

In omaggio
la guida alla scelta computer
MICROCHIP

mensile di progettazione elettronica, attualità scientifica, novità tecnologiche.

€ 6,00

Elettronica In

www.elettronica.in.it

oltre l'elettronica

Anno XX - n. 184
Marzo 2014

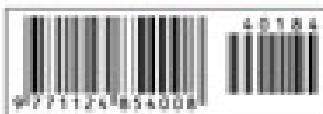


I tuoi dati meteo
su **Xively**
con Arduino e
WiFi shield



Raspberry Pi Internet Radio

- Campanello domotico con smartphone
- Shield RFTide per Arduino
- Radio controller per strip a LED
- Test Harness
- Centrale di allarme ambientale
- Essere connessi secondo Microchip
- Generatore di funzioni per myDAQ
- Corso KiCad



INTERNET RADIO CON RASPBERRY PI E SHIELD LCD

di MARCO MAGAGNIN

Un tema tradizionalmente caro agli "elettronici", da quando Guglielmo Marconi ha strabiliato il Mondo intero trasmettendo dalla nave Elettra, è quello delle comunicazioni radio e in special modo della radiofonia. Questo settore ha subito profondi cambiamenti dall'arrivo delle tecnologie digitali (DAB, per esempio) e oggi, con

lo streaming e il TCP/IP, sbarca su Internet, che diventa un vettore parallelo alle onde radio, tanto che è improprio parlare di radiodiffusione: la radio su Internet non è più una radio ma una diffusione in rete di quello che ascoltiamo in radio. Per quanto perda il fascino della radio, la

Entriamo nel mondo della radiodiffusione in streaming via Internet, trasformando la nostra Raspberry Pi prima in un ricevitore di stazioni radio, poi in un trasmettitore per broadcast su IP.



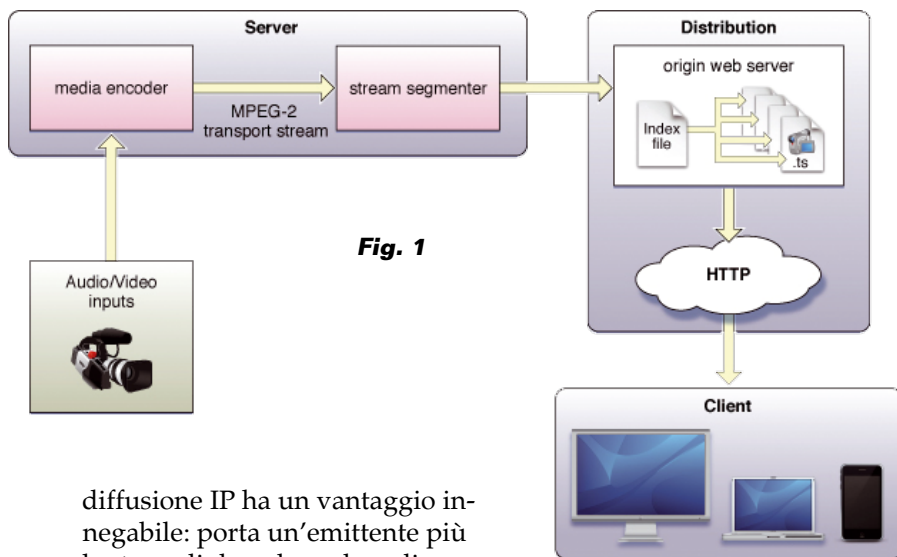


Fig. 1

diffusione IP ha un vantaggio innegabile: porta un'emittente più lontano di dove le onde radio consentirebbero.

Alla radio e, più in generale, alla diffusione sonora su IP, abbiamo deciso di dedicare alcuni articoli dal "taglio" decisamente pratico, dato che con essi vi proporremo di realizzare applicazioni incentrate su Raspberry Pi, che ormai è il cavallo di battaglia nelle applicazioni embedded. In queste pagine vi spieghiamo come realizzare una radio "internet" portatile, ossia un apparecchio in grado di ricevere stazioni radio in streaming mediante collegamento a Internet, sia via cavo che in WiFi. È anche prevista la gestione di playlist di brani audio memorizzati in precedenza su una chiavetta USB o su un supporto di memorizzazione di massa in rete. La trasmissione in "streaming" delle radio IP permette di inviare un flusso continuo di pacchetti TCP HTTP da un'architettura di server di produzione e distribuzione verso client Internet dove i brani possono essere riprodotti; lo schema di Fig. 1 è esplicativo del sistema. I contenuti possono provenire da trasmissioni dal vivo oppure registrate, nel qual caso possono anche essere richieste dai client secondo la formula "on demand". Il protocollo supporta flussi a differenti bit rate, per-

mettendo ai client di scegliere il flusso ottimale in base alla velocità della rete cui ci si collega. La gestione "fisica" della radio avviene utilizzando lo shield LCD (cod. 7100-FT1074K di Futura Elettronica, www.futurashop.it) oppure via interfaccia web.

Nel caso dello shield, utilizzeremo i pulsanti per navigare il menu delle funzionalità, selezionare brani e stazioni radio, nonché regolare il volume. Nella Fig. 2 potete vedere lo schema a blocchi d'insieme dell'applicazione. Grazie al lavoro di Adafruit, di Bob Rathbone e alla disponibilità di "mattoni" software nel mondo Open source, troviamo i componenti necessari a questa applicazione praticamente già tutti disponibili. La realizzazione pratica di questa applicazione consiste nella corretta personalizzazione (installazione e configurazione dei pacchetti necessari) del sistema operativo di Raspberry Pi, oltre che nell'installazione dello shield LCD e del dongle WiFi. Il Music Player Daemon (MPD) è, come suggerisce il nome, un server software, gestibile da remoto,

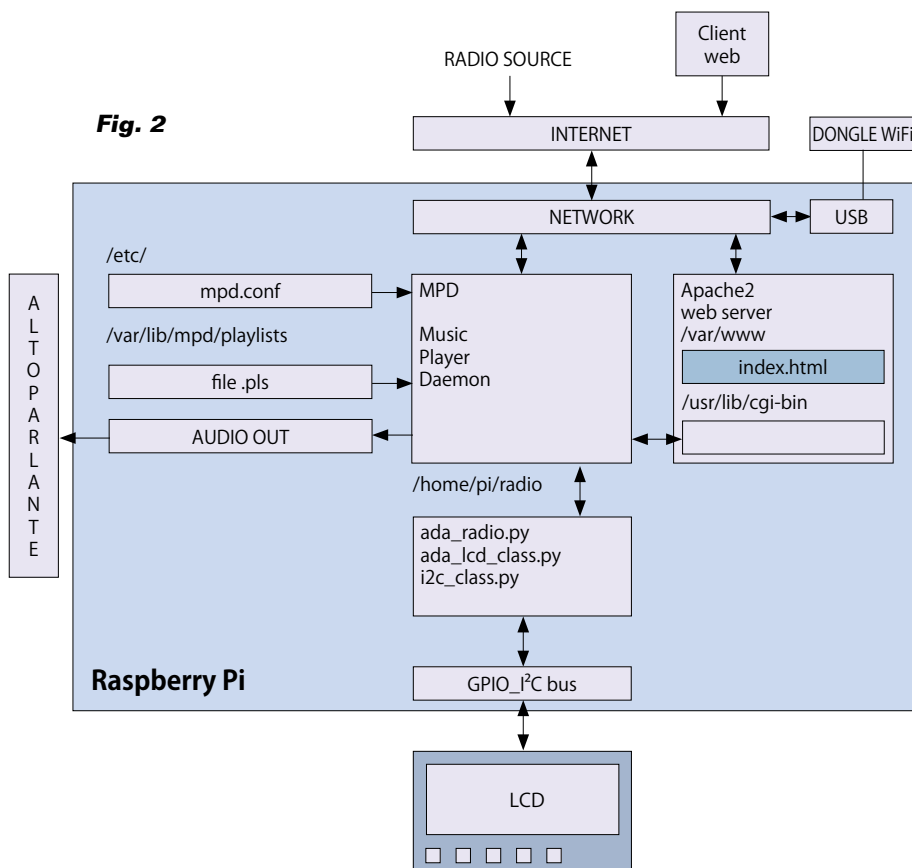


Fig. 2

utilizzato per riprodurre file musicali, compilare e riprodurre playlist e distribuire musica in streaming (HTTP STREAMS con diversi sottoprotocolli).

È stato realizzato con l'obiettivo di utilizzare la minor quantità di risorse di calcolo possibili, originariamente per creare sistemi audio stereo sui vecchi PC con processore Pentium 75. MPD è in grado di gestire riproduzioni gapless, bufferizzare l'audio in uscita e gestire il cross fading. La disponibilità dei package per Debian e il consumo di risorse minimale, ne fanno un ottimo candidato ad essere installato su Raspberry Pi. Siccome i realizzatori di MPD hanno mantenuto separata la componente server dalla componente client, è possibile scegliere quale interfaccia utente utilizzare per la gestione della propria "scatola" musica-

le. Sono disponibili molti client, di diverse tipologie (a linea di comando, grafici, web, gestibili da programma, ecc.) e destinati a diversi tipi di dispositivi come PC, tablet e smartphone. Potete trovarne un elenco all'indirizzo <http://mpd.wikia.com/wiki/Clients> e divertirvi a sperimentare. Il client web utilizzato nella nostra applicazione è una versione derivata dal client per MPD Snoopy, appositamente adattata per Raspberry Pi, sempre da Bob Rathbone.

PREPARAZIONE RASPBERRY PI

Per questa realizzazione abbiamo preferito partire da una copia "fresca" del sistema operativo di Raspberry Pi, anche perchè, molto probabilmente, chi come noi si appresta a questa realizzazione desidera ottenere un dispositivo "dedicato" all'applicazione: in sintesi, un radio internet "portatile". Per quanto riguarda la configurazione hardware di Raspberry Pi e l'impostazione dell'ambiente di configurazione e sviluppo, rimandiamo ai

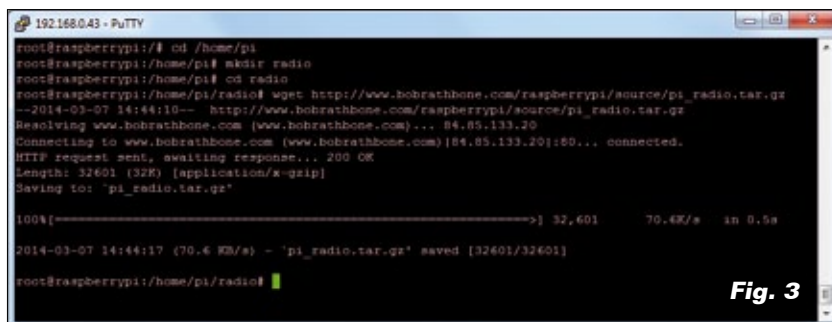


Fig. 3

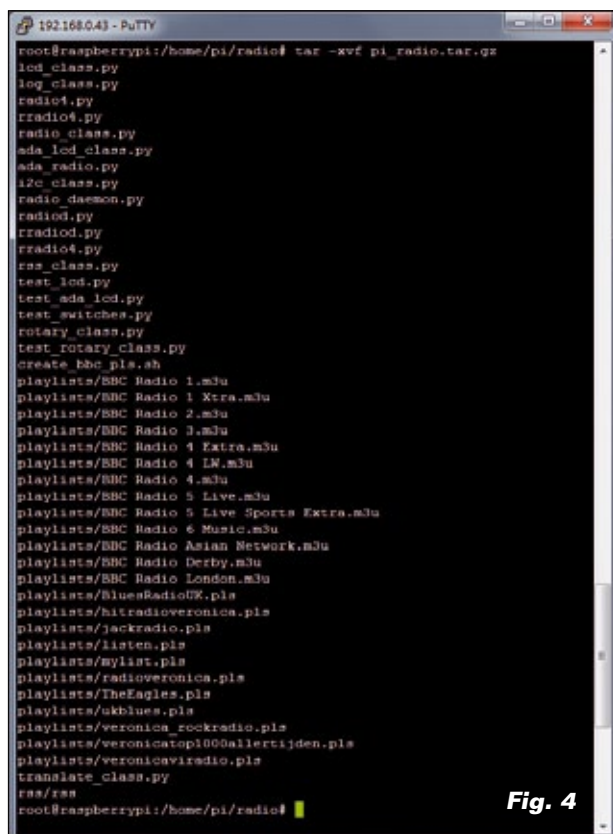


Fig. 4

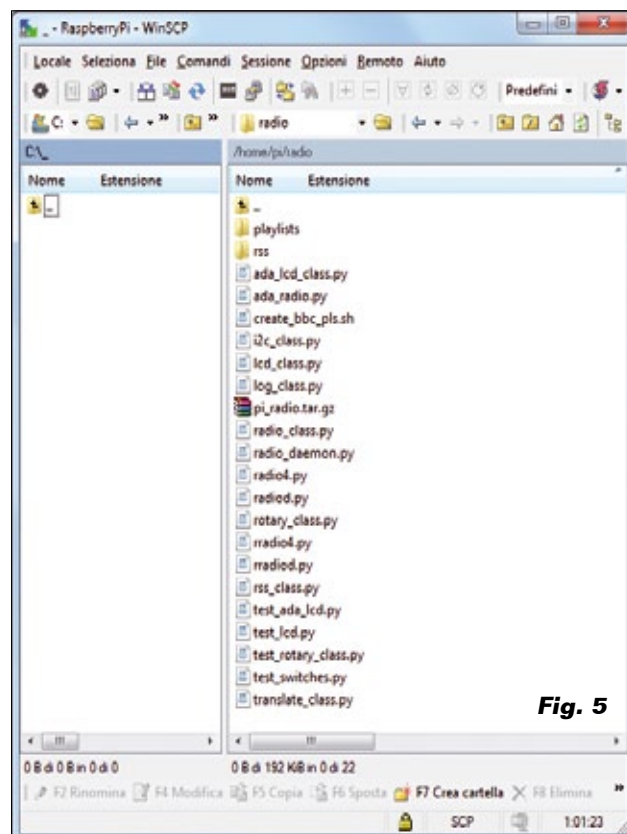


Fig. 5

```

192.168.0.43 - PuTTY
root@raspberrypi:~# i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20: 20  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
root@raspberrypi:~#

```

Fig. 6

```

192.168.0.43 - PuTTY
root@raspberrypi:/home/pi/radio# apt-get install mpd mpc
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libao-common libao4 libaudiofile1 libavcodec53 libavformat53
  libavutil51 libdirac-encoder0 libfaad2 libgsm1 liblame0 libmp3lame0
  libmpeg6 libmpdclient2 libschroedinger-1.0-0 libshout3 libspeex1
  libtheora0 libvorbis libvorbis1 libvorbisenc4
Suggested packages:
  libao0 libao-alsa0 speex avahi-daemon icecast2 mpd-client pulseaudio
The following NEW packages will be installed:
  libao-common libao4 libaudiofile1 libavcodec53 libavformat53
  libavutil51 libdirac-encoder0 libfaad2 libgsm1 liblame0 libmp3lame0
  libmpeg6 libmpdclient2 libschroedinger-1.0-0 libshout3 libspeex1
  libtheora0 libvorbis libvorbis1 libvorbisenc4 mpd mpc
0 upgraded, 24 newly installed, 0 to remove and 0 not upgraded.
Need to get 6,016 kB of archives.
After this operation, 14,2 MB of additional disk space will be used.
Do you want to continue [Y/n]?

```

Fig. 7

capitoli, dedicati a questi argomenti, del libro **“Raspberry Pi – Il mio primo Linux embedded”**, al quale ci riferiremo, nel resto dell’articolo, con il termine “libro”.

INSTALLAZIONE SOFTWARE DI BASE

Abbiamo detto che partiamo da un’installazione fresca del sistema operativo Raspian. Scarichiamo l’ultima versione di Raspbian dall’indirizzo <http://www.raspberrypi.org/downloads> e poi creiamo una SD-Card utilizzando Win32DiskImager, seguendo le indicazioni del “libro”. Una volta terminata la creazione della SD-Card, inseriamo quest’ultima in Raspberry Pi, colleghiamo un cavo di rete e l’alimentazione e forniamo tensione. Sempre seguendo le indicazioni del “libro”, scopriamo

l’indirizzo IP di Raspberry Pi e ci colleghiamo in remoto da un PC al server SSH di Raspberry Pi, utilizzando gli strumenti Putty e WINScp. Appena collegati, creiamo l’utente “root”, ci autenticiamo con quest’ultimo ed aggiorniamo la distribuzione tramite i comandi:

```

apt-get update
apt-get upgrade

```

Utilizzando il configuratore raspi-config espandiamo il filesystem di root. Intanto che stiamo utilizzando raspi-config, approfittiamone per configurare la tastiera ed impostare la giusta area geografica, in modo da poter calcolare l’ora esatta. Poi accettiamo la richiesta di reboot e ci ricolleghiamo, sempre con l’utente “root”. Con WINScp creiamo, nel percorso `/home/pi`, la cartella “radio” come visibile in Fig. 3. Il percorso completo sarà:

`/home/pi/radio`

Entriamo nella cartella e carichiamo il software realizzato da Bob Rathbone. Con “wget” scarichiamo il file compresso, che esanderemo con il comando tar. Digitiamo i comandi (Fig. 3 e Fig. 4).

```

wget http://www.bobrathbone.com/raspberrypi/source/pi_radio.tar.gz

```

```

tar -xvf pi_radio.tar.gz

```



Nella cartella `/home/pi/radio` troveremo una serie di archivi, come visibile in Fig. 5. Trattiamo per un attimo la curiosità di vederne funzionare qualcuno, perché mancano ancora dei pacchetti da installare e delle configurazioni da eseguire. Lo shield LCD, per comunicare con Raspberry Pi utilizza l'integrato MCP2307; questo rende disponibili 16 pin di I/O aggiuntivi rispetto al GPIO di Raspberry Pi, pilotabili da programma per mezzo del bus I²C. Come ormai sappiamo, il driver del bus I²C deve essere attivato manualmente, operazione che eseguiremo subito, assieme all'installazione delle utility di servizio per la gestione del bus I²C stesso. Lo shield LCD è stato presentato nel numero 178 di Elettronica In, in un articolo nel quale è contenuto un riquadro che descrive le modalità di utilizzo dell'integrato MCP23007. Ora spegniamo Raspberry Pi con il comando:

`shutdown -h now`

Togliamo l'alimentazione e montiamo lo shield sul connettore GPIO di Raspberry Pi, facendo attenzione che non vi siano contatti tra il circuito stampato e i connettori Ethernet ed USB. In caso di dubbio, copriamo detti connettori con del nastro isolante. Verifichiamo che i cursori dei DIP switch siano tutti sulla posizione "ON" (pin di address dell'integrato tutti collegati a GND). Ora possiamo riaccendere Raspberry Pi e ricollegarci come in precedenza. Utilizziamo Putty per aprire una finestra di terminale dove ci logghiamo con l'utente "root". Per abilitare il driver per il bus I²C e per riconoscere l'integrato MCP23017, rimandiamo alle istruzioni



Fig. 8

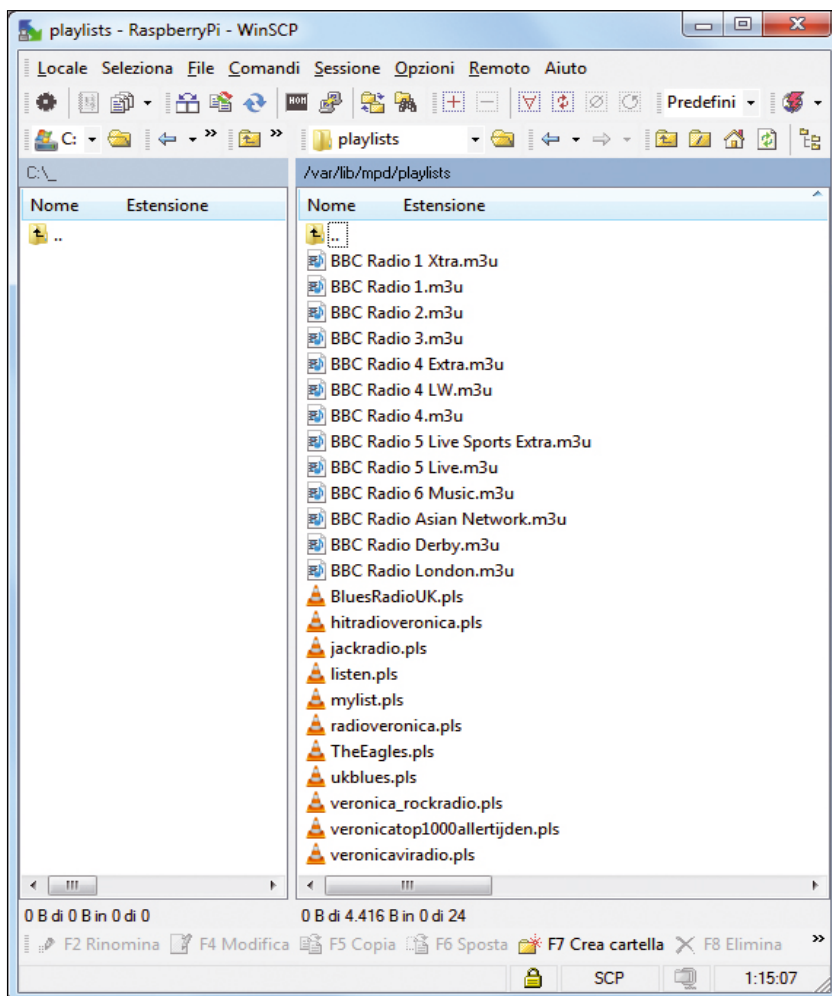


Fig. 9

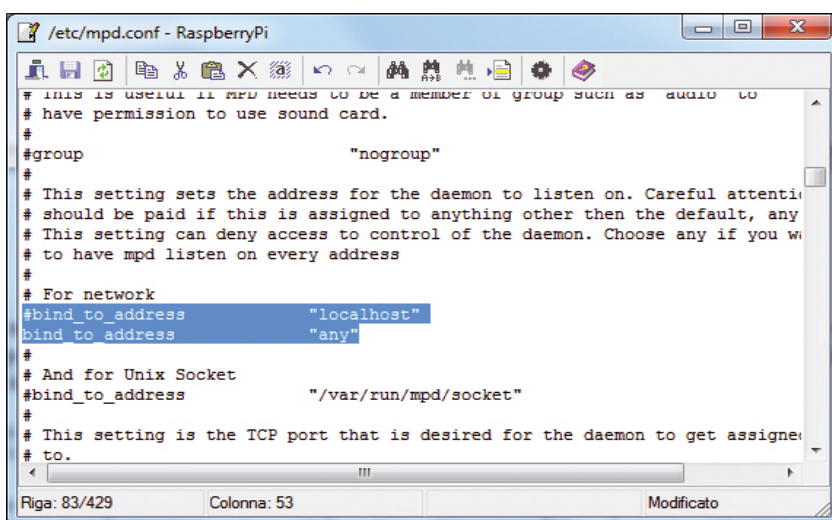


Fig. 10

Il formato dei file PLS (playlist)

Il formato file PLS non contiene direttamente i brani musicali ma i puntatori e gli indirizzi necessari a raggiungere le risorse remote. Una forma particolare di brano musicale sono gli streaming delle radio Internet. Esistono svariate sorgenti radio e musicali in streaming su internet e moltissime forniscono direttamente i record PLS da inserire nella propria playlist. Il principale scopo dei file PLS è essere utilizzati in un media player per ascoltare la propria radio preferita. Un elenco di URL di radio italiane è visibile all'indirizzo <http://simplyeng.com/2007/11/breve-lista-di-flussi-streaming-di-radio-italiane/>. Un'altra fonte di indirizzi di radio in streaming è <http://www.listenlive.eu/italy.html>.

Vediamo come comporre il nostro file PLS a partire dagli URL delle radio in streaming; i file PLS del server MPD si trovano nella cartella `/var/lib/mpd/playlists`. I file contenuti in questa cartella devono avere estensione **.pls** ed il formato descritto di seguito. I file PLS devono iniziare sempre con l'identificatore:

```
[playlist]
```

al quale segue il parametro che indica quante entrate (URL) sono presenti nello stesso file:

```
NumberOfEntries=2
```

Il parametro `Version` deve essere sempre impostato a 2. Infine segue la lista delle sorgenti in streaming. Il parametro `Filex` contiene l'URL della sorgente streaming della radio, mentre il parametro `Titlex` contiene la descrizione che vogliamo apparire sul display dell'LCD quando selezioniamo la radio.

Il parametro `Lengthx` contiene la lunghezza del brano. Per una radio indichiamo sempre "-1" che sta per "illimitato". La "x" di ciascun parametro sta per il numero progressivo della "entry", fino al numero massimo indicato in `NumberOfEntries`.

Nel Listato seguente vediamo un esempio di file playlist.

```
[playlist]
NumberOfEntries=5
File1=http://shoutcast.rtl.it:3010/stream/1/
Title1=RTL 102.5
Length1=-1
File2=http://shoutcast.rtl.it:3020/stream/1/
Title2=RTL 102.5 Classics
Length2=-1
File3=http://shoutcast.rtl.it:3040/stream/1/
Title3=RTL102.5 Groove
Length3=-1
File4=http://shoutcast.rtl.it:3030/stream/1/
```

contenute nei numeri 178 e 183 di Elettronica In (articoli "LCD shield" e "shield I/O") oppure nel "libro". Alla fine possiamo verificare se sul bus I²C è visibile l'integrato, impartendo il comando seguente:

```
i2cdetect -y 0
```

 per RaspberryPi rev. 1

Se state utilizzando Raspberry Pi rev. 1, il comando da utilizzare è:

```
i2cdetect -y 1
```

Dovrete ottenere un risultato simile a quello visibile in **Fig. 6**, dove l'indirizzo 0x20 identifica l'integrato MCP23017.

INSTALLAZIONE MUSIC PLAYER DAEMON

Ora è tempo di installare il server Music Player Daemon e la sua interfaccia client. Essendo

pacchetti compresi nella distribuzione Raspbian, li installiamo semplicemente con il comando:

```
apt-get install mpd mpc
```

Aspettiamo pazientemente che vengano installati i pacchetti e tutte le loro dipendenze (**Fig. 7**). Per configurare e provare per la prima volta il funzionamento di MPD, utilizziamo un file "playlist" presente nella cartella `/home/pi/radio/playlists` che abbiamo scaricato in precedenza: per esempio, il file `ukblues.pls`. La cartella dove vengono memorizzate le playlist di MPD è `/var/lib/mpd/playlists`. Per copiare il contenuto della cartella di origine nella cartella di destinazione digitiamo il comando di copia:

```
cp /home/pi/radio/playlists/ /var/lib/mpd/playlists/
```

come mostrato in **Fig. 8**, ottenendo il risultato visibile in **Fig. 9**. Ora apriamo, con WINScp, il file di configurazione di MPD: lo troviamo nella cartella `/etc/` con il nome `mpd.conf`. Modifichiamo il parametro `bind_to_address`, da

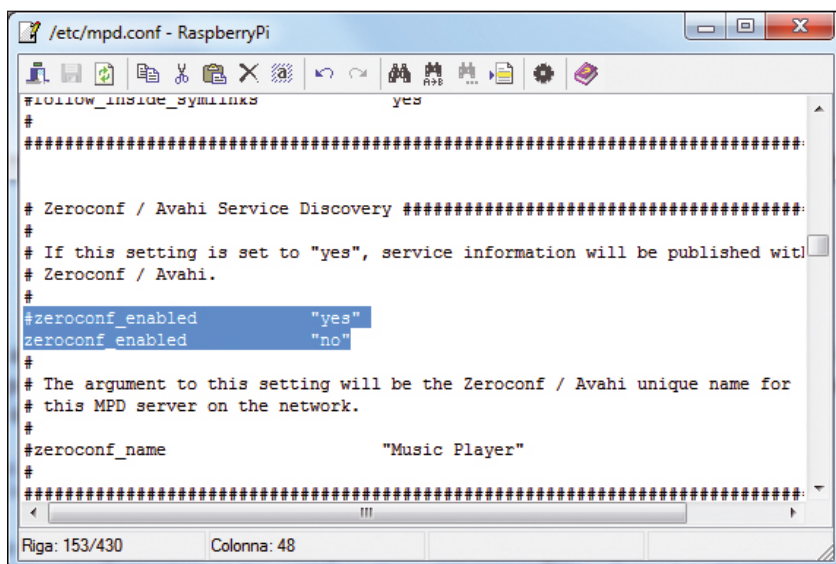


Fig. 11

Fig. B

```
Title4=RTL 102.5 Italian Style
Length1=-1
File5=http://shoutcast.rtl.it:3070/stream/1/
Title5=RTL 102.5 Cool
Length1=-1
```

Possiamo creare quanti file playlist desideriamo, ciascuno con una o più entrate. Il programma tenterà di caricare tutti i file che trova nella cartella `/var/lib/mpd/playlists`. A questo riguardo una avvertenza sul funzionamento del server MPD.

Un file playlist mal configurato può bloccare il funzionamento del server, inoltre non tutti i formati di streaming sono supportati dal server stesso.

In caso di sospetti malfunzionamenti, fermate il server MPD e andate a vedere il file di log `/var/log/mpd/mpd.log` con il comando:

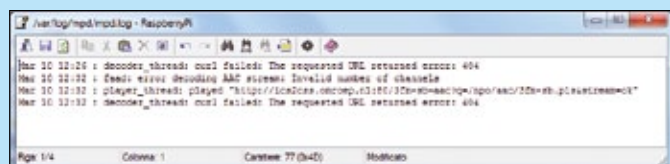


Fig. A

“localhost” ad “any”:

```
#bind_to_address "localhost"
bind_to_address "any"
```

Questa opzione permette il binding a indirizzi IPV6 (Fig. 10). Modifichiamo anche il parametro `zeroconf_enabled`, da “yes” a “no”:

```
#zeroconf_enabled "yes"
zeroconf_enabled "no"
```

in modo da poter utilizzare MPD senza esporre un nome host pubblico; in caso contrario, su Raspberry Pi, potremmo ottenere errori in fase di avvio del server (Fig. 11).

Salviamo il file di configurazione di MPD e verifichiamo se, fino a questo punto, tutto funziona a dovere. Collegiamo all’uscita audio di Raspberry Pi un altoparlante amplificato o una cuffia stereo. Attiviamo il server MPD con il comando (Fig. 12):

```
service mpd start
```

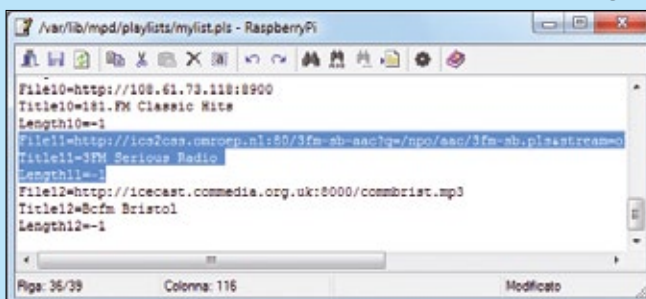
e digitiamo i due comandi seguenti (Fig. 13):

```
mpc load ukblues.pls
mpc play 1
```

Dovremmo cominciare a sentire della musica uscire dall’altoparlante o dalle cuffie. Se è così, possiamo procedere con le altre fasi dell’installazione. Il pacchetto fornito da Bob Rathbone include una serie di programmi in grado di interfacciare diversi display LCD collegati a Raspbe-

ry Pi, sia direttamente sul GPIO che via bus I²C. Come descritto nel numero 178, lo shield LCD è compatibile con le librerie Adafruit, dalle quali è stato derivato il programma di gestione `ada_radio.py`, incluso nel pacchetto citato. Nello stesso pacchetto sono inclusi i programmi necessari al funzionamento di `ada_radio.py`:

- `ada_radio.py`; è il programma principale per la gestione del



```
tail -f /var/log/mpd/mpd.log
```

in modo da leggere solo le ultime registrazioni del log stesso, oppure aprendolo da WinSCP (Fig. A). Uno degli errori più ricorrenti è che l’URL indicato nel file playlist non punta alla sorgente streaming ma a qualcos’altro. Molto spesso l’URL punta al file PLS della stazione radio.

Per risolvere il problema è sufficiente reperire il corretto URL, modificare il file e rilanciare il server MPD, oppure eliminare l’entrata errata (Fig. B).

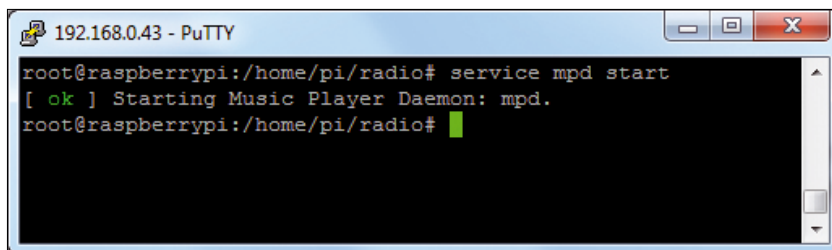


Fig. 12

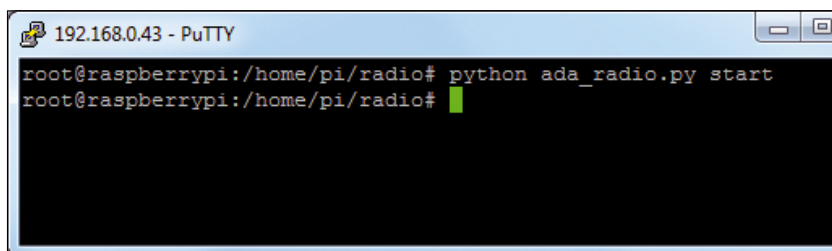


Fig. 13

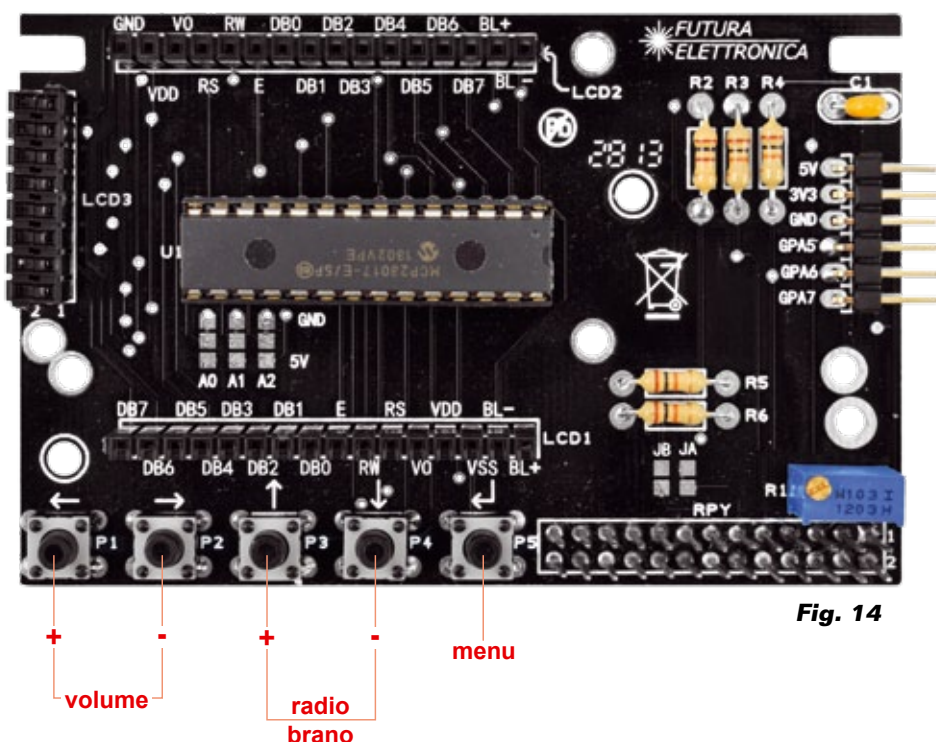


Fig. 14

display LCD;

- *ada_lcd_class.py*; è la classe di gestione LCD e pulsanti via bus I²C;
- *PC_class.py*; è la classe di gestione del bus I²C di Adafruit Industries (rinominata);
- *test_ada_lcd.py*; è un programma di test per verificare il funzionamento dell'LCD.

Come tutti i programmi realizzati in ottica "server" o "demone", *ada_radio.py* offre i comandi per la gestione del server. Di questi:

```
/home/pi/radio/ada_radio.py start
```

lancia il server di gestione dell'LCD;

```
/home/pi/radio/ada_radio.py stop
```

interrompe il funzionamento del server di gestione dell'LCD;

```
/home/pi/radio/ada_radio.py status
```

visualizza lo stato di funzionamento del server di gestione dell'LCD;

```
/home/pi/radio/ada_radio.py version
```

visualizza la versione software.

UTILIZZO DELLO SHIELD LCD

Lo shield LCD ha cinque pulsanti, uno dei quali viene utilizzato per la gestione del menu, mentre gli altri quattro permettono di impostare e selezionare le opzioni disponibili all'interno di ciascuna funzione di menu. In Fig. 14 è visibile la descrizione dei pulsanti presenti sullo shield LCD mentre nella Tabella 1 sono descritte le funzionalità assegnate ai singoli pulsanti a fronte di ciascuna scelta di menu.

INSTALLARE L'INTERFACCIA WEB

I componenti principali per

Tabella 1

Voce di menu	Destra	Sinistra	Su	Giù
Modo = TIME (Fig. 15 e 16) Linea 1: Data Ora Linea 2: Radio o Brano	Volume "+"	Volume "-"	Radio/Brano su	Radio/Brano giù
Modo = SEARCH (Fig 17) Sorgente = RADIO Linea 1: Search Linea 2: Radio Station	Volume "+"	Volume "-"	Scroll up radio station	Scroll down radio station
Modo = SEARCH Sorgente = MUSIC LIBRARY Linea 1: Search Linea 2: MusicTrack/Artist	Scorre autori in su	Scorre autori in giù	Scorre brani in su	Scorre brani in giù
Modo = SOURCE (Fig. 18) Linea 1: Input Source Linea 2: Internet Radio o Music Library	Volume "+" Mute	Volume "-" Mute	Alterna Radio e Music Library	Alterna Radio e Music Library
Modo = OPTIONS (Fig. 19) Linea 1: Selezione menu Linea 2: <opzioni> Le opzioni sono Random, Consume, Repeat or Reload Music:	Alterna modo on e off	Alterna modo on e off	Alterna Random, Consume, and Repeat	Alterna Random, Consume, and Repeat
Modo = RSS Linea 1: Data Ora Linea 2: RSS feed	Volume "+" Mute	Volume "-" Mute	Radio/Brano su	Radio/Brano giù
Modo = IP address (Fig. 20) Linea 1: IP address Linea 2: Station or Track	Volume "+" Mute	Volume "-" Mute	Radio/Brano su	Radio/Brano giù

gestire l'interfaccia web della nostra applicazione sono il server web e le pagine ed i programmi di gestione dell'applicazione vera e propria. Come server web utilizziamo Apache, una nostra vecchia conoscenza che abbiamo presentato in diversi articoli e nel libro citato all'inizio.

Per l'installazione e la configurazione del server Apache rimandiamo al "libro". Ricordiamo solo il comando principale per l'installazione del pacchetto e dei moduli aggiuntivi che ci occorrono:

```
apt-get update
apt-get install apache2 php5 libapache2-mod-php5
```

Terminata l'installazione e la configurazione, verifichiamo il corretto funzionamento del server: lo facciamo utilizzando un browser web e scrivendo nella barra degli indirizzi l'IP address di Raspberry Pi. La pagina visibile in Fig. 21 ci assicura che tutto funziona bene.

Ora è il momento di caricare l'applicazione web client di MPD: all'indirizzo <http://mpd.wikia.com/wiki/Clients> ne sono elencate diverse, perché per i più esperti può essere interessante provarne più di una. Noi abbiamo scelto ancora una volta quella personalizzata da Bob Rathbone. Dalla finestra di terminale andiamo alla cartella `/home/pi/radio` e scarichiamo le pagine mediante il comando:

```
wget http://www.bobrathbone.com/
raspberrypi/source/pi_radio_web.tar.gz
```

Scompattiamo il file compresso nella stessa cartella, usando il comando (Fig. 22):

```
tar -xvf /home/pi/radio/pi_radio_web.tar.gz
```

Al termine troveremo i file correttamente installati nelle cartelle `/var/www` e `/usr/lib/cgi-bin`.

Se necessario, lanciamo i server MPD e `ada_radio.py`, mediante i comandi (Fig. 23):

```
service mpd start
/home/pi/radio/ada_radio.py start
```

A questo punto possiamo tornare al browser e ridigitare l'indirizzo di Raspberry Pi nella finestra URL. Come risultato, dovremo vedere la pagina visibile in Fig. 24. Clicchiamo sul tab "Web interface" per ottenere la pagina visibile in Fig. 25. Facciamo clic sulle stazioni che vogliamo ascoltare: i pulsanti "Radio" e "Music" permettono di selezionare la sorgente audio.

CONFIGURIAMO L'INTERFACCIA WIFI

Per rendere portatile la nostra radio Internet, dobbiamo renderla indipendente dalla connessione di rete via cavo. Dobbiamo collegare un dongle WiFi e configurare il collegamento al nostro Access Point. Come dongle WiFi abbiamo utilizzato il modello 2846-WIFIDONGLE disponibile presso Futura Elettronica: di dimensioni minimali e particolarmente adatto ad essere utilizzato con Raspberry Pi, non richiede driver aggiuntivi e viene automaticamente

Fig. 15

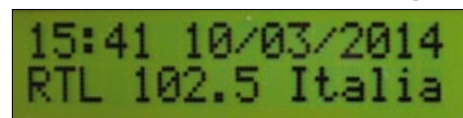


Fig. 16

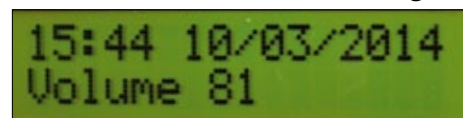


Fig. 17

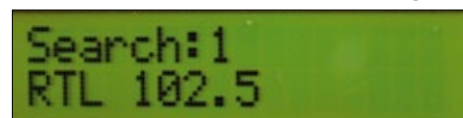


Fig. 18

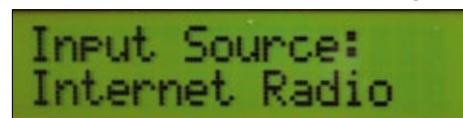


Fig. 19

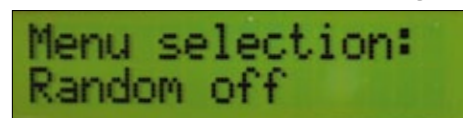


Fig. 20

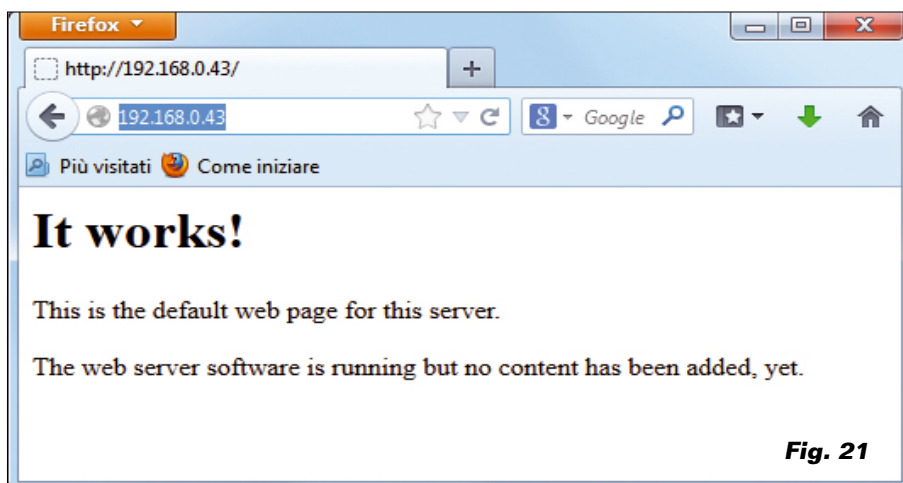
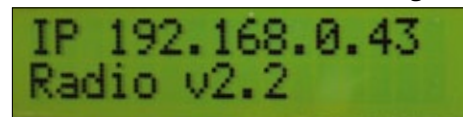


Fig. 21

```

root@raspberrypi:/home/pi/radio# wget http://www.bobrathbone.com/raspberrypi/source/pi_radio_web.tar.gz
--2014-03-07 17:36:56-- http://www.bobrathbone.com/raspberrypi/source/pi_radio_web.tar.gz
Resolving www.bobrathbone.com [www.bobrathbone.com]... 84.85.133.20
Connecting to www.bobrathbone.com (www.bobrathbone.com)[84.85.133.20]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2463140 (2.3M) [application/x-gzip]
Saving to: 'pi_radio_web.tar.gz'

100%[=====] 2,463,140 78.6K/s in 29s

2014-03-07 17:37:26 (83.3 KB/s) - 'pi_radio_web.tar.gz' saved [2463140/2463140]

root@raspberrypi:/home/pi/radio#

```

Fig. 22

```

root@raspberrypi:/home/pi/radio# tar -xvf /home/pi/radio/pi_radio_web.tar.gz

```

Fig. 23

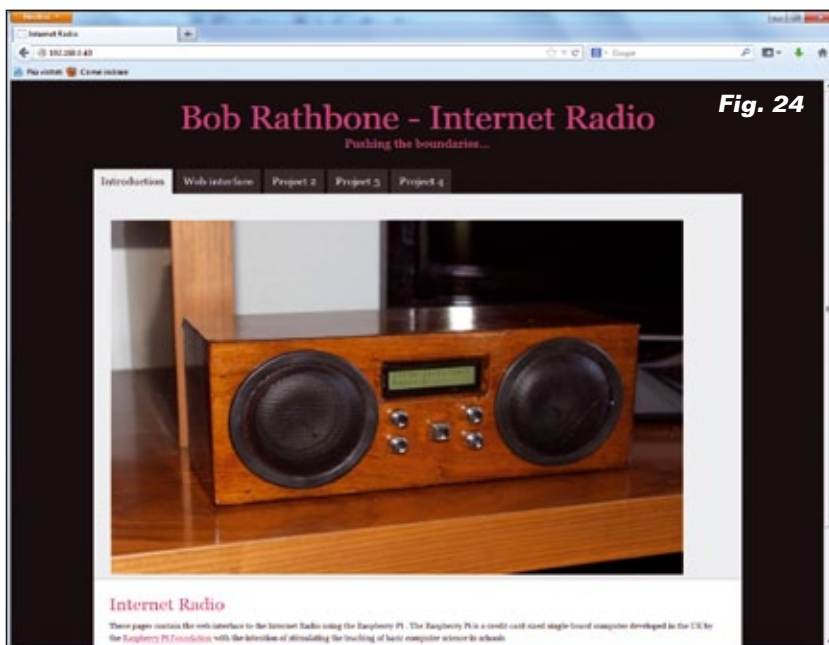


Fig. 24

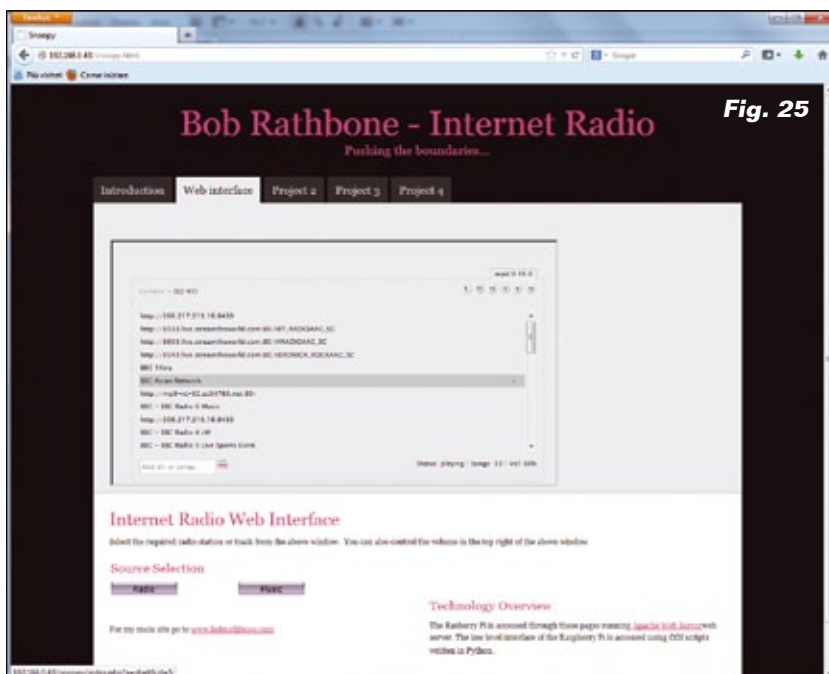


Fig. 25

riconosciuto una volta inserito in una presa USB di Raspberry Pi. Notate che quando si inserisce un dispositivo USB in Raspberry Pi, avviene il reboot del sistema operativo, probabilmente dovuto alla caduta di tensione derivante dall'inserimento. Ciò non accade, in genere, se si utilizza un Hub USB autoalimentato (ma non è detto...). Per precauzione, quindi, prima di collegare il dongle WiFi è bene spegnere correttamente Raspberry Pi, inserire il dongle e riaccenderla. Al riavvio, ricordate di lanciare nuovamente i server. Possiamo controllare il corretto riconoscimento del dongle con il comando:

lsusb

Invece con il comando:

ifconfig

verifichiamo che il dongle sia stato installato con il nome di periferica *wlan0*. Passiamo, ora, a configurare l'interfaccia wireless affinché possa connettersi al nostro Access Point: ciò richiede di intervenire su un paio di file di configurazione. Il primo è "interfaces" che si trova nella cartella */etc/network*. Lo apriamo e verifichiamo che sia così composto:

```

auto lo
iface lo inet loopback
iface eth0 inet dhcp
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp

```

Nelle nuove versioni di Raspbian, il file dovrebbe già essere configurato come descritto, mentre nelle più vecchie potrebbe essere diverso, nel qual caso copiate la sezione di configurazione di *wlan0* così come descritto. Ora configuriamo le credenziali

```

192.168.0.43 - PuTTY
root@raspberrypi:~# lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 0bda:8176 Realtek Semiconductor Corp. RTL8188CUS 802.11n WLAN Adapter
Bus 001 Device 005: ID 1a40:0201 Terminus Technology Inc. FX 2.1 7-port Hub
root@raspberrypi:~#

```

Fig. 26

```

192.168.0.43 - PuTTY
wlan0    Link encap:Ethernet HWaddr e0:4e:04:19:b7:f3
         UP BROADCAST MULTICAST MTU:1500 Metric:1
         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@raspberrypi:~#

```

Fig. 27

Tabella 2

ssid	SSID dell'Access Point
scan_ssid	Il valore 1 rende il nome SSID visibile in broadcast mentre il valore 2 nasconde il nome SSID
psk	Password WIFI
proto	Tipo di protezione RSN o WPA. RSN per WPA2 e WPA per WPA1.
key_mgmt	WPA-PSK, WPA-EAP (rispettivamente pre-shared o enterprise) oppure NONE
pairwise	CCMP o TKIP (rispettivamente WPA2 or WPA1)
auth_alg	Opzione OPEN richiesta per WPA e WPA2 (altre opzioni, SHARED e LEAP)

di accesso alla nostra rete wireless; il file interessato è *wpa_supplicant.conf*, situato nella cartella */etc/wpa_supplicant*. Aprendolo dovremmo trovarci solo un paio di linee di configurazione:

```
ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev
update_config=1
```

Sotto queste linee aggiungete le seguenti:

```
network={
ssid="VOSTRO_SSID"
scan_ssid=1
psk="VOSTRA_CHIAVE"
proto=RSN
key_mgmt=WPA-PSK
pairwise=CCMP
auth_alg=OPEN
}
```

Chiaramente dovete sostituire i valori "VOSTRO_SSID" e "VOSTRA_CHIAVE" rispettivamente con il nome del vostro SSID e la relativa password. Questa configurazione è valida per una rete protetta con chiave WPA2. Per altre configurazioni, consultate la **Tabella 2** e gli esempi di configurazione reperibili all'indirizzo web http://www.lsi.upc.edu/lcsi/Manuales/wireless/files/wpa_supplicant.conf.

