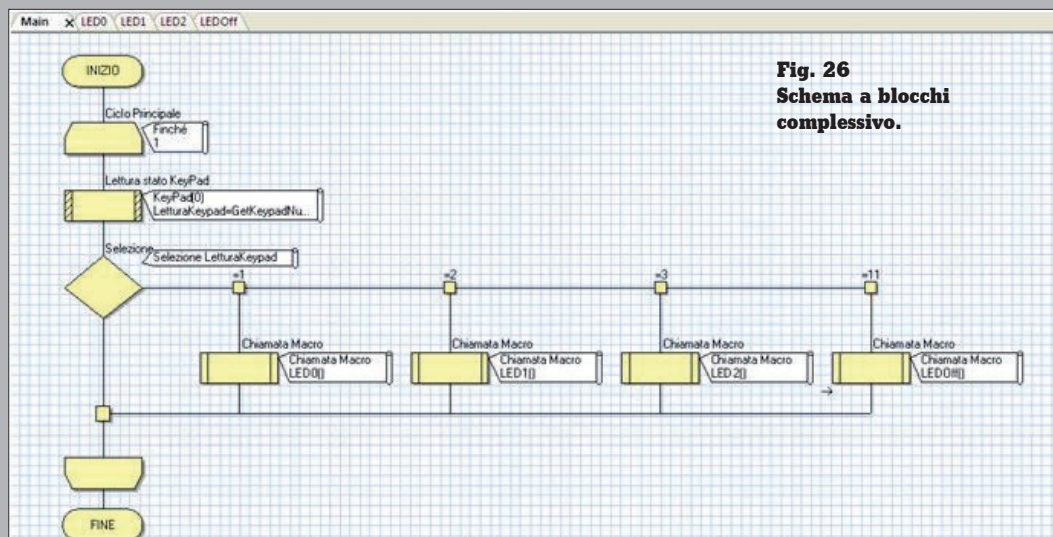


Fig. 26
Schema a blocchi complessivo.