Ora che abbiamo configurato la rete wireless, possiamo eseguire il reboot per rendere attive le configurazioni oppure possiamo disattivare e riattivare l'interfaccia WiFi con i comandi:

```
ifdown wlan0
ifup wlan0
```

Per verificare se il nostro Access Point compare nella lista degli Access Point raggiungibili, utilizziamo il comando:

```
iwlist wlan0 scanning | grep ESSID
```

```
ESSID:"APFXX"
ESSID:"Alice-XXXXXXX"
ESSID:"APFXXXXXX2"
ESSID:"APFXXXXX1"
```

Per visualizzare l'indirizzo di rete assegnato dall'Access Point al dongle WiFi, digitate il comando:

```
ifconfig
```

Per fare in modo che l'applicazione radio venga lanciata automaticamente al boot, creiamo il file *radio.sh* nella cartella */etc/init.d*. Inseriamo nel file le istruzioni:

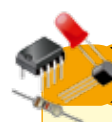
```
#!/bin/bash
/home/pi/radio/ada_radio.py start
```

Rendiamo eseguibile il file con il comando:

```
chmod a+x radio.sh
```

Tutto il processo è descritto nel "libro".

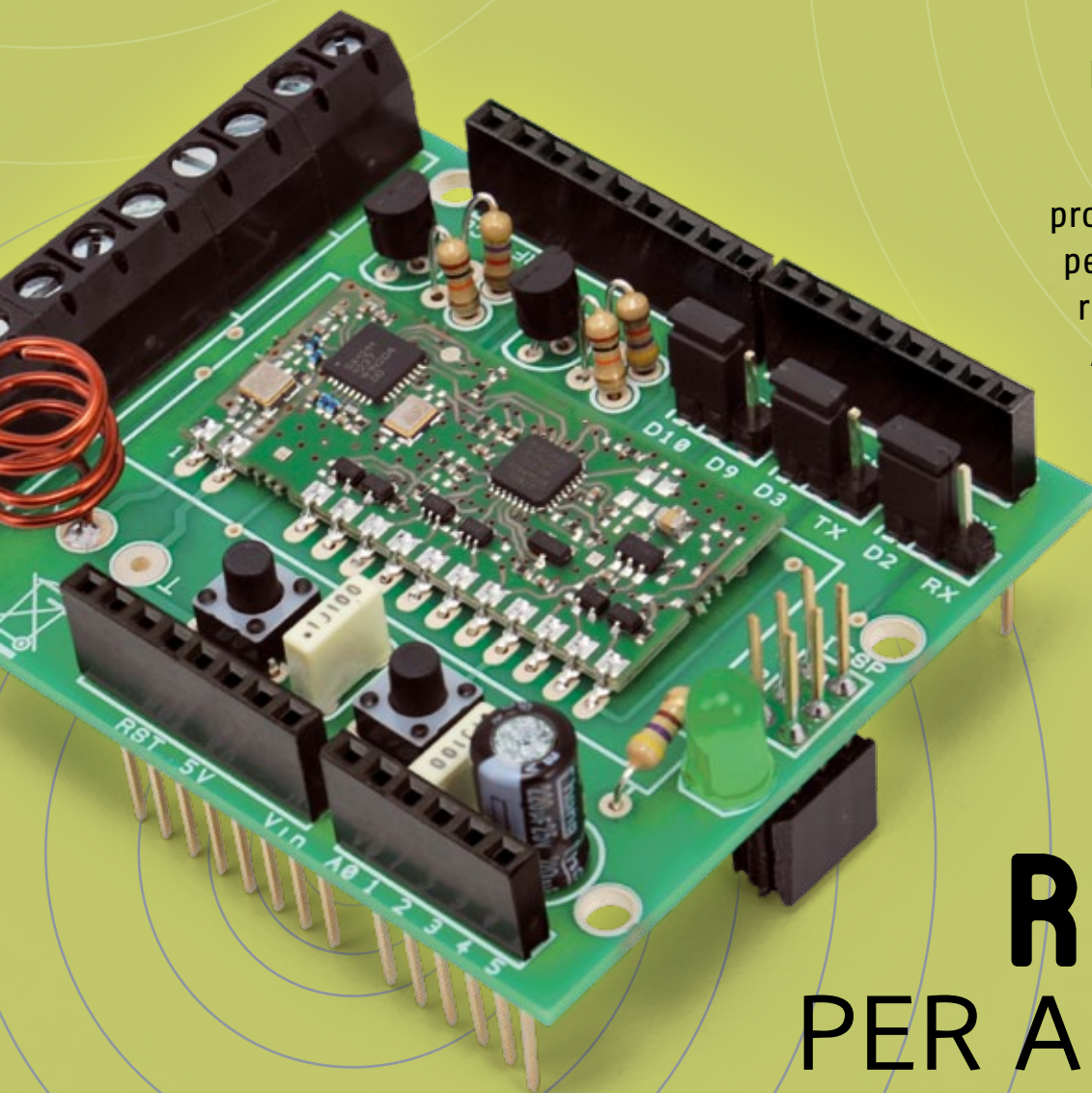
Nei prossimi articoli vedremo come aggiungere brani registrati su chiavetta USB ed introdurremo l'argomento del broadcast FM con una scheda appositamente realizzata, in modo da poter ritrasmettere via etere le nostre playlist, e non solo.



per il MATERIALE

Tutto il software utilizzato in questo progetto è scaricabile gratuitamente dai siti indicati nell'articolo. Il kit dell'LCD shield (cod. FT1074K, Euro 17,00) è stato descritto sul fascicolo 178 della rivista mentre la board Raspberry Pi Tipo B (cod. 7310-7568308RS) costa 38,00 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>



È giunto il momento di presentare lo shield che abbiamo progettato e realizzato per implementare una rete RFTide Aurel con Arduino. Seconda ed ultima puntata.

SHIELD RFTide PER ARDUINO

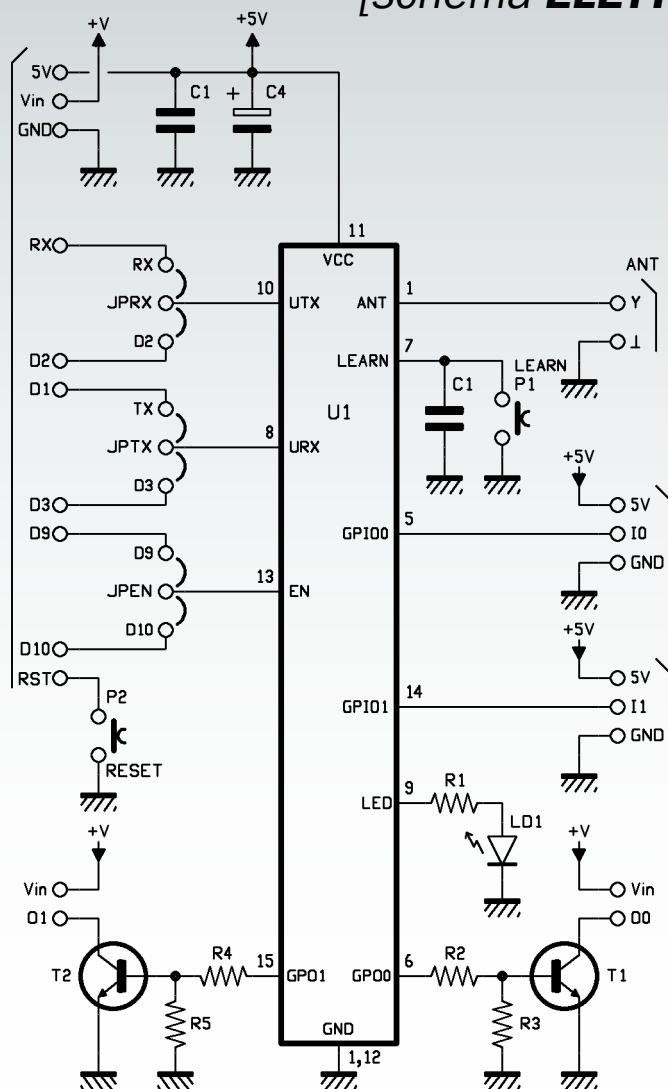
dell'Ing. MIRCO SEGATELLO

RFTide è un nuovo standard wireless promosso dall'Aurel per la realizzazione di reti PAN in banda 868 MHz, che promette nuove performance non ottenibili con i dispositivi che operano nella più affollata banda dei 2,4 GHz; ne abbiamo parlato nella scorsa puntata, facendo una panoramica sulle caratteristiche dei prodotti RFTide, ma anche proponendo una prima applicazione pratica che consiste nello sperimentare le funzioni di ricezione e trasmissione tramite una breakout board progettata per ospitare un modulo base RFTide. Ora è giunto il momento di arrivare al vero scopo di questo articolo, ossia interfacciare i moduli dell'Aurel con una scheda Arduino. Mettere in

contatto il mondo wireless delle reti mesh con il mondo Arduino, permetterà agli sviluppatori di creare nuove ed interessanti applicazioni, soprattutto nel campo della domotica.

Per raggiungere lo scopo abbiamo realizzato un apposito shield (chiamato "shield RFTide") fatto appositamente per gestire in modo semplice i moduli base RFTide con Arduino; montato su Arduino, mediante un'apposita libreria permette di rice-trasmettere dati ed emulare il funzionamento sia del Master che dello Slave, interfacciandosi tramite porta seriale e supportando il protocollo RFTide. Nell'applicazione specifica, faremo in modo che l'insieme Arduino-shield realizzi un'unità Master,

[schema **ELETTRICO**]



dato che come Slave, il modulo base RFTide può funzionare anche da solo, pilotando LED o relé direttamente. L'unità Master che realizzeremo provvederà alla creazione e alla gestione della rete, nonché alla configurazione con il sistema e al coordinamento dei moduli Slave. Arduino a sua volta può essere connesso a reti esistenti sia in modalità cablata che wireless, per un accesso da remoto alla rete RFTide con possibilità di sviluppo davvero notevoli; proprio l'abbinamento dello shield RFTide e di un Ethernet shield potrebbe consentire ad Arduino

di realizzare un Gateway RFTide. Meglio ancora, sarebbe montare il nostro shield su una Yùn, più adatta di Arduino tradizionale a gestire i protocolli di rete (TCP/IP) e di Internet (HTTP).

L'HARDWARE

Lo shield che abbiamo realizzato supporta i moduli con alimentazione a 5V in versione sia serializzata che non, e dispone di alcuni ponticelli per l'impostazione dell'interfaccia di comunicazione; ciò conferisce al progetto grande versatilità. La comunicazione con i moduli RFTide avviene in modalità se-

riale, con livelli TTL, alla velocità di 19.200 bps, e può facilmente essere gestita tramite la porta seriale hardware o software di Arduino.

I jumper JPRX e JPTX permettono di selezionare a quali pin di Arduino saranno connesse le linee dell'UART interno del modulo RFTide: posizionando il ponticello JPRX su RX, la linea TX del modulo sarà indirizzata verso il contatto 0 di Arduino, ovvero il pin di ricezione del modulo hardware di Arduino, al quale, lo ricordiamo, è connesso anche il modulo convertitore USB/seriale della scheda. Posizionando il jumper JPTX su TX, la linea RX del modulo sarà indirizzata verso il contatto 1 di Arduino, ovvero il pin di trasmissione del modulo hardware di Arduino, al quale è connesso nuovamente il modulo convertitore USB-seriale della scheda Arduino.

Come vedete, la comunicazione seriale si basa su un collegamento incrociato DTE-DTE, dove la linea di trasmissione di Arduino si connette a quella di ricezione del modulo e viceversa. In questa modalità Arduino può comunicare con il modulo RFTide, tuttavia un'eventuale comunicazione tra Arduino e il PC causerà un conflitto, in quanto il modulo UART del microprocessore di Arduino è già impiegato dal modulo RFTide.

Per consentire ad Arduino di comunicare liberamente con il PC, sia per il download dei programmi, sia per la funzione di Serial Monitor, lo shield prevede la possibilità di connettere le due linee TX ed RX del modulo RFTide a due generici pin digitali di Arduino D3 e D2, i quali, grazie alla libreria Softserial integrata nell'IDE di Arduino, possono essere utilizzati come seriale sof-

ware (UART virtuale) lasciando libera la comunicazione tra Arduino ed il PC.

Il jumper JPENI permette di assegnare la linea di Enable del modulo alla linea digitale D9 o D10 di Arduino; questa linea, portata a livello logico alto, permette di abilitare il modulo, che altrimenti rimarrebbe inattivo (modo a basso consumo di energia, ossia power down).

L'alimentazione dello shield viene prelevata direttamente dal pin +5V di Arduino; l'assorbimento di corrente è dell'ordine di soli 10 mA in ricezione e 33 mA in trasmissione, mentre in modalità di power down, la corrente assorbita non supera i 4 µA. I moduli Aurel necessitano di un'antenna esterna che può

essere connessa al pin Y dello shield; il modo più semplice per realizzare l'antenna, suggerito anche dalla stessa Aurel, è quello di saldare direttamente sulla piazzola uno spezzone di filo di rame rigido del diametro di 1 mm lungo 85 mm. La lunghezza di 85 mm permette di avere un'antenna accordata ad $\frac{1}{4}$ d'onda sulla frequenza dei 868 MHz. La portata è davvero notevole ed arriva a qualche centinaio di metri in aria libera; è più che sufficiente per coprire un'abitazione anche molto grande, senza ricorrere ai ripetitori.

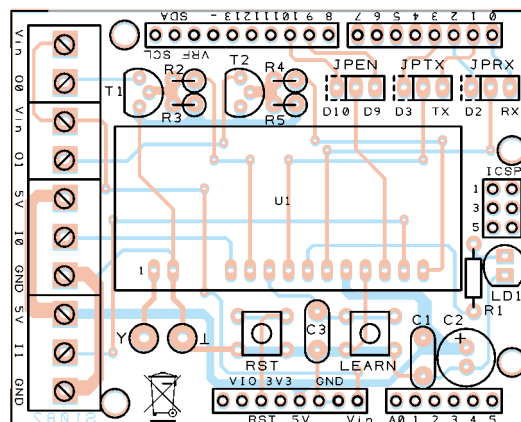
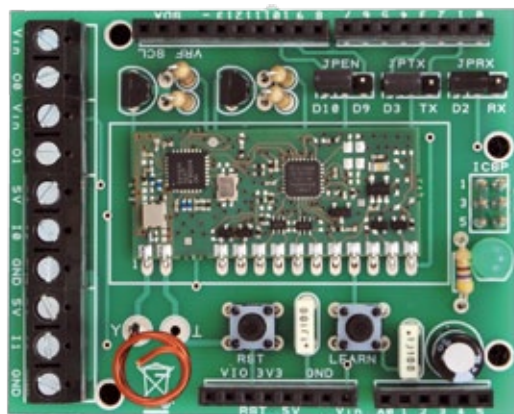
Se l'insieme Arduino-shield va inserito in un contenitore metallico, visto che farebbe da schermo bloccando le onde radio, conviene utilizzare un'antenna a

stilo flessibile specifica per la frequenza degli 869 MHz (come ad esempio la 8110-AG868, oppure la 8110-AG868-TA commercializzata dalla Futura Elettronica - www.futurashop.it).

Comunque, dalle prove effettuate abbiamo verificato che anche la semplice antenna a stilo realizzata con uno spezzone di filo di rame ripiegato sullo shield, assicura una portata più che ottimale, permettendo nel contempo di ridurre gli ingombri, così da poter montare ulteriori shield sopra quello RFTide. Va però detto che coprire lo shield RFTide con altri può ridurre la portata, a causa dell'effetto schermante che le piste degli shield applicati sopra determinano.

Sulla scheda sono disponibili

[piano di MONTAGGIO]



Elenco Componenti:

R1: 470 ohm
R2: 4,7 kohm
R3: 10 kohm
R4: 4,7 kohm
R5: 100 kohm
C1: 100 nF multistrato
C2: 220 µF 25 VL elettrolitico
C3: 100 nF 100 VL poliestere
T1: BC547

T2: BC547
P1: Microswitch
P2: Microswitch
LD1: LED 5 mm verde
U1: Modulo RFTIDE (RFT-868-5V)

Varie:
- Morsetto 2 poli (2 pz.)
- Morsetto 3 poli (2 pz.)

- Strip maschio 3 poli (3 pz.)
- Jumper (3 pz.)
- Strip M/F 3 poli (2 pz.)
- Strip M/F 6 poli (1 pz.)
- Strip M/F 8 poli (2 pz.)
- Strip M/F 10 poli (1 pz.)
- Antenna accordata 868 MHz
- Circuito stampato

Tabella 1 - Esempio di utilizzo con LCDshield della Futura Elettronica.

Shield	Pin utilizzati
RFTide	D2, D3, D10
LCD	AN3, D4, D5, D6, D7, D8, D9

Tabella 2 - Esempio di utilizzo con la Motorshield della Futura Elettronica.

Shield	Pin utilizzati
RFTide	D2, D3, D10
Motor	D4, D5, D8, D9

i due ingressi GPIN0 e GPIN1, funzionanti in logica positiva, quindi un eventuale contatto dovrà essere connesso tra il pin di ingresso e l'alimentazione +5V. Data la configurazione elettrica degli ingressi dei moduli Aurel (vedasi la puntata precedente) essi funzionano anche fornendo tra GND e ingresso una tensione continua di valore compreso tra qualche volt e 12 volt. Le due uscite GPOUT0 e GPOUT1 sono connesse alla base di due transistor utilizzati in modalità open-collector, utili per attivare piccoli carichi alimentati sia a +5V sia con Vin, come ad esempio dei relé.

La scheda riporta un pulsante di Reset collegato in parallelo a quello di Arduino, che sarebbe difficilmente accessibile una volta applicato lo shield. Con la configurazione degli I/O da noi consigliata, lo shield RFTide può funzionare senza conflitti anche montando su Arduino il Motorshield e l'LCDshield, entrambi prodotti e distribuiti dalla Futura Elettronica; la compatibilità è garantita impostando i jumper delle varie shield secondo la **Tabella 1** per il Motorshield e la **Tabella 2** per l'LCDshield. Lo shield RFTide non impegna la porta SPI e l'PC-Bus, che restano quindi a disposizione per ulteriori shield o per connettere Arduino ad altri sistemi.

I moduli RFTide non permettono una comunicazione radio in

transparent-mode, per cui non è possibile utilizzarli per sostituire una comunicazione seriale cablata con una wireless, come invece poteva avvenire con i moduli Xbee; infatti la trasmissione avviene solo mediante pacchetti di lunghezza fissa pari a 8 byte. I moduli Aurel sono programmati tramite comandi seriali che implicano l'invio di apposite stringhe (assieme a dei parametri) secondo uno standard proprietario non compatibile con lo standard AT utilizzato per i moduli Bluetooth e GPRS. Per una gestione semplice ed immediata, abbiamo pertanto deciso di scrivere un'apposita libreria che svincoli l'utente dalla conoscenza dettagliata dei moduli, se non per impieghi specifici. La libreria comunica con i moduli Aurel tramite la seriale software, pertanto i jumper saranno impostati nella modalità D2, D3 e la linea di Enable su D10. La libreria consente comunque di definire a piacere i pin usati per gestire i moduli in modo da lasciare la massima libertà nella gestione dei pin hardware, purché non siano quelli usati dalla seriale hardware; questo per agevolare anche quanti testeranno i moduli utilizzando semplicemente una basetta sperimentale e alcuni jumper.

Per i particolari dei collegamenti a livello hardware, si consiglia di leggere la puntata precedente, dove abbiamo spiegato come sono fatti i moduli Aurel RFTide. Vediamo ora, in dettaglio, come utilizzare la libreria, i cui sorgenti sono disponibili, assieme ai file di questo articolo, sul sito della rivista, ma possono essere scaricati anche presso il repository all'indirizzo <https://github.com/mircose/RFTide-shield-Arduino>. La disponibilità di un repository consente una più facile distribuzione di versioni successive della libreria, la quale, essendo ancora in fase di sviluppo, potrebbe subire modifiche.

Una volta scaricati i file, è sufficiente scompattarli all'interno della cartella *Libraries* di Arduino; per ogni chiarimento a riguardo, fate riferimento al sito <http://arduino.cc/en/Guide/Libraries>. Al termine delle operazioni vi ritroverete installata una cartella di nome RFTide, contenente i file *function.txt*, *keywords.txt*, *readme.txt*, *RFTide.h* e *RFTide.cpp* e una cartella di nome *Examples*.

Il file *readme.txt* contiene alcune informazioni sulla versione della libreria, mentre il file *function.txt* contiene un elenco di tutte le funzioni implementate; il file *RFTide.h* costituisce la libreria vera e propria.

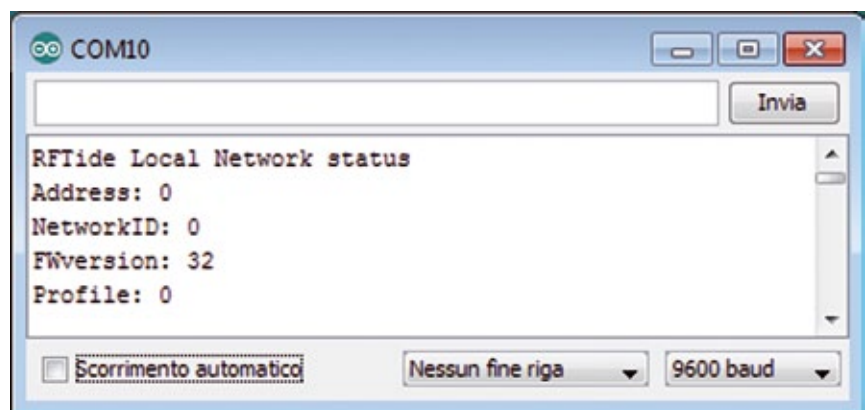


Fig. 1 - Schermata di Serial Monitor relativa all'esempio LocalStatus.

Durante il primo avvio dell'IDE di Arduino verrà caricata la nuova libreria e saranno resi disponibili gli esempi presenti nella cartella *Examples*.

Connettete lo shield ad Arduino e impostate correttamente i jumper, quindi collegate il tutto al PC servendovi dell'apposito cavo. Se il modulo RFTide è stato appena acquistato presenterà le impostazioni di fabbrica; se non conoscete le sue impostazioni, vi conviene prima resettarlo e riportarlo alla situazione di fabbrica (parametri predefiniti) tramite l'apposita procedura:

- 1) premete (per circa 2 secondi) il pulsante di LEARN presente sullo shield finché il LED LD1 si accende a luce fissa;
- 2) entro 15 secondi, premete (sempre per circa 2 secondi) di nuovo il pulsante di LEARN finché il LD1 inizia a lampeggiare.

Terminata la procedura di reset, il LED LD1 si spegne.

A questo punto è opportuno verificare che il modulo funzioni correttamente; allo scopo, interroghiamo il modulo locale utilizzando l'esempio chiamato *LocalStatus.ino*. Caricate l'esempio ed aprite Serial Monitor di Arduino, impostando una comunicazione a 9.600 bps; nella finestra di Serial Monitor sarà riportata la configurazione del modulo (Fig.1). Nel caso dell'esempio, in cui viene usato un modulo non serializzato, l'indirizzo e il NetworkID saranno zero, il firmware sarà la versione 3.2 e non andrà programmato alcun profilo specifico. Ora possiamo andare a gestire direttamente lo stato delle uscite e leggere lo stato degli ingressi caricando l'esempio *LocalIO*. È opportuno connettere almeno un LED e un pulsante allo shield, come si

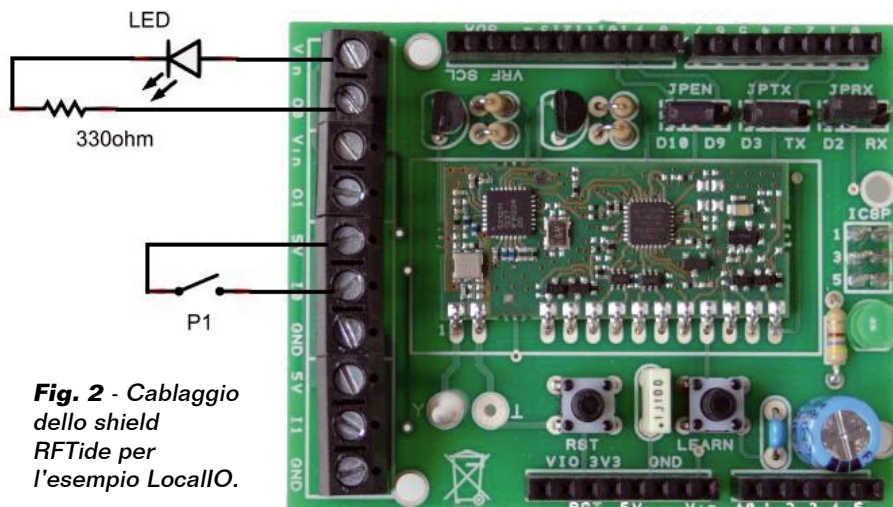


Fig. 2 - Cablaggio dello shield RFTide per l'esempio LocalIO.

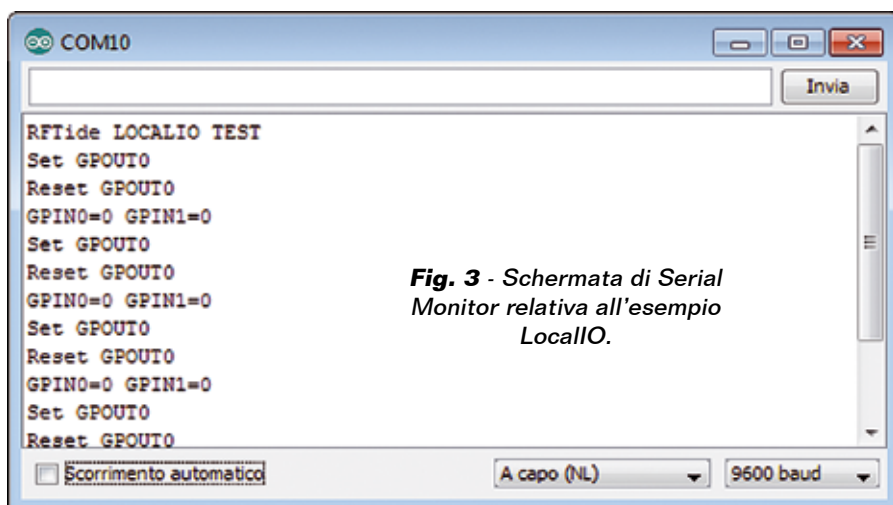
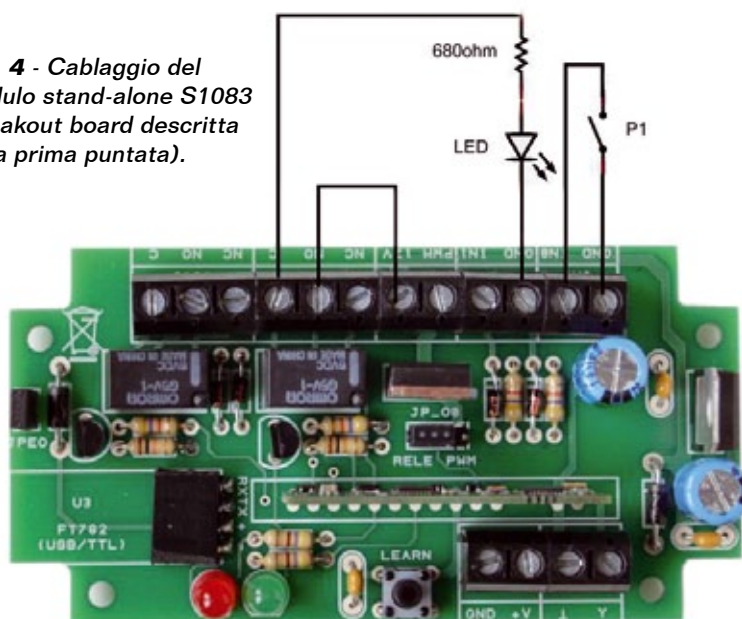


Fig. 3 - Schermata di Serial Monitor relativa all'esempio LocalIO.

Fig. 4 - Cablaggio del modulo stand-alone S1083 (breakout board descritta nella prima puntata).



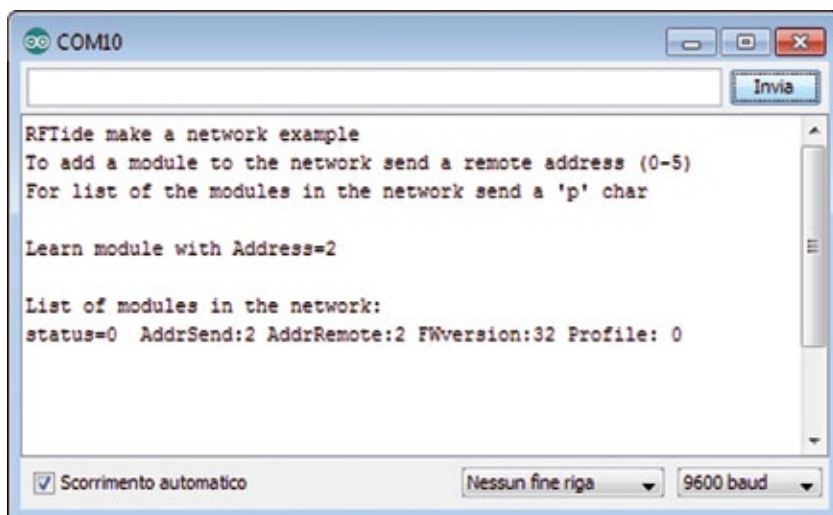


Fig. 5 – Schermata di Serial Monitor per l'esempio Network.

vede in **Fig. 2**. Lo sketch provvede ad azionare alternativamente l'uscita GPOUT0 e riportare lo stato dei due ingressi GPIN0 e GPIN1 come visibile in **Fig. 3**. A questo punto è interessante utilizzare un secondo modulo per creare una piccola rete wireless: il modulo sarà uno Slave realizzato mediante un circuito stand-alone come quello distribuito dalla ditta Futura Elettronica con il codice S1083. Come nel caso precedente, conviene interfacciare un LED ed un pulsante come riportato in **Fig. 4**. L'alimentazione potrà avvenire tramite un semplice alimentatore non stabilizzato che fornisca 9 volt cc. Assicuratevi che il modulo presenti le impostazioni di fabbrica; se così non fosse, applicate la procedura di reset descritta in precedenza. Adesso è necessario associare il nuovo modulo alla rete e per fare questo utilizziamo l'esempio chiamato Network: avviate SerialMonitor impostando una comunicazione a 9.600 baud e sul modulo remoto premete il pulsante di LEARN finché il LED lampeggia, quindi entro 15 secondi inviate, tramite il comando SEND di SerialMonitor, l'indirizzo da assegnare al modulo (ad esempio 2); allora il LED

lampeggerà per alcuni secondi e poi si spegnerà. L'indirizzo deve essere diverso da quello del modulo locale rilevato con l'esempio *LocalStatus*; per evitare ripetizioni potete visualizzare l'elenco dei moduli connessi alla rete RFTide: basta inviare, con il comando SEND, il carattere "p". Notate che il modulo locale non viene rilevato perché i moduli non possono trasmettere a loro stessi (**Fig. 5**).

A questo punto avete creato una rete con due moduli ed è possibile interagire con quello remoto così come abbiamo fatto in locale. Aprite e caricate l'esempio *RemoteIO*, che provvede ad attivare al-

ternativamente l'uscita GPOUT0 sul modulo remoto, oltre a leggere lo stato degli ingressi (**Fig. 6**). L'indirizzo del modulo con il quale interagire è riportato nella riga *byte remoteAddress = 2* che, ovviamente, dovrà essere modificata riportando l'indirizzo del modulo remoto assegnato con la precedente procedura di LEARN. Con la stessa procedura di associazione potete aggiungere alla rete PAN tutti i moduli che volete, purché ciascuno abbia un indirizzo univoco, poi è sufficiente specificare l'indirizzo del modulo con il quale si vuole interagire. Una particolarità della libreria è consentire diverse modalità per l'accesso allo stato dei pin GPIO. Nell'esempio *LocalIO*, la lettura dello stato degli ingressi può avvenire in tre modi differenti, il primo dei quali è:

```
IOSTATUS InOutStatus = rftide.localIoRead();
Serial.print(InOutStatus.getGPIN0);
```

In questo caso viene usata la funzione *localIoRead* per leggere lo stato di tutti i GPIO, mentre la funzione *INOUTStatus.getGPIN0* riporta il valore del pin GPIN0 estrapolato dalla lettura precedente.

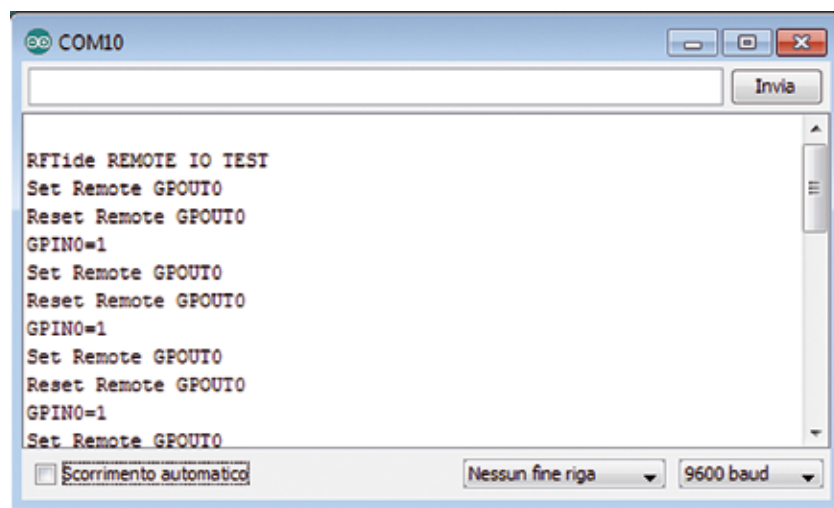


Fig. 6 – Schermata di Serial Monitor relativa all'esempio RemoteIO.

Elenco funzioni della libreria v 1.0

In questo riquadro trovate tutte le funzioni contenute nella versione 1.0 (da noi usata) della libreria per la gestione della rete con moduli RFTide.

Il secondo modo è il seguente:

```
Serial.println(rftide.localloRead().getGPIN0());
```

In questo caso si utilizza la funzione precedente *localloRead* ma con la modalità di accesso al contenuto della variabile *getGPIN0*. Il terzo modo, molto più intuitivo e immediato è:

```
Serial.println(rftide.getGPIN0());
```

dove viene usata la funzione *getGPIN0()* che riporta direttamente lo stato del pin GPIN0.

FUNZIONI READ/SEND PACKET

Vediamo ora come usare le funzioni avanzate per inviare e ricevere delle semplice stringhe di dati mediante la rete PAN. Utilizzate un modulo RFTide stand-alone ed inseritevi nell'apposito connettore l'adattatore USB-Seriale FT782M (si trova su www.futurashop.it) a sua volta connesso al PC con l'apposito cavo; esso alimenta anche la scheda stand-alone. Verificate quale porta COM virtuale viene creata alla connessione del convertitore USB-Seriale, quindi avviate il software ufficiale distribuito dalla Aurel di nome RFTide-v1.0 e specificate la porta COM di comunicazione di conseguenza (pulsante SET). Tramite il comando *Local Status*, verificate che il modulo venga correttamente riconosciuto. Dall'ambiente Arduino aprite l'esempio *ReadPacket* e caricatelo sulla scheda Arduino con inserito uno shield RFTide, aprite Serial Monitor ed impostate una velocità di comunicazione di 9.600 baud.

Usate il software della Aurel per inviare un pacchetto di dati allo shield: inserite nella casella Command Address l'indirizzo dello shield (ad esempio 0),

RFTide(byte RX_pin, byte TX_pin, byte EN_pin)	Constructor
void localloSet(byte payload) payload=0 set GPOUT0 payload=1 set GPOUT1	Set Output in local module
void localloReset(byte payload) payload=0 reset GPOUT0 payload=1 reset GPOUT1	Reset Output in local module
IOSTATUS localloRead() byte getGPIN0 (0=OFF, 1=ON) byte getGPIN1 (0=OFF, 1=ON) byte getGPOUT0 (0=OFF, 1=ON) byte getGPOUT1 (0=OFF, 1=ON) int getADC0 int getADC1	Return IO status in local module
NTSTATUS localStatus() byte getAddress unsigned long getNetworkID int getFWversion byte getProfile	Return status in local module
byte getLocalAddress()	Return address of the local module
int getLocalFWversion()	Return firmware version of the local module (format XYY)
byte getLocalProfile()	Return profile type of the local module (0=none 1=motor 2=PWM 3=metering)
void resetLocalDevice()	Reset local module
unsigned long getLocalNetworkID()	Return NetworkID of the local module
void remoteloSet(byte nodeAddress, byte payload) payload=0 set GPOUT0 payload=1 set GPOUT1	Set IO in remote module
void remoteloReset(byte nodeAddress, byte payload) payload=0 reset GPOUT0 payload=1 reset GPOUT1	Reset IO in remote module
IOSTATUS remoteloRead(byte nodeAddress) byte getGPIN0 (0=OFF, 1=ON) byte getGPIN1 (0=OFF, 1=ON) byte getGPOUT0 (0=OFF, 1=ON) byte getGPOUT1 (0=OFF, 1=ON) int getADC0 int getADC1	Return IO status in remote module
void progMessage(byte nodeAddress, byte nodeProfile)	Program remote module for associate to network nodeProfile: none, motor, PWM, metering, repeater
NTSTATUS ping(byte nodeAddress) byte getAddress unsigned long getNetworkID int getFWversion byte getProfile	Ping remote module and return remote status
byte getGPIN0()	Read value of GPIN0
byte getGPIN1()	Read value of GPIN1
byte getGPOUT0()	Read value of GPOUT0
byte getGPOUT1()	Read value of GPOUT1
int version()	Return a version of library format is XYY (100 is V1.00)
void enable()	Enable module
void disable()	Disable module
void sendPacket(byte nodeAddress, char payload[]) nodeAddress = address of remote module payload = 8 byte of data	Send packet to remote module
byte readPacket(byte inbyte[]) If valid data is available return address of remote module else return 255. Data received is saved in the inbyte array	Read fixed 8byte Packet received from remote module
byte status() ANSWER NOANSWER BADANSWER BRKANSWER RECPACKET	Return status of the last operation receive correctly answer no answer received bad data on receive buffer incomplete data on receive buffer receive packet from remote

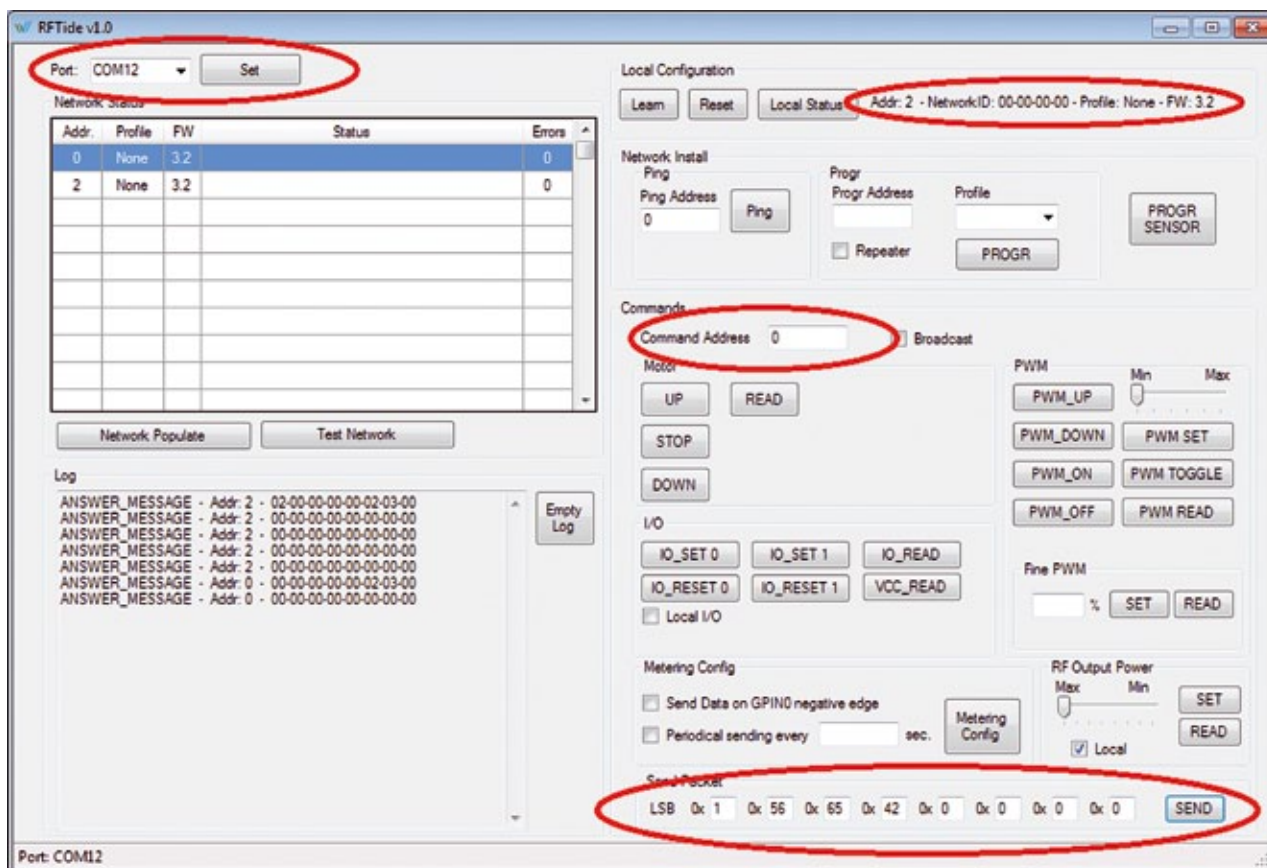


Fig. 7 – Schermata del software Aurel per inviare una stringa.

inserite i byte da inviare nelle caselle Send Packet, quindi fate clic sul pulsante SEND (Fig. 7). Su Serial Monitor saranno visualizzati i dati ricevuti e verrà indicato da quale indirizzo della rete sono stati spediti (Fig. 8). Aprite ora lo sketch SendPacket e specificate sulla riga `byte remoteAddress = 2` l'indirizzo del modulo remoto al quale volete inviare i dati, poi aprite Serial Monitor ed impostate come modalità di invio "A capo (NL)", quindi inserite la stringa da inviare (massimo 8 caratteri) e fate clic su SEND (Fig. 9 e Fig. 10). La stringa sarà inviata al modulo remoto che la ripeterà sulla porta seriale e potrà essere visualizzata attraverso il software Aurel (Fig. 11 e Fig. 12). Vediamo più in dettaglio la struttura della libreria

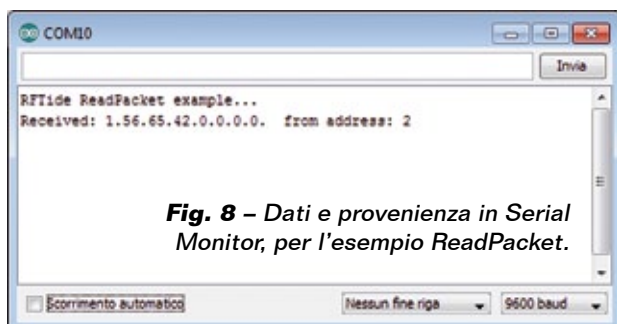


Fig. 8 – Dati e provenienza in Serial Monitor, per l'esempio ReadPacket.

e come utilizzarla al meglio. Le prime righe che è importante riportare nello sketch sono quelle che includono le librerie:

```
#include <RFTide.h>
#include <SoftwareSerial.h>
```

La sintassi che permette di creare l'oggetto RFTide è la seguente: `RFTide rftide(RX, TX, EN)`, in cui RFTide è il nome associato all'oggetto, mentre i valori RX, TX ed EN specificano i pin utilizzati per la comunicazione tra Arduino e il modulo; possono essere specificati tutti i pin digitali tranne 0 e 1 già occupati dalla seriale hardware. Ad esempio, la funzione che attiva un'uscita sul modulo remoto è chiamata `remoteIoSet` e la giusta sintassi prevede che siano specificati come parametri l'indirizzo remoto del modulo al quale si vuol accedere e il pin remoto che si vuole attivare:

```
rftide.remoteIoSet(remoteAddress, GPOUT0)
```

Notate che il termine GPOUT0 si può usare al posto della costante numerica 0 (zero) in quanto all'interno della libreria sono specificate le parole chiave:

```
//define RFTide pin
```

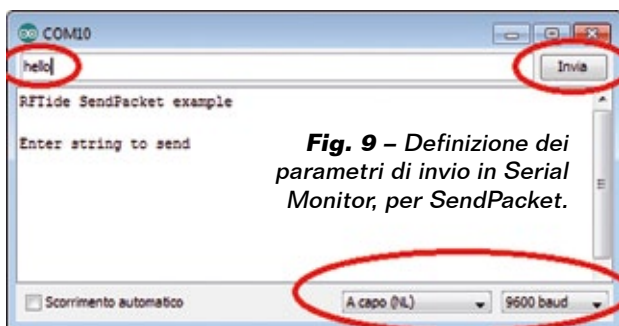


Fig. 9 – Definizione dei parametri di invio in Serial Monitor, per SendPacket.

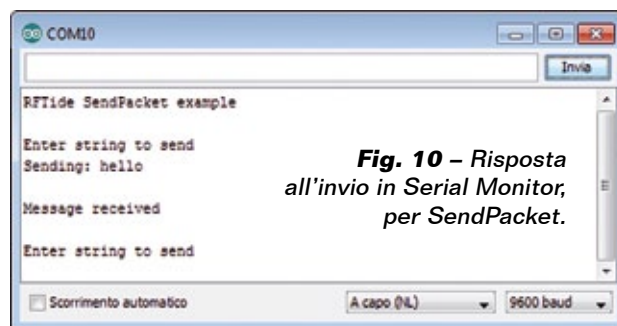


Fig. 10 – Risposta all'invio in Serial Monitor, per SendPacket.

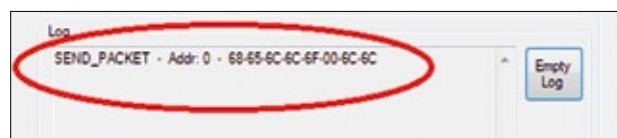
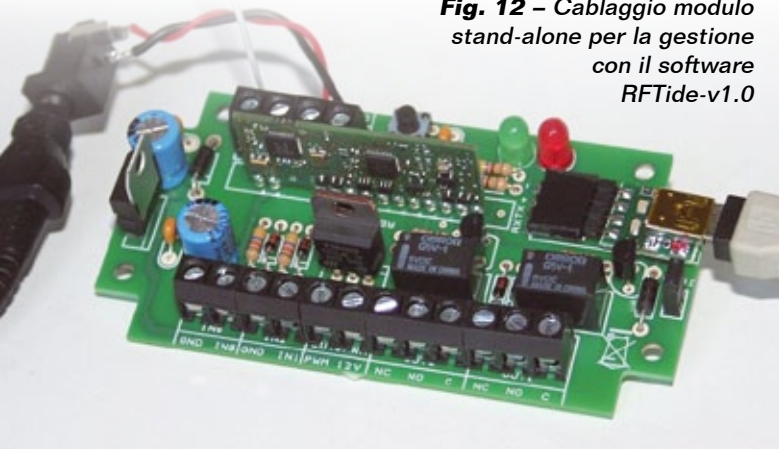


Fig. 11 – Schermata del software Aurel alla ricezione della stringa.

Fig. 12 – Cablaggio modulo stand-alone per la gestione con il software RFTide-v1.0



```
#define GPINO 0
#define GPIN1 1
#define GPOUT0 0
#define GPOUT1 1
```

Appena richiamata questa funzione, la libreria crea il pacchetto di 11 byte che sarà spedito al modulo remoto:

```
outPacket[CommandLSB]=CMD_RIOSET;
outPacket[CommandMSB]=0x00;
outPacket[Address]=nodeAddress;
outPacket[Byte0]=payload;
```

Il pacchetto così composto viene spedito al modulo remoto e subito è attivata la funzione che permette di analizzare il pacchetto di ritorno, sempre inviato dal modulo remoto a conferma della ricezione dei dati. L'analisi del pacchetto di ritorno permette di capire come è andata la trasmissione: se il pacchetto arriva nei tempi prestabiliti e contiene i byte identificativi di un pacchetto di risposta, la trasmissione è andata buon fine, altrimenti la variabile STATUS contiene il codice dell'errore in trasmissione: pacchetto perso, incompleto o corrotto.

```
//define status value
```

```
#define ANSWER 0//receive correctly answer
#define NOANSWER 1//no answer received
#define BADANSWER 2//bad data on receive buffer
#define BRKANSWER 3//incomplete data on receive buffer
#define RECPACKET 4//receive packet
```

Se viene ricevuto un pacchetto di dati la variabile STATUS riporta il valore RECPACKET. La variabile STATUS può essere sempre interrogata dopo ogni trasmissione, al fine di verificarne l'efficacia, tramite la funzione `rftide.status()`. Chi ha pratica di librerie Arduino potrà ampliare le funzioni implementate, in quanto il codice è ampiamente commentato e già suddiviso in funzioni facilmente richiamabili. ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità; il master del circuito stampato può essere scaricato gratuitamente dal sito della rivista (www.elettronica.in.it). Lo shield descritto in questo articolo è disponibile in scatola di montaggio al prezzo di 31,00 Euro (cod. FT1082K); il kit comprende la basetta forata e serigrafata, tutti i componenti ed il modulo radio RFT-868-5V. Non è compresa l'antenna AG868 che costa 9,00 Euro. Ricordiamo che il modulo base di questa rete è stato descritto sul fascicolo del mese scorso ed è disponibile in kit (cod. FT1083K, Euro 34,00). 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

Channels Last added a month ago

Graphs

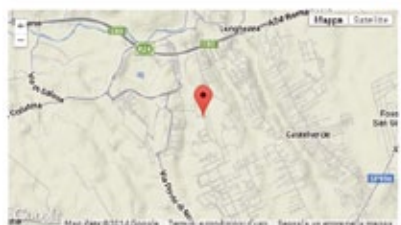
Humidity 60.1 %

Pressure 1058.95 hPa

Temperature 19.9 °C

+ Add Channel

Location



Location Name

Latitude 46.508455053258

Longitude 12.665936704847

Elevation

Request Log

Pause

200 GET feed 10:04:47 +0100

API Keys

Auto-generated TestArduinoMeteo device key for feed 1070345129

256vf0QFuhMq0Tpdj2Wf/pkYfyu9

permissions READ,UPDATE,CREATE,DELETE

device success

+ Add Key

Triggers

Triggers provide 'push' capabilities by sending HTTP POST requests to a URL of your choice when a condition has been satisfied.

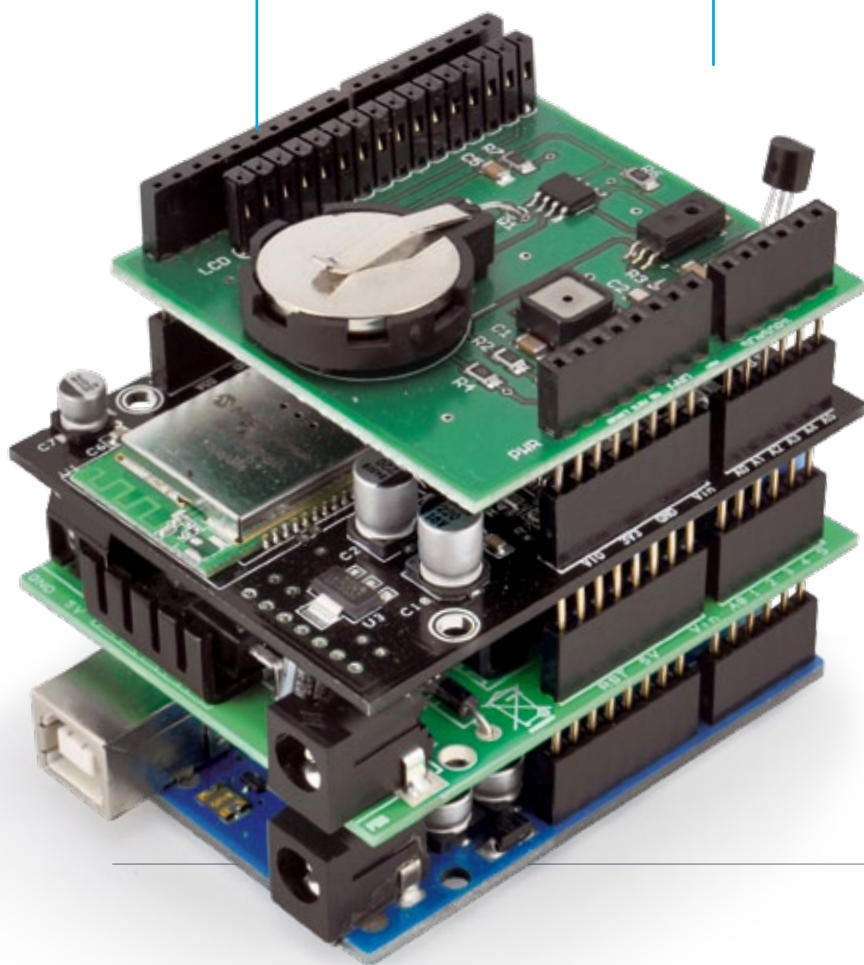
+ Add Trigger

dell' Ing. DANIELE DENARO

STAZIONE METEO ARDUINO LOW POWER

Acquisiamo dati meteo e inviamoli al portale Xively tramite WiFi shield e protocollo client HTTP.

La connettività WiFi è una porta che Arduino deve tenere aperta per accedere con facilità al mondo di Internet e delle applicazioni rese possibili dal web; ne abbiamo parlato pubblicando, nel fascicolo n° 182, il progetto di uno shield WiFi ad alte prestazioni e spiegando come utilizzarlo come server o come client web. Le due modalità, lo abbiamo visto, spiana-



Internet delle cose

"Internet of Things" (IoT) è un termine coniato da Kevin Ashton nel 1999 che, guarda caso, si occupava di RFID (Radio Frequency Identification) al MIT (Massachusetts Institute of Technology). In sostanza consiste nella visione di un mondo in cui ogni oggetto, che abbia un qualche concetto di identità o di funzionalità, possa comunicare.

A questa esplosione di comunicazione contribuisce anche la riduzione di costo e di dimensione degli apparati di rete. Il passaggio dalla codifica dell'indirizzo IP da 4 byte (versione IPv4; circa 10^9 possibilità) a 16 byte (versione IPv6; circa 10^{38} possibilità) è stato voluto anche per supportare l'immensa quantità di dispositivi presto in Rete.

no il terreno a una vastissima serie di applicazioni appoggiate alla connettività di rete. Diversamente da altri shield, il nostro integra un modulo in grado di farsi carico di quei compiti che, se lasciati ad Arduino, impedirebbero a quest'ultimo di svolgere il proprio lavoro in tempi ragionevoli, precludendo la possibilità di implementare applicazioni che implicano grandi moli di calcoli. Il nostro WiFi shield gestisce, oltre al colloquio in WiFi, tutta la parte del protocollo HTTP, indispensabile per lavorare sul web; questo significa che permette ad Arduino di concentrarsi sul ruolo di server/client, dimenticandosi del task imposto dalle connessioni web. In queste pagine, basandoci su Arduino e sul WiFi shield, proponiamo di realizzare una stazione meteo Arduino-based, implementabile aggiungendo un Meteo Shield e un timer ciclico molto semplice, implementato anch'esso in un apposito shield: il nostro stack sarà quindi composto da Arduino, WiFi shield, Meteo shield e dallo shield timer.

Conosciamo già i primi due elementi, quindi spieghiamo gli altri due. Ma prima è il caso di anticipare qual è lo scopo del nostro progetto: già, perché la cosa che non abbiamo detto è che si tratta di un'applicazione web "con tutti i crismi", dato che non ci proponiamo semplicemente di rilevare dati climatici, ma piuttosto pubblicheremo i rilevamenti sul web appoggiandoci al sito www.xively.com.

IL SITO XIVELY

Il sito www.xively.com nasce per

servire "l'Internet delle cose" ed è uno spazio pubblico virtuale su Cloud, dedicato alla raccolta e condivisione dei dati rilevati da sensori e non solo. Naturalmente ha obiettivi commerciali, ma permette comunque l'utilizzo gratuito sotto la forma di "Account di sviluppo", così da permettere ai potenziali utenti di provare le funzionalità fornite. Se volete utilizzare questo sito per scopi dilettantistici e realizzare l'applicazione qui descritta, vi dovete registrare come "Free Developer Account" inserendo poche essenziali informazioni.

A questo punto si tratta di definire la sorgente dei vostri dati; ovvero quella che viene chiamata genericamente "device". Questa sorgente di dati è chiamata anche "feed" (alimentatore) ed è contraddistinta da un numero identificativo (Feed Id), una URL di riferimento ed una URL da utilizzare con le API di collegamento. Inoltre prevede una lunga KEYWORD che deve essere utilizzata nei collegamenti per

inserire e gestire i dati. Tutto ciò è automaticamente definito quando create un device. Un device può avere diversi canali (datastreams), ognuno dei quali corrisponde ad un tipo di misura (temperatura, pressione ecc.). Quindi, dopo aver definito il vostro device dovete creare almeno un canale, che sarà identificato dal suo nome (ad esempio: Temperatura). Definita la destinazione dei vostri dati, si tratta di decidere quale modalità utilizzare per i collegamenti. Infatti è possibile utilizzare il protocollo http, ma anche il collegamento tramite socket. Noi, per semplicità utilizzeremo l'accesso Web.

Per spedire i dati bisogna effettuare una "request" con metodo "PUT". La chiamata va inoltrata all'indirizzo api.xively.com e, più precisamente, alla risorsa `/v2/feeds/{id}devostrofeed`.

Però bisogna anche definire, nel pacchetto http, i seguenti header:

- `Host: api.xively.com`
- `X-APIKey: 39KvFo30fuhM...` (qui compare abbreviata).

In realtà, laddove incontraste problemi con i DNS, potete indirizzare la chiamata alla pagina api.xively.com utilizzando il corrispondente indirizzo numerico "173.203.98.29", invece di quello

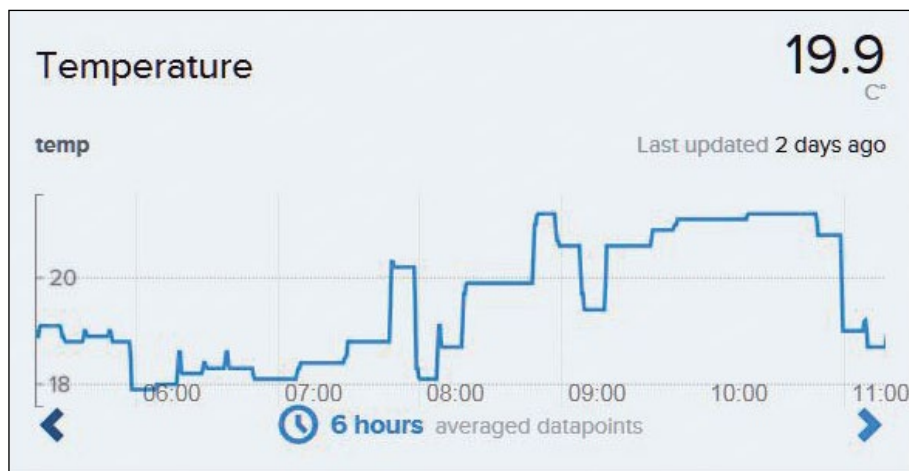


Fig. 1 - Esempio di dati di temperatura (abitazione).

alfabetico. I dati possono essere forniti in diversi formati: XML, JSON, csv. Noi abbiamo scelto, per semplicità, il formato csv (Comma Separated Values).

Riepilogando, dovendo pubblicare dati meteorologici, si tratta per esempio di fornire tre righe del tipo:

Temperature, 19

Pressure, 1022

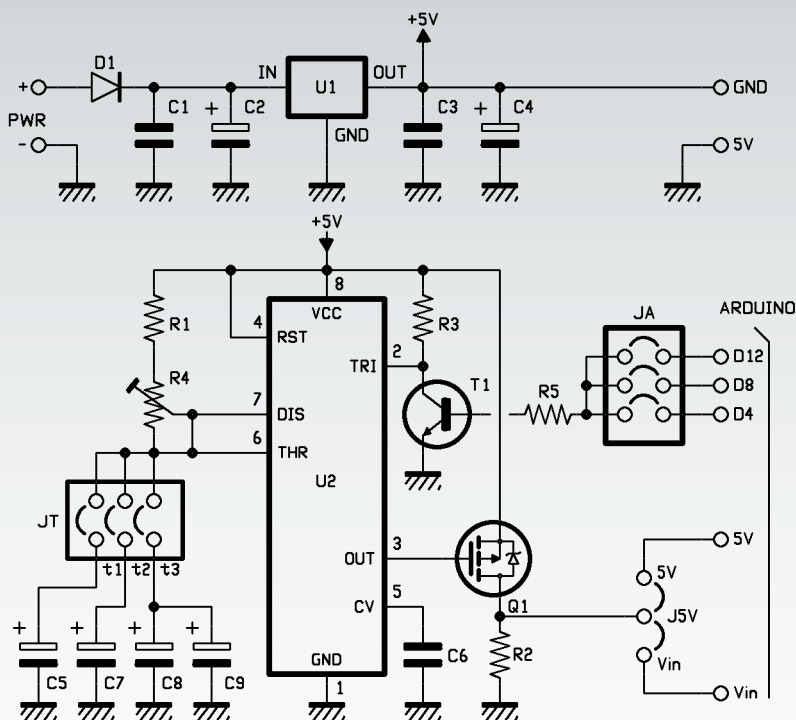
Humidity, 62

È possibile visualizzare i dati raccolti sotto forma di grafico (Fig. 1), direttamente sul sito, ma sono possibili diverse altre modalità di visualizzazione.

Il sito Xively fornisce anche delle librerie per i sistemi più conosciuti e quindi anche per Arduino; ma naturalmente la libreria fornita fa riferimento alla WiFi di base di Arduino e quindi non è utilizzabile nel nostro caso. In realtà, come vedrete, grazie alla semplificazione della nostra libreria nel trattare il protocollo http, il codice necessario alla connessione consiste in poche righe.

ARDUINO COME HTTP CLIENT

Per utilizzare lo shield WiFi, avevamo proposto un'apposita libreria da caricare in Arduino, che l'autore aggiorna in continuazione e che oggi è disponibile, migliorata e aggiornata, nella versione 2.2. In particolare, nella modalità **client** (sendRequest), è stata aggiunta la possibilità di definire degli header nel pacchetto http, in modo da permettere il collegamento con server che richiedono la presenza di particolari header. Inoltre sono stati aggiunti i metodi PUT e DELETE, previsti, tra le altre cose, dall'approccio REST (REpresentational State Transfer) utilizzato nei server per la raccolta pubblica dei dati. Nella modalità **server** (getRequest), invece, è stata aggiunta l'autenticazione.



Negli esempi della nostra libreria è già presente uno sketch per realizzare un client http che dialoghi con un server personalizzato. Ora, con l'obiettivo di illustrare la possibilità di collegarci ad un server pubblico, approfondiamo meglio la conoscenza della libreria nell'utilizzo come client, rimandando al prossimo articolo l'approfondimento della modalità server. Ecco la procedura da compiere.

1. Ci colleghiamo al Router WiFi.
2. Ci colleghiamo all'indirizzo IP del server (con una certa porta).
3. Formiamo l'eventuale stringa con i dati da spedire.
4. Definiamo la risorsa da indirizzare sul server.
5. Definiamo il metodo da utilizzare (GET, POST, PUT, DELETE).
6. Definiamo eventuali header.
7. Spediamo (sendRequest) fornendo metodo, risorsa, header e dati (se usiamo POST o PUT).
8. Attendiamo risposta (getResponse) e la utilizziamo (se abbiamo usato GET).

9. Chiudiamo il socket con il server.

Applicando questa scaletta nel collegamento con Xively, abbiamo, nella fattispecie, la seguente serie di statement.

1. `WiFi.ConnectWPAwithKey(ACCESSPOINT,key);`
2. `csocket=WiFi.openSockTCP(REMOTEIP,PORT);`
3. `snprintf_P(recValues,64,FVal,step,sumid,spress);`
4. `#define RESOURCE "/v2/feeds/1070345129.csv"`
5. il metodo utilizzato è PUT
6. `#define HOST "Host: api.xively.com"`
7. `#define KEY "X-APIKey: 39KvFo30fu..."` (qui solo accennata)
7. `WiFi.sendRequestPUT(csocket,headers,2,RESOURCE,recValues);`
8. `WiFi.getResponse(csocket,1000);` (con timeout di 1s)
9. `WiFi.closeSock(csocket);`

La connessione web a Xively è fatta in unico passo e sfrutta la

chiave che è stata elaborata una tantum per il router WiFi utilizzato. La chiave è elaborata in funzione della password e, una volta calcolata, può essere utilizzata per un accesso quasi immediato. Invece l'accesso tramite password può durare anche un minuto a causa del calcolo della chiave (se la chiave è stata già memorizzata). In questa applicazione, che prevede lo spegnimento periodico di Arduino (vedi più avanti), la chiave è stata memorizzata in EEPROM. Riguardo alla connessione al server, non c'è molto da aggiungere rispetto a quanto detto nell'articolo di descrizione della libreria. Il numero di socket (csocket) deve essere, affinché significhi socket connesso, un numero minore di 255. La costruzione della stringa dati è stata fatta utilizzando una funzione C di AVR che permette l'utilizzo di format in memoria di programma, in modo da economizzare sulla RAM; infatti, quello della scarsità di RAM (2kB, su Arduino Uno), è uno dei problemi maggiori che si devono affrontare quando si realizzano programmi complessi su Arduino. Per questo la libreria stessa fa largo uso della

memoria di programma per le stringhe costanti e per i messaggi di testo. Sulla definizione della risorsa da indirizzare, sul metodo e sugli header necessari, non c'è molto da dire, oltre al fatto che sono quelli indicati da Xively. Come si è detto precedentemente, alla libreria sono state aggiunte le modalità `sendRequestPUT` e `sendRequestDELETE` per completare le possibili esigenze dei server pubblici. Al `getResponse` è stato aggiunto un timeout, così da essere sicuri dell'arrivo della risposta.

GESTIONE DEL CONSUMO

L'applicazione qui proposta prevede la spedizione periodica di dati meteorologici raccolti mediante il Meteo shield (commercializzato da Futura Elettronica -www.futurashop.it- con il codice 8190-METEO_SHIELD); si tratta di temperatura, pressione atmosferica e umidità dell'aria. Ai sensori corrispondenti, nello shield è stato affiancato un RTC (Real Time Clock) realizzato con l'integrato DS1307 della Maxim. Con il Meteo shield, Arduino implementa una vera e propria stazione meteo anche migliore di quelle commerciali, in quanto la

nostra piattaforma è espandibile e configurabile cambiando il codice e aggiungendo ulteriori periferiche e shield, come nel nostro caso, dove otteniamo la connettività Internet tramite lo shield WiFi. Il sensori montati nello shield Meteo sono:

- MCP9700A della Microchip per misurare la temperatura;
- HIH-5030-001 della Honeywell per determinare l'umidità relativa;
- MPXH6115A6U della Freescale per disporre dei dati inerenti la pressione atmosferica assoluta.

Per gestire il Meteo shield abbiamo previsto un'apposita libreria scaricabile dal nostro sito web www.elettronica.in.it, che provvede anche alla calibrazione dei sensori. I dati meteo desunti dal sistema, tipicamente vengono acquisiti con periodicità di diversi minuti o addirittura ore. Per questo motivo non ha senso e utilità tenere acceso il sistema e tantomeno il Meteo shield in continuazione; lasciare continuamente alimentato il tutto significa consumare più di un centinaio di milliampere/ora (tra Arduino e WiFi shield) per utilizzarlo solo per circa 4 secondi (questo è il tempo occorrente all'upload dei dati su Xively) ogni 10 minuti o addirittura ogni ora. Significa sprecare elettricità, sempre preziosa ma che diventa vitale nel caso il sistema sia alimentato a batterie e collegato unicamente tramite WiFi. Pertanto abbiamo pensato di progettare uno shield di power management molto elementare, il cui compito è accendere il sistema solamente quando serve e spegnerlo negli intervalli. Questo shield ha un ingresso regolato per alimentazione variabile da 7 a 15 volt ed un ingresso diretto a 5 V; l'alimentazione può essere anche 6V, ma bisogna cortocircuitare il diodo D1 dello shield. L'alimentazione di Arduino avviene tramite

Una nuvola di dati

Il "cloud computing" vi può fornire:

- programmi da utilizzare in remoto (SaaS: *Software as a Service*)
- uno spazio di appoggio per dati (DaaS: *Data as a Service*)
- risorse hardware da gestire in remoto (HaaS: *Hardware as a Service*).

Ed è evidente il tentativo di ricreare un "business" che era sfuggito con il proliferare dell'informatica distribuita. I grossi centri di calcolo composti da una miriade di potenti server, oltre a creare qualche problema di "privacy" hanno l'indubbio merito di fornire una

potente alternativa al farsi tutto in casa, soprattutto in una fase di start-up. In questo numero della rivista vi proponiamo un utilizzo della seconda opzione tramite uno di questi siti del "cloud computing".

Xively non è l'unico a fornire questi servizi; altri siti che forniscono servizi analoghi sono per esempio: Google stesso e Nimbits (www.nimbits.com). Stiamo parlando di siti che prevedono il collegamento con protocollo HTTP (o HTTPS) o con apertura di socket dedicati per la raccolta di dati elementari cioè di misure.

questo shield, che fornisce la tensione sul contatto 5 V degli strip. Lo shield è sostanzialmente un timer basato sul classico 555 in configurazione monostabile, che controlla un MOSFET a canale P usato come interruttore statico per accendere e spegnere Arduino secondo il tempo di carica del condensatore di temporizzazione; in breve, il 555 quando il suo piedino di trigger (2) è a livello alto (T1 interdetto) il condensatore visto tra i piedini 6 e 7 si carica e l'uscita (piedino 3) fornisce un impulso a livello logico alto, la cui durata dipende dalla resistenza vista tra i pin 6,7 e l'alimentazione, oltre che dalla capacità presente tra detti piedini e massa. Ciò non consente la polarizzazione del gate del MOSFET, il quale resta interdetto e non alimenta il contatto 5V degli strip di Arduino. Lo schema concettuale della gestione è rappresentato in Fig. 2. Completata la carica, il piedino 3 del 555 si porta a zero logico e determina l'entrata in stato di ON del MOSFET, il quale fornisce alimentazione ad Arduino mediante il contatto 5 V. Una volta alimentato, Arduino esegue le operazioni e poi si porta

Dieci volte AUGURI, Arduino!

Per qualcuno è giovane, per qualcun altro ha raggiunto la maturità; sta di fatto che ci sembra ieri quando Arduino ha fatto la sua comparsa nel mondo dell'elettronica e dei Maker, invece la sua nascita risale a 10 anni fa. Per festeggiare i 10 anni della popolare scheda di prototipazione open source, il 29 marzo al tempio di Adriano a Roma, dalle 10 alle 18.30 si terrà Arduino Day, una giornata (organizzata in tutto il mondo) dedicata ad Arduino e al mondo dei maker. L'appuntamento è a cura di DiScienza e Officine Arduino e vedrà allestiti spazi dove ognuno potrà portare ed esporre i propri prototipi basati su Arduino, spazi di prototipazione libera, stand di maker e di aziende open, laboratori per bambini e ragazzi, workshop e dimostrazioni pratiche. Il programma completo dell'evento si trova su www.discienza.org. Alla pagina web <http://day.arduino.cc/#events> si può invece vedere e ascoltare un'intervista a Massimo Banzi proprio sull'evento 2014.

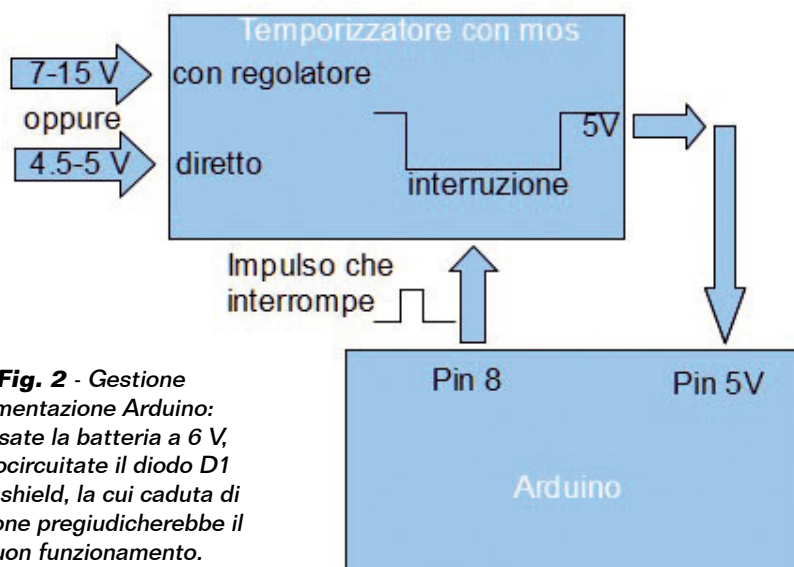


Fig. 2 - Gestione alimentazione Arduino: se usate la batteria a 6 V, cortocircuitate il diodo D1 dello shield, la cui caduta di tensione pregiudicherebbe il buon funzionamento.

in sleep, ponendo a livello logico alto il contatto 8 dello shield; ciò causa la saturazione del T1, il quale cortocircuita per un istante il piedino 2 del 555, triggerando il monostabile e ricominciando il ciclo: il piedino 3 dell'integrato si riporta a livello logico alto e fa interrompere l'alimentazione diretta ad Arduino, in attesa della ricarica della capacità. Poiché il Meteo shield contiene anche un RTC (DS1307) (con batteria di backup), abbiamo pensato di far svegliare Arduino ogni minuto per fargli controllare velocemente se è tempo di trasmettere (ogni 10 minuti). Nel caso non fossero

Il programma per ricevere dati

Se non avete a disposizione un sito dove scaricare i dati come Xively, ma disponete di un computer sempre acceso, potete fare tutto "in casa". Infatti nella cartella "JavaServicePrograms" della libreria MWiFi trovate due programmi Java: uno di questi è "SocketLog.jar". Lanciato sul vostro computer, esso si mette in ascolto sulla porta stabilita da voi (5000 è la predefinita) e può visualizzare i record che eventualmente gli inviate da Arduino ed anche salvarli sul file deciso da voi. Inoltre potete salvare insieme al record un "timestamp" (data ed orario) fornito dal vostro computer nel caso rinunciate a montare lo shield RTC.

trascorsi 10 minuti, il sistema si rimette in sleep. In questo modo si mantiene una risoluzione al minuto senza doversi basare su tempi di carica del condensatore lunghi ed imprecisi. Il tempo necessario per il controllo è bassissimo e comunque non viene accesa la WiFi, per cui il consumo è irrisorio. Quando è trascorso l'intervallo stabilito, Arduino si connette e trasmette i dati. Il tutto avviene in circa 4 secondi. Ne consegue un consumo così basso nel tempo, che permette di alimentare il sistema anche con un piccolo pannello solare con batteria.

Come si può vedere dallo schema elettrico, nello shield c'è una sezione di alimentazione ed una di gestione del pin 5 V di Arduino. La sezione di alimentazione permette di fornire una tensione di 5 volt stabilizzati partendo da quella applicata all'ingresso PWR, al quale possiamo applicare da 4,5 a 15 Vcc. Nel caso volessimo alimentare Arduino con tre batterie stilo per un totale di 4,5 V o, infine, quattro batterie stilo ricaricabili (NiCd o NiMH) per un totale di 4,8 V, potremo farlo tramite la morsettiera 5V di Arduino (Arduino può funzionare anche a tensioni inferiori a 5 volt). La stessa cosa vale nel caso volessimo alimentare Arduino con

una sorgente che prende tensione da una presa USB. L'alimentazione a batteria è possibile (purché la tensione sia almeno 6 volt) anche da PWR.

Analizziamo lo stadio regolatore, che fa capo all'integrato U2 (LF50AB) che è un regolatore a basso dropout (450 mV) e bassa corrente di riposo (<1mA), capace di erogare 5 V stabilizzati con una corrente massima di 1 A. Il diodo di ingresso D1 è stato messo per proteggere il circuito dall'inversione di polarità. Notate che tra la morsettiera di uscita a 5 V e l'out del regolatore non è stato inserito alcun diodo, in quanto l'LF50AB è in grado di sopportare i 5 volt senza danneggiarsi (l'uscita è costituita da un transistor PNP montato a base comune, quindi espone il collettore e la relativa giunzione, che sopportano sicuramente questa tensione).

La sezione di gestione dell'alimentazione, si occupa di fornire tensione ad Arduino autonomamente (a carica dei condensatori del 555 avvenuta) e toglierla quando Arduino stesso fornisce il comando: l'impulso di spegnimento può essere dato a scelta da uno dei pin di I/O digitale D4, D8, D12 di Arduino; il piedino si sceglie mediante il jumper JA sulla base di qual è la linea disponibile e implementata nello sketch. In pratica, se una delle D4, D8, D12 è occupata, impostate JA in modo da usare una linea libera e modificate di conseguenza lo sketch. L'impulso proveniente da Arduino comanda il transistor T1, che opera come interruttore statico invertente e amplifica in corrente, invertendo il livello logico fornito da Arduino, i cui I/O non potrebbero, da soli, pilotare il piedino di trigger del 555. Il MOSFET Q1 è un IRLB9343, caratterizzato da bassa $R_{ds(on)}$ e ridotta tensione di soglia gate-source V_{gs} (appena 3 volt) e funziona da in-

teruttore statico collegato a source comune.

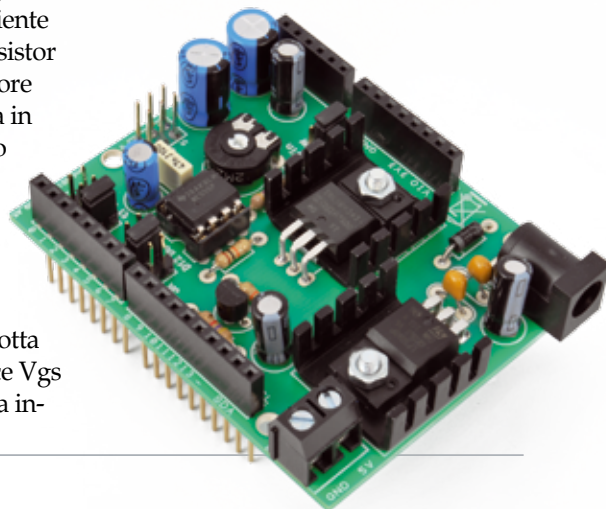
Il 555, appena riceve l'impulso di trigger spegne il MOSFET e carica il condensatore tramite R1 e R2 (essendo questo un trimmer, il tempo si può variare a piacere); caricato il condensatore, il 555 scatta, porta l'uscita a 0 logico attivando Q1 e scarica il condensatore. Il tempo T (in secondi) durante il quale l'alimentazione di Arduino viene interrotta è:

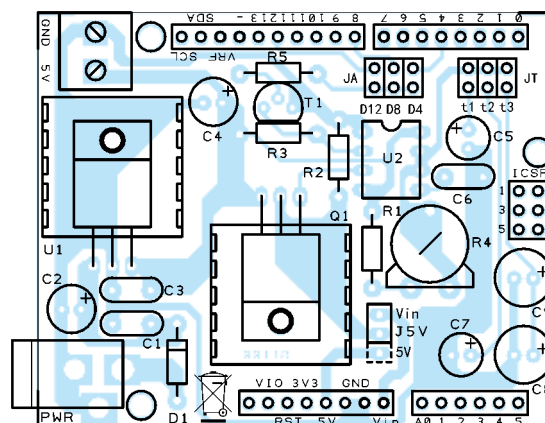
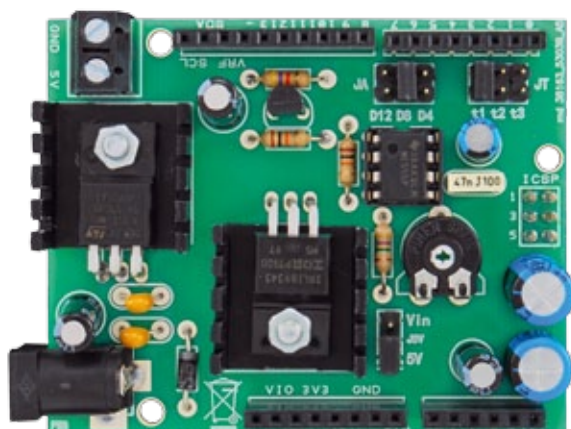
$$T=1,1 \cdot R \cdot C$$

Per rendere il circuito più versatile, abbiamo predisposto tre diversi valori di capacità gestibili tramite jumper, che consentono altrettanti intervalli (supponendo il cursore di R2 a metà corsa): 10 μ F (T di 60 secondi) 100 μ F (T di 10 minuti) e 600 μ F (T pari a 3.600 s, ossia un'ora).

LO SKETCH DI ARDUINO

Lo sketch documentato (in inglese) si può trovare sul nostro sito www.elettronica.in insieme agli altri file del progetto; qui ne riassumiamo la struttura generale. Il Meteo shield è sensibile alla tensione di alimentazione perché non ha un riferimento di tensione interno, perciò i dati raccolti dai sensori sono anch'essi relativi all'alimentazione e alle sue fluttuazioni. Perciò è necessario calibrare il sistema misurando la





Elenco Componenti:

R1: 4,7 Mohm
R2, R3: 10 kohm
R4: Trimmer 2,2 Mohm MO
R5: 4,7 kohm
C1, C3: 100 nF multistrato
C2, C4, C7: 100 μ F 25 VL elettrolitico
C5: 10 μ F 63 VL elettrolitico
C6: 47 nF 100 VL poliestere

C8, C9: 470 μ F 16 VL elettrolitico
D1: 1N4007
U1: LF50AB
U2: NE555
Q1: IRLIB9343PbF
T1: BC547

Varie:
- Plug alimentazione
- Zoccolo 4+4

- Morsetto 2 poli
- Jumper (3 pz.)
- Strip maschio 3 poli (5 pz.)
- Strip maschio/femmina 6 poli (1 pz.)
- Strip maschio/femmina 8 poli (2 pz.)
- Strip maschio/femmina 10 poli (1 pz.)
- Vite 10 mm 3 MA (2 pz.)
- Dado 3 MA (2 pz.)
- Dissipatore
- Circuito stampato

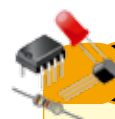
tensione presente sul piedino 5V di Arduino. Questa può variare anche di alcuni decimi di volt, per cui è stata prevista una voce "define VADC" relativa a tale grandezza, che va aggiornata per ogni installazione dopo aver applicato l'alimentazione. Inoltre vengono fatte delle medie su diversi sample consecutivi per migliorare la precisione delle misure.

Lo sketch comprende routine per la rilevazione dei dati e la gestione dell'alimentazione. Nello sketch è implementato il solo metodo **setup()** mentre il metodo **loop()** è lasciato vuoto; infatti non deve eseguire nessuna azione ripetutamente, ma piuttosto agire e spegnersi. Quindi lo shield esegue nell'ordine queste operazioni:

- legge la chiave WiFi, se è presente in EEPROM;
- si collega all'RTC del Meteo shield utilizzando la libreria a corredo e legge l'ora;

- verifica che siano passati 10 minuti (define TIMEINT) e in caso contrario lancia la routine di spegnimento;
- se sono trascorsi 10 minuti, effettua tutti i passi di connessione e spedizione dei dati;
- si spegne comandando il contatto 8 di Arduino.

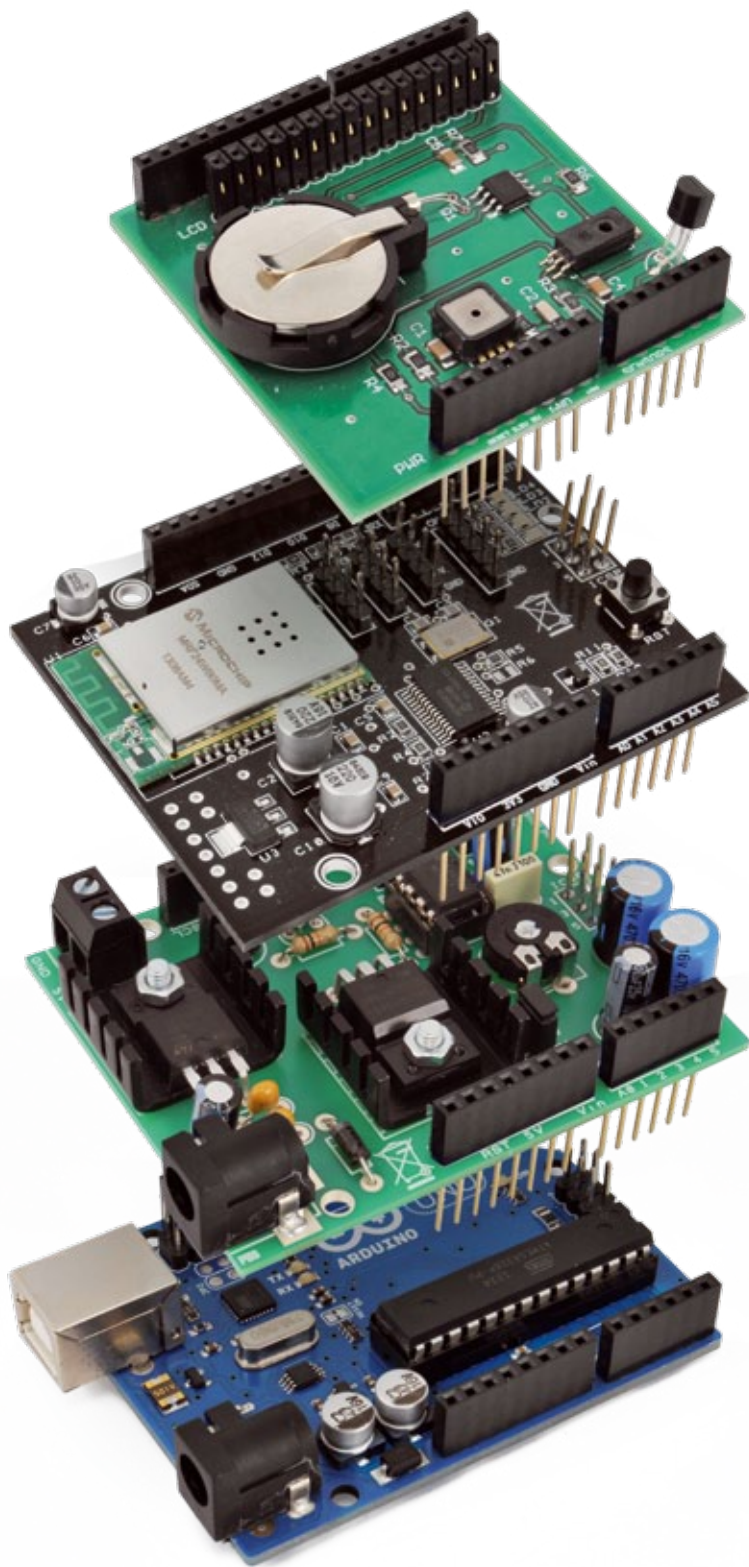
La routine di spegnimento opera in questo modo: attiva il pin 8 che provoca lo stop dell'alimentazione per circa un minuto. Al risveglio, riparte come dopo un reset. Nel caso non fosse presente la scheda di power management, è stata prevista una riduzione dell'assorbimento ponendo Arduino in modalità SLEEP_MODE_PWR_DOWN, dalla quale viene svegliato, dopo circa un minuto, da un reset provocato dal "watchdog counter". In realtà viene effettuata una serie ripetuta di watchdog interrupt con il massimo tempo (8 s) consentito al watchdog timer, fino al raggiungimento del minuto.



per il MATERIALE

Tutti i componenti utilizzati per realizzare lo shield del timer descritto in questo articolo sono facilmente reperibili in commercio. Di questo progetto esiste anche un kit prodotto da Futura Elettronica (cod. FT1100K, Euro 15,00). Il software per realizzare l'intero progetto è scaricabile gratuitamente dal sito della rivista (www.elettronicaonline.it) mentre anche gli altri elementi sono reperibili presso Futura Elettronica: Meteo Shield per Arduino (cod. 8190-METEO_SHIELD, Euro 42,00), Wi-Fi Shield per Arduino (cod. 7100-FT1076M, Euro 56,00) e Arduino Uno Rev.3 (Euro 24,50). 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
<http://www.futurashop.it>



REALIZZAZIONE PRATICA

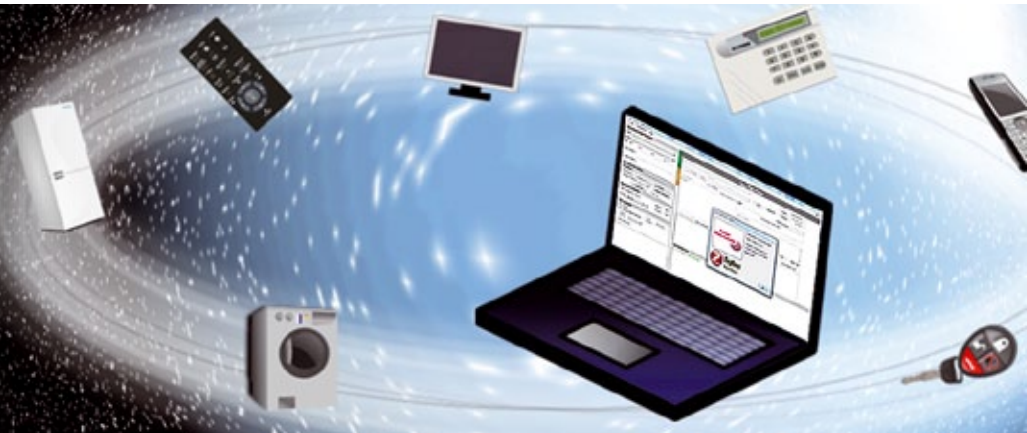
Per mettere insieme il sistema occorre una scheda Arduino Uno, sulla quale montare il Meteo Shield, il WiFi shield e lo shield di power management; Arduino si compera già pronta, come gli altri due shield. Ciò che dovete costruire è lo shield di power mana-

gement, operazione alla portata di tutti dal momento che il circuito è un singola faccia e si può ottenere per fotoincisione senza difficoltà; infatti avrebbe i fori metallizzati con piazzole sul lato componenti, ma vi si può rinunciare, pertanto basta impiegare la sola traccia lato rame. Preparata la basetta, bisogna montarvi i componenti in ordine di altezza, quindi resistenze e diodo al silicio per primi, cui seguiranno lo zoccolo per il 555, i condensatori non polarizzati, il transistor T1 e il trimmer; il regolatore LF50AB e il MOSFET vanno applicati ciascuno a un dissipatore di calore ad U per TO-220 (va bene il classico ML26) da non più di 20 °C/W di resistenza termica. I componenti vanno appoggiati ai rispettivi dissipatori dal lato opposto a quello delle scritte e fissati con delle viti 3MA (lo stampato è già forato allo scopo) con di dado. Lo shield si completa montando gli elettrolitici, i jumper (si ottengono con dei pin strip a tre poli), la morsettiera, il plug di alimentazione e i connettori SIL (strip femmina) per l'accatastamento dello shield (ne servono due da 8 poli, uno da 6 e uno da 10). Il piano di montaggio pubblicato in queste pagine mostra il corretto orientamento degli elementi polarizzati.

Completato lo shield di power management, realizzate l'insieme, ricordando che per avere la maggiore distanza possibile di connessione WiFi, lo shield WiFi dovrebbe essere posto in cima alla catasta; comunque ponendo in cima il Meteo shield, il modulo WiFi è abbastanza libero e le onde radio vengono minimamente schermate. L'alimentazione principale va applicata al plug dello shield di power management mediante un alimentatore che fornisca 7÷12 V in continua (6V se saltate D1) e una corrente di almeno 200 mA; la morsettiera va usata in alternativa (cioè se avete già un alimentatore che fornisca 5 V stabilizzati. Prima di alimentare il tutto decidete se il MOSFET del timer deve alimentare il contatto 5V o il Vin di Arduino; se avete una YÚN, dove Vin è a 5 volt, chiudete il jumper j5V su Vin, mentre se avete altre Arduino, dato che Vin fa capo all'ingresso del regolatore interno alla scheda potete scegliere 5V o Vin: nel primo caso il MOSFET alimenta direttamente Arduino e i suoi shield, mentre nel secondo il MOSFET dà e toglie tensione ad Arduino, il cui regolatore alimenta gli shield (escluso il nostro) con il suo contatto 5V. Non dimenticate di impostare il jumper JT per ottenere la durata voluta di spegnimento di Arduino, tra 60 (t1) 600 (t2) e 3.600 secondi (t3). Ultima cosa: verificate se nel sistema è già usata una delle linee D4, D8 o D12 e impostate il jumper JA per collegare il transistor T1 a quella libera; altrimenti si verificherà un conflitto di attribuzione delle risorse e il timer non funzionerà correttamente. ■

TEST HARNESS

Verifichiamo se il nostro devices ZigBee è conforme allo standard ed in particolare - nel contesto della Casa "Intelligente" - al profilo Home Automation.



..... a cura della REDAZIONE

Test Harness è un test tool che, grazie ad una suite di strumenti più o meno automatizzati, coadiuva lo sviluppatore di dispositivi ZigBee a verificare la conformità di nuovi prodotti alle specifiche di questo standard wireless che sta sempre più prendendo piede nel campo della domotica e della Home Automation.

LO STANDARD ZIGBEE

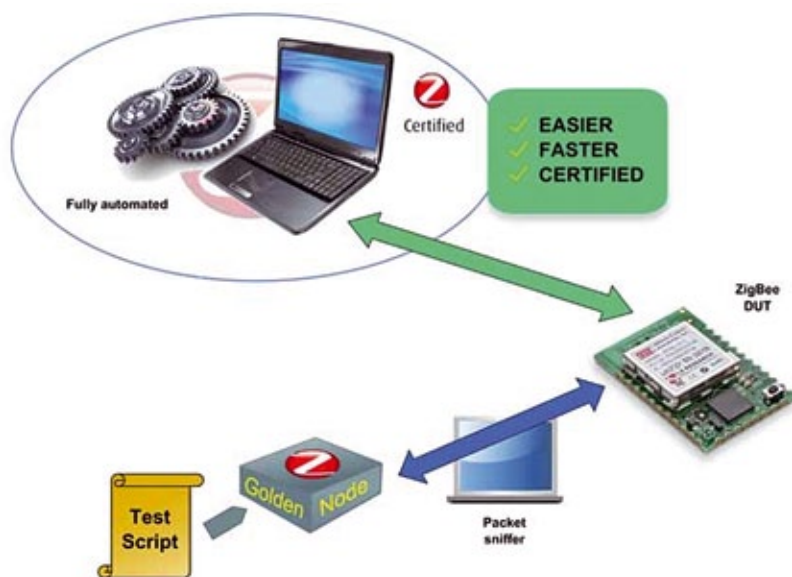
ZigBee è uno standard dedicato alla definizione di soluzioni di sensori e attuatori wireless a basso costo e a basso consumo energetico, i cui paradigmi fondamentali sono flessibilità e interoperabilità. Nello standard vengono definiti diversi scenari applicativi e tipologie di prodotto: sistemi di Home Automation, Building Automation, Health Care, sistemi di illuminazione, prodotti per la connettività, servizi location-based, SmartEnergy e tanti altri. Con questa premessa, è chiaro che la definizione di specifiche standardizzate, per la realizzazione delle soluzioni proposte all'utente finale, è un requisito fondamentale se si vuole garantire la realizzazione di un ecosistema di dispositivi tra loro interoperanti, anche quando vengono proposti da costruttori diversi. Questi stessi costruttori possono a loro volta cooperare nella ZigBee Alliance, un'associazione aperta di aziende, istituti universitari ed enti di ricerca che, dal 2002, cooperano nella definizione di nuove specifiche tecniche e nuove funzionalità. Ad oggi, la ZigBee Alliance raccoglie più di 400 so-

cietà worldwide, che hanno immesso sul mercato più di 600 prodotti certificati ZigBee, basati sul protocollo radio IEEE 802.15.4, anch'esso standardizzato a livello globale e che, oltre all'interoperabilità, garantiscono i necessari meccanismi per avere bassi consumi energetici e standard di sicurezza adeguati.

La certificazione ZigBee, è il passaggio che assicura il livello qualitativo di un prodotto e la sua conformità alle specifiche funzionali e protocollari definite per un particolare "profilo" applicativo. A livello di marketing, questo si traduce nella possibilità di apporre il bollino ZigBee, relativo ai profili per cui si è ottenuta la certificazione e, per fare un esempio pratico, questo elemento distintivo permette all'utente che acquista un prodotto con il bollino ZigBee, di avere una garanzia di compatibilità e interoperabilità con eventuali altre componenti certificate già in uso nel suo sistema di Home Automation.

La Casa "Intelligente"

Ma quali sono, ad esempio, i vantaggi pratici di un sistema di Home Automation? Innanzitutto grazie alla domotica abbiamo a disposizione abitazioni sempre più "intelligenti", in cui gli impianti e i dispositivi - come elettrodomestici, sistemi d'allarme e videocontrollo, per fare qualche esempio - sono collegati e gestiti in modo automatizzato. In questo modo le case sono più confortevoli e sicure, con un notevole risparmio grazie ad una gestione più razionale dei consumi energetici. Limitare i consumi significa



anche ecosostenibilità: si riducono le emissioni di anidride carbonica nell'atmosfera contribuendo alla salvaguardia dell'ambiente. ZigBee permette tutto ciò e, per questo motivo, è una delle opportunità più promettenti per le aziende che vogliono contribuire alla realizzazione di un nuovo concetto di abitazione e ambiente domestico: la Casa "Intelligente".

Test Harness

Consoft Sistemi (www.gruppo-consoft.com) è attiva nel campo della domotica sin dal principio della sua diffusione: un'azienda del gruppo, la CS Domotica, ha realizzato Mercury, uno dei primi gateway domotici proposti sul mercato. Il progetto, basato su diverse tecnologie di connettività, prevedeva la realizzazione di una soluzione completa e modulare per la gestione della casa, anche da remoto.

L'evoluzione delle tecnologie ha poi portato alla diffusione dei sistemi basati sullo standard ZigBee, che hanno quindi potuto disporre di una infrastruttura di rete univoca e standardizzata diventando più attraenti per i mercati Consumer e Business, ma introducendo anche la grossa incognita della reale interopera-

bilità tra i dispositivi.

Dall'esigenza di far evolvere Mercury e altri prodotti verso il mondo ZigBee, nonché dalla consapevolezza di quanto lo stesso standard ZigBee possa rappresentare un'opportunità per molte aziende nel campo delle Wireless Sensor Network, Consoft Sistemi ha realizzato, in collaborazione con Telecom Italia, "Test Harness": un software che permette di verificare, attraverso appositi test plans, sia ufficiali (utilizzati dalla ZigBee Alliance nei suoi processi di certificazione), sia personalizzati (creati ad hoc dall'utente per testare singole funzionalità), la conformità dei devices allo standard ZigBee, e in particolare - nel contesto della Casa "Intelligente" - al profilo Home Automation.

DESCRIZIONE DI TEST HARNESS

Test Harness è un test tool che, grazie ad una suite di strumenti più o meno automatizzati, coadiuva lo sviluppatore di dispositivi ZigBee a verificare la conformità di nuovi prodotti alle specifiche della ZigBee Alliance. Una caratteristica che rende Test Harness innovativo rispetto agli altri software è la possibilità di creare test list personalizzate e dettagliate, anche su aspetti non coperti dalle

specifiche ufficiali, diventando a tutti gli effetti un vero tool di sviluppo.

Al momento della stesura di questo articolo sono state implementate le specifiche funzionali per il profilo ZigBee Home Automation 1.2 (pensato per applicazioni di domotica in scenari di interazione fra dispositivi domestici), presto saranno disponibili anche altri profili.

Perché utilizzare Test Harness

Come si è già detto, ogni prodotto realizzato in modo conforme ad un profilo ZigBee (come Home Automation), può essere sottoposto a certificazione, garantendo all'utente finale che l'oggetto è parte di un ecosistema aperto di dispositivi tra loro compatibili. Questo fornisce all'utente una certa flessibilità nella scelta dei componenti per il proprio sistema domotico; cosa che altrimenti non sarebbe possibile.

Test Harness è stato progettato come strumento che permette di eseguire, "in proprio", tutti i test previsti dalla ZigBee Alliance per un determinato profilo. Questa pre-certificazione può quindi essere eseguita prima di sottoporre i dispositivi all'ente validatore ufficiale.

Essendo Test Harness un tool che è stato a sua volta sottoposto in modo rigoroso agli stessi test plans previsti per i dispositivi reali, il completo superamento della pre-certificazione effettuata con Test Harness su un dispositivo, facilita la successiva acquisizione del bollino ZigBee senza reiterazioni del processo di validazione, riducendo notevolmente le spese legate alle attività di testing ed evitando di dover ricorrere alla consulenza di aziende (Test House) specializzate.

Il sito ufficiale della ZigBee Alliance (www.zigbee.org) raccomanda Test Harness come soluzione per

lo sviluppo ed i test nella sezione "ZigBee Home Automation testing & development solutions".

Ma Test Harness non si ferma qui: essendo stato pensato anche come strumento utile per le fasi di sviluppo di un prodotto, è possibile far eseguire dal tool singoli test customizzati ad hoc, dove si può stimolare il "prototipo" del dispositivo anche con azioni (comandi e payload) non previste dallo standard e, di conseguenza, andando a verificare il comportamento dello stesso secondo i parametri dettati dal costruttore stesso. Se necessario questi test custom possono essere organizzati in intere test list personalizzate che permettono di ridurre notevolmente il tempo destinato a complesse verifiche funzionali e protocollari. Test Harness diventa quindi uno strumento utile per chi realizza nuovi prodotti perché è in grado di mettere a disposizione dello sviluppatore una serie di strumenti avanzati che abilitano, oltre al testing vero e proprio, la visualizzazione dei pacchetti generati e/o ricevuti dal dispositivo, la verifica di timeout, di condizioni, ecc.

A chi è rivolto

Test Harness è indirizzato alle seguenti categorie di utenti:

- aziende produttrici che vogliono testare la conformità dei

propri apparati alle specifiche ZigBee;

- sviluppatori e produttori che vogliono verificare la corretta realizzazione di prototipi basati su uno standard sempre più importante nel mondo dell'automazione domestica.

Anche se è stato realizzato pensando ai grossi player del mercato delle wireless sensor network e della Home/Business Automation, con l'evolversi delle caratteristiche funzionali di Test Harness e grazie ad una politica commerciale orientata a facilitare l'introduzione della tecnologia ZigBee nelle case degli utenti, anche dal punto di vista economico, Test Harness diventa a tutti gli effetti un tool adatto a chiunque si interessi di prodotti ZigBee.

TEST HARNESS COME STRUMENTO DI SVILUPPO E DEBUG

Come funziona

Test Harness genera una sequenza di messaggi ZigBee, li trasmette attraverso un dongle ZigBee Freescale modello MC13226V al dispositivo da testare (DUT: device under test) e verifica che le risposte ricevute siano conformi con quanto previsto dalle speci-

che del profilo Home Automation 1.2 e dai relativi cluster ZigBee supportati. Con il termine Cluster si intende un set di funzionalità di base, necessario a definire il comportamento di un device: ad esempio il cluster ON/OFF definisce tutte le interazioni tra un interruttore ZigBee e il dispositivo che lo controlla. Chiaramente un prodotto reale potrà implementare più cluster contemporaneamente. È abbastanza comune l'esempio della presa elettrica intelligente, dove i cluster ON/OFF (per la gestione - acceso/spento - del carico) e Meter (per la misura dei consumi) coesistono in un unico dispositivo. Test Harness, nella sua configurazione di default (senza aver quindi impostato test customizzati) interpreta tutti i messaggi appartenenti ai vari cluster e capisce se l'interazione tra due elementi del sistema è corretta o meno.

Per poter far fronte alle esigenze di chi sviluppa i dispositivi e non solo di chi deve verificarne il funzionamento, Test Harness è stato reso flessibile e configurabile: nel seguito viene mostrata brevemente l'interfaccia utente ed alcune delle principali funzionalità. Per personalizzare ulteriormente le impostazioni si faccia riferimento al manuale utente scaricabile dalla sezione "Documentazione" nella Home Page del sito



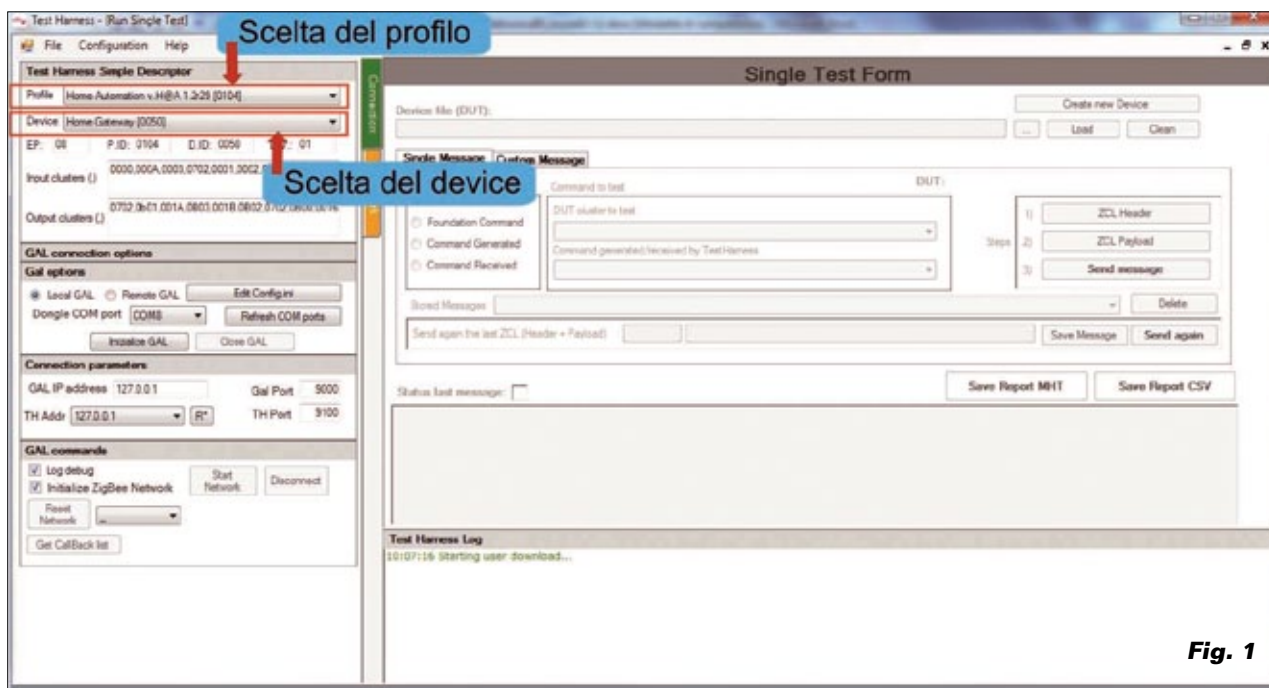


Fig. 1

www.testharness.it.

Sempre dalla Home Page del sito si può visualizzare un filmato in cui si eseguono passo a passo alcune attività d'esempio.

Come già anticipato, quindi, Test Harness permette di effettuare due attività distinte:

- Pre-certificazione
- Sviluppo e Verifica

Nel primo caso ci si affida all'esecuzione di Test plans ufficiali e si ottiene come output un Report che elenca le funzionalità non conformi allo standard. Nel secondo caso si utilizzando test custom (eventualmente aggregati in una test list) che hanno lo scopo di verificare funzionalità specifiche di un dispositivo.

L'interfaccia utente

Per accedere al software è necessario inserire le credenziali che vengono fornite nel corso della registrazione al sito; l'interfaccia utente per l'esecuzione dei test singoli è illustrata in Fig. 1. Per prima cosa è necessario selezionare un profilo e quindi indicare a Test Harness quale tipo

di dispositivo dovrà emulare nelle fasi successive.

Dopo aver scelto tra i nodi appartenenti alla rete quello che si desidera testare, è possibile eseguire un test plan ufficiale o definire test custom e test list (elenco di test eseguiti in modo automatico o semi-automatico), che possono essere salvate e riutilizzate permettendo di risparmiare progressivamente tempo nell'esecuzione delle attività successive.

Supponendo di dover stimolare un prototipo con dei comandi specifici, la schermata "Single test

form", illustrata in Fig. 3, permette di inserire velocemente i parametri che si vogliono verificare per costruire ed eseguire un test personalizzato. Questa funzione è fondamentale quando, durante le fasi di sviluppo, si vuole testare una singola interazione (un comando) da o verso un dispositivo il cui software non è ancora completo, oppure, quando in un sistema completo, si vuole debuggare un comportamento anomalo difficile da replicare o che coinvolge funzioni generalmente annidate in scenari più comples-

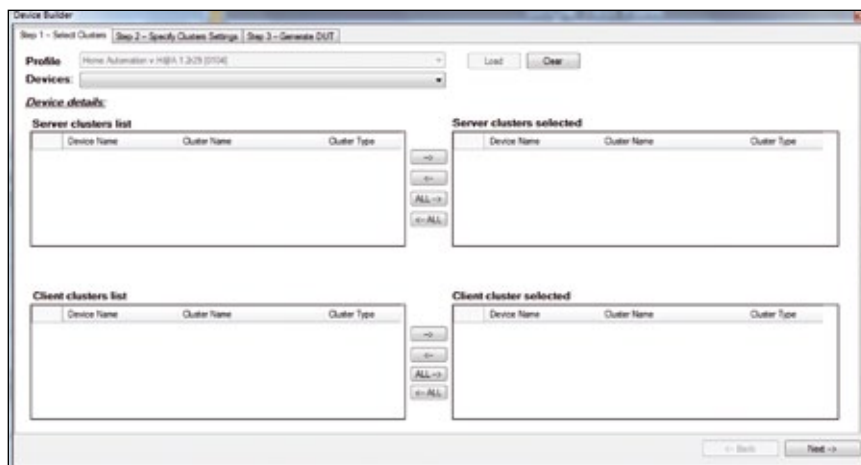


Fig. 2

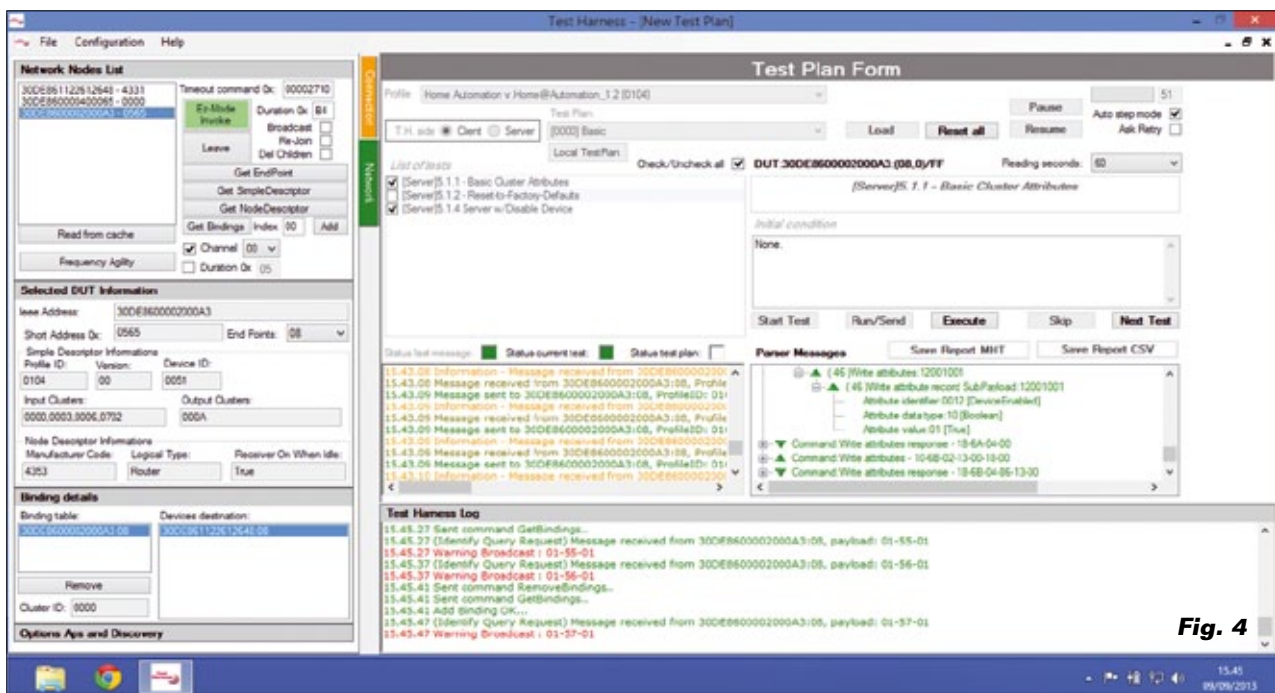


Fig. 4

si. Ad esempio, se in un sistema di Home Automation (basato su prodotti ZigBee Standard e certificati), un attuatore non si comporta correttamente, passando da uno scenario giorno ad uno scenario notte, è possibile utilizzare Test Harness per comandare direttamente quell'attuatore sulla base dei cluster che supporta, verificando se il problema è nell'attuatore o nel sistema di controllo. Se è necessario lavorare su un device prototipale che aggrega differenti cluster, è possibile definire all'interno di Test Harness, in

pochi passaggi, il nuovo device (DUT) attraverso il "device builder"- illustrato in Fig. 2 - un'interfaccia a tab che guida passo a passo nella scelta dei Cluster e nell'associazione ad ogni Cluster di attributi e comandi (ricevuti ed inviati). Il Device generato viene salvato in un file XML e può essere quindi richiamato in seguito. Dopo aver realizzato (e salvato) il device si può quindi procedere con l'invio di messaggi preparati in precedenza, oppure si possono utilizzare i custom test formatando opportunamente le infor-

mazioni necessarie per costruire i messaggi ZigBee nei campi del pannello "Custom message", illustrato in Fig. 3.

Se si sceglie invece di eseguire un test plan ufficiale, a scopo di pre-certificazione, la schermata "Test plan form" apparirà come quella illustrata in Fig. 4.

In questo caso, durante l'esecuzione dei test, l'esito dei singoli passaggi viene evidenziato da un led (come illustrato in Fig. 5) che sarà di colore verde se il test ha avuto esito positivo, giallo se è stato saltato volontariamente dall'utente e rosso se sono stati rilevati errori. In caso di esito negativo del test, una finestra di debug fornisce i dettagli dei problemi riscontrati, in questo modo l'utente può facilmente identificare e risolvere eventuali anomalie. Al termine dell'esecuzione di un test plan, Test Harness produce un Report (illustrato in Fig. 6) contenente i singoli passaggi eseguiti e l'esito globale della sessione di prove.

Come ottenere Test Harness

Test Harness è disponibile sul sito di e-commerce www.testharness.it dove è possibile acquistare on-line

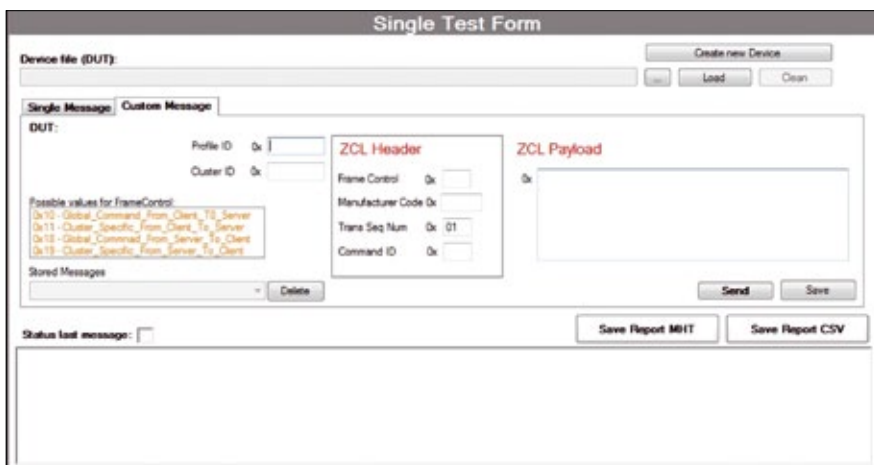


Fig. 3

Fig. 5

una licenza del software disponibile con diverse modalità, consentendo in questo modo l'acquisto delle licenze necessarie per testare i profili ZigBee desiderati, con la possibilità di acquistarne altri in un secondo momento. L'acquisto delle licenze dà diritto al supporto

tecnico. Le licenze disponibili sono:

- Illimitata – licenza completa
- Limitata nel tempo – 90 giorni

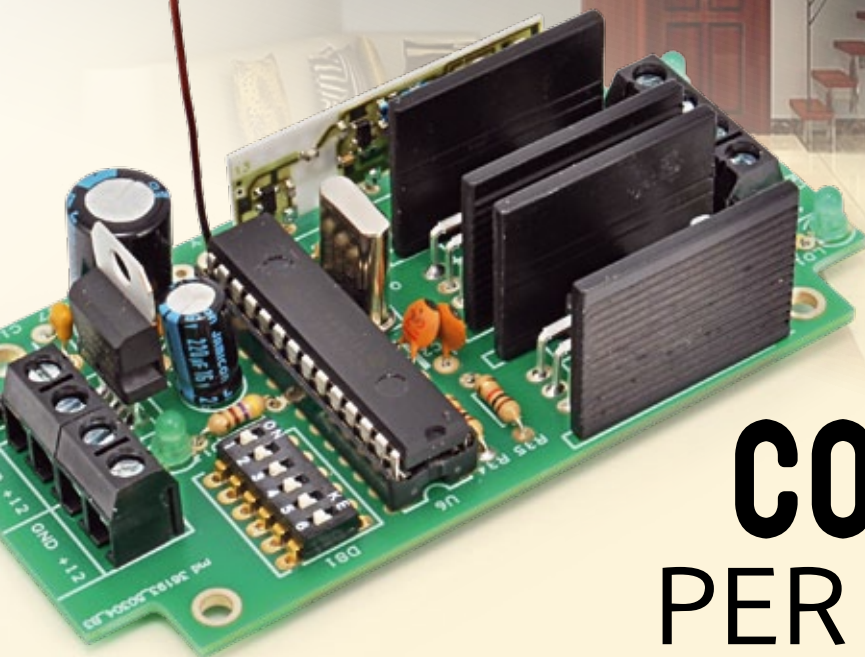
È disponibile inoltre, una versione trial di Test Harness che si può scaricare gratuitamente dallo stesso sito (www.testharness.it). È

sufficiente registrarsi per ottenere una versione dimostrativa che implementa il Basic Cluster, comune a molti profili ZigBee. La demo è utilizzabile per un tempo limitato (10 giorni) e richiede l'utilizzo di una chiavetta USB-ZigBee Free-scale modello MC13226V.

TEST REPORT					Test result: PASS
Date of issue: 03/04/2013 14:47:17					
Test Harness Side: Client Profile: Home Automation v.H@A 1.2 [0104] Cluster: [0000] Basic					
INITIAL CONDITIONS	TEST PERFORMED				RESULT
None	Command	Message	Execution	Error Message	
	Read attributes	10-00-00-00	✓		
				Parser	
				• (0) Read attributes 10-00-00-00	
				• (0) Attributes Identifier SubPayload 0000	
				• Attribute Identifier 0000 [ZCLVersion]	
	Read attributes response (Standard Type)	18-00-01-00-00-00-20-01	✓		
				Parser	
				• (1) Read attributes response (Standard Type) 18-00-01-00-00-00-20-01	
				• (1) Read attribute status record SubPayload 0000000001	
				• Attribute Identifier 0000 [ZCLVersion]	
				• Status 00 [SUCCESS] [00]	
				• Attribute data type 20 [Unsigned 8-bit integer]	
				• Attribute value 01 [01]	
	Read attributes	10-0A-00-07-00	✓		
				Parser	
				• (2) Read attributes 10-0A-00-07-00	
				• (2) Attributes Identifier SubPayload 0700	
				• Attribute Identifier 0007 [PowerSource]	
	Read attributes response (Standard Type)	18-0A-01-07-00-00-30-01	✓		
				Parser	
				• (3) Read attributes response (Standard Type) 18-0A-01-07-00-00-30-01	
				• (3) Read attribute status record SubPayload 0700003001	
				• Attribute Identifier 0007 [PowerSource]	
				• Status 00 [SUCCESS] [00]	
				• Attribute data type 30 [8-bit enumeration]	
				• Attribute value 01 [01]	
Legend					
Status	OK	Image	✓		
	NO		✗		
	Skipped		⚠		

Fig. 6

Alimentando in PWM una strip a coppie di LED bianchi a tonalità calda e fredda, permette di ottenere una luce con temperatura di colore variabile dalla più calda a quella più fredda; il circuito si comanda a distanza via radio mediante un trasmettitore palmare standard a 433 MHz, 4 canali.



RADIO CONTROLLER PER STRIP A LED

di ALESSANDRO SOTTOCORNOLA

Da quando sono disponibili LED ad alta luminosità emittenti luce bianca, il passaggio di consegne dai tradizionali corpi illuminanti alle lampadine allo stato solido è diventato irreversibile: i LED presentano un'efficienza (espressa in lumen/watt) superiore a quella di quasi tutte le lampade tradizionali (fanno eccezione, al momento, le grosse lampade a vapori di sodio impiegate nell'illuminazione stradale, che però negli ambienti chiusi sono inutilizzabili per potenza e per l'aberrazione cromatica che causano) ed un costo ormai meno proibitivo

di quanto non fosse qualche anno fa. Inoltre sono molto robusti e presentano un rapporto tra flusso luminoso e ingombro decisamente accettabile. In comune con le lampade tradizionali, almeno con quelle fluorescenti e al neon, i LED bianchi hanno la caratteristica di essere prodotti per emettere, a seconda del modello, una tonalità di bianco che varia dal caldo (circa 3.000 K) al freddo (oltre 8.000 K); per questa ragione, quando si vuole realizzare una soluzione lighting a diodi luminosi bisogna fare attenzione a cosa si compera, altrimenti se si

CARATTERISTICHE TECNICHE

- Alimentazione scheda: 12/24Vcc
- Alimentazione strip: 12/24Vcc
- Corrente: 2A per canale
- Codifica RF: MM53200/HT12 con autoapprendimento
- Ripristino stato in caso di black-out
- Funzioni: Dimmer, On/Off
- Gestione di due strip (2 canali) ad anodo comune

prende una lampada qualunque ci si può trovare ad avere un'illuminazione che non è proprio quella desiderata.

Un altro problema sorge se si desidera realizzare un'illuminazione in grado di variare la tonalità del bianco in base al gusto o all'umore del momento, o anche solo per adattarla alle esigenze degli ospiti; in questo

caso è d'obbligo un sistema in grado di variare la tonalità della luce emessa, tuttavia ogni LED nasce per una certa temperatura di colore e non può mutarla. C'è però una soluzione, che consiste nell'impiegare diodi a luce bianca calda e fredda insieme, mischiandone la luce in proporzione variabile.

Questo è ciò che fa il sistema de-

scritto in questo articolo, il quale, sostanzialmente, è un controller per gruppi di LED bianchi o per strisce composte da LED bianchi caldi e freddi pilotabili separatamente (noi l'abbiamo provato con questa soluzione) in PWM, così da variare il flusso luminoso individuale delle due componenti luminose e ottenere una luce che sia la miscela di caldo e freddo e quindi una tonalità a piacere.

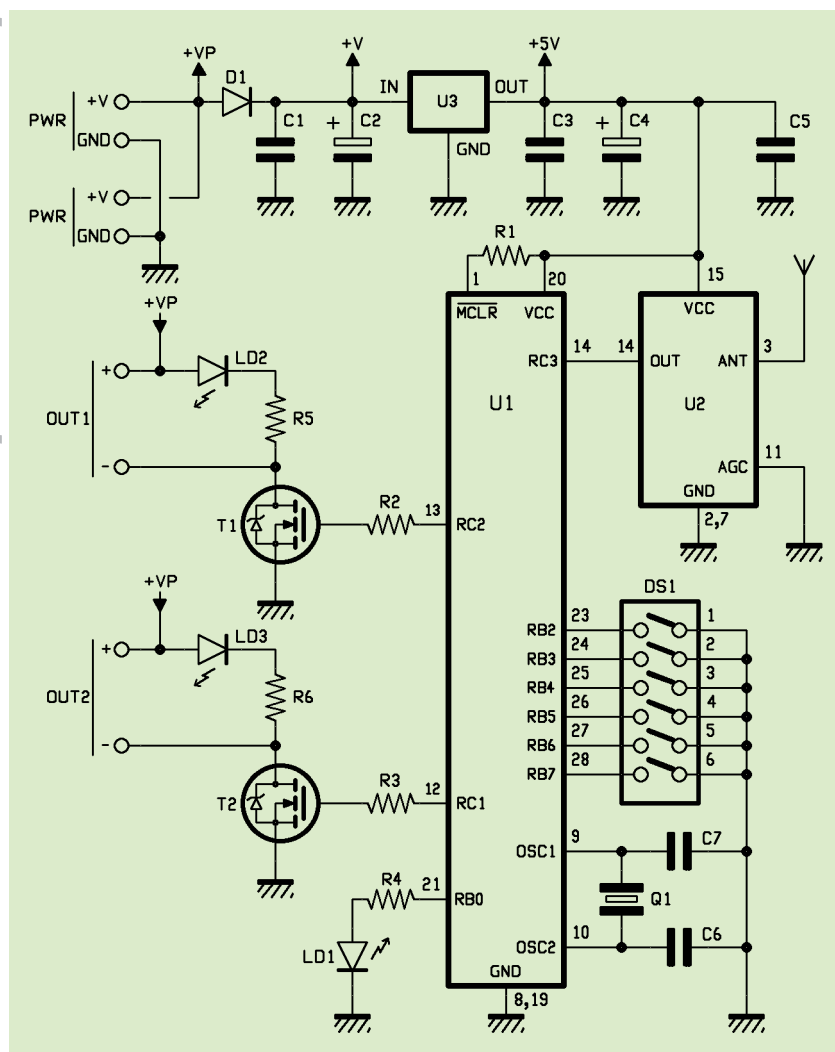
Il sistema consta di un circuito di controllo per strip a LED dotato di ricevitore del radiocomando, controllato a distanza via radio in banda UHF a 433,92 MHz mediante un TX standard (TX3750-4CS) commercializzato da Futura Elettronica; il radiocomando è a modulazione d'ampiezza a quattro canali e con gli altrettanti pulsanti del trasmettitore è possibile gestire individualmente i canali, ovvero alzare e abbassare distintamente la luminosità della fila di LED a luce calda e di quella a luce fredda.

Visto che il trasmettitore è un modello standard commerciale basato sull'integrato UMC UM3750 (un'evoluzione dello storico MM53200 a 4.096 combinazioni con codifica binaria a 12 bit) non lo descriveremo; ci concentreremo, invece, sul ricevitore/controller, di cui pubblichiamo e descriviamo lo schema elettrico nei prossimi paragrafi.

SCHEMA RICEVITORE

Il controller è l'insieme di un ricevitore ibrido Aurel AC-RX2 sintonizzato sui 433,92 MHz e di un microcontrollore PIC16F876A programmato per fungere da decoder del codice del tasto premuto sul telecomando (ma anche per apprendere in automatico un codice di base dello stesso senza impostare dip-switch, se non quello di avvio dell'autoappren-

[schema ELETTRICO]

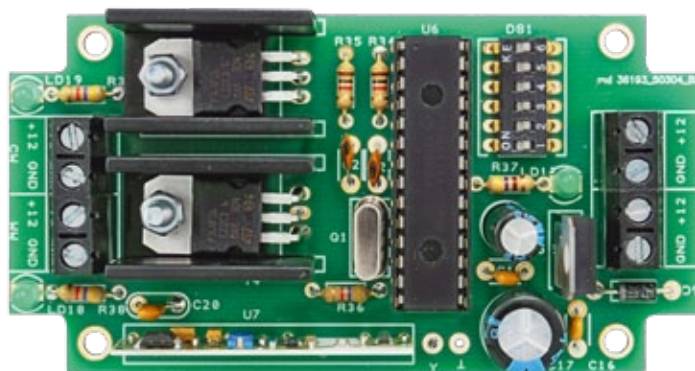
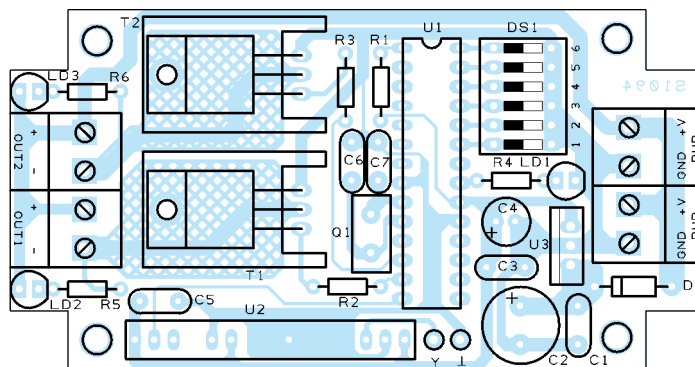


Elenco Componenti:

R1, R2: 1 kohm
 R3: 4,7 kohm
 R4: 470 ohm
 R5, R6: 4,7 kohm
 C1: 100 nF multistrato
 C2: 470 μ F 35 VL elettrolitico
 C3: 100 nF multistrato
 C4: 220 μ F 16 VL elettrolitico
 C5: 100 nF multistrato
 C6, C7: 22 pF ceramico
 D1: 1N4148
 LD1 ÷ LD3: Led 3 mm verde
 U1: PIC16F876A-I/P (MF1094)
 U2: AC-RX2
 U3: 7805
 Q1: Quarzo 4 MHz
 DS1: Disp-Switch 6 vie
 T1, T2: P36N06

Varie:

- Morsetto 2 poli (4 pz.)
- Zoccolo 14+14
- Dissipatore ML26 (2 pz.)
- Vite 12mm 3 MA (2 pz.)
- Dado 3 MA (2 pz.)
- Filo 17 cm per antenna
- Circuito stampato S1094



dimento), generare i due segnali PWM indipendenti e controllare con essi i gate di due MOSFET enhancement-mode a canale N, cui è affidato il compito di pilotare le due file di LED dello strip o, più genericamente, gruppi di LED bianchi. Siccome i transistor sono configurati a source comune, il circuito è adatto a pilotare strip di LED ad anodo comune: infatti sull'apposita morsettiera rendiamo disponibile il positivo dell'alimentazione e i drain dei MOSFET, quindi il circuito può pilotare i diodi portandone verso massa i catodi con i tempi e modi previsti dal PWM.

Diamo uno sguardo allo schema elettrico: il microcontrollore Microchip (montato nella configurazione con oscillatore interno quarzato mediante Q1 -da 4 MHz- e ausiliato dai conden-

satori ceramici C6 e C7), dopo il power-on reset inizializza i propri I/O e imposta RC3 come input senza pull-up dedicato ad acquisire i dati in arrivo dal ricevitore radio U2, mentre sempre come ingressi, ma stavolta dotati di pull-up interno, setta RB2, RB3, RB4, RB5, RB6, RB7, che serviranno a leggere il dip-switch DS1; invece RB0 è impostata come uscita per il comando del LED di segnalazione durante le procedure ed RC1 ed RC2 sono settate come output del segnale PWM che pilota i gate dei MOSFET T1 e T2.

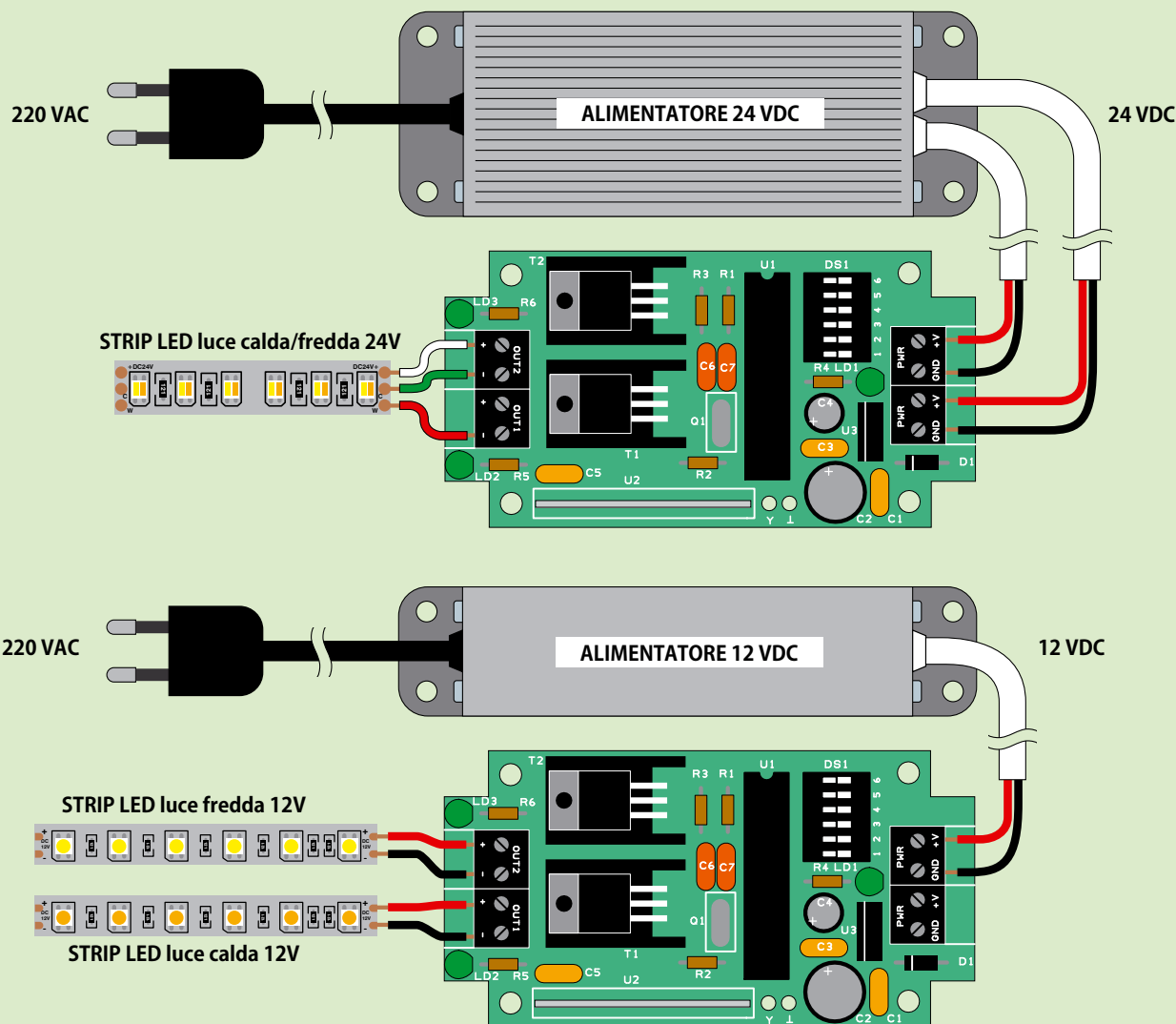
Conformemente a quanto consigliato da Microchip, il piedino / MCLR è collegato all'alimentazione (Vcc, pin 20) mediante un resistore di pull-up.

Prima di proseguire, è opportuno specificare che l'uscita OUT1

(quindi T1) comanda i LED a luce fredda (Cold white) mentre OUT2 agisce su quelli a luce calda (Warm white).

Il modulo radiorecettore U2 è un Aurel AC-RX2 provvisto di amplificatore del segnale d'antenna (che conferisce una sensibilità di ben -106 dB), stadio di sintonia superrigenerativo sintonizzato a 433,92 MHz mediante compensatore tarato in fabbrica e dotato di filtro RF (il filtro serve a migliorare la selettività, che nei superrigenerativi non è mai granché...) e demodulatore d'ampiezza; completa la dotazione del modulo, uno squadratore a comparatore del segnale digitale (a livello TTL) uscente dal piedino 14 e un amplificatore LF del segnale in uscita dal demodulatore AM. Quando premiamo uno dei pulsanti sul trasmettitore, il segnale

Il cablaggio del sistema



Come collegare il circuito del controller/ricevitore all'alimentatore e alle strip a LED: in alto è illustrato il cablaggio con strip unica a due file di LED (bianco caldo e bianco freddo) e tre fili, dove le file di diodi luminosi funzionano ognuna a 24 volt; di conseguenza l'alimentatore da usare è a 24 volt c.c. L'esempio si riferisce a una strip di LED con-

figurati ad anodo comune, quindi il filo dell'anodo si deve collegare al contatto - di una sola delle morsettiere d'uscita (l'altro - va lasciato sconnesso). In basso è invece mostrato il controllo di due distinte strip a 12 volt, ognuna delle quali è formata da un solo tipo di LED; in questo caso l'alimentatore collegato alla scheda è da 12 Vcc.

RF irradiato raggiunge l'antenna ricevente del modulo AC-RX2, il quale demodula la componente dati e la invia al piedino 14; da qui, il microcontrollore prende gli impulsi TTL, li colloca in RAM e li analizza con un'apposita routine firmware che prima di tutto discerne, tra i tanti segnali captati dall'etere, quello compatibile con il formato della codifica UM3750, quindi, in caso affer-

mativo verifica se il codice è uno di quelli memorizzati durante la fase di autoapprendimento, mentre in caso contrario elimina i dati dalla RAM e si predispone a una nuova analisi. Vedremo in un secondo momento come abbinare il trasmettitore al circuito, ricorrendo all'autoapprendimento; per ora vi basti sapere che il firmware prevede di apprendere tutti e quattro i

codici dei comandi, che sono incremento e diminuzione della luminosità dei LED a luce fredda e incremento e riduzione della luminosità dei LED a luce calda. Questa scelta vi permette di adattare il sistema a codifiche sia UM3750 (o MM53200/UM86409) sia Holtek HT-12, oltre che di gestire il circuito mediante due diversi trasmettitori, senza quindi il vincolo di restare legati ad

un unico trasmettitore.

Ora vediamo cosa accade se il segnale ricevuto contiene uno dei codici appresi e mantenuti in EEPROM di lavoro dal PIC16F876: in questa evenienza, si avvia l'apposita routine di gestione del segnale generato dal modulo PWM interno al microcontrollore, segnale che, dopo l'avvio del main program, è caratterizzato da un duty-cycle del 50% (a meno che non stata attivata la funzione di ripristino che spiegheremo qualche paragrafo più avanti). Se il codice ricevuto viene riconosciuto e corrisponde al tasto appreso come UP dell'OUT1 (consigliamo quello in alto a sinistra del radiocomando - canale 2), il PWM destinato al controllo dell'OUT1 subisce un incremento del duty-cycle alla volta, ovvero proporzionale al tempo di pressione; se corrisponde al tasto appreso come DOWN dell'OUT1 (suggeriamo quello in basso a sinistra) diminuisce il duty-cycle del PWM relativo al canale 2. Ancora, se corrisponde al codice appreso come UP dell'OUT2 (pulsante in alto a destra del trasmettitore) causa un incremento del duty-cycle del PWM che pilota OUT2 e infine, se è il codice appreso come DOWN dell'OUT2, causa una riduzione del duty-cycle del PWM dell'OUT2. Resta inteso che questo è il funzionamento in modo *dimmer*; se il codice ricevuto si protrae per meno di un secondo (come verrà spiegato più avanti) si ottiene l'accensione o spegnimento della luce corrispondente. La verifica della durata del codice è effettuata dal firmware.

Ciascuna delle uscite dispone di un LED spia che ne segnala l'attività: per OUT1 si tratta di LD2 e per OUT2 di LD3; ciascun diodo è dotato di resistenza di limitazione della corrente (R5 per

LD2 e R6 per LD3).

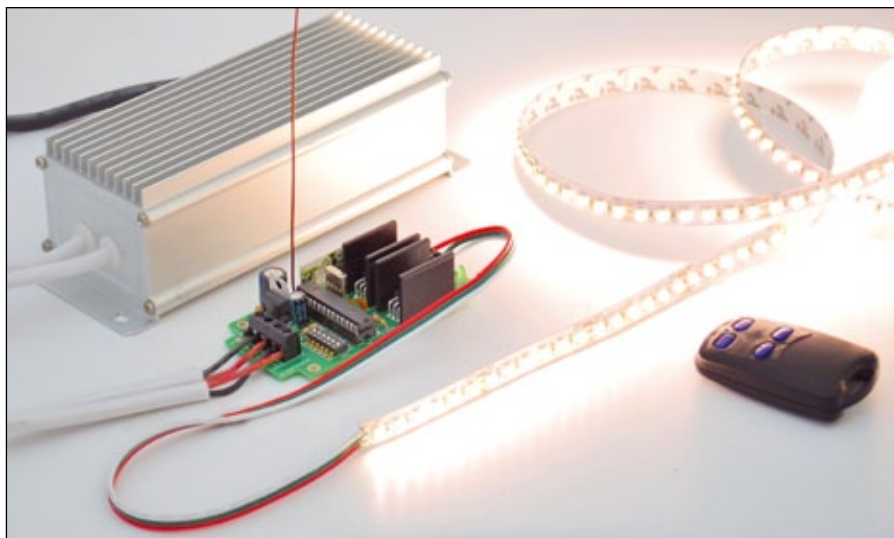
Concludiamo la descrizione dello schema elettrico con lo stadio di alimentazione, che fa capo alle due coppie di morsetti PWR, le quali sono in parallelo tra loro; questa connessione inusuale e apparentemente ridondante, ha in realtà due scopi: il primo è consentire il collegamento di più controller/ricevitori dello stesso tipo, semplicemente portando ad una delle morsettiere i cavi dell'alimentatore e all'altra quelli in arrivo dal circuito seguente. Il secondo scopo è fissare i cavi in eccesso quando si usa un alimentatore per lampade a LED come quello che abbiamo impiegato per le prove (nello specifico è un prodotto della Futura Elettronica, www.futurashop.it, dotato di due uscite a cavo bipolare in guaina, internamente connesse in parallelo): in tal caso, infatti, senza le due morsettiere si connetterebbe un cavo ad essa e l'altro andrebbe isolato mediante nastro isolante, mentre disponendo di due morsetti, uno fa sostanzialmente da "parcheggio".

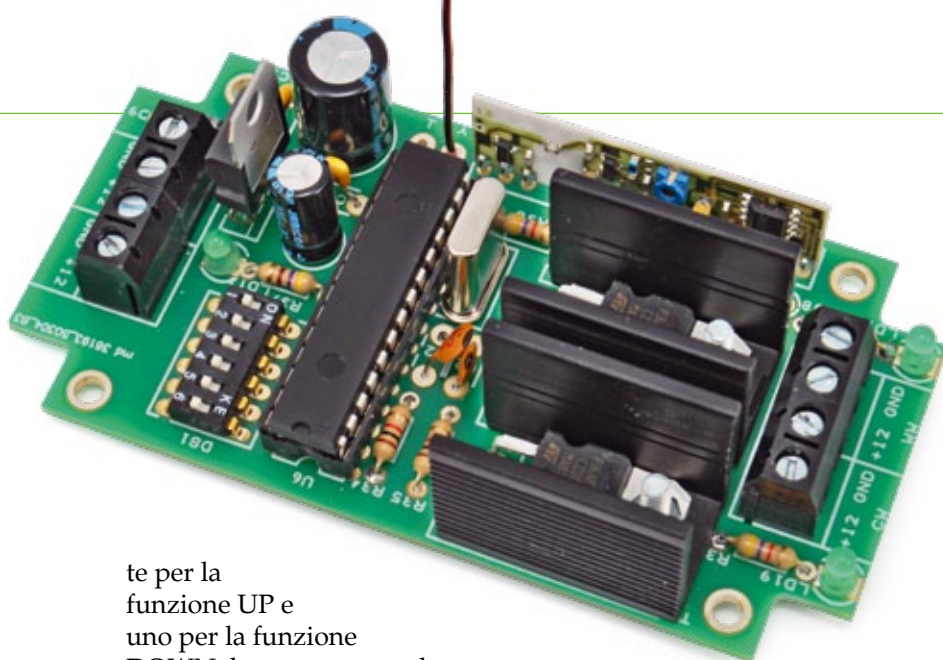
Dai morsetti PWR + e -, tramite il diodo di protezione D1 (evita il danneggiamento del circuito in

caso di inversione della polarità della tensione di alimentazione) la corrente arriva al regolatore integrato a tre terminali U3 (un comune lineare di tipo 7805) ben filtrata da ripple e disturbi impulsivi tramite C1 e da C2; l'integrato ricava una tensione continua ben stabilizzata, che fornisce al microcontrollore e al modulo ricevitore radio U2 tramite i piedini U ed M (quest'ultimo, connesso a massa, è comune al circuito d'ingresso) opportunamente filtrate da C3, C4 e C5.

FUNZIONAMENTO

Il ricevitore prevede il controllo dei due canali, unicamente mediante telecomando a 433,92 MHz avente codifica MM53200/HT12 o compatibile. Ogni canale viene gestito mediante due pulsanti (anche di trasmettitori distinti), pertanto per gestire entrambi i canali sono necessari quattro pulsanti e perciò un trasmettitore da radiocomando quadricanale o due bicanale, da destinare uno al controllo della luminosità della luce calda e uno per il controllo della fredda. Ad esempio, per il canale 1 deve essere memorizzato un pulsante





te per la funzione UP e uno per la funzione DOWN; la stessa cosa vale per il secondo canale. Le due funzioni UP e DOWN permettono, rispettivamente, di incrementare la luminosità e decrementare la luminosità.

Nel caso del trasmettitore quadricanale TX3750-4CS, consigliamo di assegnare i tasti di sinistra al canale OUT1 e quelli di destra ad OUT2; inoltre, allo scopo di rendere il comando più intuitivo, è bene che i pulsanti in alto di destra e sinistra corrispondano alle funzioni UP e quelli in basso alle DOWN dei rispettivi canali. Resta inteso che, siccome il sistema apprende distintamente i codici per ciascuna funzione di ogni canale, se vi viene comodo potete fare diversamente ed assegnare ai pulsanti del trasmettitore (o dei trasmettitori) un ordine diverso da quello consigliato. Riguardo al normale funzionamento, ovvero quello che si ottiene quando i primi 4 DIP sono su Off, l'unità controlla la luminosità in funzione dei comandi via radio impartiti dal TX palmare:

una pressione breve (meno di un secondo circa) del pulsante del radiocomando, sia UP o DOWN, permette di spegnere o accendere la rispettiva uscita in base all'attuale luminosità, quindi se la striscia a LED è accesa, premendo uno dei due tasti del canale, la striscia stessa si spegne e premendolo nuovamente, si riaccende. Se invece si desidera regolare la luminosità, bisogna tenere premuto il tasto per più di un secondo, automaticamente verrà variata la luminosità passo dopo passo, fino al raggiungimento della luminosità massimo o minima.

Qualora venga ricevuto un segnale RF appartenente alla codifica accettata, il LED LD1 si illuminerà emettendo brevi lampeggi, ad indicarne la ricezione. Ogni uscita dispone di un LED (LD2 e LD3) che segue lo stato del dimmer, pertanto può essere utile per verificarne il funzionamento prima di collegare le strip

a LED o per cercare, se le strip collegate non si accendono, qual è la causa (ad esempio, se la strip non si illumina, potrebbe essere saltato il MOSFET o invertita la polarità).

L'unità prevede alcune impostazioni del modo di funzionamento, accessibili tramite un dip-switch da 6 vie che dà accesso alla procedura di memorizzazione, oppure consente di attivare o meno la funzione di ripristino in caso di blackout per il canale desiderato. La corrispondenza tra i dip e le funzioni è esplicita dalla **Tabella 1**.

Per poter memorizzare, ad esempio, il pulsante di un TX da radiocomando e associarlo all'incremento di luminosità (UP) dell'uscita OUT1 (controllo luminosità del bianco caldo), è necessario portare il DIP1 su ON (lasciare il DIP2, DIP3, DIP4 su Off): a questo punto il LED LD1 verde si illuminerà fisso. A LED acceso fisso, premere il pulsante che si desidera associare e rilasciarlo quando il LED verde inizierà a lampeggiare, segno che la memorizzazione è avvenuta correttamente. La memorizzazione di un codice sovrascriverà l'eventuale codice memorizzato in precedenza. La medesima procedura dovrà essere effettuata per il DIP2, il DIP3 e DIP4, che permettono di apprendere i codici assegnati ai tasti 2, 3, 4, del radiocomando, ovverosia a quelli che intendete associare alle funzioni UP e DOWN rimanenti.

Se si desidera rimuovere un pulsante dalla memoria, è possibile farlo a proprio piacimento, in quanto si possono cancellare tutti i pulsanti dalla memoria, oppure solo quello desiderato. Per effettuare la cancellazione, togliere l'alimentazione, quindi portare ad ON uno dei 4 DIP associati

Tabella 1 - Significato dei dip-switch del blocco DS1.

Dip	Stato	Funzione
1	ON	Memorizzazione tasto "Up" uscita OUT1
2	ON	Memorizzazione pulsante "Down" uscita OUT1
3	ON	Memorizzazione pulsante "Up" uscita OUT2
4	ON	Memorizzazione pulsante "Down" uscita OUT2
5	ON	Funzione ripristino attiva su uscita OUT1
6	ON	Funzione ripristino attiva su uscita OUT2

La App calcola il sistema a LED

Vista la diffusione e considerato il business originato dalle soluzioni di illuminazione a LED, i vari costruttori e commercianti di lampade e corpi illuminanti a diodi luminosi si sono attrezzati per fornire ai tecnici del settore strumenti e prospetti on-line che li guidino nella progettazione rapida dell'illuminazione dell'ambiente e nella scelta dei prodotti adatti a realizzarla. Tra questi si distingue Philips, che propone oltre a strumenti software di calcolo (ad esempio CalcuLux, che vale per



ambienti chiusi e aperti) anche App per iPhone e Android come LED Road Calculator e Lighting Hub App; questa App si trovano alla pagina web www.lighting.philips.it/connect/lighting_apps.wpd.

al pulsante di cui rimuovere il codice (oppure anche più di un DIP insieme, se intendete cancellare in un sol colpo più codici) e poi fornire l'alimentazione alla scheda. Una volta alimentato il circuito, il LED LD1 rimarrà acceso fisso per circa 2 secondi a indicare l'avvenuta cancellazione del pulsante dalla memoria, dopodiché il dispositivo comincerà a funzionare in modo normale. Quando il circuito sarà entrato nel normale funzionamento, bisognerà aprire i DIP che avete chiuso, ovvero togliere alimentazione, riaprire i DIP e, trascorsi almeno dieci secondi, alimentare di nuovo il circuito.

Notate che normalmente, quando il controller/ricevitore viene alimentato in "normale accensione" (vale a dire con i primi quattro DIP su Off, cioè aperti), il LED LD1 emetterà una serie di lampeggi ad indicare il corretto avvio del sistema.

Concludiamo con il DIP5 e il DIP6, i quali permettono di impostare o meno la funzione di ripristino rispettivamente sulla prima e sulla seconda uscita; per ripristino si intende che in caso di blackout, alla riaccensione del circuito le uscite tornano come prima dell'interruzione dell'alimentazione. Tale funzione viene ottenuta semplicemente

salvando in un apposito spazio della EEPROM del PIC16F876A lo stato di ciascuna uscita ogni volta che da remoto, tramite il trasmettitore del radiocomando, viene modificata; alla riaccensione, dopo l'inizializzazione degli I/O il main program del microcontrollore va a vedere se in EEPROM è stato settato il bit flag della funzione ripristino e in caso affermativo guarda nella stessa EEPROM se ci sono dati riguardanti l'impostazione del duty-cycle del PWM diverse dal 50%: in caso affermativo reimposta i duty-cycle come scritto in memoria, mentre se non è così, parte con il valore predefinito del 50%. Ricordiamo che la funzione di ripristino è attiva se il rispettivo DIP è su On (vedere la solita **Tabella 1**).

REALIZZAZIONE PRATICA

Bene, spiegato il sistema, visto lo schema elettrico e descritto il funzionamento, non ci resta che affrontare la costruzione, della quale descriviamo solo quella riguardante il controller/ricevitore, dato che il trasmettitore si acquista già pronto all'uso. Trovate la traccia per realizzare il circuito stampato, insieme agli altri file del progetto, nell'apposita sezione del nostro sito web www.elettronica.in.it; scaricato il

file potete, anche con una semplice stampa su acetato, ottenere la pellicola da cui iniziare la preparazione (per fotoincisione) della basetta, la quale ad incisione completa dovrà essere forata con punta da 1 mm di diametro per tutti i fori ad eccezione di quelli per le morsettiere (da 1,5 mm di diametro) e quelli di fissaggio (punta da 3 mm).

Fatto ciò, disponete i componenti partendo da quelli a basso profilo (resistenze, diodi al silicio, zoccolo per il microcontrollore) e procedendo in ordine di altezza; terminate con il regolatore 7805, che va montato in piedi seguendo l'orientamento indicato nel piano di montaggio che trovate in queste pagine e che dovete seguire per orientare in maniera corretta i diodi, i MOSFET, il dip-switch. Quanto all'AC-RX2, si monta soltanto nel verso giusto e va tenuto in piedi.

Questo componente va provvisto di un'antenna ricevente, che prevedendo un uso domestico, può essere un semplice spezzone di filo di rame nudo saldato nella piazzola con il simbolo di antenna situata vicino all'U2, nello spazio tra questo e l'elettrolitico C2 (attenzione che la piazzola a destra, a ridosso di questo condensatore, è la massa dell'antenna e in questo caso non serve).

Alimentatori per LED e strip a LED



Per alimentare i sistemi di illuminazione a LED esistono alimentatori AC/DC che si collegano con l'ingresso alla rete elettrica a 220 Vca e forniscono tipicamente da 12 a 24 volt in continua; per il nostro circuito, avendo montato una strip a LED che richiede un'alimentazione di 24 Vcc, abbiamo fatto uso di un alimentatore da 60 W di uscita (cod. 8220-LPS24060, Euro 36,00) commercializzato da Futura Elettronica (www.futurashop.it). La versione a 12 Vcc da 45 W (cod.

8220-LPS04512R) costa 34,00 Euro. La strip a LED che abbiamo provato è a due tonalità di bianco (modello con LED a tono caldo e freddo montati su linee distinte, ad anodo comune); anche questa si acquista da Futura Elettronica, con il codice 8220-LQ24N550CWW al prezzo di 199,00 Euro. Sono anche disponibili strip separate, più economiche, come il modello 8220-LS12M210CW (bianco freddo, Euro 69,00) o il modello 8220-LS12M210WW (bianco caldo, Euro 69,00).

Va bene anche uno spezzone di filo in rame smaltato da 1 mm e lungo 17÷18 cm (così ottenete un'antenna a 1/4 d'onda) saldato nello stesso punto, avendo però l'accortezza di raschiare via lo smalto con una lama di un paio di forbici o con della tela smeriglio l'estremità da saldare, perché diversamente la lega saldante non si attacca.

A montaggio completato, potete racchiudere il circuito in un contenitore non metallico (noi abbiamo previsto il Coffer 1 della Teko); se optate per un contenitore metallico, dovete montare all'esterno un'antenna a stilo accordata sui 433,92 MHz (434 MHz) e connessa al circuito stampato con del cavetto schermato RG59, usando il conduttore interno per il segnale (cioè la piazzola d'antenna vicina all'AC-RX2) e quello di massa per lo schermo. La calza, a meno che l'antenna non abbia un piano di massa (non sia, cioè, una Ground Plane) va collegata solo allo stampato e isolata dal lato dell'antenna stessa; curate anche che lo stilo non venga in contatto con il metallo della scatola, che di norma dovete collegare alla

massa presa dalla morsettiera di alimentazione (una o l'altra è irrilevante).

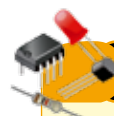
Quanto ai collegamenti, ricordate che a + e - PWR va applicata l'alimentazione prelevata da un alimentatore (va benissimo uno switching, anche perché ha un rendimento molto più elevato di un lineare, delle cui doti di pulizia della tensione fornita, in questa applicazione, non sapremo che farcene...) capace di erogare da 12 a 24 Vcc (dipende da qual è la tensione nominale della strip di LED da voi utilizzata) e una corrente di 4 A (o inferiore, in base all'assorbimento della striscia di LED utilizzata).

La striscia va collegata alle uscite OUT1 e OUT2 badando che il positivo comune vada sul + di una morsettiera tra OUT1 o OUT2 e che gli altri due fili si connettano al - delle morsettiere OUT1 e OUT2 ricordando che quello corrispondente ai LED a luce calda va sul - di OUT1, mentre quello dei LED a luce fredda deve finire su OUT2. Se invertite questi due fili, i pulsanti del canale OUT1 controlleranno OUT2 e viceversa.

La connessione descritta vale per la strip che abbiamo usato noi nel-

le nostre prove, la quale ha tre fili; se invece ne trovate una a quattro, ossia con + e - distinti per le file di LED a luce calda e fredda, connettete i relativi + e - alle rispettive morsettiere.

Terminiamo ricordando che il circuito può controllare anche gruppi di LED messi insieme da voi a piacimento, rammentando però la tensione disponibile (quindi quella dell'alimentatore) e tenendo presente che la caduta su un LED bianco a piena luminosità supera i 4 volt; perciò con 12 V si possono accendere bene due LED in serie (il resto deve cadere sul MOSFET e sul resistore serie che collegherete) mentre con 13 V ne accendete 3 in serie (sempre interponendo una resistenza calcolata sulla corrente nei LED) e con 24 V ne fate accendere 5 (con la solita resistenza serie). Inoltre, si possono connettere anche più di due strip a LED: dipende dal loro assorbimento; l'importante è che complessivamente non chiedano più di 4 ampere (2 A per il bianco caldo e altrettanti per il bianco freddo). ■



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista (www.elettronicaonline.it) così come il firmware utilizzato per programmare il microcontrollore. Il controller è anche disponibile in scatola di montaggio (cod. FT1094K, Euro 27,00), comprensiva di tutti i componenti, C.S. forato e serigrafato, micro programmatore, modulo radio e tutte le minuterie. Il telecomando da utilizzare (già montato e collaudato) costa 22,00 Euro (cod. 8110-TX3750-4CS). 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>

Conoscere e usare



kicad

Iniziamo il nostro viaggio alla scoperta di KiCad, un CAD elettronico open source che permette di coprire l'intero ciclo di sviluppo di una scheda elettronica, dalla realizzazione degli schemi elettrici fino allo sbroglio del PCB e alla realizzazione dei file di fabbricazione. Prima Puntata.

di Francesco Ficili e Vincenzo Germano



Da buoni tecnici, sappiamo che il primo passo di un progetto elettronico è la realizzazione dell'hardware sul quale esso si basa. Lo sviluppo di una scheda elettronica può essere affrontato a diversi livelli di complessità e la progettazione può essere più o meno professionale e richiedere o no l'utilizzo di strumenti CAD (Computer Aided Design) più o meno complessi. La progettazione assistita dal computer è una realtà assodata da decenni in vari settori, che vanno dall'edilizia alla meccanica e all'aeronautica, e, naturalmente, all'elettronica. I programmi CAD per elettronica (in questo

caso si parla di ECAD -Electronic CAD- oppure di EDA, ossia Electronic Design Automation) esistono da almeno trent'anni, ma mentre un tempo si trovavano solo prodotti per il settore professionale (molto costosi e inaccessibili al grosso del pubblico), da diversi anni sviluppatori e semplici appassionati hanno creato software di progettazione freeware o comunque di basso costo che hanno portato il CAD nei computer di molti sperimentatori. Per il settore professionale esistono ottimi software come ad esempio OrCAD della Cadence, Altium dell'omonima azienda australiana; nel campo operano altre software house come Mentor Graphics,

FIG. 1
Logo di
KiCad.



Synopsys, Zuken e molti altri. Questi tools, che sono indubbiamente molto performanti, costano cifre che solo aziende con un certo giro d'affari possono sostenere: le licenze dei tools di EDA possono, infatti, costare da alcune migliaia ad alcune decine di migliaia di euro (per licenze multi postazione).

Fortunatamente la diffusione dell'open source ha toccato anche questo mondo, venendo in soccorso dell'hobbista che vuole cimentarsi nella progettazione elettronica, ma anche del professionista e della piccola azienda che hanno necessità di progettare schede elettroniche destinate all'uso interno o a progetti che non producono un ritorno economico tale da giustificare la spesa per l'acquisto di un ECAD professionale.

Tra i software open source spicca KiCad, una Electronic Design Automation Suite completamente open e multi piattaforma, sviluppato e distribuito sotto licenza GNU da Jean-Pierre Charras. Chiaramente il livello di questo CAD non può essere paragonato a quello degli strumenti elencati in precedenza, ma è sicuramente paragonabile a tool ECAD a pagamento, come ad esempio Eagle CAD di Cadsoft.

Sebbene completamente open source, il tool si

dimostra estremamente valido e gode di una fama che va crescendo nel panorama hobbistico. In Fig. 2 è possibile osservare il trend di crescita della richiesta del software dall'anno del suo primo rilascio, nel 2007, comparato con quello di ECAD simili.

KiCad è utilizzato da moltissimi appassionati e piccole aziende in tutto il mondo, grazie alla semplicità di utilizzo che lo caratterizza, al fatto di essere completamente free ed al taglio professionale che ha raggiunto nel corso degli anni.

INTRODUZIONE A KiCad

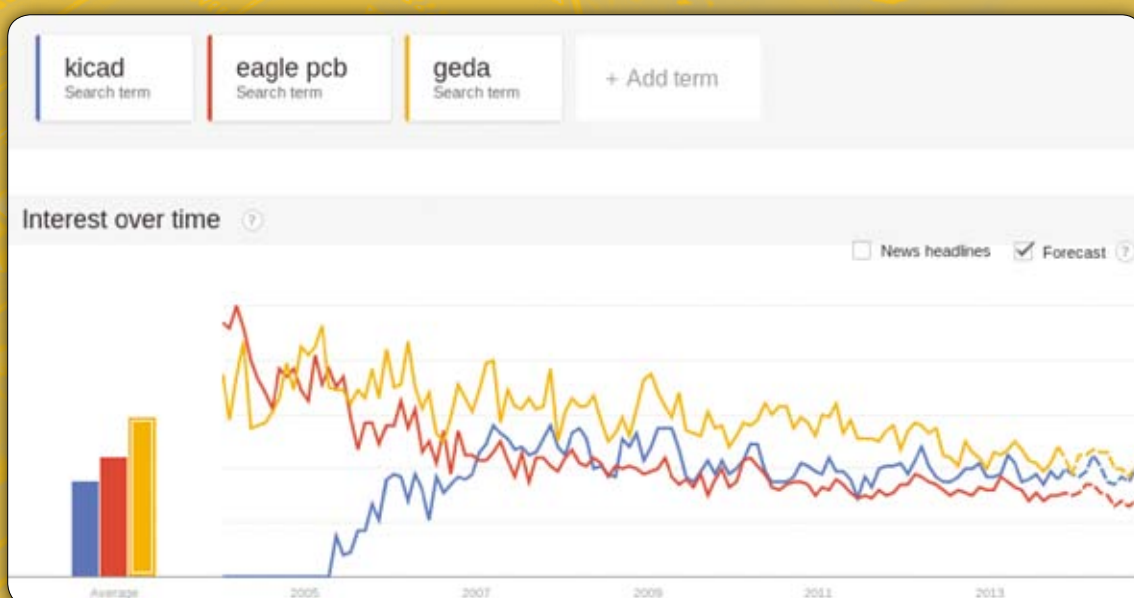
KiCad è un tool CAD specificatamente pensato per il disegno di schemi e circuiti elettronici. Attualmente KiCad è distribuito sotto licenza GNU GPLv2 in maniera completamente gratuita, ed è disponibile per Windows, Linux e Mac OS X.

Tramite KiCad è possibile realizzare schemi elettrici, distinte componenti e PCB fino a 16 layer, quindi tutti i master possibili. All'interno di KiCad è presente un gestore integrato delle librerie dei componenti, completo di editor sia per gli schemi elettrici che per i footprint.

Nel CAD sono inoltre integrati un visualizzatore di file gerber ed altri strumenti di supporto alla progettazione.

KiCad è un IDE (Integrated Development

FIG. 2
Diffusione
di KiCad a
confronto
con quella
di altri
ECAD di
prestazioni
simili.



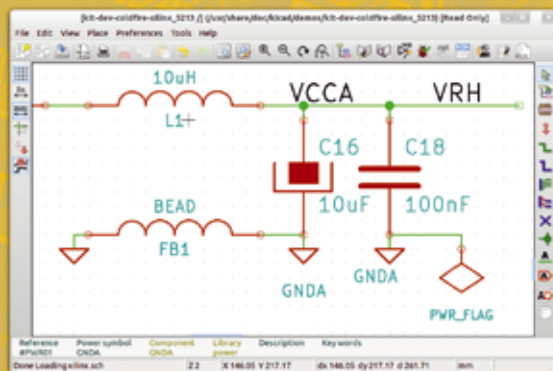


FIG. 3 - Esempio di schema elettrico realizzato con Eeschema.

Environment), ed è quindi costituito da un set di tools software interdipendenti e racchiusi all'interno di un Project Manager, che provvede alla gestione dei file di progetto e all'avvio dei singoli programmi da un'unica interfaccia utente.

All'interno del Project Manager sono racchiusi quattro programmi principali di seguito descritti:

- *Eeschema*: è l'editor degli schemi elettrici. Integra l'editor per le librerie degli schemi

Component	Footprint
1 ABRT_SW1	SW_PUSH : SW_PUSH_SMALL
2 ALLPST1	CONN_1 : PINTST
3 BDM_EN1	JUMPER : PIN_ARRAY_2X1
4 BDM_PORT1	CONN_13X2 : pin_array_13x2
5 C1	10pF : SM0805
6 C2	10pF : SM0805
7 C3	1nF : SM0805
8 C4	100nF : SM0805
9 C5	100nF : SM0805
10 C6	100nF : SM0805
11 C7	100nF : SM0805
12 C8	100nF : SM0805
13 C9	100nF : SM0805
14 C10	100nF : SM0805
15 C11	100nF : SM0805
16 C12	100nF : SM0805
17 C14	100nF : SM0805
18 C15	100nF : SM0805
19 C16	10uF : SM1206POL
20 C17	100uF : SM1206POL
1 C1	
2 C1-1	
3 C2	
4 SM0402	
5 SM0603	
6 SM0603_Capa	
7 SM0603_Resi...	
8 SM0805	
9 SM1206	
10 SM1206POL	
11 SM1210	
12 SM1210L	
13 SM1812	
14 SM1812E	
15 SM1812L	
16 SM2010	
17 SM2112L	
18 SM2512	
19 SM2403	
20 SM2403/VF	

FIG. 4 Esempio di associazione componente footprint.

elettrici. In Fig. 3 è rappresentato uno screenshot di una porzione di schema elettrico realizzata con Eeschema.

- *Cvpcb*: è il programma che effettua l'associazione componente-footprint (il footprint è il layout del componente sul circuito stampato, ovvero il suo ingombro), permette di associare lo schema di un componente ad un footprint. In Fig. 4 è riportato uno screen-

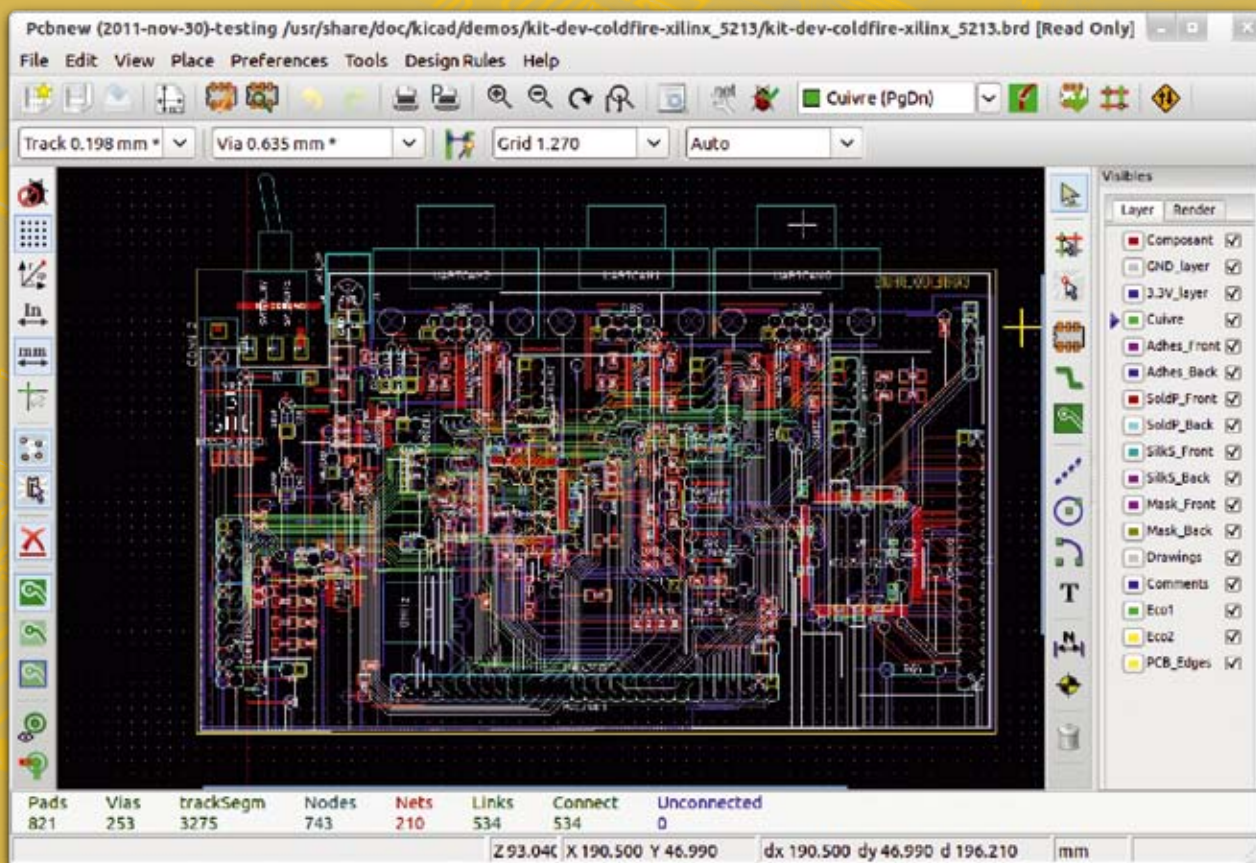


FIG. 5 - Esempio di layout realizzato con Pcbnew.

FIG. 6 - Download di KiCad.



- shot di una distinta elaborata con Cvp pcb.
- *Pcbnew*: è l'editor del PCB. Può gestire PCB fino a 16 layer e integra un tool per la generazione dei gerber e un editor per le librerie dei footprint. Inoltre dispone di un tool di visualizzazione 3D del PCB. In Fig. 5 è riportato uno screenshot di un layout di PCB realizzato con Pcbnew.
- *Gerberview*: è il visualizzatore di file gerber.

Nelle versioni più recenti sono inoltre disponibili altri due strumenti aggiuntivi:

- *Bitmap2Component* è un convertitore bitmap-footprint, utile per inserire i loghi all'interno dei PCB.
- *PcbCalculator* è un tool di calcolo per PCB, contenente diversi utili fogli di calcolo tra cui un calcolatore per il dimensionamento dei dissipatori e relativo ingombro, un calcolatore per la larghezza piste, per l'isolamento elettrico (distanza tra le piste) ed altro ancora.

Durante il corso analizzeremo in dettaglio tutti gli strumenti di lavoro, con il supporto di un progetto pratico che si svilupperà lungo tutta la durata del corso stesso e che vi porterà a realizzare una demoboard per microcontrollori Microchip.

DOWNLOAD E INSTALLAZIONE

Passiamo adesso alla fase operativa del corso,

iniziando con il download e l'installazione del tool: KiCad può essere liberamente scaricato dal web collegandosi al suo sito ufficiale <http://www.kicad-pcb.org/>. Una volta all'interno della pagina, selezionare la voce "Download" dal menu che si trova sulla sinistra, così da accedere alla pagina di download. Una volta entrati, dall'apposito menu in basso (Fig. 6) selezionare il tipo di download desiderato. Sono disponibili, per il download, versioni per Windows, Linux e Mac OS X (una versione per Ubuntu è attualmente in lavorazione) ed è inoltre disponibile il download del codice sorgente di KiCad, così che chi lo desidera può sviluppare una propria versione con funzioni personalizzate. In questo corso faremo riferimento solo alla versione per Windows. Cliccando sul pulsante con il logo di Windows comparirà la finestra rappresentata in Fig. 7, che rappresenta il contenuto della directory di

FIG. 8 - Welcome window dell'installer di KiCad.



FIG. 7
Contenuto
della directory
di download.

lut-tice.ujf-grenoble.fr - /cao/

[To Parent Directory]

28/11/2011	19:33	1873	AUTHORS.txt
24/05/2013	09:37	18635	COPYRIGHT.txt
10/11/2008	13:22	569	full_version_linux-install.txt
25/01/2012	21:15	1137	how_to_download_sources.txt
23/01/2012	11:00	7042	install.txt
07/07/2013	17:13	280723247	kicad-2013-07-07-BZR4022-stable-UBUNTU 12.04 full with components doc.tarx
16/03/2013	08:12	119979606	kicad-docs+translations_only_for-stable 2013-03-15_BZR4003.zip
18/05/2013	19:50	15025159	kicad-sources-stable 2013-05-18_BZR4017.zip
07/07/2013	15:18	207245212	KiCad stable-2013.07.07-BZR4022_Win_full_version.exe
07/07/2013	17:36		old_versions
08/03/2013	11:16		sources
24/11/2008	10:03	168	web.config

FIG. 9A - Finestra di "License Agreement".



FIG. 9B - Finestra di selezione dei componenti.



FIG. 10 - Selezione del path di installazione.

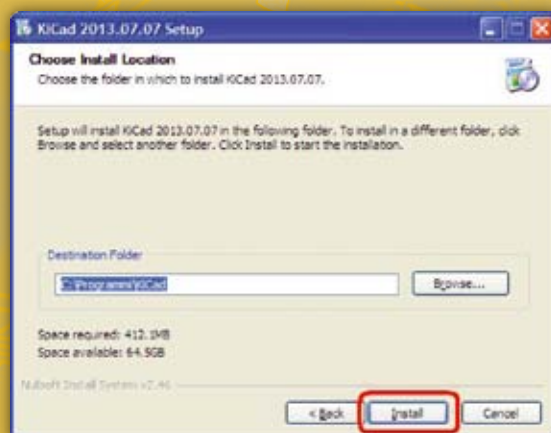


FIG. 11 - Processo di installazione.

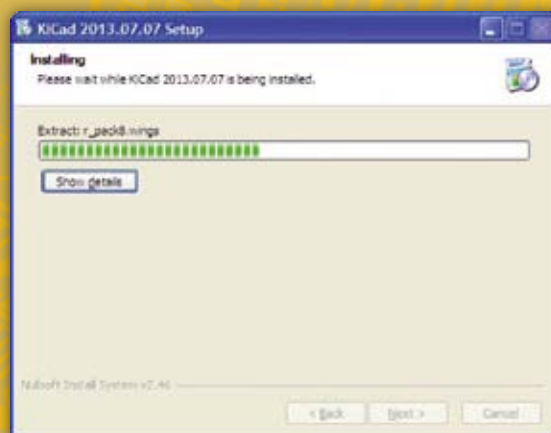


FIG. 12 - Installazione eseguita con successo.



download. Selezioniamo il file *KiCad_stable-2013.07.07-BZR4022_Win_full_version.exe*, che costituisce l'ultima versione stabile disponibile al momento della scrittura di questo corso. Dopo aver cliccato sul link, si avvierà il download del pacchetto di installazione. Completato il download, possiamo lanciare l'eseguibile, in modo da avviare la procedura di installazione di KiCad. Dopo aver lanciato l'installer, apparirà la "Welcome Window" illustrata in Fig. 8. Qui bisogna cliccare sul pulsante "Next" per procedere. A questo punto comparirà la finestra di "License Agreement", dove bisognerà cliccare sul pulsante "I Agree" e procedere. Nella finestra successiva verrà richiesto quali componenti si desidera installare: oltre ai file binari dei vari programmi, vengono installate delle librerie predefinite e tutta la documentazione. Questa finestra può risultare utile se in futuro nasce la necessità di reinstallare i binari senza sovrascrivere le librerie (che potreste avere modificato nel corso del tempo), evitando così di sovrascrivere quelle esistenti. Per il momento

limitatevi a selezionare tutti i componenti installabili per poi procedere (Fig. 9b). A questo punto non vi resta che selezionare il path di installazione del software (come illustrato in Fig. 10) e procedere, cliccando sul pulsante "Install". L'installer inizierà l'installazione del software e dei vari componenti (Fig. 11). Se tutto andrà

Breve storia di KiCad

KiCad è stato realizzato nel corso degli anni da Jean-Pierre Charras, un ricercatore del GIPSA-LAB e docente allo IUT de Saint Martin d'Hères in Francia. Abbiamo scritto a Jean-Pierre per rivolgergli alcune domande circa la storia di KiCad.

FF: Come nasce questo progetto?

JPC: Il progetto nasce nel 1992, anno in cui ho iniziato a scrivere il codice di quello che oggi è Pcbnew. All'epoca ero solo interessato a capire come un editor di layout potesse funzionare (usavo OrCAD per MS-DOS per i miei progetti elettronici). Ho iniziato a lavorare quando i primi compilatori a 32-bit free (DJGPP) e le prime librerie grafiche (libgrx) sono apparse nel web. Due anni dopo avevo reso utilizzabile (solo da me, perché all'epoca era soltanto un esperimento...) una prima versione di editor per layout.

FF: Poi cosa è successo?

JPC: A quel punto avevo un editor per layout ma non un editor per schematici. Un giorno ho trovato i sorgenti di un editor per schematici, molto primitivo e privo di diverse parti, che però era un punto di partenza. Così ho iniziato a lavorare su quello che oggi è Eeschema. Quando wxWindows è divenuto utilizzabile l'ho usato per rimpiazzare la prima libreria grafica. Qualche anno dopo vedeva la luce una prima versione di un tool completo (schematic + layout): era un tool base, ma utilizzabile. Era KiCad.

FF: Qualche aneddoto divertente?

JPC: Beh... quando ero docente a Grenoble, ho chiesto a un gruppo di studenti di provare a disegnare una scheda elettronica usando KiCad, per validare la prima release. Loro hanno accettato ed hanno realizzato prima lo schema e poi il layout. Tutto perfetto, nessun bug, nessun problema. Solo alla fine ne abbiamo trovato uno: il comando di salvataggio in Pcbnew non funzionava, così la prima scheda realizzata con KiCad non ha mai visto la luce...

a buon fine, al termine del processo vedrete comparire la schermata di **Fig. 12**. Nella schermata conclusiva, l'installer vi avvisa che il software è stato correttamente installato e che i componenti 3D presenti in KiCad sono stati realizzati con Wings3D; inoltre vi informa che per creare i vostri elementi 3D personalizzati è necessario installare tale componente software.

In questo corso non cureremo tale aspetto, limitandoci ad utilizzare la visualizzazione 3D con i modelli esistenti, senza crearne di nuovi. Lasciamo ai lettori più esperti ed intraprendenti l'esplorazione di questa possibilità aggiuntiva, che permette di personalizzare la libreria di elementi grafici del 3d Viewer.

KiCad PROJECT MANAGER

Ora che abbiamo completato l'installazione del nostro ambiente, possiamo lanciarlo ed analizzare l'interfaccia, in modo da comprenderne la struttura.

Siccome KiCad è un IDE, dispone di una serie di tool integrati all'interno di una singola interfaccia detta Project Manager. Questo applicativo si occupa della gestione dei file di progetto e dell'integrazione dei vari applicativi, offrendo all'utente un'interfaccia unica per la gestione del progetto.

Il Project Manager di KiCad è costituito da quattro componenti principali.

- **Project Bar:** questa è l'area di gestione del progetto, da qui è possibile salvare o caricare un progetto, comprimerne i file o aggiornare la visualizzazione della Navigation Bar. Sono inoltre disponibili tutte le opzioni di sistema, come ad esempio le opzioni relative alla lingua ed è possibile accedere al manuale.
- **Navigation Bar:** è l'area di gestione del file system del progetto, da qui è possibile vedere tutti i file relativi al progetto ed aprirli.
- **Tool Bar:** è l'area dei tool che costituiscono l'IDE, i principali, come descritto anche in precedenza sono Eeschema, Cvpcb, Pcbnew e Gerbview.
- **Output Window:** questa è la finestra di output di KiCad, da questa finestra il tool ci comunica che tipo di operazioni sono

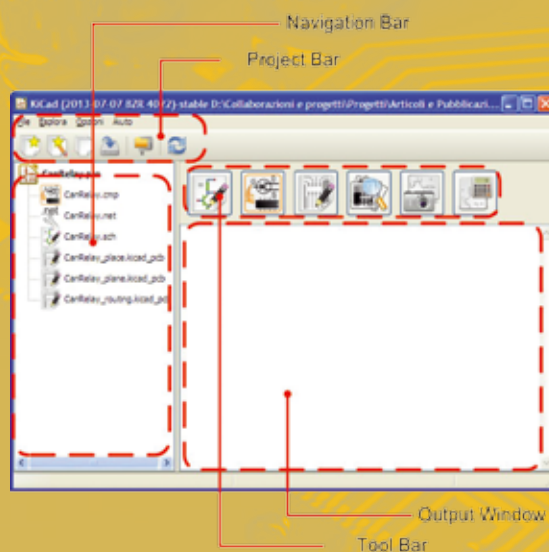


FIG. 13 - Project Manager di KiCad.

state eseguite e ci segnala eventuali errori o avvertimenti.

In **Fig. 13** è riportata la finestra del Project Manager e sono mappate le aree descritte in precedenza.

Il Project Manager è anche l'interfaccia utilizzata per creare un nuovo progetto; per fare ciò è sufficiente cliccare sull'icona "Nuovo Progetto", posizionata all'estrema sinistra della "Project Bar", come illustrato in **Fig. 14**.

Una volta fatto, verrà presentata una finestra di dialogo per la selezione del path e del nome del progetto; scelti questi parametri, è possibile cliccare sul pulsante "Salva" per salvare il file con estensione **.pro**, che costituisce il file progetto di KiCad. A questo punto è possibile utilizzare i vari tools messi a disposizione dal CAD. Nei successivi paragrafi chiariremo come viene gestito il flusso di progetto con KiCad.

FLUSSO DI PROGETTO KiCad

KiCad è impostato in modo da agevolare l'utilizzo di un certo work-flow, che parte con la tracciatura dello schema elettrico fino a terminare con la generazione dei file gerber. L'accesso a uno step successivo avviene solo dopo il completamento di quello precedente, in modo da minimizzare il più possibile la generazione di errori durante il progetto, che sarebbero molto onerosi da individuare e recuperare nelle fasi finali. Il flusso di progetto di KiCad è schematizzato in **Fig. 15**: qui, i blocchi indicati in grigio rappresentano tool che non sono integrati all'interno del CAD, ma che generano

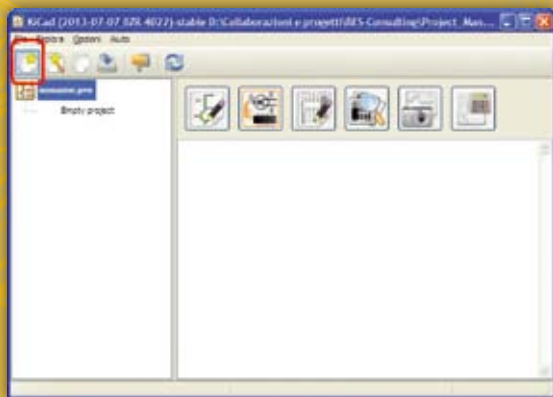


FIG. 14 - Creazione di un nuovo progetto.

External Tools

KiCad dispone di una serie di tool esterni per agevolare gli utenti in alcuni compiti. Un elenco di alcuni di questi tool si trova sul sito web di KiCad, all'interno della sezione External Tools. Eccone una breve descrizione.

Quick KiCad Library Component Builder: permette la generazione di componenti per Eeschema. Ha un'interfaccia web molto intuitiva, che semplifica la generazione del componente.

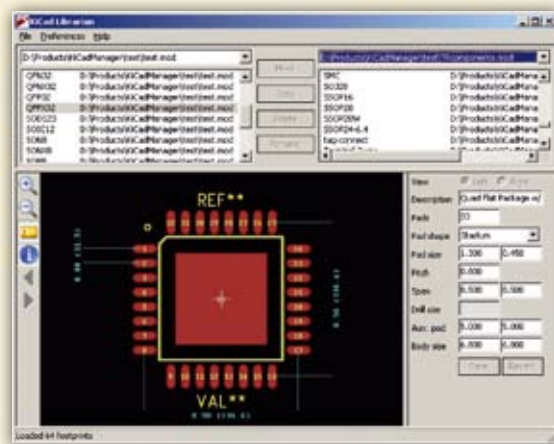
KiCad Librarian: tool di library management per librerie di schemi e footprint di KiCad. È stand-alone e disponibile per le stesse piattaforme per cui è disponibile KiCad. È un ottimo tool per mantenere in ordine le librerie.

CycleRecorder's footprint builder: tool per la generazione di footprint per Pcbnew.

Madparts footprint editor: tool per la generazione di footprint. Supporta KiCad ed anche Eagle. Disponibile per Windows, Linux e Mac OS X.

KiCad 3D Models generator: tool per la generazione di modelli 3D parametrici per KiCad 3D viewer.

FIG. 16 - KiCad Librarian.



degli output compatibili con questo (una lista di questi tool è presente sul sito stesso di KiCad, e ce ne sono molti altri disponibili gratuitamente in rete). Come si può vedere, il primo passo dello sviluppo di un progetto in KiCad (dopo la creazione del progetto stesso) è il disegno dello schema elettrico. Il punto di partenza di questo blocco è una libreria di componenti esistente o la creazione dei componenti necessari tramite il "Component Library Editor". Si consideri che la creazione di componenti per Eeschema non necessita della definizione del footprint, in quanto in KiCad gli schemi elettrici e i footprint sono due entità distinte, che fanno capo a due diverse librerie (e ad altrettanti editor), aventi file di tipo differente (**.lib** per gli schemi elettrici e **.mod**

per i footprint). Questo approccio è differente rispetto ad altri ECAD similari e rappresenta una delle particolarità di KiCad.

Una volta procurata o generata la necessaria libreria di componenti è possibile passare al disegno effettivo dello schematico su Eeschema: questo processo produce due differenti output: un file **.sch** che costituisce il file di lavoro di Eeschema ed un file **.net** (netlist) che costituisce il punto di partenza per il tool successivo, ossia Cvpcb. Il compito di Cvpcb è associare agli schemi elettrici un footprint; per assolverlo, Cvpcb acquisisce in ingresso, oltre alla netlist, anche le librerie dei footprint, che anche in questo caso possono essere pre-esistenti o generate tramite un altro tool (il "Module Editor" di Pcbnew o un tool esterno per la generazione di footprint per KiCad).

Cvpcb costituisce quindi l'anello di congiunzione tra gli schematici ed i footprint, mettendo insieme queste due entità, che fino ad ora risultavano separate. Tale approccio, se da un lato apparentemente introduce uno step aggiuntivo (solo teorico, in quanto la selezione del footprint deve essere fatta comunque, anche se eseguita a livello di schematic design), dall'altro riduce notevolmente le dimensioni delle librerie. L'output del processo di associazione di Cvpcb è il file **.net** di partenza, arricchito con le informazioni relative ai footprint. Questo file viene preso come input da Pcbnew, che lo utilizza come

file di partenza per la generazione del file **.brd**, ossia il file di lavoro di Pcbnew stesso.

A questo punto può avvenire lo sbroglio del PCB, che produce come output un file Gerber o Postscript (PS/PDF) da utilizzare per la fabbricazione della scheda stessa. Lo step finale del lavoro è l'ispezione dei file di output generati tramite Pcbnew, utilizzando Gerbview. Questo step, sebbene opzionale, è di fondamentale importanza per assicurare che non ci siano stati errori non visti durante il disegno, che potrebbe portare alla produzione di un c.s. sbagliato. L'analisi dei file gerber è lo step conclusivo del work-flow di KiCad; chiaramente alcuni step possono subire delle iterazioni, derivanti da errori che vengono corretti tornando allo step precedente, effettuando le modifiche necessarie e rigenerando i file di ingresso corretti per lo step successivo. Questo processo, denominato in genere *backward annotation*, permette di mantenere un'assoluta coerenza tra tutti gli step di progetto, in maniera che non si possano mai verificare situazioni di inconsistenza tra le varie fasi della progettazione, come ad esempio tra lo sviluppo dello schematico e lo sbroglio del PCB. Siamo quindi giunti al termine di questa puntata. Abbiamo introdotto l'ambiente, fornito una guida per il download e l'installazione e descritto l'interfaccia del Project Manager e il work-flow.

Dalla prossima puntata inizieremo ad occuparci della progettazione, introducendo l'uso di Eeschema su di un progetto pratico.

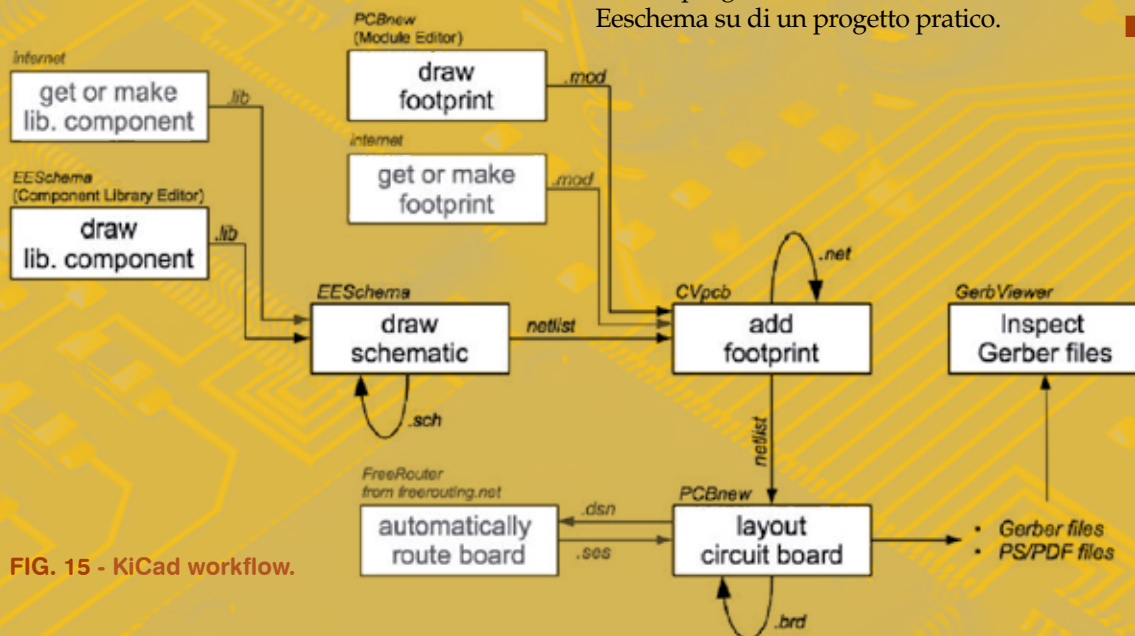


FIG. 15 - KiCad workflow.



di DAVIDE SCULLINO

Sistema di monitoraggio di parametri ambientali composto da sensori eterogenei collegati otto alla volta a concentratori, con una centralina che ne riceve i segnali e, in base alle impostazioni, attiva segnalazioni e allarmi locali. La disponibilità di 5 ingressi sulla centralina consente di rilevare anche condizioni di allarme da sensori antintrusione.

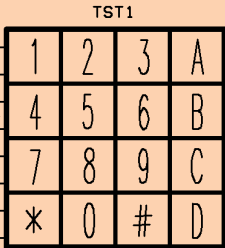
CENTRALE DI ALLARME AMBIENTALE

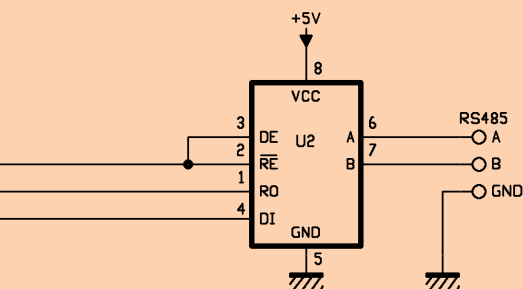
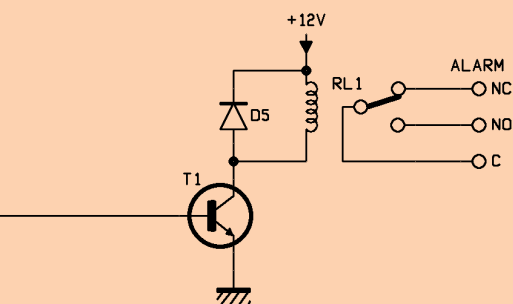
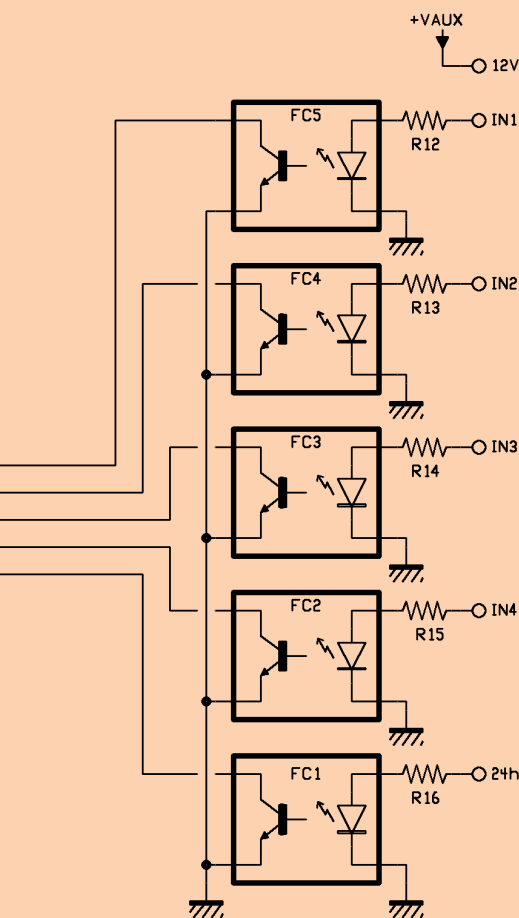
Gli odierni impianti di allarme sono sempre più sofisticati e vanno differenziandosi per applicazione: ad esempio, esistono, oltre alle più comuni centraline antifurto, centrali di Allarme Tecnologico: queste non nascono per proteggere i locali dalle intrusioni, ma piuttosto per monitorare parametri ambientali come ad esempio temperatura, presenza d'acqua e gas, illuminazione, distanza di corpi e oggetti ecc. e dare l'allarme se tali parametri escono da valori o finestre preimpostate.

La funzione di allarme tecnologico potrebbe, in verità, essere svolta anche da centrali antifurto particolarmente avanzate, sebbene un sistema dedicato sia la soluzione ottimale. Per questa ragione abbiamo pensato di progettare un sistema basato su una centralina specifica (che poi è una versione modificata di quella d'allarme antifurto wireless che vi abbiamo proposto nei fascicoli n° 172 e n° 173 di Elettronica In) e su un massimo di otto moduli periferici di interfacciamento ai sensori, che

dialogano con essa sfruttando il bus RS485; questo standard di connessione seriale consente il superamento di distanze relativamente grandi e la trasmissione sicura di dati anche in







ambienti relativamente disturbati. In queste pagine ci occuperemo della centralina, rimandando a un apposito articolo, nel prossimo numero di Elettronica In, la descrizione del modulo di interfacciamento dei sensori.

LA CENTRALE

Per spiegare come funziona il sistema dobbiamo innanzitutto vedere l'hardware della centrale d'allarme, meglio descritto dallo schema elettrico che trovate in queste pagine; come appena accennato, si tratta del circuito

di ingressi e azioni, cambiano anche le schermate del display. Abbiamo mantenuto, tuttavia, i cinque ingressi optoisolati a filo, che ci consentono di generare segnalazioni locali di allarme in caso di anomalia; ad essi è possibile collegare vari sensori o contatti puliti, per rilevare, oltre a condizioni ambientali, anche l'effrazione di porte o finestre o il verificarsi di eventi come l'incendio ed altro ancora.

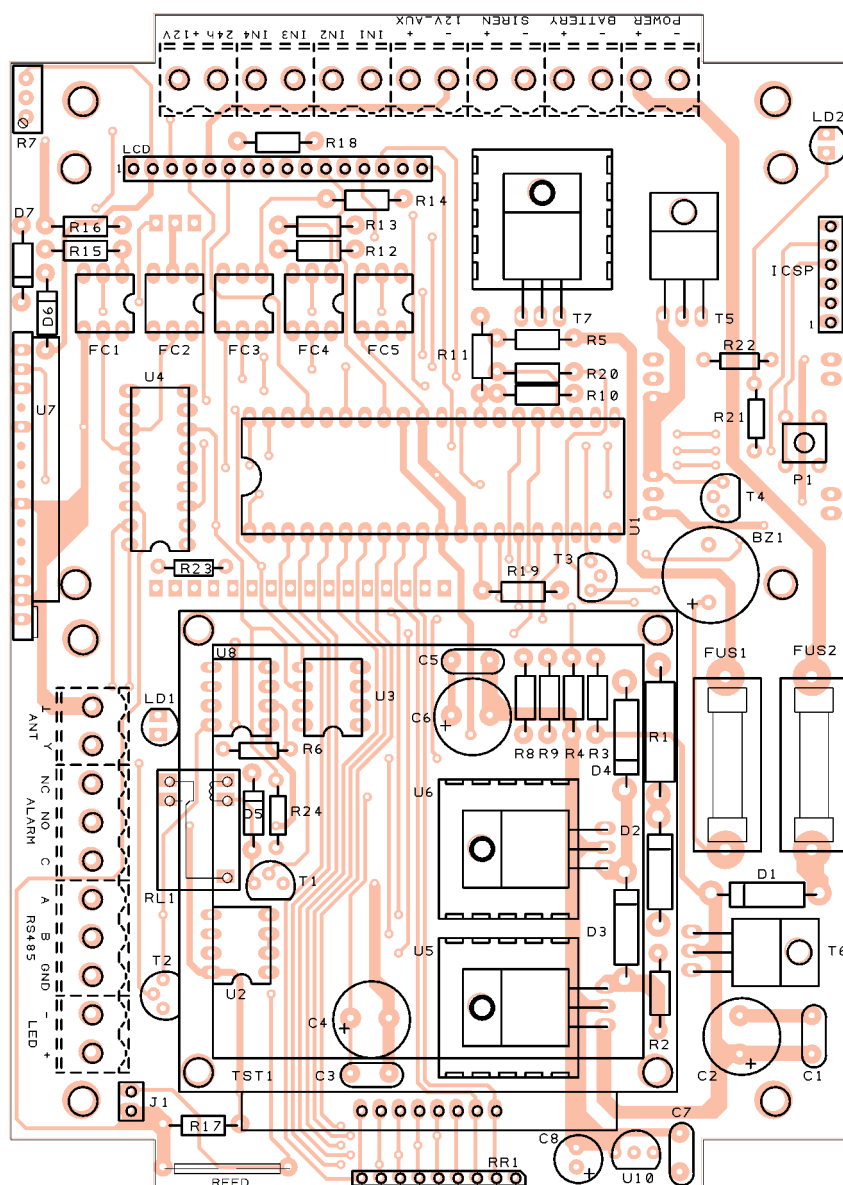
La centralina prevede varie azioni di output che possono essere distinte in allarmi, segnalazioni

CARATTERISTICHE TECNICHE

- Display 4 x 20
- Tastiera 16 tasti
- Pulsante di reset (P1)
- Bus 485 per accesso a 8 schede periferiche (+, GND, A, B)
- 5 linee d'ingresso cablate, di cui 4 n.a. e 1 n.c.
- LED di attività (LD2 verde, che lampeggia ad ogni interrogazione di periferica)
- Antimanomissione di centrale (tamper, J1 e relé Reed)
- Linea di uscita allarme (relé, C, NC, NA)
- Linea di uscita segnalazione (LED, + e -)
- Linea di allarme tamper di centrale o di periferica (+ - SIREN)
- Buzzer di segnalazione attività varie (cambio di schermata, fasi di test, pressione di un tasto, cicli di attesa e riavvio, reset e partenza da spento)

della centrale antifurto descritta nei fascicoli n° 172 e n° 173, nel cui microcontrollore è stato però caricato un nuovo firmware. Le funzioni non sono più le stesse, in quanto, per limitazioni nella memoria Flash del micro, non abbiamo potuto integrare l'allarme tecnologico nell'antifurto e abbiamo dovuto fare una scelta, eliminando il grosso del firmware che riguarda la gran parte delle funzioni di antifurto e inserendo quello per l'allarme tecnologico. Ne deriva che, oltre alla gestione

di stato o tamper; sono di allarme quelle segnalazioni che devono richiamare l'attenzione su personale di sicurezza o Vigili del Fuoco, perciò quelle causate da fumo, allagamento, gas, mancanza di corrente e che quindi attivano l'uscita dedicata alla sirena e le azioni corrispondenti (ad esempio la memorizzazione dello stato e la segnalazione mediante l'apposita uscita). Le cinque linee optoisolate generano sempre e solo un'azione di allarme, mentre i sensori collegati alle schede connesse alla cen-



Elenco Componenti:

- R1: 220 ohm 2W
- R2: 390 ohm
- R3, R4: 10 kohm
- R5: 6,8 kohm
- R6: -
- R7: Trimmer multigiri 10 kohm
- R8÷R10: 4,7 kohm
- R11: 10 kohm
- R12÷R17: 4,7 kohm
- R18: 82 ohm
- R19, R20: 4,7 kohm
- R21: 330 ohm
- R22: 10 kohm
- R23, R24: 4,7 kohm
- RR1: Rete resistiva 8+C 10 kohm
- C1, C3, C5, C7: 100 nF multistrato
- C2: 470 µF 35 VL elettrolitico
- C4: 470 µF 25 VL elettrolitico
- C6: 470 µF 16 VL elettrolitico
- C8: 100 µF 16 VL elettrolitico
- U1: PIC18F46K20-I/P (MF1015M)
- U2: MAX485
- U3: -
- U4: PIC16F88-I/P (MF1015I)
- U5: 7815
- U6: 7805
- U7: -
- U8: -
- U9: Modulo RTC (FT674M)
- U10: MCP1702-3302E/TO
- T1÷T4: BC547
- T5÷T7: BD137
- LD1: -
- LD2: LED 5 mm verde
- D1÷D4: BY254
- D5: 1N4007
- D6: -
- D7: -
- REED: Sensore reed NO (PPS24)

trale via RS485 possono generare allarme o segnalazione in base all'impostazione fatta dall'utente mediante le procedure (descritte più avanti) assistite dalle "schermate sensore".

Le linee di segnalazione sono quelle che non devono generare un allarme ma solo un promemoria di qualche tipo: ad esempio il sensore di distanza usato per rilevare la presenza dell'autovettura in garage, rilevata applicando al

soffitto un sensore ad ultrasuoni (ne parleremo nell'articolo dedicato all'interfaccia multisensore); in questo caso la segnalazione ricorda la presenza del veicolo. Infine, il tamper riguarda la segnalazione di manomissione sia della centralina, sia delle schede ad essa collegate via RS485.

IL CIRCUITO

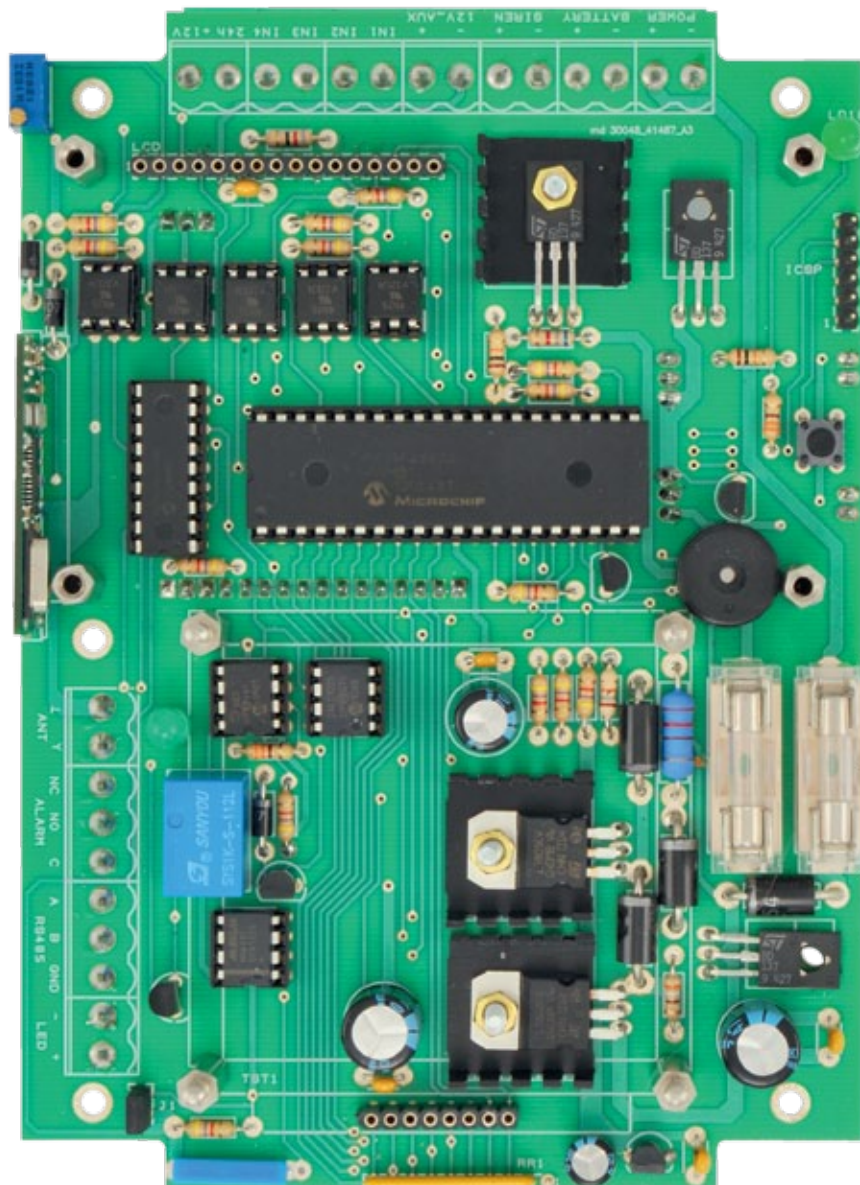
Il cuore del circuito è il PIC18F4620 (U1), che verifica la

pressione dei tasti della tastiera a matrice, aggiorna costantemente il display e, all'occorrenza, provvede ad attivare il LED di stato e l'eventuale uscita a relé. Inoltre, comunica costantemente con un secondo microcontrollore (U11) che monitora tutti gli ingressi optoisolati IN1÷IN4 (oltre a quello siglato 24h, ovvero IN5) cablati sulla centralina stessa; di questi ingressi, i primi quattro (IN1÷IN4) sono normalmente a ri-

P1: Microswitch
 BZ1: Buzzer 12V con elettronica
 FC1÷FC5: 4N25
 RL1: Relé 12V
 GSM: -
 LCD: Display LCD 20x4 (LCD20X4)
 F1: Fusibile 2A
 F2: Fusibile 1A
 TST1: Tastiera a matrice 4x4

Varie:

- Porta fusibile da CS (2 pz.)
- Morsetto componibile 2 poli verticale (9 pz.)
- Morsetto componibile 3 poli verticale (2 pz.)
- Zoccolo 3+3 (5 pz.)
- Zoccolo 4+4 (1 pz.)
- Zoccolo 9+9
- Zoccolo 20+20
- Dissipatore (3 pz.)
- Strip femmina 2 pin (2 pz.)
- Strip femmina 3 pin (2 pz.)
- Strip femmina 8 pin
- Strip femmina 16 pin (1 pz.)
- Strip maschio 2 pin
- Strip maschio lungo 8 pin
- Strip maschio lungo 16 pin
- Vite 8 mm 3 MA testa conica (22 pz.)
- Vite 12 mm 3 MA testa conica (3 pz.)
- Dado 3 MA (3 pz.)
- Torretta F/F 10 mm (1 pz.)
- Torretta F/F 12 mm (4 pz.)
- Torretta F/F 15 mm (4 pz.)
- Torretta M/F 20 mm (4 pz.)
- Jumper
- Magnete ad anello cod. MAGNETRING
- Contenitore plastico
- Circuito stampato

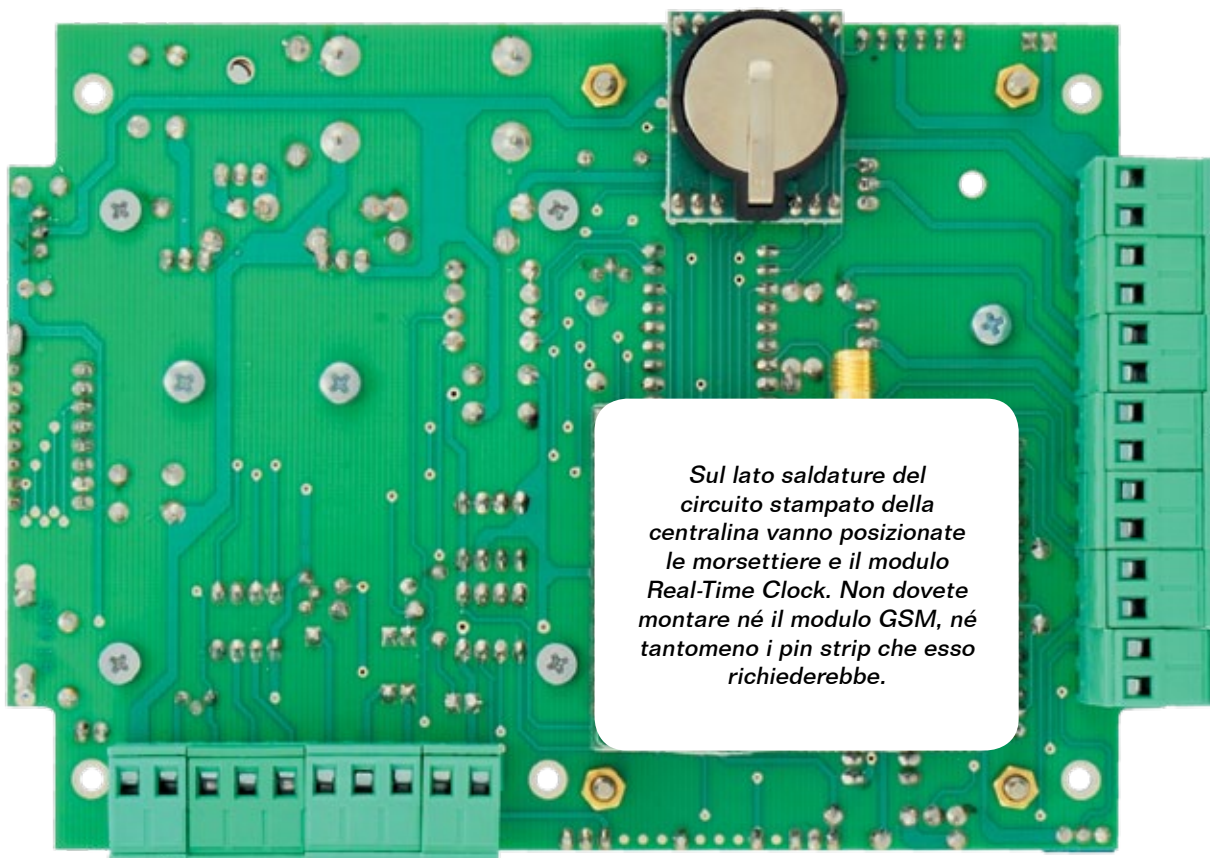
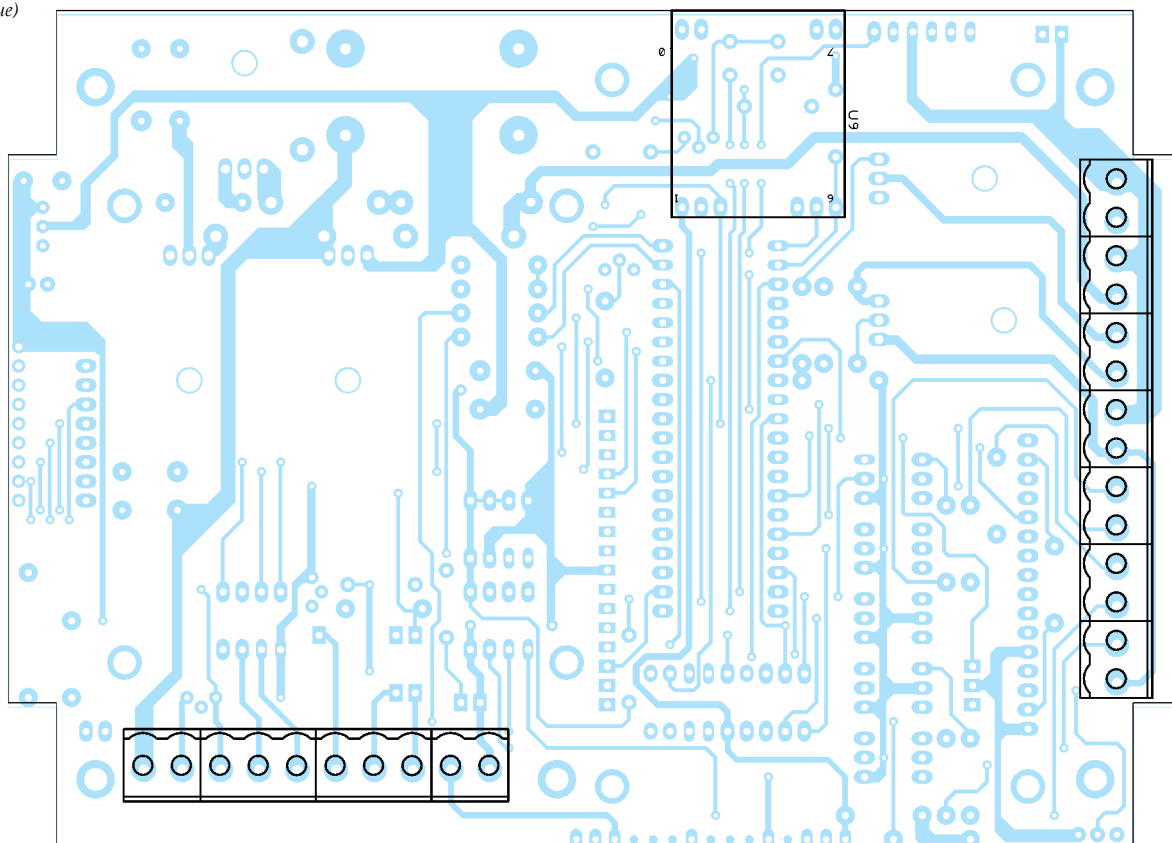


poso (cioè la condizione di quiete è quando non viene loro applicata tensione o la tensione presente non supera la minima che innesca l'optoisolatore) e il quinto (24h) è normalmente a riposo pure lui (cioè in condizioni di riposo deve essere alimentato con più di 4 volt). L'ingresso IN5 (24h) può essere impiegato per collegare alla centralina sensori di gas. Le linee di I/O destinate alla lettura della tastiera dispongono di un

pull-up esterno al micro, ottenuto con la rete resistiva RR1. La centralina impiega in tutto tre microcontrollori: oltre a quello principale (U1), abbiamo (U4), che gestisce interamente tutti gli ingressi, ed un terzo che è montato sul modulo RTC (U9) e provvede a ricavare e passare al micro principale la data e l'ora del sistema. Gli ingressi a filo hanno un'interfaccia a transistor e vengono

alimentati con il positivo +12V generale del circuito; ad essi possiamo collegare contatti normalmente chiusi o aperti di ogni genere, quindi qualsiasi tipo di sensore. Se si desidera usare solo alcuni ingressi, quelli inutilizzati vanno lasciati liberi (se sono IN1÷IN4) o collegati al +12V e la massa (24h). Il micro U1 gestisce il cicalino BZ1, utile esclusivamente per avere un riscontro della pressione di un tasto sulla tastiera. Lo stesso pro-

(Segue)



cessore comanda il relé RL1, che verrà attivato in caso di allarme e rimarrà attivo fino a quando la centralina non verrà portata in standby.

Per quanto concerne la sirena, è stato previsto un apposito morsetto SIREN sul quale è possibile collegare una sirena a 12 V; in caso di allarme, tramite il transistor T5 la morsettiera alimenterà e farà suonare la sirena. In questo caso si è pensato, per comodità, di utilizzare una comune sirena, ma è decisamente meglio prevedere di utilizzare una sirena autoalimentata, così da garantire al sistema una maggiore affidabilità in caso di interruzione dei collegamenti causata da un incidente o un guasto locale.

Il morsetto LED serve a comandare un diodo luminoso che ci indicherà, dal punto dove vorremo installarlo, se il sistema d'allarme è attivo oppure no; ad allarme attivo, il transistor T2 alimenterà il diodo facendolo illuminare. Evitate di collegare a tale morsettiera utilizzatori che assorbano più di 300 mA, visto che il transistor è un BC547 e al massimo può erogarne 500.

Dopo aver fatto una panoramica generale, soffermiamoci sul PIC dedicato agli ingressi: il PIC16F88 si occupa di monitorare i 5 ingressi di bordo, dotati di fotoaccoppiatore 4N25, e il tamper (anti-sabotaggio) che in questo caso è di tipo NC, quando il magnete è di fronte al contatto. Il jumper J1, posto in parallelo al sensore Reed, permette di forzare l'ingresso a chiuso, per gestire i casi in cui il sensore si è guastato o vada momentaneamente escluso; ricordate che il tamper protegge la centralina dall'apertura del contenitore quando è armata, quindi va, di norma, usato.

Per dialogare con il gruppo di sensori remoti la centralina

dispone di un'interfaccia RS485 implementata con il transceiver U2 e facente capo ai morsetti RS485; l'integrato è un MAX485 ed è interfacciato al microcontrollore principale mediante i propri piedini RO (uscita del ricevitore/traslatore RS485/TTL) e DI (ingresso del traslatore TTL/RS485). Le linee di controllo DE ed /RE servono per attivare distintamente trasmettitore e ricevitore: la prima è attiva a livello alto e la seconda a zero; con l'attuale configurazione possiamo gestirle tramite un'unica linea di I/O del microcontrollore, anche perché, visto che la comunicazione avviene in simplex (cioè alternando trasmissione e ricezione su un'unica coppia di fili) va più che bene che il trasmettitore e il ricevitore funzionino uno alla volta, prima uno e poi l'altro.

Concludiamo la descrizione dell'elettronica dicendo che il circuito è stato pensato per essere alimentato a 15 V, in quanto per caricare la batteria, che è al piombo da 12 volt (consigliamo una capacità di 800 mAh), ci occorre una tensione di almeno un paio di volt sopra questo valore.

Nel circuito, notate la presenza di due fusibili: il primo, F1, è inserito a protezione di tutto il sistema; in questo caso si tratta di un fusibile da 2A. Il secondo fusibile è F2, da 1A, ed è utilizzato per proteggere esclusivamente tutta la centralina da eventuali sovraccarichi esterni, dovuti ad esempio a dispositivi/sensori collegati al morsetto 12V_AUX.

REALIZZAZIONE PRATICA

La centralina va costruita procurandosi innanzitutto il circuito stampato a doppia faccia, ottenibile per fotoincisione partendo dalle tracce lato rame che trovate nel nostro sito Internet www.elettronica.in. Incisa e forata la ba-

setta, procedete seguendo il piano di montaggio illustrato in queste pagine; prima di cominciare, vi facciamo notare che quest'ultimo e la lista dei componenti corrispondente si riferiscono alla centralina antifurto da cui la nostra è derivata, perciò non dovete montare quegli elementi che nella lista dei componenti sono contrassegnati da un trattino (trovate - al posto del valore o sigla).

Iniziate quindi montando le resistenze tranne R1, poi procedete con gli zoccoli, facendo coincidere la tacca di ciascuno con la corrispondente in serigrafia (ricordate di non montare gli zoccoli per U3 e U8). Proseguite montando il pulsante P1 e tutti i condensatori ceramici/multistrato, quindi la rete resistiva RR1, il sensore Reed, i LED e il pin strip ICSP seguito da quello femmina a 16 poli per l'LCD e da quello ad 8 poli femmina per la tastiera TST1. Ora saldate i diodi D1, D2, D3, D4, D5, quindi procedete con i transistor; siccome T7 dovrà essere provvisto di dissipatore, piegatene a 90 gradi i terminali e appoggiatelo col lato metallico al dissipatore, poi stringete il tutto con una vite 3MA con apposito dado. Fate altrettanto con i regolatori di tensione 7812 e 7805.

Montate il buzzer BZ1 facendo attenzione alla polarità (il pin più lungo dovrà essere inserito in corrispondenza del "+" presente sul c.s.) poi passate al portafusibile e a tutti i condensatori rimanenti, prestando sempre attenzione alla polarità, quindi sistemate il relé. Completato il montaggio dal lato componenti, girate la basetta e montate il pin strip femmina a due poli per l'RTC; concludete inserendo tutti i morsetti facendo in modo che, una volta inseriti, la loro vite rimanga rivolta verso l'esterno del circuito stampato, così da renderli accessibili co-

modamente. Per completare la centralina abbiamo previsto l'uso del contenitore plastico Futura Elettronica G313; il montaggio al suo interno richiede il fissaggio sullo stampato di torrette distanziali da 18 mm sul lato componenti -necessarie per montare la tastiera- e quattro torrette da 15 mm per sostenere il display. Notate che per fissare la tastiera conviene utilizzare viti a testa svasata, allo scopo di ridurre l'ingombro: sotto lo stampato prenderà posto la batteria ed è pertanto importante ridurre al minimo gli spessori. Prendete ora il contenitore plastico e avvitate sull'interno della base quattro torrette distanziali da 30 mm; realizzate tutti i collegamenti del caso, quindi appoggiate lo stampato (con sopra tastiera e display) sulle quattro torrette, dopo bloccate con le viti e chiudete il contenitore, il cui coperchio dovrà essere stato prima forato per ricavare le finestre dalle quali far uscire, display, tastiera e LED di segnalazione.

COLLEGAMENTI

La centralina è dotata di due file di morsetti che permettono di ef-

fettuare tutti i collegamenti e che per comodità sono stati scelti tutti ad innesto, perciò ogni singola parte può essere scollegata dalla centralina anche semplicemente staccando la parte ad innesto del morsetto, risparmiandosi di dover svitare la vite. Al morsetto "PWR" bisogna collegare l'alimentazione principale della centralina, ovvero un alimentatore in grado di fornire 15 Vcc e 2 A. Il dato è puramente indicativo, dato che potrebbe bastare anche meno corrente: tutto dipende da ciò che la centralina dovrà a sua volta alimentare. Per la sola carica della batteria, sono sufficienti 100 mA; a ciò va aggiunto il consumo della sirena e di eventuali dispositivi esterni collegati al morsetto ausiliario "12V_AUX". Tenete comunque presente che, in ogni caso, l'assorbimento complessivo dall'alimentazione non può superare i 2 A. Tipicamente, considerando i consumi medi, si può optare per utilizzare un alimentatore in grado di fornire almeno 1 A. Al morsetto "BATTERY" andrà collegata - rispettando la polarità - una batteria al piombo da 12V/800 mAh che, tra l'altro, può essere alloggiata nel

contenitore stesso della centralina. Ai contatti "SIREN" collegate una sirena a 12V o qualsiasi altro segnalatore ottico/acustico desiderato, senza scordare una certa attenzione al consumo di corrente: il transistor che la pilota è un BD137 e non può sopportare più di 1,5 A; considerando che per questo componente non abbiamo previsto un dissipatore, è meglio evitare di collegare sirene che richiedano più di 700 mA. Al morsetto "12V_AUX", come anticipato in precedenza, è presente una tensione di circa 12 volt con la quale è possibile alimentare dispositivi ausiliari o sensori; la corrente erogata non può eccedere 1,4 A perché ad erogarla c'è ancora un BD137 (è il T7), ma questa volta è provvisto di dissipatore e possiamo quindi permetterci di prelevare con continuità gli 1,4 ampere. L'uscita "12V_AUX" è inoltre protetta da un fusibile (F2). Per terminare, sono stati previsti anche altri cinque morsetti facenti capo agli ingressi di allarme a filo IN1, IN2, IN3 e IN4 e 24h. Il terminale +12V è un'uscita a 12 volt protetta anch'essa dal fusibile F2; il +12V può essere utilizzato in combinazione con i cinque ingressi, dato che ogni LED del fotoaccoppiatore è già connesso da un lato a massa, mentre l'anodo è collegato attraverso una resistenza al terminale a filo dell'ingresso. Se il sensore cablatto non fornisce tensione ma ha in uscita un contatto pulito, basta collegare il +12V a un capo del contatto stesso e il capo libero all'ingresso (IN1, IN2 ecc.); lo stesso dicasi se il sensore ha un'uscita a transistor: in tal caso bisogna sapere se l'uscita è NPN o PNP e se è open-collector oppure a inseguitore di emettitore. La open-collector può essere usata se basata su transistor PNP: in questo caso si collega il +12V all'emettitore e il collettore

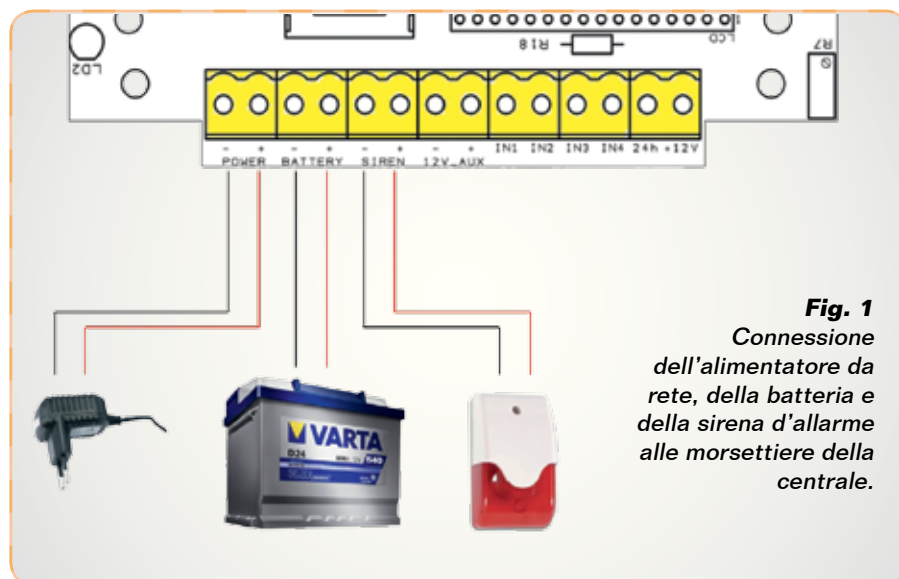


Fig. 1
Connessione dell'alimentatore da rete, della batteria e della sirena d'allarme alle morsettiere della centrale.

all'ingresso a filo del circuito. Il discorso funziona se il transistor è polarizzato in base con 12 volt, mentre se la tensione è minore è meglio evitare.

In generale, l'uso di uscite a transistor complica un po' le connessioni perché richiede la conoscenza di molti parametri, mentre il contatto pulito si usa senza alcuna complicazione.

Il connettore "ALARM" fa capo a un relé utilizzabile per attivare un sistema di vario genere quando la centrale rileva un evento che determina un allarme e questo evento è in corso: ad esempio una spia che informi visivamente dell'avvenuto allarme; bisogna però tenere presente che a differenza della sirena -che suonerà per un tempo programmabile- il relé rimarrà eccitato anche ad allarme terminato. Attraverso i contatti del relé si può gestire un carico da 12 V che assorba al massimo 1 A. Infine, con il morsetto "LED" è possibile collegare un LED al +12V per sapere, quando si è all'esterno dei locali in cui è installata la centrale, se questa è attivata oppure no.

Adiacente a quest'ultimo morsetto è presente un jumper usato per la gestione del sensore anti manomissione: il sensore è un Reed di tipo NC e il jumper (J1, visibile in Fig. 2) serve per bypassarlo nel caso, ad esempio, non si voglia utilizzarlo, si debba effettuare un intervento di manutenzione a scatola aperta o, ancora, se il Reed si guasta e nell'attesa di poterlo cambiare si vogliono evitare false segnalazioni di allarme. Nessuno vieta, comunque, di utilizzare un sensore differente dal Reed (magari fuori scheda), fermo restando che deve essere di tipo normalmente chiuso.

DISPLAY E COMANDI: LA TASTIERA E IL DISPLAY

La centralina si interfaccia con l'utente mediante la tastiera e

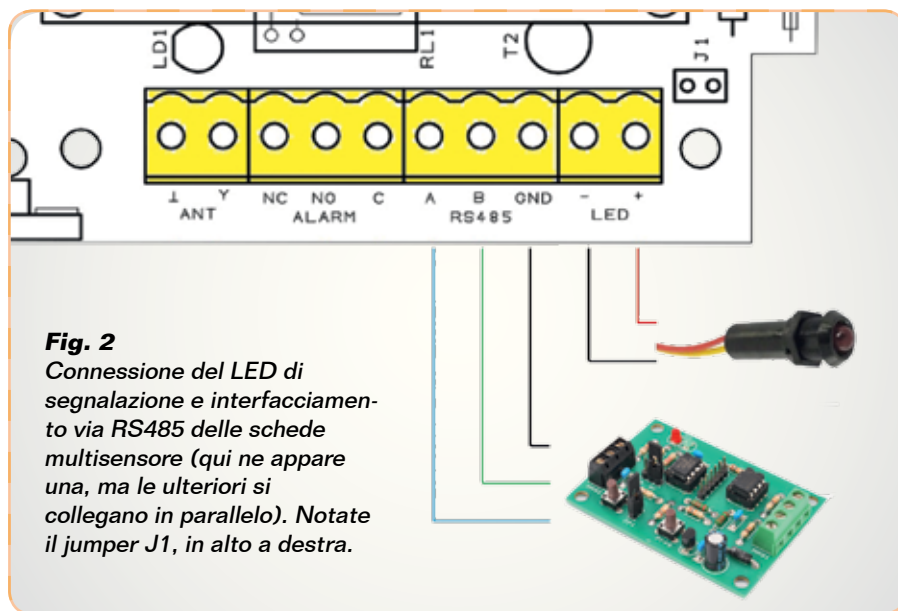


Fig. 2
Connessione del LED di segnalazione e interfacciamento via RS485 delle schede multisensore (qui ne appare una, ma le ulteriori si collegano in parallelo). Notate il jumper J1, in alto a destra.

il display. Nella tastiera ci sono 16 tasti; i numeri servono per richiamare le schermate riguardanti l'attività delle varie schede di interfaccia (1÷8) con i sensori collegati via RS485, oppure per immettere dei valori numerici durante la configurazione. I tasti A, B, C gestiscono le varie funzioni. Notate che il polling alle periferiche è lento e richiede circa 0,6 secondi a periferica più un secondo di sincronismo, pertanto quando il sistema è completo (PIC_IO, RTC, 8 schede multisensore) il tempo necessario alla scansione per la lettura dei sensori, l'elaborazione, la visualizzazione e l'eventuale segnalazione di allarme a schermo e sulle uscite, quindi l'interrogazione della tastiera è di circa 8 secondi. Questo significa che quando si preme un tasto da 1 a 9 per entrare in qualche schermata di configurazione, oppure B o C, la lettura della tastiera avviene sempre e solo prima dell'inizio della scansione delle periferiche. Quindi bisogna tenere premuto il tasto di interesse fino ad avvenuta reazione della centralina, il che significa che nella peggiore delle ipotesi (tutte le schede periferiche presenti) si potrebbero dover

attendere fino a 8 secondi. In ogni caso, se si sa che la scheda 5 (ad esempio) è l'ultima di quelle collegate sul bus, premendo il tasto di interesse quando si vede Lett:5 a display nella schermata 0 significa che in teoria tra un secondo il sistema leggerà il tasto, essendo arrivato a fine scansione.

Passiamo adesso al display, che ci permette di introdurre il discorso sulle funzioni della centralina.

Il display può fornire diverse informazioni, ovvero permette di conoscere lo stato della centralina quando si trova in standby, se è stato rilevato un allarme, se è stata avviata la procedura di allarme, oppure, nel caso del menu, indica in quale e a quale livello ci si trova.

La schermata generale è la 0 e viene mostrata all'avvio o al timeout di una qualsiasi delle altre schermate che vedremo; può anche essere richiamata tenendo premuto 0 fino a che l'unità non risponde mostrandola. Essa si presenta così:

```
Gio. 09/01/14 22:04$
Sens 12345678 Lett:5
Tipo -$--$--- U:AST$
All. S A F12345
```

Dove c'è "\$" significa che al suo posto viene stampato un carattere speciale, cioè non uno di quelli del set predefinito del display, ma costruito apposta dal firmware della centrale per generare un simbolo di facile e intuitiva comprensione.

La prima riga contiene data e ora. L'ultimo carattere, se presente, è una doppia freccia che indica l'esistenza del link con centrale attiva.

La seconda riga indica le periferiche e l'operazione in corso (sincronizzazione, lettura scheda 0, 1... invio ad antifurto ecc.). La terza riga indica che tipo di sensore è associato alla scheda il cui numero è riportato in fondo alla seconda riga; sono disponibili quattro caratteri speciali: frecce di quota (freccia a doppia punta) per la distanza, potenziometro (contatto o potenziometro), termometro e fiammella (sensore gas). Seguono U(scita): A(llarme) S(egnalazione)T (amper) e un carattere speciale che ricorda una batteria (quando la centrale sta funzionando a batteria a causa della mancanza della tensione di rete).

La quarta riga mostra il tipo di allarme fornito dalla scheda corrispondente: nell'esempio, sono presenti solo la scheda 2 e la 5; la 2 è in segnalazione, quindi attiva l'uscita S, la 5 è in allarme, quindi attiva l'uscita allarme.

L'esempio mostra inoltre il rilevamento di una manomissione da qualche parte, più la mancanza della tensione di rete). Seguono poi F e i numeri da 1 a 5 ad indicare quale linea cablata (cioè quale ingresso on board della centralina) è in allarme. Ricordate che le linee a filo sono sempre prioritarie sui sensori collegati via RS485 e generano sempre e solo un allarme.

Le schermate di sensore (da 1 a 8)

si richiamano premendo il tasto corrispondente; per schermata di sensore si intende quella che riguarda la condizione e l'attività delle schede d'interfaccia collegate via RS485. Il loro aspetto, se la scheda corrispondente è presente, è il seguente:

```
ID.2: Dist$ T:All A!  
#=imposta *=OK AB=+-  
C=<> D=AS Val: xxxxx  
All:*> di (xxx%) xxxxx
```

Se la scheda manca, il display mostra per 2 secondi l'indicazione "Periferica non presente", poi torna a mostrare la schermata 0. Nella prima riga si trovano, nell'ordine, l'identificativo numerico di periferica (calcolato, non imposto), il tipo di sensore (letto, testo e simbolo grafico), stato del tamper di quella scheda (All o Ok) A o S per impostazione allarme o segnalazione, ! per indicare che la scheda è nella situazione che deve segnalare.

Nella seconda riga appaiono le istruzioni, ossia il display segnala quali comandi impartire per gestire la scheda: # entra in programmazione, * salva e fa abbandonare la programmazione, mentre A e B servono rispettivamente a incrementare e decrementare la soglia di allarme.

Le istruzioni proseguono nella terza riga, la quale nella parte a destra (Val:) riporta il valore letto dalla scheda. Per quanto riguarda le istruzioni, il tasto C fa commutare tra < (si ritiene allarme la discesa sotto il valore di soglia) e > (allarme per valori maggiori di quello di soglia). Il tasto D determina se la scheda debba generare un Allarme o una Segnalazione. Val: il valore fornito dalla scheda periferica. Il formato dipende dal tipo di sensore.

Nella quarta riga il display

mostra informazioni relative all'impostazione della soglia: * se presente indica che si è in programmazione (si è premuto prima #) e qui non c'è timeout. Da qui si esce solamente confermando con *. Segue il simbolo >/< (allarme se minore o maggiore di); segue tra parentesi un valore % compreso tra 2 e 98, che rappresenta la soglia di allarme percentuale del valore letto immediatamente dopo. I livelli di intervento corrispondono alla soglia $\pm 1\%$, quindi se impostiamo, ad esempio, che l'allarme debba avvenire con un valore letto da scheda < del 50%, significa che l'allarme viene rilevato al 49% (o al disotto di questo valore) e non viene rilevato al 51% o più. Nelle schermate da 1 a 8 il link verso la centrale antifurto viene disabilitato, anche se quando si torna alla schermata 0 rimane attivo, qualora lo fosse stato prima di entrare in 1÷9. Esiste poi la schermata 9 (richiamata con il tasto 9), riguardante l'impostazione di data e ora di sistema:

```
Gio. 09/01/14 22:04  
^  
#=imposta *=salva $  
AB=+- CD=<>
```

Nella prima riga appaiono data e ora che sono le stesse della schermata 0; non appare mai, come ultimo carattere, il simbolo del link.

Nella seconda riga si trova il puntatore, il quale, se la centralina si trova in impostazione, indica quale campo si varia agendo su + e -.

La terza e la quarta riga riportano le istruzioni: # per entrare in impostazioni, * per salvare (Appare Salvataggio seguito da Terminato), A e B incrementa o decremen-

ta, C e D per spostarsi a destra o sinistra (rotazione continua in entrambi i versi). Nel firmware è implementato un rudimentale controllo di errore per i dati immessi nella data, ma non è in grado di correggere tutto, quindi fate comunque attenzione (ad esempio non verifica la coerenza tra il giorno e la quantità di giorni nel mese).

COMANDI SPECIALI

La tastiera permette di operare comandi diretti dalla schermata corrispondente al normale funzionamento; questi comandi sono A, B e C. La pressione del tasto B blocca il ciclo di scansione delle schede multisensore: la centralina si ferma e non interroga più le schede perché ritiene si stia facendo qualche intervento sulle periferiche. Il display mostra una barra tipo "supercar" che scorre a destra e sinistra. Durante il blocco, un PC affacciato sul bus RS485 tramite un apposito convertitore RS232 o USB potrebbe interrogare le varie schede (tramite Hyperterminal o simile) a scopo diagnostico e a patto che una alla volta possa rispondere (una accesa e le altre spente). Questa interrogazione verrà spiegata nell'articolo dedicato alle schede multisensore, che verrà pubblicato nel prossimo numero di Elettronica In.

Il tasto A riavvia il ciclo normale quando esso è stato interrotto dalla pressione di B.

Il tasto C, ad ogni pressione attiva/disattiva l'emissione dei dati indirizzati a un'eventuale centrale antifurto interfacciata con essa via RS485.

Se il link non è attivo, la centralina invia i caratteri 0 (sincronismo), pausa 500 ms, N(interrogazione prima scheda presente), pausa 500 ms,..... N(interrogazione ultima scheda), pausa 500 ms. Se il link è attivo, la centralina

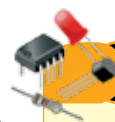
invia 0 (sincronismo), pausa 500 ms, N(interrogazione prima scheda presente), pausa 500 ms, N(interrogazione ultima scheda presente ...), P, pausa 50 ms, P,"dati per centrale...) pausa 1.100 ms.

Il pulsante di reset è un ulteriore comando speciale: se viene premuto all'accensione, forza un reset di prima accensione con ripartenza da zero e cancellazione EEPROM. Nel ciclo normale funziona da riavvio del sistema. Alla partenza da zero (sia dopo un reset che dopo la prima programmazione in assoluto con ICSP) con accensione, reset o altro, o comunque quando il programma riparte dal principio, la scheda emette un avviso acustico mediante il cicalino ed esegue in sequenza la serie di operazioni preliminari qui descritte.

- 1) Distingue tra primo avvio, in cui attende la pressione di un tasto per dare il tempo di configurare le varie periferiche, e avvii successivi, in cui esegue il suo ciclo normale. Se si tratta di un primo avvio, immediatamente dopo la pressione del tasto invia due volte R alle periferiche per forzare in esse il riavvio.
- 2) Controlla il pulsante di reset, che se premuto fino a conferma da parte della centralina, causa il reset della EEPROM e il riavvio come primo avvio.
- 3) Rileva in autotest la presenza delle linee a filo (PIC_IO, software nuovo, che legge due comandi: T per test e < per lettura linee; linee da 1 a 4 NA, linea 24H NC, tamper centralina a riposo se pulsante premuto). Se si rileva un errore di PIC I/O, viene segnalato sull'LCD e si inibisce la lettura delle linee a filo.
- 4) Rileva in autotest l'RTC.
- 5) Rileva in autotest indirizzo

e tipo sensore delle schede periferiche presenti. La centrale interrogherà solamente quelle rilevate. Ogni cambio di informazione è accompagnato da un messaggio chiaro a display e da un tono acustico breve di attenzione emesso dal BZ1.

In ogni schermata del sistema esistono delle parti statiche (comuni a tutte le schermate o specifiche di ciascuna) e delle parti dinamiche. Vengono scritte a display solamente le variazioni, poiché il software controlla che talune parti comuni siano già state aggiornate oppure no. Tipicamente il firmware controlla un flag: se è a zero modifica la schermata e lo setta a 1 per non modificare nuovamente la schermata. Il flag viene azzerato solamente all'uscita da quella schermata, ma chiaramente si avvia un ciclo identico per la nuova schermata in cui si è entrati.



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono facilmente reperibili in commercio. Il master del circuito stampato ed il firmware col quale programmare i microcontrollori sono scaricabili gratuitamente dal sito della rivista. Il modulo RTC cod. FT674M costa 18,00 Euro ed è reperibile presso Futura Elettronica così come il display LCD 20x4 (cod. 1446-LCD20X4, Euro 26,00) e la tastiera a 16 tasti (cod. 5150-TST16, Euro 6,90). 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>

PER UNA GIUSTA CONNESSIONE

La connettività wireless sta diventando rapidamente una funzione 'must' in tutti i settori, offrendo le più svariate finalità, dal rimpiazzo di cavi alla connettività Internet; persino convertendo oggetti d'uso comune in web-server con funzionalità complete.

di FRANCESCO DONI

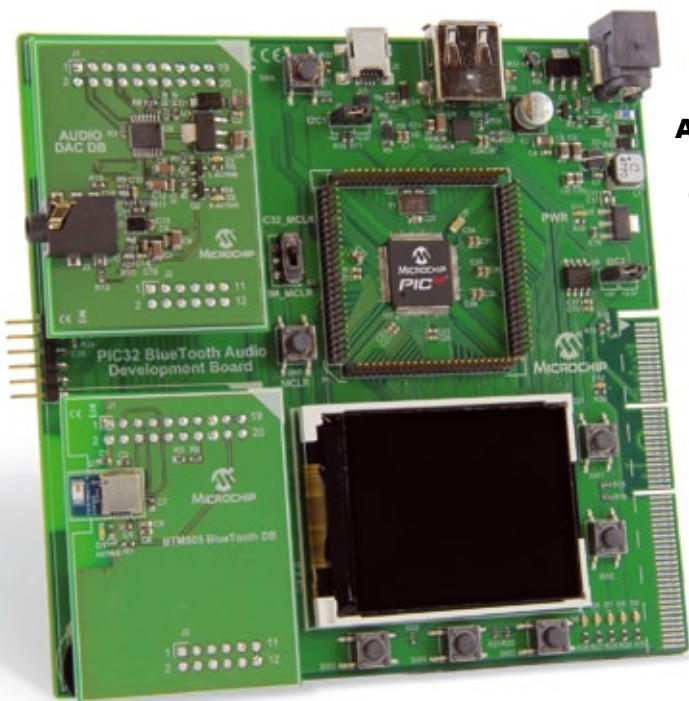
La connettività wireless è realmente onnipresente ed è dovuta, e non in piccola parte, ai continui sforzi di gruppi di lavoro e associazioni industriali, impegnate nello sviluppo di standard.

Attraverso l'adozione di tali standard, i produttori di semiconduttori possono ipotizzare situazioni di business credibili per investire nello sviluppo di soluzioni integrate; la disponibilità di queste soluzioni in un certo senso semina il mercato e, se la tecnologia è quella giusta - come è il caso per molti aspetti della connettività wireless - anche altri produttori molto presto seguono, creando un mercato competitivo e spingendo verso il basso il prezzo per singolo prodotto. Questo ovviamente tende ad aumentare la domanda e in qualche caso il risultato è ciò che ora viene definito 'mega-trend'.

Il mega-trend per la

connettività wireless non mostra segni di cedimento, infatti sta diventando sempre più forte; l'*Internet of Things (IoT)* verrà fondamentalmente mosso da soluzioni wireless *low power* e analisti del settore prevedono che ci saranno oltre 30 miliardi di nuovi dispositivi connessi all'IoT entro il 2020. La stragrande maggioranza di questi dispositivi utilizzerà sistemi wireless a basso consumo, condizione fondamentale per rendere possibile la connettività e per soddisfare la tipologia di utilizzo dell'IoT; associati ai sistemi ad intelligenza distribuita contribuiranno a rendere più semplice la vita. La tecnologia wireless a basso consumo, basata su standard consolidati, è in forte crescita; Bluetooth, ZigBee (nei suoi molti profili) e WiFi sono ora normalmente integrati - spesso in combinazione - in sistemi embedded che, persino recentemente, potevano





PIC32 Bluetooth Audio Development Kit (Part #DV320032)

Fig. 1

anche non offrire, del tutto, alcun tipo di connettività.

BLUETOOTH

Probabilmente, la tecnologia wireless più presente oggi in dispositivi consumer è il Bluetooth; un solido protocollo a corto raggio di trasmissione che offre bande passanti relativamente elevate per dati impegnativi. Una delle tipologie più impegnative è l'audio, che è eccezionalmente sensibile nella qualità a causa della capacità delle cuffie di rilevare persino piccolissime fluttuazioni e variazioni. Pertanto, l'utilizzo di Bluetooth in applicazioni audio, come dispositivi a mani libere e più di recente anche streaming audio, è dovuto proprio ai suoi punti di forza intrinseci.

La richiesta di riproduzione audio da Bluetooth sta crescendo, insieme al desiderio di fornire funzionalità *value-added* come elaborazione di segnali digitali e supporto per codec multipli. Nei dispositivi consumer, è anche importan-

te offrire interfacce utente coinvolgenti e da questo punto di vista Microchip ha recentemente presentato un *Bluetooth Audio Development Kit* basato sul suo MCU PIC32 (Fig. 1).

WIFI

Nel prendere in esame lo sviluppo di dispositivi connessi a Internet, una tecnologia wireless sta davanti a tutte: il WiFi. Aggiungere connettività WiFi può dare la possibilità praticamente a qualsiasi dispositivo di diventare parte di un più ampio network, ampliando le proprie capacità ben oltre quelle di un dispositivo standalone. Produttori di ogni settore stanno ora aggiungendo connettività WiFi per svariati motivi: accesso remoto, diagnostica e controllo logistico sono tutti esempi del perché il WiFi viene usato per portare i benefici di Internet ad oggetti d'uso quotidiano.

Aggiungere WiFi sta anche divenendo via via più semplice, grazie ai dispositivi altamente integrati e a *proto-*

col stack "pronti all'uso" che supportano svariate modalità di networking, funzionalità di sicurezza, e servizi Internet quali FTP, SMTP e HTTP. È ora possibile integrare un completo web server in un singolo, potente microcontroller, per esempio, consentendo pressoché a qualsiasi dispositivo, connesso in rete, di fornire dati ovunque nel mondo.

ZigBee vs PROPRIETARI

Per connettività a corto raggio o a larghezza di banda ridotta, una singola tecnologia sottostà alla crescente varietà di soluzioni standard e personalizzate.

L'interfaccia fisica IEEE 802.15.4, e il suo MAC standardizzato (attraverso ZigBee Alliance), usa la stessa tecnologia radio delle altre soluzioni 2,4 GHz a bassa potenza, che sono libere di utilizzare una vasta gamma di protocolli proprietari e *bit-rate*. Questa conosciuta *license-free* bandwidth supporta diverse tecnologie wireless, rendendo importante la scelta di un protocollo in grado di coesistere in uno spettro affollato, che siano *standard-based* o proprietari. Microchip offre diversi *protocol stack* in questo spazio, compresi ZigBee PRO, RF4CE e il proprio MiWi wireless mesh protocol. Oggi offre anche l'MRF24XA, un transceiver radio single-chip che per la prima volta supporta tutti questi protocolli, ad un *data-rate* compreso tra 125 kbit/s e 2 Mbit/s

SOLUZIONI MODULARI

Mentre operare nello spettro *license-free* rende la commercializzazione di tecnologie wireless più semplice, è ancora presente l'esigenza di certifi-

care le radio. Una sempre più diffusa e vitale soluzione per semplificare questo stadio della progettazione è quella di usare moduli preconfigurati.

I moduli hanno il vantaggio di essere "conformi per progettazione" ed in quanto tali rendono la certificazione molto meno difficoltosa. Microchip ha recentemente annunciato una gamma di nuovi moduli wireless mirati per Bluetooth, WiFi, ZigBee e connettività proprietaria.

Questa include la gamma di moduli MRF WiFi progettati per operare a stretto contatto di MCU che eseguano TCP/IP stack, come pure la serie RN XV di moduli WiFi e Bluetooth che integrano il TCP/IP stack sul modulo.

La Fig. 2 mostra uno schema a blocchi dei moduli MRF24WG0MA/MB, che integrano una radio IEEE 802.11 b/g in grado di supportare data-rate fino a 54 Mbit/s e al contempo raggiungendo un consumo di potenza di soli 100µA in modalità ibernazione. Il TCP/IP stack, con funzionalità ampie e complete, viene fornito come codice sorgente per gli sviluppatori, mentre *demonstration*

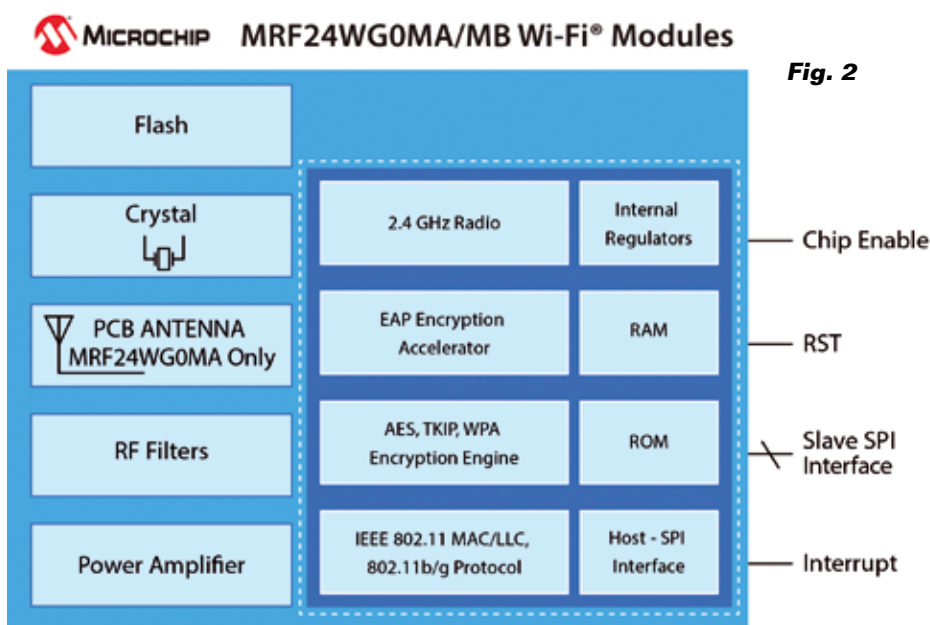


Fig. 2

board sono disponibili per gli ingegneri hardware. Microchip sta anche offrendo agli sviluppatori un percorso di migrazione tra WiFi e Bluetooth, rendendo disponibili una gamma di moduli compatibili con lo standard industriale *XBee footprint*. La presenza di entrambi i moduli plug-in WiFi e Bluetooth *through-hole* rende persino più semplice aggiungere connettività wireless a progetti sia nuovi che preesistenti.

L'Internet of Things verrà spinto da connettività wireless *low-power*, con miliardi di dispositivi che si collegano tra loro in un *network ad-hoc, peer-to-peer*, o che formino una rete *personal-, home-, metro- e wide-area-*. Questa perfetta aggregazione e distribuzione di dati potrà contare su semplici e robuste connessioni wireless mosse da un crescente numero di soluzioni modulari altamente integrate e a basso consumo di potenza. ■

RM ELETTRONICA

Forniture per hobbisti, scuole e industria

Distributore autorizzato:

**FUTURA
ELETTRONICA**

**ELSE
KIT**

ELETTRONICA

Il tuo punto
di riferimento
per l'elettronica
a ROMA

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



C'E' QUALCUNO ALLA PORTA? ME LO DICE LO SMARTPHONE

degli Ingg. TOMMASO E ALESSANDRO GIUSTO

Collegandone gli ingressi al campanello o al citofono, quando qualcuno suona alla porta comanda via Bluetooth uno smartphone e ne attiva la vibrazione.

Ideale per persone anziane o con problemi di udito, dispone di ingressi aggiuntivi per l'integrazione con sistemi di allarme.

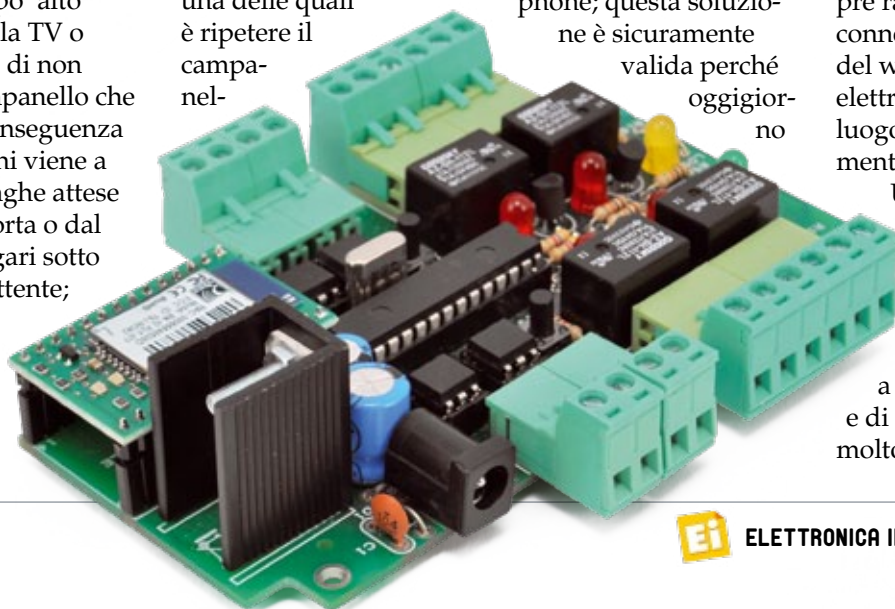
Sarà certamente capitato a tutti noi, almeno una volta, di tenere un po' alto il volume della TV o dello stereo e di non sentire il campanello che suona e di conseguenza costringere chi viene a trovarci a lunghe attese fuori dalla porta o dal cancello, magari sotto la pioggia battente; per evitare situazioni del genere e gli inevitabili accidenti

scagliatici contro dai malcapitati di turno, esistono varie soluzioni, una delle quali è ripetere il campanello

lo o il pulsante del citofono sotto forma di vibrazione dello smartphone; questa soluzione è sicuramente valida perché oggi giorno

molti di noi tengono il cellulare in tasca anche in casa per essere sempre raggiungibili o per connettersi al mondo del web e alla posta elettronica in qualsiasi luogo e in ogni momento.

Una soluzione del genere è proposta in queste pagine e può indubbiamente essere utile anche a chi soffre d'udito e di conseguenza tiene molto alto il volume



del televisore o, semplicemente, non sempre avverte il campanello, il telefono ecc.

Il progetto nasce sfruttando come hardware un circuito già pubblicato nel fascicolo n° 180, ossia l'Interfaccia I/O 4 canali Bluetooth basata su Android e come firmware una rielaborazione di quello originale di tale scheda. Ricordiamo che questo sistema consta di una scheda base dotata di quattro relé ed altrettanti ingressi a livello di tensione, alla quale è abbinata la connettività Bluetooth ottenuta mediante un modulo Roving Network RN-42 montato a bordo. Proprio la disponibilità di quattro ingressi, permette di utilizzare la scheda per rilevare non solo la pressione del pulsante del campanello, ma anche l'attivazione di allarmi antincendio o di altri rilevatori (per esempio campanelli dotati di sensore P.I.R. come quelli montati all'ingresso dei negozi per rilevare l'entrata delle persone); in questo modo, sullo smartphone si verrebbe avvertiti di varie condizioni e non semplicemente del suono del campanello. Questa condizione è provvidenziale, ad esempio, per chi ha gravi problemi di udito e fatica a sentire le segnalazioni acustiche: convertendo queste ultime in vibrazione del telefonino, si risolve il problema. Tramite modulo Bluetooth RN-42, la MCU della scheda è in comunicazione costante con lo smartphone Android sul quale gira una App realizzata ad hoc; nel caso che un ingresso si porti al livello rilevato come allarme, segnala la condizione allo smartphone, il quale tramite vibrazione avvisa l'utente.

COME FUNZIONA

L'elettronica utilizzata, oltre ai quattro input digitali, dispone anche di 4 uscite a relé e 2 pul-

santi che, per il nostro progetto, abbiamo pensato di utilizzare nel modo seguente: visto che per un funzionamento ottimale i contatti dei pulsanti di citofono e campanello devono essere sconnessi dal loro circuito originario e collegati a due input della nostra scheda, abbiamo cercato un modo per fargli comandare comunque gli avvisatori acustici che dovrebbero attivare. Questo modo consiste nel far ripetere ai relé RL1 e RL2, grazie a un'apposita routine firmware, la pressione del pulsante di chiamata del citofono (IN1) e quello del campanello (IN2); in sostanza, quando IN1 viene attivato, il firmware attiva RL1 affinché ne ripeta la condizione (cioè a fotoaccoppiatore in conduzione corrisponde l'attivazione del relé) e quando è attivo IN2, il firmware fa scattare RL2 per lo stesso intervallo di attivazione di tale ingresso. Entrambi i relé ripetono, dunque, la condizione degli ingressi corrispondenti; ciò permette di usarne gli scambi per comandare il campanello (RL2) e la suoneria del citofono di casa (RL1). Riguardo al campanello va fatta una precisazione: se è di quelli alimentati direttamente dalla rete, occorre un ulteriore relé la cui bobina va comandata mediante lo scambio di RL2; ciò perché i relé montati nella nostra scheda sono costruiti per operare al massimo a 120 Vca o 24 Vcc. Se invece il vostro campanello è da 24 Vcc (dotato quindi di trasformatore riduttore) basta il relé su scheda.

Passiamo oltre dicendo che i due relé rimanenti e i pulsanti sulla scheda permettono di replicare funzioni come l'apriporta (nel caso la porta di casa o del negozio avesse un'elettroserratura) e l'apricancello elettrico o l'azionamento dell'elettroserratura del portone o cancelletto; allo scopo,

abbiamo previsto di abbinare il pulsante P2 al relé RL3 e il P1 al relé RL4; in questo modo. Collegando ai contatti di P1 i fili del pulsante dell'apriporta e a quelli di P2 i fili in arrivo dal pulsante apricancello, basta poi alimentare le elettroserrature corrispondenti tramite i relé, per ottenere le stesse funzioni (i relé vanno utilizzati come interruttori inserendone gli scambi in serie ai fili che vanno alle elettroserrature).

Il nostro sistema permette anche l'apertura di cancelli e porte provviste di elettroserratura tramite un'apposita App Android, cosicché, ad esempio, una volta ricevuto l'avviso che hanno suonato al citofono sarà possibile aprire.

SCHEMA ELETTRICO

L'hardware del sistema è meglio descritto dallo schema elettrico riportato nella pagina accanto; il tutto è centrato intorno al microcontrollore utilizzato (modulo U2, PIC18F2520) e al modulo Bluetooth RN-42 (U1). I quattro relé vengono comandati dal PIC tramite altrettanti I/O configurati come uscite e i relativi BJT che alimentano, quando la loro base è polarizzata dal livello logico alto presente sulla rispettiva uscita, le rispettive bobine. In parallelo a ciascuna bobina si trova il diodo che protegge la giunzione base-collettore dall'extratensione che essa genera, a causa della sua natura induttiva, quando viene sospesa la corrente; infatti in tal caso la bobina reagisce all'interruzione, determinata dal passaggio del BJT dalla conduzione all'interruzione, tentando di mantenere il passaggio di corrente e determinando un picco di tensione di polarità inversa, che applicato al transistor lo danneggerebbe. Il diodo assorbe l'extratensione. In parallelo alla bobina di ciascun

relé si trova anche un LED, con relativa resistenza di limitazione della corrente, usato per indicare se lo stesso è scattato o a riposo. Quanto agli ingressi, fanno capo ognuno ad un fotoaccoppiatore utilizzato per trasferire al microcontrollore dei livelli logici mantenendo gli input separati galvanicamente dal circuito e ovviamente dal PIC.

Ogni ingresso è attivo quando riceve una tensione continua di valore compreso fra 8 e 30 volt: tensione presente significa livello logico alto sulla rispettiva linea di I/O del microcontrollore, mentre se la tensione applicata è di valore insufficiente a far eccitare il fotoaccoppiatore (o manca, cioè con l'ingresso a riposo) il piedino 5 dello stesso è a livello logico alto (per effetto del pull-up implementato nelle linee RB0, RB1, RB6, RB7 dal firmware) e l'ingresso viene considerato a riposo. Completa il circuito, una sezione di alimentazione formata dal regolatore di tensione U3 (un LD1086-3.6 che fornisce i 3,6 volt alla logica); ai punti + e - PWR va collegata l'alimentazione principale, che è previsto sia una tensione continua di valore compreso tra 12 e 15 Vcc. I relé funzionano con la tensione d'ingresso passata attraverso il diodo D1, che protegge dall'inversione di polarità.

IL FIRMWARE

Il firmware che gira nel microcontrollore gestisce la rilevazione dello stato i due degli input fotoaccoppiati (IN1 e IN2) nonché di quello dei due pulsanti on-board; provvede quindi a ripetere sui relé l'attivazione di pulsanti e ingressi IN1 e IN2. Anche la comunicazione con il modulo U1 viene gestita dal PIC, sia a basso livello (comandi AT previsti dal chip RN-42) sia ad alto livello (comandi previsti per l'attivazione

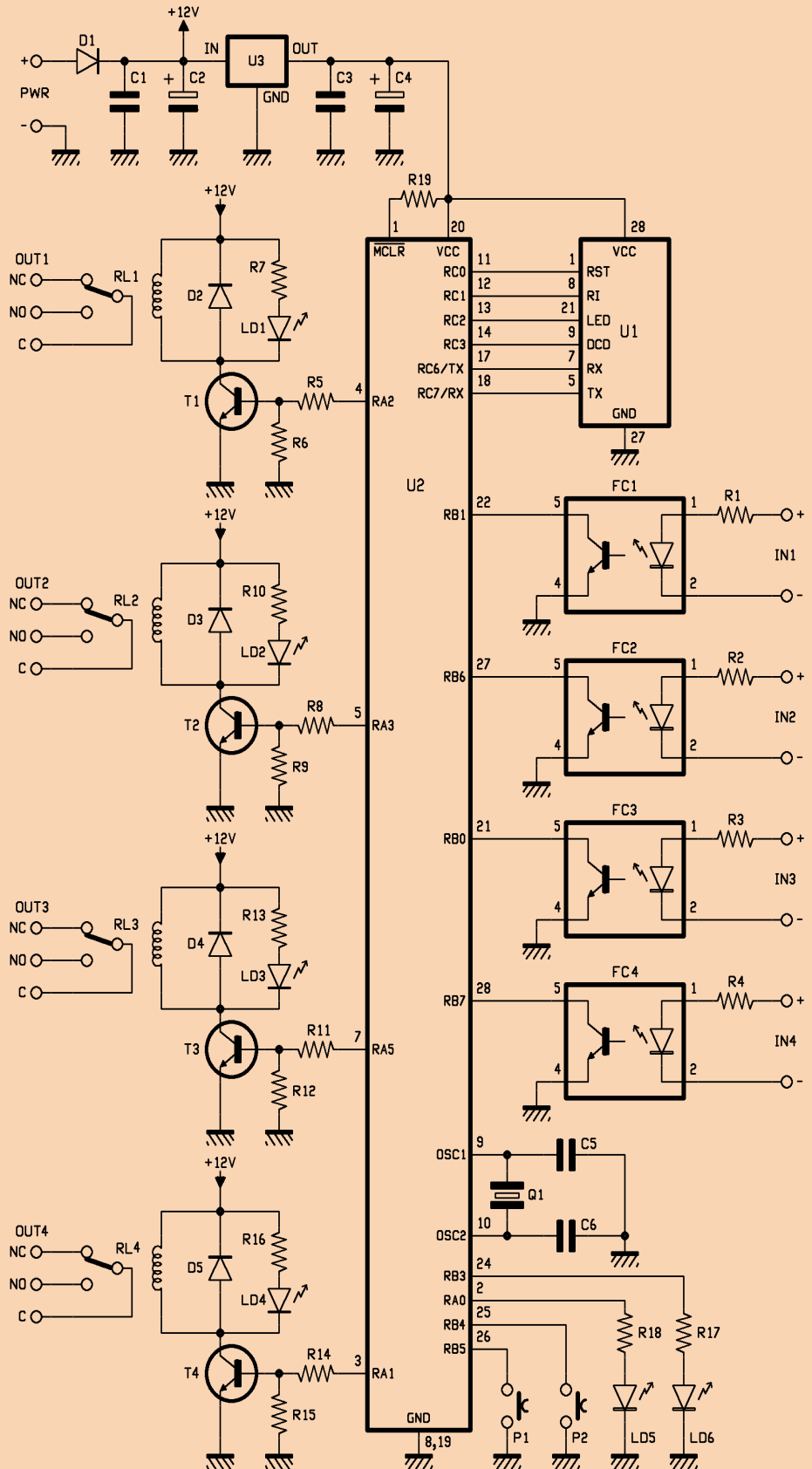




Fig. 1



Fig. 2

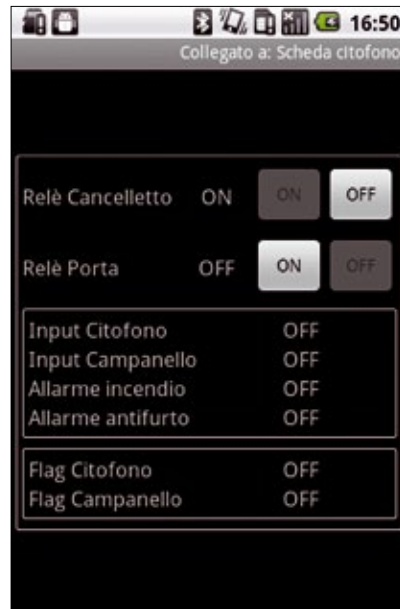


Fig. 3

degli output e lettura degli input). Una volta applicata l'alimentazione al circuito, il firmware che gira nel PIC inizializza le proprie linee impostando RA1, RA2, RA3 ed RA5 come uscite per il comando dei relé; RB0, RB1, RB6 ed RB7 diventano ingressi con pull-up interno dedicati alla lettura dei livelli di uscita dei fotoaccoppiatori. Le linee RA0 ed RB3 vengono inizializzate come uscite usate per

accendere i LED di segnalazione ed RB4 ed RB5 diventano input a pull-up interno, per la lettura dei pulsanti; infine il firmware utilizza l'UART interno per dialogare con il modulo Bluetooth.

In seguito si entra nella sezione di gestione vera e propria del programma, il cui algoritmo di funzionamento può essere schematizzato nel modo seguente.

- Si testano gli input digitali

(fotoaccoppiatori e pulsanti) e si gestiscono correttamente gli output dei relé o LED (il LED giallo è usato come segnalazione per indicare che è stato premuto il pulsante citofono e/o campanello, mentre il LED verde indica che è attivo il link bluetooth).

- Si verifica se è necessario eseguire l'inizializzazione dell'RN-42 (si fa all'avvio, op-



Fig. 7



Fig. 8



Fig. 9



Fig. 4



Fig. 5



Fig. 6

- pure in caso di aggiornamento del nome o PIN code);
- Si identifica se è attiva la comunicazione Bluetooth; in caso contrario si attende dall'UART la relativa segnalazione; in caso positivo, invece, si verifica se sono stati ricevuti dei caratteri (byte) dal modulo. In tal caso si leggono e si analizzano per l'identificazione dei comandi.



Fig. 10

Segnaliamo che sono state utilizzate due variabili per memorizzare la condizione di pressione del citofono e campanello; questo perché potrebbe accadere che la pressione di un pulsante sia troppo breve e l'App Android non faccia in tempo a rilevarla. Memorizzando la pressione all'interno di variabili, queste vengono mantenute e quindi anche se lette con alcuni secondi di ritardo, permettono alla App di rilevare e segnalare la condizione. In conclusione, vi facciamo notare che nel firmware sono stati implementati diversi timeout per la comunicazione utilizzati per evitare blocchi del software nel caso in cui il collegamento bluetooth realizzato cada all'improvviso (sempre possibile quando si utilizzano tecnologie di trasmissione senza fili).

SOFTWARE PER ANDROID

L'App sviluppata per smartphone Android ("RingerControl") permette il collegamento in Bluetooth con la scheda di I/O qui descritta e segnala tramite vibrazione ed allarme la condizione di citofono o campanello premuto, nonché l'attivazione dei

restanti input optoisolati; inoltre consente di inviare i due comandi per attivare i relè collegati al cancelletto o alla porta.

All'avvio, dopo il tipico splash screen di presentazione del software (Fig. 1), vengono avviati la scansione e il rilevamento di tutti i dispositivi Bluetooth presenti all'interno della zona di copertura dello smartphone. Nella lista visualizzata (Fig. 2) vengono mostrati sia tutti i sistemi già accoppiati, sia i nuovi rilevati (nell'esempio è presente solo un elemento con cui è già stata eseguita la procedura di accoppiamento); toccando lo schermo sulla voce della scheda ("Scheda" nel nostro esempio) la si seleziona e si richiede l'avvio del collegamento. Nella schermata si trova inoltre un pulsante per richiedere una nuova esecuzione della scansione e l'aggiornamento delle liste.

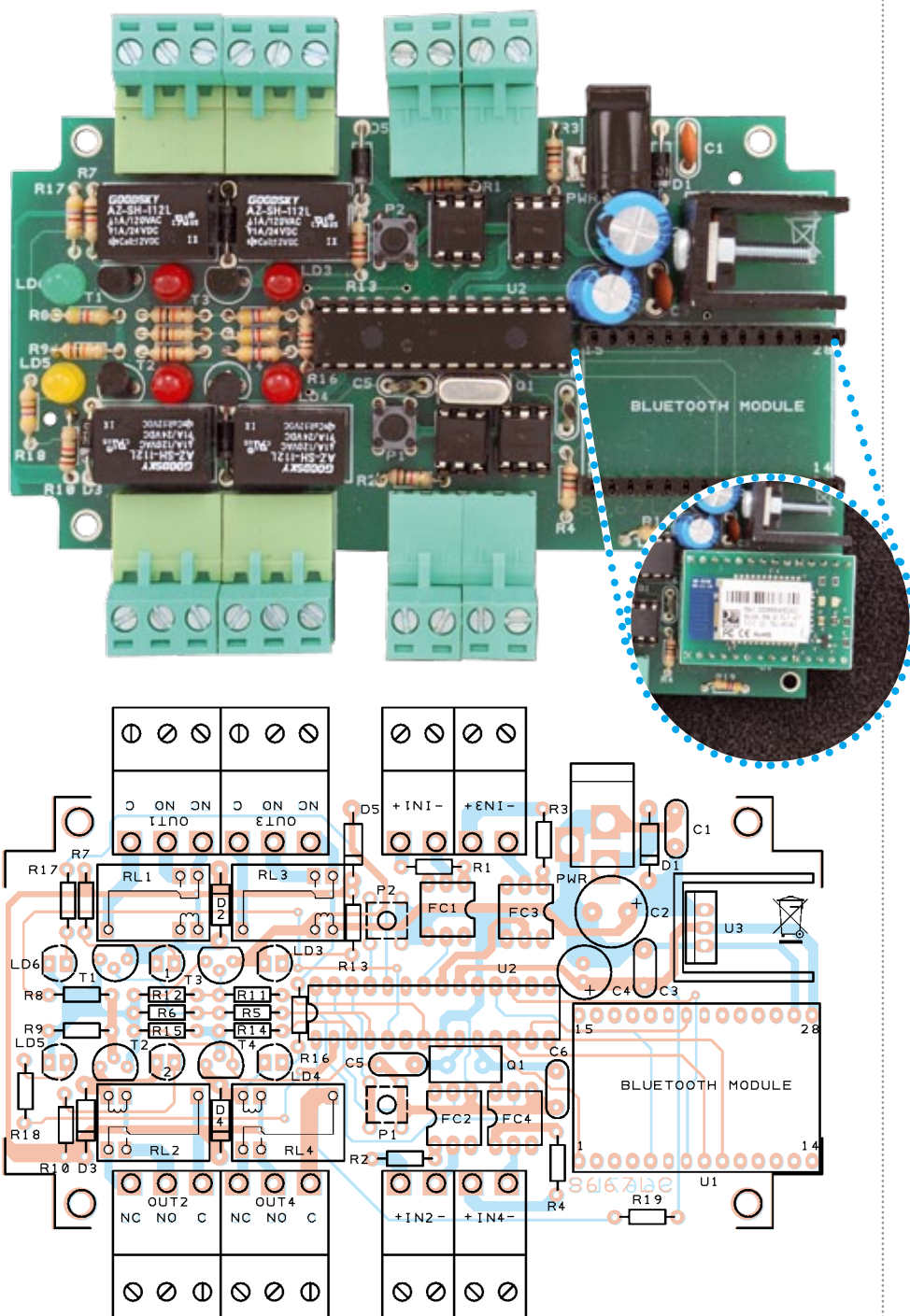
A questo punto, se è la prima volta che il dispositivo viene rilevato, viene richiesta la digitazione del PIN code ("5555" è il predefinito) e in seguito si passa nella schermata principale del programma (Fig. 3) nella quale, partendo dall'alto, tramite etichette di testo

Elenco Componenti:

R1÷R4: 1 kohm
 R5: 4,7 kohm
 R6: 10 kohm
 R7: 1 kohm
 R8: 4,7 kohm
 R9: 10 kohm
 R10: 1 kohm
 R11: 4,7 kohm
 R12: 10 kohm
 R13: 1 kohm
 R14: 4,7 kohm
 R15: 10 kohm
 R16: 1 kohm
 R17, R18: 470 ohm
 R19: 1 kohm
 C1, C3: 100 nF multistrato
 C2: 470 µF 25 VL elettrolitico
 C4: 470 µF 16 VL elettrolitico
 C5, C6: 10 pF ceramico
 U1: Socket RN-42 (FT1018M)
 U2: PIC16F876A (MF1095)
 U3: LD1086-3.6
 D1÷D5: 1N4007
 Q1: quarzo 20 MHz
 LD1÷LD4: LED 5 mm rosso
 LD5: LED 5 mm giallo
 LD6: LED 5 mm verde
 T1÷T4: BC547
 RL1÷RL4: Relé 12V 1 scambio
 P1: microswitch
 P2: microswitch
 FC1÷FC4: 4N25

Varie:

- Plug alimentazione
- Morsettiera componibile 2 poli 90° (4 pz.)
- Morsettiera componibile 3 poli 90° (4 pz.)
- Zoccolo 3+3 (4 pz.)
- Zoccolo 14+14
- Strip femmina 14 poli (2 pz.)
- Vite 10 mm 3 MA
- Dado 3 MA
- Dissipatore (ML26)
- Circuito stampato codice S667



e pulsanti viene visualizzato lo stato delle uscite a relé del cancelletto e della porta (nel nostro esempio il relé del cancelletto è attivo perché abbiamo premuto il pulsante P2 sulla scheda, mentre quello della porta è a riposo).

Più sotto, nel successivo riquadro, vengono visualizzati gli stati dei quattro ingressi fotoaccoppiati che corrispondono, in sequenza, alle funzionalità *pulsante citofono*, *campanello*, *allarme antincendio* ed *antifurto*.

Infine, l'ultimo box visualizza lo stato delle due variabili di memorizzazione dello stato (flag) citofono o campanello. I pulsanti presenti nella schermata permettono di attivare (per circa 2 secondi) i relé collegati

rispettivamente alla gestione dell'apertura del cancelletto e dell'elettroserratura della porta. Per richiedere l'attivazione è sufficiente toccare lo schermo su un pulsante attivo; a questo punto il comando selezionato viene inviato alla scheda (nello schermo appare una finestra indicante l'invio del comando, come in Fig. 4) e, al completamento, viene aggiornata l'interfaccia grafica.

All'interno del software per Android è stato previsto anche un thread sempre attivo, che ciclicamente si collega alla scheda via Bluetooth, legge lo stato corrente degli input fotoaccoppiati (IN1÷IN4), delle uscite a relé e dei flag relativi all'arrivo dei segnali corrispondenti a citofono e campanello, aggiorna la grafica secondo le ultime informazioni ed eventualmente segnala gli allarmi.

Gli allarmi presenti possono essere relativi alla pressione del pulsante del citofono (Fig. 5) o di quello del campanello (Fig. 6), ma anche all'allarme antincendio (Fig. 7) oppure a quello in arrivo dall'antifurto (Fig. 8).

La vera utilità degli allarmi non è legata tanto all'audio o alle finestre di allarme che generano, ma alla segnalazione tramite vibrazione che viene eseguita in contemporanea alla visualizzazione delle schermate mostrate. Facciamo notare che le schermate rimangono visualizzate fino a quando non vengono chiuse tramite il relativo pulsante "Close"; allo stesso modo, la vibrazione viene continuamente eseguita (con intervalli di circa 5 secondi) fino a quando le finestre non vengono chiuse. Questo accorgimento serve per evitare che gli allarmi vengano "persi" se la persona che ha con sé lo smartphone non si accorge di vibrazioni brevi o brevi perché tardive.

Ultima cosa da notare è la presenza di una pagina di configurazione attivabile tramite i tipici menu delle applicazioni Android (Fig. 9) dalle quali è possibile configurare alcune impostazioni Bluetooth e in particolare il nome del sistema e il PIN code di accesso (Fig. 10).

REALIZZAZIONE PRATICA

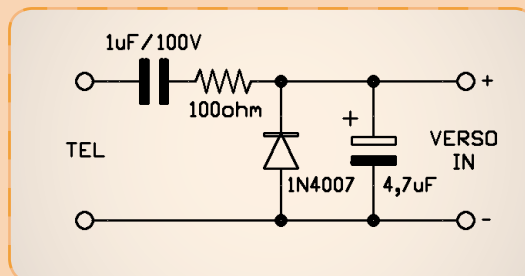
Il sistema è, dal punto di vista hardware, formato da una scheda base e da un modulo che tramite pin-strip le si inserisce sopra in appositi connettori SIL. Il modulo contiene il componente RN-42 della Roving Network, il quale viene fornito dal costruttore in versione SMD, che non è facile saldare; per facilitarne l'integrazione nella scheda base, la Futura Elettronica (www.futurashop.it) lo rende disponibile già montato

sull'apposito modulo (socket, ossia un c.s. che lo ospita) provvisto di pin-strip, quindi raccomandiamo vivamente di utilizzare il modulo di Futura Elettronica, piuttosto che il componente nudo e crudo, perché con quest'ultimo dovrete realizzare da voi il circuito stampato.

Detto ciò, possiamo vedere la costruzione della scheda base, che sicuramente potete realizzare da voi, anche senza particolare esperienza, in quanto è composta in toto da componenti elettronici a montaggio tradizionale, quindi non dagli SMD; pertanto vi basta disporre di un saldatore per elettronica da 25÷30 W (a punta sottile) e un po' di filo di lega saldante da 0,75 mm a base di stagno. Al solito, disponete per primi i diodi al silicio e le resistenze, quindi gli zoccoli per i fotoaccop-

Anche con il telefono fisso

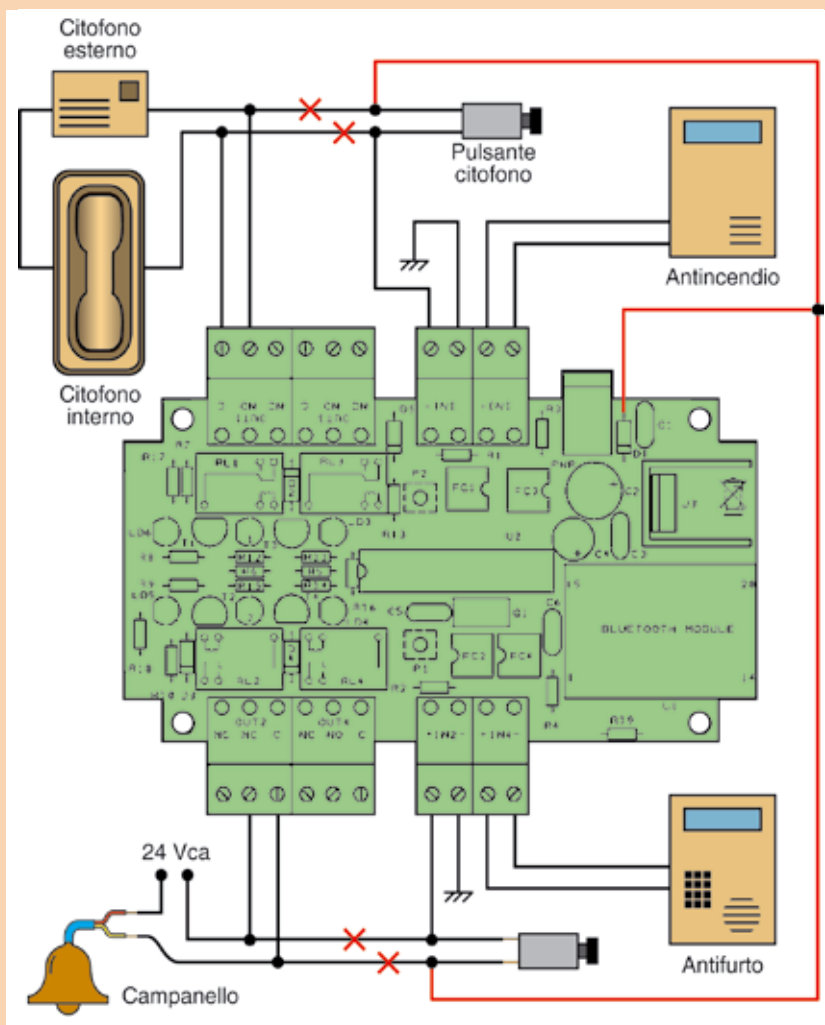
Gli ingressi optoisolati consentono alla scheda di rilevare segnali provenienti da qualsiasi fonte di tensione che sviluppi almeno 8 volt e un massimo di 30 Vcc; collegando ad uno di essi il semplice circuito illustrato qui sopra, la scheda di I/O Bluetooth può riconoscere l'arrivo di una chiamata sul telefono fisso, che normalmente determina la presenza sulla linea telefonica (doppino) di una tensione di circa 70 Veff alternati. Rad-



tensione mediante il diodo 1N4007 e filtrandola mediante l'elettrolitico, possiamo collegare i punti + e - a una delle morsettiere IN1÷IN4 dopo aver cortocircuitato la resistenza in serie al piedino 1 del relativo optoisolatore. All'arrivo dell'alternata di chiamata,

il fotoaccoppiatore andrà in conduzione e comunicherà la chiamata entrante al microcontrollore determinando l'invio dell'avviso allo Smartphone; la App dovrà essere adattata per segnalare la chiamata telefonica invece della condizione prevista per il rispettivo IN.

Collegamento della scheda



Cablaggio della scheda: il positivo comune dei pulsanti campanello e citofono (che vanno isolati dai cavi dell'impianto) si prende dall'anodo del D1 (+PWR); l'alimentazione serve perché gli input richiedono tensione. Le uscite a relè chiudono le linee dei pulsanti quando questi vengono premuti. Nel caso esemplificato in questo articolo, la corrispondenza tra ingressi, pulsanti e relè è quella indicata

in tabella. La corrispondenza vale indipendentemente dall'applicazione, nel senso che anche se interfacciate la scheda di I/O Bluetooth con altri dispositivi, RL1 segue il comando dato a IN1, RL2 replica lo stato di IN2, RL3 segue P2 e RL4 il P1. Agli input IN3 e IN4 possiamo applicare uscite a livello di tensione di qualsiasi sistema: non è necessario si tratti di centraline d'allarme.

input fotoaccoppiato 1	pulsante citofono
input fotoaccoppiato 2	pulsante campanello
input fotoaccoppiato 3	allarme antincendio
input fotoaccoppiato 4	allarme antifurto
output relè 1	replica pulsante citofono
output relè 2	replica pulsante campanello
output relè 3	comando apertura cancelletto
output relè 4	comando apertura porta

piatori e il microcontrollore, nonché i pulsanti miniatura; quindi passate ai transistor e ai LED, poi ai condensatori (prima i non polarizzati e poi gli elettrolitici) ed alla presa plug per l'alimentazione.

Montate quindi due file di contatti femmina a passo 2,54 mm da 14 contatti ciascuna in corrispondenza dei fori per l'inserzione del modulo Bluetooth, i quattro relè e le morsettiere, concludendo con il regolatore di tensione U3, che dovete mantenere verticale e sollevato quanto basta a fissarlo ad un radiatore di calore ad U avente resistenza termica di circa 20 °C/W (va bene il classico ML26). Per l'orientamento dei componenti polarizzati (elettrolitici, transistor, LED, optoisolatori, regolatore di tensione e microcontrollore) riferitevi al piano di montaggio pubblicato in queste pagine.

Quanto al modulo Bluetooth, sia sul c.s. socket su cui è montato, sia sullo stampato della scheda base è presente la numerazione dei pin (sono indicati i pin numero 1, 14, 15 e 28) allo scopo di agevolare il montaggio.

Il modulo socket RN-42 viene fornito da Futura Elettronica (codice FT1018M) con i quattro piccoli ponticelli sotto lo stampato, che all'occorrenza (in base all'utilizzo che si intende fare del modulo) vanno chiusi con una goccia di lega saldante.

Nell'applicazione proposta in queste pagine, tali ponticelli vanno chiusi tutti altrimenti la configurazione del modulo RN-42 non è compatibile con il firmware del PIC della scheda base.

Per completare il montaggio del sistema, inserite il microcontrollore Microchip nell'apposito zoccolo, dopo averlo programmato (il PIC si trova anche già programmato presso Futura Elettronica).

INSTALLAZIONE

Il progetto prevede di abbinare le risorse in/out della scheda secondo quanto descritto nella tabella in fondo a questa pagina, ovvero:

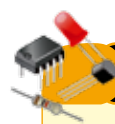
- all'input IN1 si connette il pulsante del citofono, ricordando di staccare quest'ultimo dai fili dell'impianto originario;
- all'IN2 si connette il pulsante del campanello, sconnettendo i cavi dell'impianto originario;
- l'input IN3 è considerato allarme antincendio e ad esso va collegata un'uscita della centralina capace, in caso di allarme) di fornire una tensione continua (rispettare la polarità, altrimenti l'input non rileva alcunché) di valore tra 8 e 30 volt;
- all'ingresso IN4 è abbinato l'antifurto e vi va collegata un'uscita di allarme della centralina capace di fornire una tensione continua di valore compreso tra 8 e 30 V (anche in questo caso il rispetto della polarità è determinante);
- l'OUT 1 replica la condizione del pulsante del citofono, perciò dobbiamo utilizzarne i contatti C ed NO per cortocircuitare i fili che abbiamo staccato dal pulsante di chiamata del citofono quando abbiamo collegato il pulsante all'IN1;
- l'OUT 2 replica la condizione del pulsante del campanello, pertanto possiamo utilizzare i contatti C ed NO del rispettivo relé per cortocircuitare i fili sconnessi dal pulsante stesso quando l'abbiamo collegato all'IN2;
- l'uscita OUT 3 è previsto che comandi l'apertura del cancelletto o portone; lo scambio del rispettivo relé può chiudere il circuito dell'elettroserratura, facendo però attenzione che la tensione e la corrente in questo circuito non eccedano la corrente (1 A) e la tensione (24 Vcc o 120 Vca)

che il relé può sopportare;

- l'output OUT 4 è previsto che comandi l'apertura dell'elettroserratura della porta di casa o altro, come ad esempio un apricancello motorizzato; anche in questo caso bisogna tenere conto dei limiti del relé, ma nei due casi esemplificati non ci sono problemi, dato che l'elettroserratura di solito funziona a 24 V cc o ca, mentre la centralina dell'apricancello di solito ha ingressi di comando a bassa tensione, gestibili tramite scambi di relé o contatti puliti che reggano anche solo 24 V.

Per realizzare il vostro impianto considerate che tipicamente i pulsanti dei citofoni e campanelli non sono altro che contatti puri che si aprono o chiudono a seconda del fatto che si eserciti o meno pressione, ma non è detto che questo sia valido per tutti i sistemi; per questa ragione, visto che in ogni ingresso della nostra scheda è presente un fotoaccoppiatore (integrati FC1÷FC4) al cui diodo emettitore di ingresso (pin 1 e 2 del componente) è collegata una sola resistenza. Questo significa che per attivare il fotoaccoppiatore (e quindi l'input sul PIC) è necessario che all'ingresso sia presente una tensione. Essendo, i pulsanti dei citofoni e dei campanelli, dei contatti puri, è necessario realizzare una linea di alimentazione che alla chiusura del pulsante applichi una tensione (attenzione alla corretta polarità) ai contatti degli ingressi desiderati. La tensione portata ai pulsanti deve essere continua. La maniera più semplice di collegare al circuito i pulsanti, consiste nel prendere il positivo dalla connessione di alimentazione (+ PWR, che è accessibile sull'anodo o anche sul catodo del diodo di protezione D1; il negativo corrisponde alla massa del circuito, che deve

essere collegata ai piedini 2 (IN-) dei fotoaccoppiatori interessati dal comando con i pulsanti di citofono e campanello. I piedini 1 (IN+) dei fotoaccoppiatori degli ingressi interessati vanno collegati ciascuno al capo libero (vale a dire a quello opposto al capo su cui avete applicato il positivo) del rispettivo pulsante. Per quanto riguarda il collegamento ad eventuali centraline di allarme antincendio o allarme, in linea generale si possono fare le stesse osservazioni fatte in precedenza anche se per questi dispositivi c'è maggiore disomogeneità tra i diversi produttori e modelli. Alcuni dispongono di uscite a relé (quindi contatti puri), altre di uscite in tensione, altre di uscite open-collector, altre, infine, non sono provviste di uscite verso sistemi esterni. Le uscite a livello di tensione sono da preferire (rispettando la polarità) ai contatti puliti, perché questi richiedono la realizzazione di un circuito di alimentazione. ■



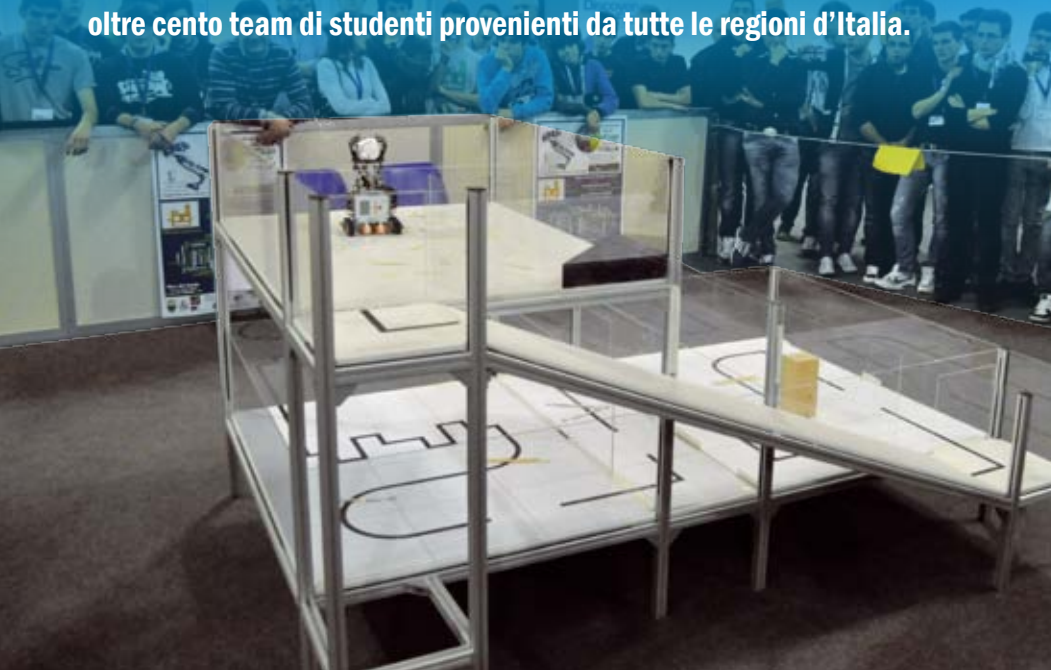
per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato, la app ed il firmware col quale programmare il micro sono scaricabili gratuitamente dal sito della rivista (www.elettronica-in.it). Il circuito è anche disponibile in kit (cod. FT1095K) al prezzo di 64,00 Euro; il firmware utilizzato nel kit (di impiego generico, descritto sul numero 180) va sostituito con quello relativo a questa applicazione. Ricordiamo che il kit comprende anche il modulo Bluetooth FT1018M già montato; quest'ultimo è disponibile anche separatamente al prezzo di 29,00 Euro. 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>

NELLA TERRA DEI ROBOT LA FINALE DELLA **RoboCup JUNIOR 2014**

Ai blocchi di partenza la VI edizione "Pisa 2014" della RoboCup Junior Italia, organizzata in collaborazione col Dipartimento di BioRobotica della Scuola Superiore Universitaria Sant'Anna di Pisa e che vedrà la partecipazione di oltre cento team di studenti provenienti da tutte le regioni d'Italia.



a cura della REDAZIONE



Tutto pronto per la sesta edizione della finale italiana della RoboCup Junior Italia, la manifestazione che vedrà i robot di 114 team di studenti scontrarsi tra loro dal 9 al 12 aprile 2014 per affermarsi nelle categorie in programma: Soccer, Rescue e Dance/Theatre. Quest'anno - dopo Torino, Vicenza, Catania, Riva Del Garda e Pescara - la manifestazione arriva in terra di Toscana, grazie al Dipartimento di BioRobotica della Scuola Superiore Universitaria Sant'Anna di Pisa a cui è stata affidata l'organizzazione dell'evento dalla "Rete di scuole per la Robocup Jr ITALIA" che rappresenta

l'espressione operativa del Ministero dell'Istruzione che, attraverso l'Autonomia, ha affidato agli Istituti scolastici le funzioni di ricerca e sperimentazione.

L'unione di più Istituti (la Rete) permette di affrontare percorsi impegnativi al di là delle forze delle singole scuole: dagli 8 Istituti scolastici fondatori nel 2008/09, la Rete è cresciuta sino a contare nell' a.s. 2012/13 ben 43 scuole in 10 regioni e 2 province autonome, 24 Istituti superiori e 19 Istituti del primo ciclo.

RoboCup è un'iniziativa scientifica internazionale con l'obiettivo di migliorare lo stato dell'arte dei robot intelligenti; nata nel 1997, la

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

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

La manifestazione si articola in tre differenti categorie:

- Soccer
- Rescue
- Dance/Theatre

Nella categoria Soccer (a sua volta suddivisa in due sottocategorie)



i due robot in gara si muovono autonomamente in un campo di calcio in miniatura inseguendo una speciale pallina che emette raggi IR, nel tentativo di mettere a segno quanti più goal possibile.

Nella categoria Rescue A i robot sono impegnati ad individuare in modo rapido e preciso le vittime all'interno di uno scenario che simula le rovine di un disastro. All'interno di un'arena i robot si dovranno confrontare con problemi di varia difficoltà dal seguire una linea su una superficie piana (percorso guidato), al muoversi attraverso gli ostacoli di un terreno

irregolare (percorso libero), al percorrere una rampa ed afferrare e porre in salvo la vittima.

Nella categoria Rescue B i robot sono impegnati nell'esplorazione di un labirinto con l'individuazione al suo interno di vittime.

Infine, nella Dance, uno o più robot vestiti in costume si muovono e danzano seguendo una coreografia, in armonia con la musica di sottofondo.

Tutte le categorie sono a loro volta suddivise in Under 19 (riservata ai ragazzi delle Scuole Medie Superiori) e in Under 14 (per i ragazzi delle Medie Inferiori).

Come per l'edizione 2013, anche il calendario della manifestazione toscana si articola in 4 giorni: dall'apertura ufficiale del mercoledì pomeriggio alla cerimonia di premiazione del sabato a mezzogiorno.

Tre stampanti 3D ai vincitori

La nostra società supporta la RoboCup Junior offrendo numerosi premi alle squadre che vinceranno e si piegheranno ai primi posti nelle varie discipline previste.

Tra i premi in palio ben tre Stampanti 3D in scatola di montaggio con le quali realizzare facilmente chassis, ruote, ingranaggi, pezzi di robot e tanto altro ancora. Offriremo anche schede e sistemi di prototipazione per Arduino e Raspberry Pi nonché libri e riviste sullo stesso argomento. Un modo per premiare le qualità dei team migliori ma anche per fornire strumenti moderni e innovativi per l'attività in campo robotico nei prossimi anni.



Di seguito il programma della manifestazione:

- **Mercoledì 9 aprile 2014:**

Ore 15.00 Apertura registrazione squadre

Ore 15.00 – 18.00 prove sui campi di gara

Ore 18,30 Inaugurazione ufficiale VI edizione RoboCup Jr Italia

- **Giovedì 10 aprile 2014:**

Registrazione squadre

Gare Soccer

Gare Rescue A e B (1° turno)

Gare Dance/Theatre

Eventi collaterali (workshop, dimostrazioni, incontri per i docenti)

- **Venerdì 11 aprile 2014:**

Registrazione squadre

Gare Soccer

Gare Rescue A e B (2° turno)

Gare Dance/Theatre (finale)

Gare Under 14: non competitiva di Rescue Theatre/Dance

ORE 15.00 – Convegno a cura della RCJ Italia in collaborazione con UNITO – UNIFE – SSSUP: la Robotica Educativa, discussione tesi del Corso conduttori LRE nelle scuole del I settore dell'Istruzione.

- **Sabato 12 aprile 2014**

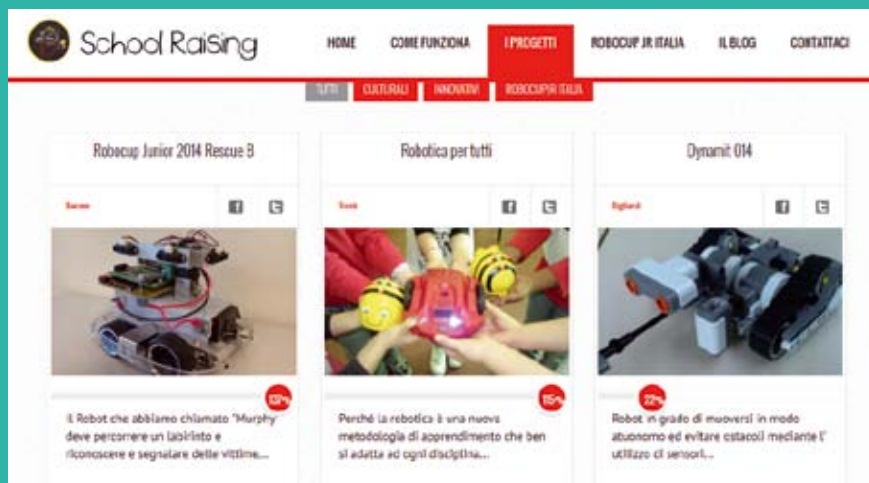
Ore 9.00 Finali Dance Under 14

Ore 10.30 Cerimonia



Anche la scuola italiana scopre il crowdfunding

Uno degli ostacoli che frenano la diffusione di iniziative quali la RoboCup Junior - e il coinvolgimento di un numero sempre maggiore di scuole - è la mancanza di fondi pubblici per questo genere di attività (e non solo per queste); lo stesso problema riguarda gli studenti e le loro famiglie che debbono farsi carico delle spese di partecipazione alle gare e dei materiali necessari per realizzare i prototipi di robot. Quest'anno, per la prima volta, per supportare gli studenti italiani che partecipano ai progetti di robotica, il crowdfunding ha fatto il suo ingresso nella scuola italiana con l'iniziativa School Raising 4 RoboCup



Jr Italia il cui obiettivo è proprio quello di aiutare i team nella ricerca dei fondi necessari alla realizzazione dei prototipi di robot.

Il progetto, nato dalla collaborazione tra School Raising e la Rete di scuole per la RoboCup jr Italia ha visto la partecipazione di 12 progetti di robotica presentati dagli studenti di diverse città d'Italia (Vicenza, Verona, Padova, Giulianova, Bari, Torino, Urbino, Roma, Pavia, Venezia). Differenti i budget individuati per ognuno di loro: si parte dai 495 Euro per costruire Robot Ballerini fino ai 12,000 Euro per un'iniziativa mirata all'utilizzo, durante la didattica, di robot costruiti dagli studenti stessi, presupponendo un innalzamento del livello di motivazione all'apprendimento.

In questo caso viene applicata la formula "Keep it all", ovvero alla fine della campagna (2 Aprile) il budget raccolto verrà comunque versato ai team, anche senza che sia stato raggiunto l'obiettivo.

Il 2014 sarà l'anno di prova per una collaborazione più stretta, prevista nel 2015, con più tempo a disposizione e maggiore integrazione dei due progetti.

di premiazione U14 e a seguire U19.

Tutte le gare si svolgeranno a Pontedera (Pisa) nella zona conosciuta come «dente Piaggio», dal 9 al 12 aprile 2014.

Le gare di Dance e Theatre U14 e U19 (anche non competitiva) si svolgeranno nell'auditorium del Museo Piaggio, in

viale Rinaldo Piaggio 7 che si trova a circa 100 m dalla sede dell'Istituto di BioRobotica.

Le gare di Rescue A U14 e U19 si svolgeranno nel Centro Culturale 7 Soli 7 Lune, che dista circa 200 m dall'Istituto di BioRobotica.

Le gare di Rescue B U14 e U19 nonché le gare non competitive di Soccer B Light si svolgeranno alla



Palestra dell'Istituto Pacinotti in Via Dante, 42 (nella stessa sede si svolgerà anche la gara di Soccer B Light).

Alcune gare sono valide per la qualificazione ai Campionati mondiali che quest'anno si svolgeranno dal 19 al 25 luglio a João Pessoa (Brasile).

FLL Italia, ecco chi ha vinto

A rinfoltire il nutrito panorama degli eventi dedicati alla robotica educativa e all'autocostruzione e sperimentazione in campo robotico, la finale italiana della FIRST LEGO League, trampolino di lancio - per le due squadre uscite vincitrici - per la finale europea e quella mondiale, ma anche l'edizione 2014 di Officine Robotiche in programma a Roma. Due eventi diversi per natura (FLL Italia è più orientata alla robotica educativa, mentre Officine Robotiche nasce come laboratorio di idee e del DIY in campo robotico) ma accomunati dalla condivisione delle conoscenze.

Dopo le fasi regionali di qualificazione che si sono svolte il 18 gennaio a Pistoia per il Centro Italia, il 26 gennaio a Cuneo (per il Nord-Ovest), il 2 febbraio a Udine (per il Nord-Est) e il 6 febbraio a Catania per il Sud Italia, FLL Italia si è conclusa a Udine, il 14-15 marzo, dove la fase finale di qualificazione per l'Italia ci ha svelato i nomi

dei team che -classificati rispettivamente primo e secondo- hanno conquistato il diritto a partecipare alla finale mondiale di Nature's Fury che si terrà negli Stati Uniti e alla finale europea, in programma in Spagna. Anche quest'anno il Liceo Rosmini di Rovereto ha fatto l'en-plein, facendo qualificare ai primi due posti le proprie squadre: il team primo classificato - quello del Liceo Rosmini Team A di Rovereto - andrà alla finale mondiale (World Festival) prevista dal 23 al 26 aprile a St. Louis (Stati Uniti); il Liceo Rosmini Team B di Rovereto (secondo classificato) potrà partecipare alla finale europea che si terrà il 23 marzo a Barcellona (Spagna). Oltre ai premi assegnati alle prime qualificate, la finale italiana ha conferito menzioni e premi speciali ai team Marconi B (Rovereto) per il miglior progetto tecnico, DBP Team Mirandola (Mirandola) per il miglior progetto scientifico, Melting pot (Genova) per ispirazione e scoperta, Erre come Robot (Lari) per il fair play. È stato premiato come miglior team, Rairob di Bolzano, ed ha conquistato il Premio Media Direct final four, la squadra Pluchirobot di Sant'Agata Li Battiati. Infine, il Premio Speciale Giuria è stato assegnato al team Firestorm di Piacenza, che si è aggiudicato anche un premio speciale per la categoria Robot

game. Ulteriori dettagli sono disponibili sul sito ufficiale della manifestazione italiana, www.fll-italia.it.



Officine Robotiche

Il mese di aprile regalerà agli appassionati di robotica e del DIY tecnologico un nuovo evento, nato per far incontrare chi è accumulato dalla passione per la robotica e dalla voglia di condividere le proprie conoscenze. Passione per la robotica e condivisione sono i temi principali della manifestazione, che nasce dalla volontà di un gruppo di amici sparsi per l'Italia con la voglia di confrontarsi e divulgare le loro conoscenze a chi, come loro, è appassionato o anche solo incuriosito da queste tematiche.

Officine Robotiche è un evento poliedrico che andrà in scena il 12 e 13 aprile a Roma e che prevede un'area espositiva dove sarà possibile vedere alcuni Robot in azione, fare domande ai loro costruttori e scambiare esperienze al riguardo, ma anche un nutrito ciclo di laboratori pratici e workshop, dove i partecipanti potranno vedere dal vivo le soluzioni da cui nasce la progettazione e la realizzazione di robot. A connotare maggiormente la sua natura pratica, Officine Robotiche ospiterà il Fablab SPQwoRk, che è insieme un laboratorio di fabbricazione digitale ed un makerspace. Il Fablab SPQwoRk nasce nel 2013 dalla community dell'omonimo Coworking, struttura situata nel cuore direzionale di Roma nei pressi della Stazione Tiburtina, cuore direzionale della Capitale.

Nell'area espositiva, maker, studenti di istituti scolastici e progettisti di start-up esporranno robot da loro realizzati e ne sveleranno i trucchi; i loro creatori sveleranno al pubblico partecipante i segreti e le soluzioni che hanno trasformato le loro idee nelle realtà che ognuno potrà, durante l'evento, toccare con mano e vedere in azione. Alla pagina web <http://2014.officinerobotiche.it/category/espositori/> potete vedere già qualche anticipazione: droni, robot di servizio, ROV (Remote Operated Vehicle), quadricotteri, robot cingolati, sono solo alcuni esempi.

Officine Robotiche si terrà presso lo spazio SPQwoRk, nato come contenitore di

talenti destinato alla produzione, promozione e valorizzazione della creatività in genere, con particolare attenzione per la creatività giovanile. SPQwoRk si trova nel cuore direzionale di Roma, nei pressi della nuova Stazione Tiburtina, nell'ex Pastificio Scanzani, oggi diventato una grande struttura di archeologia industriale.

Concludendo la panoramica sugli eventi del mese, non possiamo non spendere due parole su RomeCup 2014, da cui sono usciti i nomi dei vincitori -dell'VIII Trofeo Internazionale Città di Roma di Robotica- e quelli dei primi team selezionati per la RoboCup 2014.

Al termine delle fasi finali, svoltesi il 21 marzo, nel suggestivo scenario della Protomoteca in Campidoglio, sono stati consegnati i premi ai team vincitori delle sei categorie in gara per l'VIII Trofeo Internazionale Città di Roma di Robotica:

- Dance Primary - "Funiculì Funiculà" dell'IC Gonzaga di Eboli;
- Dance Secondary; - "Democrito team" del Liceo scientifico Democrito di Roma;
- Rescue Primary - "Spoltore Insieme" dell'IC Spoltore di Pescara;
- Rescue Secondary - "Rangers" dell'IIS Volta di Pescara;
- Explorer Junior - "Gerolamo 2.0" dell'IIS Cicerone di Sala Consilina;
- Explorer Senior - "Red 1" dell'IIS Rosatelli di Rieti.

Per le tre categorie valevoli per le selezioni nazionali RoboCup 2014, i vincitori sono stati:

- Soccer Open League - "Robot Magut" dell'IIS Righi di Treviglio;
- Soccer Light Weight - SPQR Levis dell'IIS Galilei di Roma;
- Cospace Theatre - "Democrito Team" dell'IIS Democrito di Roma.



GENERATORE DI FUNZIONI PER myDAQ

Abbiniamo a myDAQ di National Instruments un generatore di funzioni per compiere misure e test su circuiti analogici. La scheda integra anche un sensore di temperatura lineare.



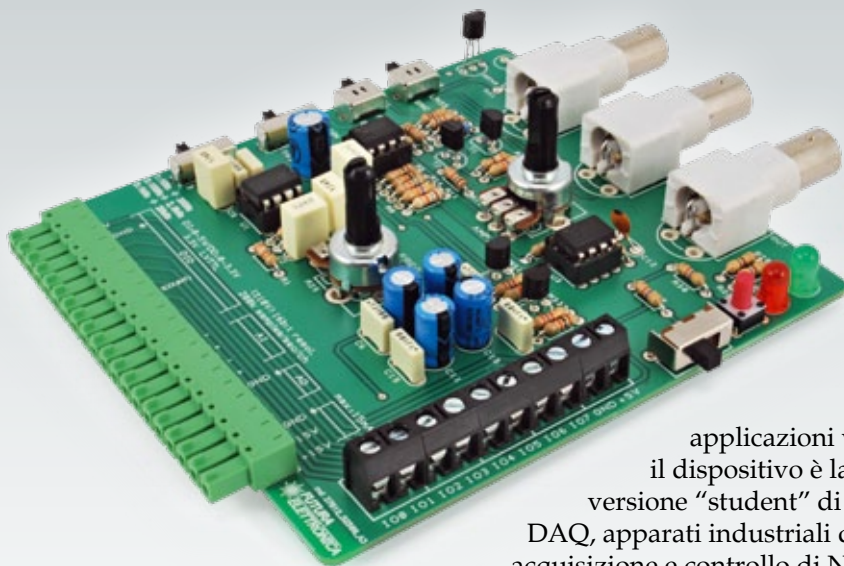
dell'Ing. VINCENZO MENDOLA

Chi lavora in ambito elettronico avrà sicuramente utilizzato molte volte un generatore di funzioni e probabilmente ne ha autocostruito almeno uno nella propria carriera di tecnico e sperimentatore; questo strumento è indispen-

sabile per effettuare molte prove di laboratorio, quali ad esempio la verifica della banda passante o l'analisi della risposta ai transienti di un amplificatore, la risposta in frequenza di un filtro e via di seguito. Essendo, il generatore di funzioni, uno

strumento che molti dei nostri lettori hanno, vi starete chiedendo come mai stiamo presentando nuovamente qualcosa del genere; ebbene, la risposta è che non si tratta di un generatore

progettato per essere utilizzato come strumento stand-alone. Infatti, sebbene dal punto di vista delle prestazioni



applicazioni varie; il dispositivo è la versione "student" di NI DAQ, apparati industriali di acquisizione e controllo di National Instruments. Con myDAQ abbiamo pubblicato e stiamo tuttora pubblicando una serie di esercitazioni pratiche mediante myKIT, il set per esperimenti appositamente creato.

IL CIRCUITO

Oggi i progressi nella produzione dei semiconduttori hanno permesso la realizzazione di circuiti integrati molto complessi in contenitori sempre più piccoli e a prezzi veramente accessibili a tutte le tasche; in quest'ottica rientrano i generatori di funzioni integrati, tra cui il MAX038 della Maxim e il popolare XR2206 della Exar, che permette di realizzare un completo generatore di funzioni utilizzando una manciata di componenti esterni di contorno. Tali integrati sono ormai da tempo fuori produzione, anche se è possibile trovarne ancora in vendita da quanti ne hanno fatto scorta in passato. Durante la definizione delle caratteristiche salienti del nostro progetto, abbiamo cercato degli integrati in grado di assolvere alle stesse funzioni e abbiamo trovato nel più recente XR2209 (sempre prodotto dalla Exar) ciò che soddisfa le nostre esigenze. Ci siamo quindi orientati verso una soluzione "ibrida", affiancando all'XR2209, che utilizza-

mo per generare l'onda quadra e quella triangolare, ulteriore componentistica che ci permette di generare l'onda sinusoidale ricorrendo a un circuito di pulse-shaping che parte dall'onda triangolare.

Entriamo nel dettaglio, osservando lo schema elettrico del circuito e diciamo subito che questo nasce per essere interfacciato con myDAQ, inserendolo con l'apposito connettore nella connessione laterale del dispositivo di National Instruments.

Il circuito è essenzialmente costituito da tre blocchi: l'alimentazione, il generatore di funzioni vero e proprio e quella che potremmo definire la parte prettamente "didattica", contenente un trasduttore di temperatura LM35 ed alcuni LED e pulsanti gestiti dai GPIO del myDAQ. Quest'ultima parte ci servirà per sperimentare un po' con LabVIEW.

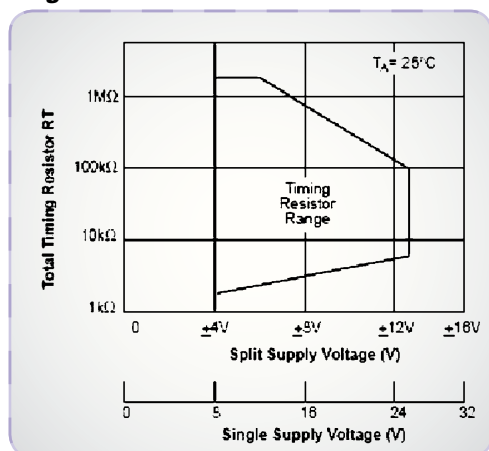
La sezione di alimentazione è costituita da due regolatori di tensione lineari regolabili, entrambi in case TO-92: sono uno positivo (U4, un LM317L) e uno negativo (U3, di tipo LM337L). Essi provvedono a ricavare, dalla tensione duale di $\pm 15V$ in uscita dal myDAQ, una componente simmetrica del valore di $\pm 6V$, necessaria per alimentare l'XR2209 (U1). In questo modo possiamo alimentare direttamente l'intero circuito attraverso le tensioni in uscita dal myDAQ, senza alcuna necessità di fare ricorso ad un'alimentazione esterna dedicata, a tutto vantaggio della compattezza dell'insieme.

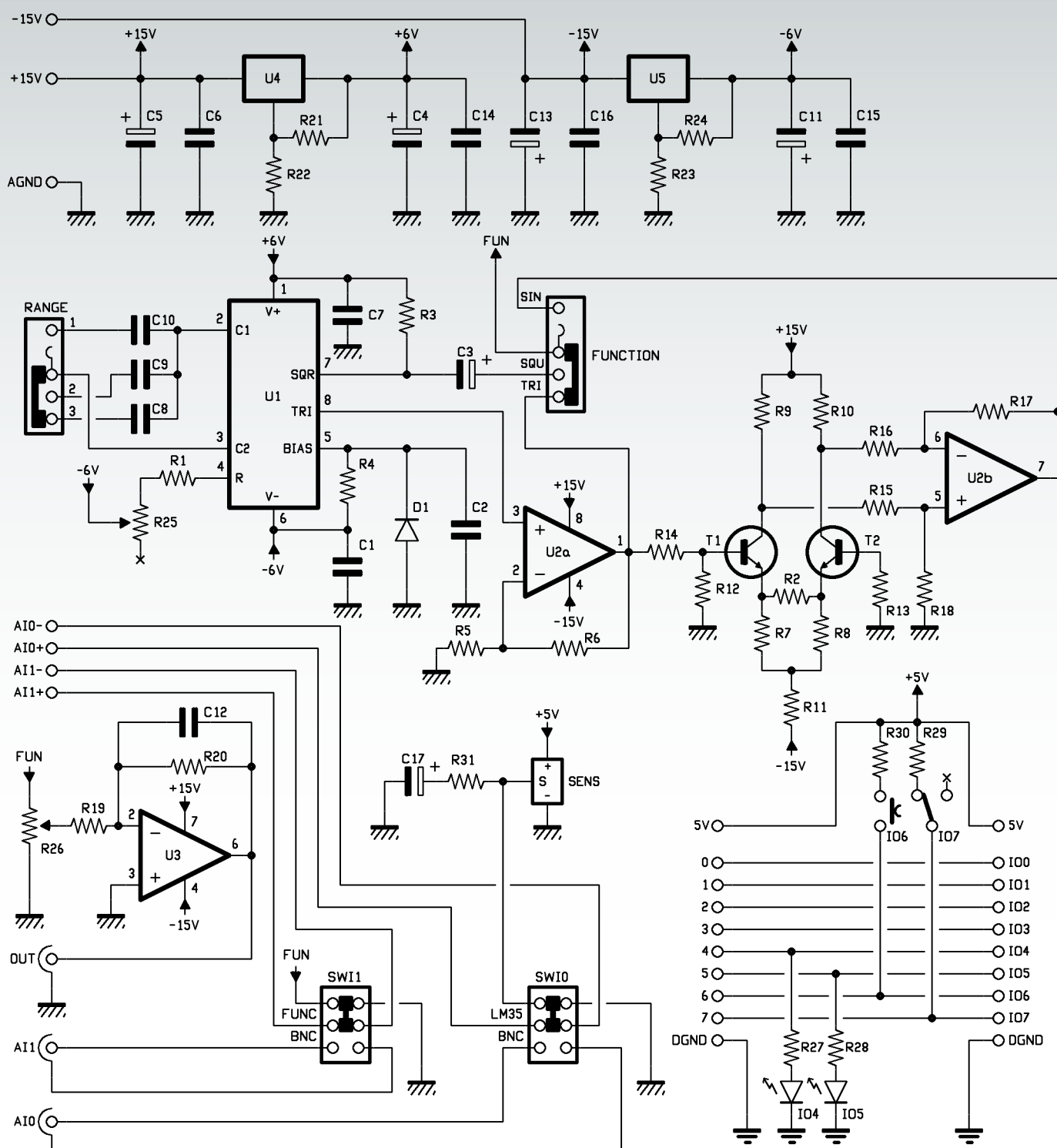
Veniamo ora alla parte più importante e più complessa del circuito, ossia il cuore del nostro dispositivo: il generatore di funzioni vero e proprio. U1 ci permette di generare un'onda quadra e un'onda triangolare di ampiezza e di frequenza variabi-

e delle caratteristiche non rappresentano certamente lo stato dell'arte in questo campo, il generatore che vi proponiamo in queste pagine introduce delle soluzioni didatticamente interessanti e, soprattutto, è stato appositamente sviluppato per essere utilizzato in combinazione con il myDAQ della National Instruments, come utile complemento nell'apprendimento dei fondamenti dell'elettronica.

NI myDAQ è uno strumento di acquisizione e output di segnali utilizzabile con l'ambiente LabVIEW di National Instruments per compiere vari esperimenti o interfacciare un computer con segnali di vario genere in misure, controlli e

Fig. 1





le. La configurazione utilizzata, è la stessa riportata nel data-sheet ufficiale della Exar e raccomandata per l'utilizzo con alimentazione duale (punti [1] e [2] dei Riferimenti); in pratica la frequenza viene regolata attraverso

la capacità presente tra i piedini 2-3 e la resistenza (*timing resistor*) compresa tra il piedino R e il -6V. La Fig. 1 riporta il campo dei valori di resistenza di temporizzazione ammessi, in funzione della tensione di alimentazione. Come

si vede, utilizzare un'alimentazione di $\pm 6V$ ci permette di operare praticamente alla massima dinamica di valori di resistenza ammessi. Inoltre questo valore di tensione garantisce il minimo drift in frequenza, come riportato

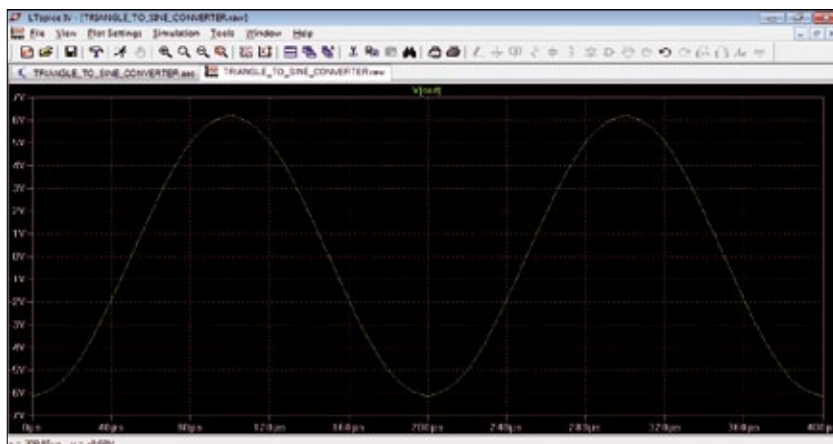


Fig. 2

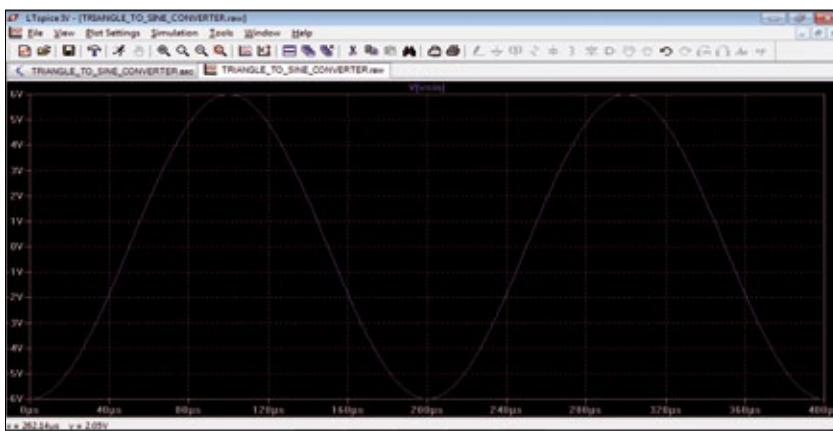


Fig. 3

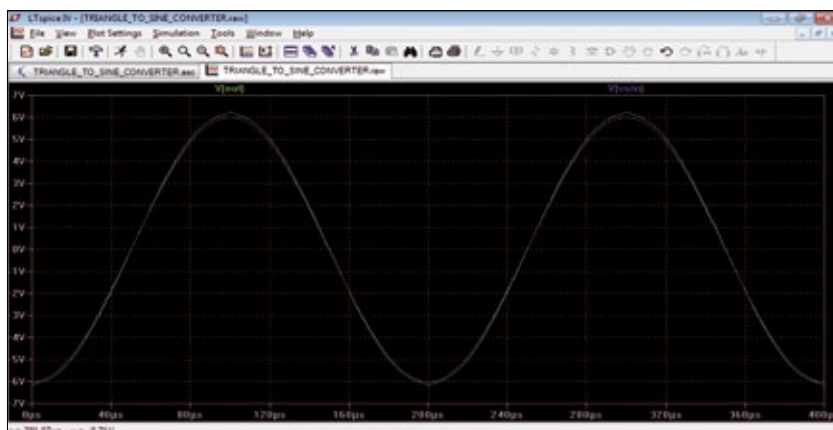


Fig. 4

dal costruttore nel data-sheet. L'utilizzo dell'alimentazione duale di ± 6 volt permette, inoltre, di non assorbire troppo dal myDAQ, dato che è questo dispositivo a fornire l'alimentazione all'intera scheda; ricordiamo, a proposito, che myDAQ può erogare correnti di piccola entità, dato che la potenza complessiva

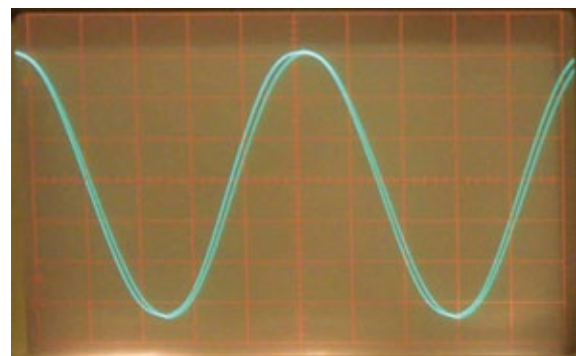
disponibile al modulo di alimentazione, alle uscite analogiche e digitali, è limitata complessivamente a 500mW, come indicato nel manuale fornito da National Instruments. La frequenza della forma d'onda in uscita è determinata dalla capacità del condensatore, tuttavia con un solo valore avremmo potuto coprire

un campo ristretto; ecco perché abbiamo previsto la possibilità di cambiare condensatore mediante un selettore, così da ottenere tre "portate", corrispondenti a C8, C9, C10. All'interno di esse, il potenziometro R25 ci permette di spaziare dalla frequenza più bassa a quella più alta. In serie ad R25 è stata posta R1 (3.9K) per assicurare di restare all'interno del range di resistenza di **Fig. 2** anche nel caso in cui il potenziometro sia nella posizione di minima resistenza. Anche se la resistenza ammessa per il potenziometro R25 va ben oltre i 100 kohm utilizzati, questo valore è stato scelto per permettere una facile regolazione della frequenza in tutto il range ammesso, in modo da poter apprezzare variazioni in uscita adeguate all'escursione del potenziometro. La selezione della frequenza della forma d'onda viene effettuata oltre che da R25, dalla posizione del selettore "RANGE", che commutando i condensatori C8, C9, C10, ci permette di ottenere una gamma che si estende tra 10 Hz e 25 kHz, così suddivisa:

- C8 fa da 10 a 250 Hz;
- C9 fa da 100 Hz a 2,5 kHz;
- C10 fa da 1 a 25 kHz.

Il diodo D1 ha la funzione di garantire che l'onda triangolare in uscita al pin 8 dell'integrato sia simmetrica rispetto alla massa, con un'ampiezza pari a metà del-

Fig. 5



la tensione di alimentazione (+/-3V). L'onda quadra, disponibile sul piedino 7, ha un'escursione pari alla tensione di alimentazione (+/-6V); R3, da 10 kohm, è la resistenza di pull-up dell'uscita, la quale è di tipo "open-collector". C3 blocca la componente continua che determinerebbe un offset dell'onda quadra acquisita dal myDAQ; serve perché quest'ultimo prevede solamente un accoppiamento DC del segnale. La sua capacità è stata scelta sufficientemente alta da rendere trascurabile l'attenuazione del segnale alla più bassa frequenza di funzionamento ammessa. A tal proposito si ricorda che l'impedenza di un condensatore ideale è data dalla formula:

$$z = \frac{1}{j\omega C}$$

quest'impedenza è una reattanza capacitiva. Nella formula, ω vale $2\pi f$, dunque il modulo dell'impedenza capacitiva è pari a:

$$z = \frac{1}{2\pi f C}$$

Da questa formula vediamo che a 10 Hz, un condensatore di capacità pari a 22 μF presenta un'impedenza di circa 14 m Ω , quindi del tutto trascurabile.

Lo stadio U2a del doppio amplificatore operazionale a basso rumore NE5532, in configurazione non invertente con guadagno pari a 2, ha la funzione di equalizzare il livello delle uscite di U1, in modo da avere la stessa dinamica per entrambe le forme d'onda. La parte più interessante dal punto di vista circuitale, è quella inerente all'amplificatore differenziale, costituito da T1 e T2 e dalle relative resistenze i cui valori sono stati determinati attraverso le simulazioni al computer. U2b ser-

Fig. 6



Fig. 7

Fig. 8

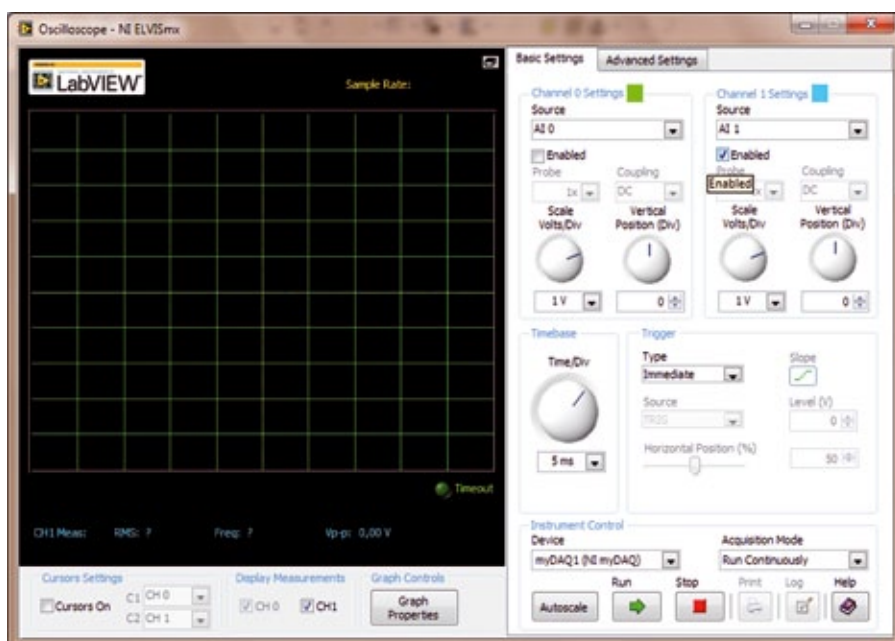
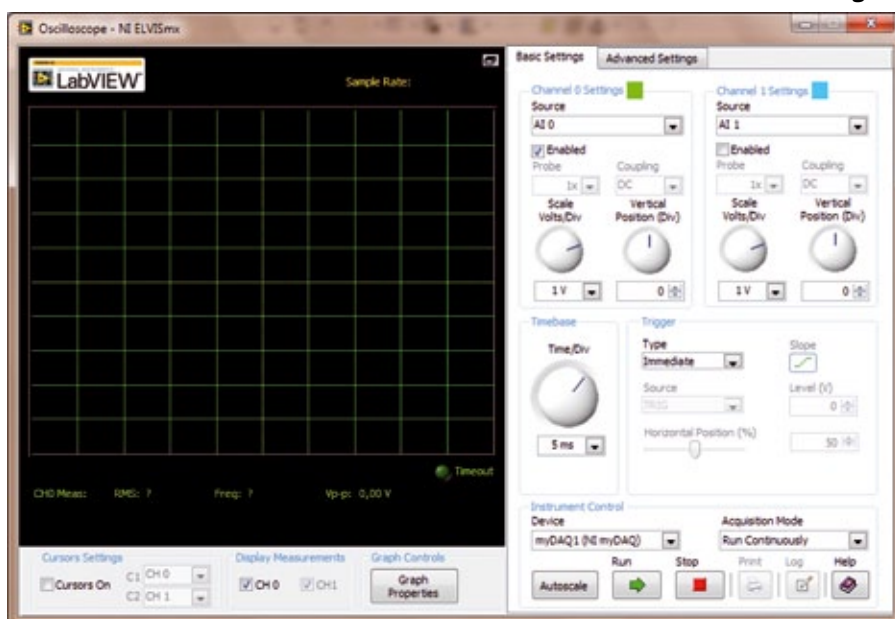
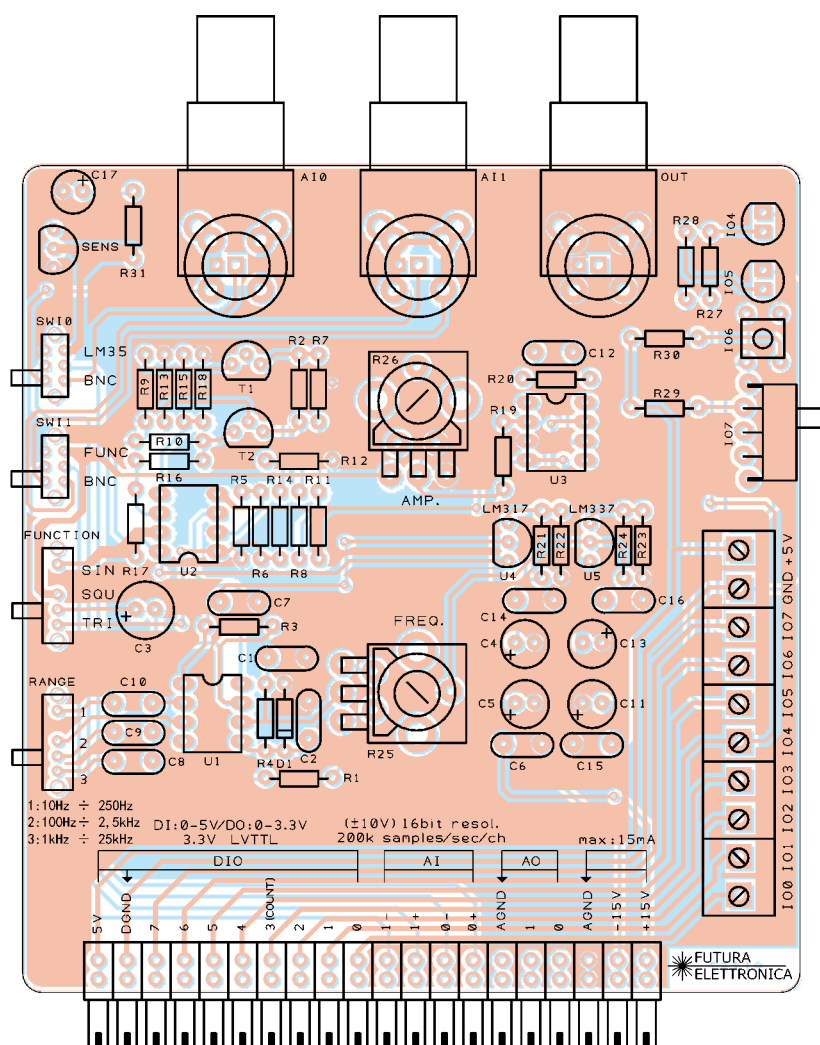


Fig. 9



Elenco Componenti:

R1: 3,9 kohm
 R2: 390 ohm
 R3÷R6: 10 kohm
 R7, R8: 47 kohm
 R9, R10: 10 kohm
 R11: 18 kohm
 R12, R13: 1,8 kohm
 R14: 56 kohm
 R15, R16: 1 kohm
 R17, R18: 20 kohm 1%
 R19, R20: 10 kohm
 R21: 270 ohm
 R22, R23: 1 kohm
 R24: 270 ohm
 R25: Potenzimetro 1 Mohm lineare (Mouser: 652-PDB12-M4251105BF)
 R26: Potenzimetro 100 kohm lineare (Mouser: 652-PDB12-H4251104BF)
 R27, R28: 470 ohm
 R29, R30: 4,7 kohm
 R31: -

C1, C2: 1 μ F 63 VL poliestere
 C3: 220 μ F 35 VL elettrolitico
 C4: 47 μ F 63 VL elettrolitico
 C5: 100 μ F 25 VL elettrolitico
 C6: 100 μ F 63 VL poliestere
 C7, C8: 1 μ F 63 VL poliestere
 C9: 100 nF 63 VL poliestere
 C10: 10 nF 63 VL poliestere
 C11: 47 μ F 63 VL elettrolitico
 C12: 22 pF ceramico
 C13: 100 μ F 25 VL elettrolitico
 C14÷C16: 100 nF 63 VL poliestere
 C17: -

ve ad amplificare l'onda pseudo-sinusoidale in uscita dallo stadio differenziale, che effettua la conversione dell'onda triangolare in sinusoidale, come dettagliato nel WIKI di Analog Devices (punto [3] dei Riferimenti). Per chi vuole approfondire, in rete si trovano molti articoli teorici che spiegano nel dettaglio la stessa tecnica. Lo stadio differenziale è stato simulato attraverso LT Spice IV della Linear Technology, già presentato sul n° 166 di Elettronica In. La Fig. 2 mostra l'onda pseudo-sinusoidale in uscita, mentre la Fig. 3 illustra un'

onda sinusoidale "pura"; in Fig. 4 è possibile vedere le due forme d'onda sovrapposte: sono praticamente identiche! La Fig. 5 mostra la sovrapposizione di un'onda sinusoidale con l'onda pseudo-sinusoidale del nostro generatore, sullo schermo di un oscilloscopio, leggermente sfasate per essere meglio distinguibili: la qualità dell'onda sinusoidale del nostro generatore è sufficientemente alta per i nostri scopi. Ci sono vari modi per generare un'onda sinusoidale; considerando lo scopo del circuito proposto, la soluzione adottata ci è sem-

brata adeguata, in quanto ci ha permesso di ottenere tre forme d'onda isofrequenziali e in fase, disponibili in uscita semplicemente spostando un commutatore a slitta, il cui morsetto comune "FUN" (funzione) è connesso all'ultimo blocco del circuito, lo stadio d'uscita, costituito dall'amplificatore operazionale U5 (TL071), in configurazione invertente, il cui scopo è quello di variare con continuità l'ampiezza del segnale in uscita tramite il potenziometro R26, disaccoppiando al contempo gli stadi precedenti dall'uscita.

D1: 1N4148

U1: XR2209

U2: NE5532N

U3: TL081

U4: LM317LZ (Mouser: 512-LM-317LZ)

U5: LM337LZ (Mouser: 926-LM-337LZ/NOPB)

IO4: LED 5 mm verde

IO5: LED 5 mm rosso

IO6: Microswitch

IO7: Deviatore a slitta 90°

T1, T2: BC547

SENS: LM35

SWI0: Deviatore Doppio 2 vie (Mouser: 611-OS202011MA0QN1)

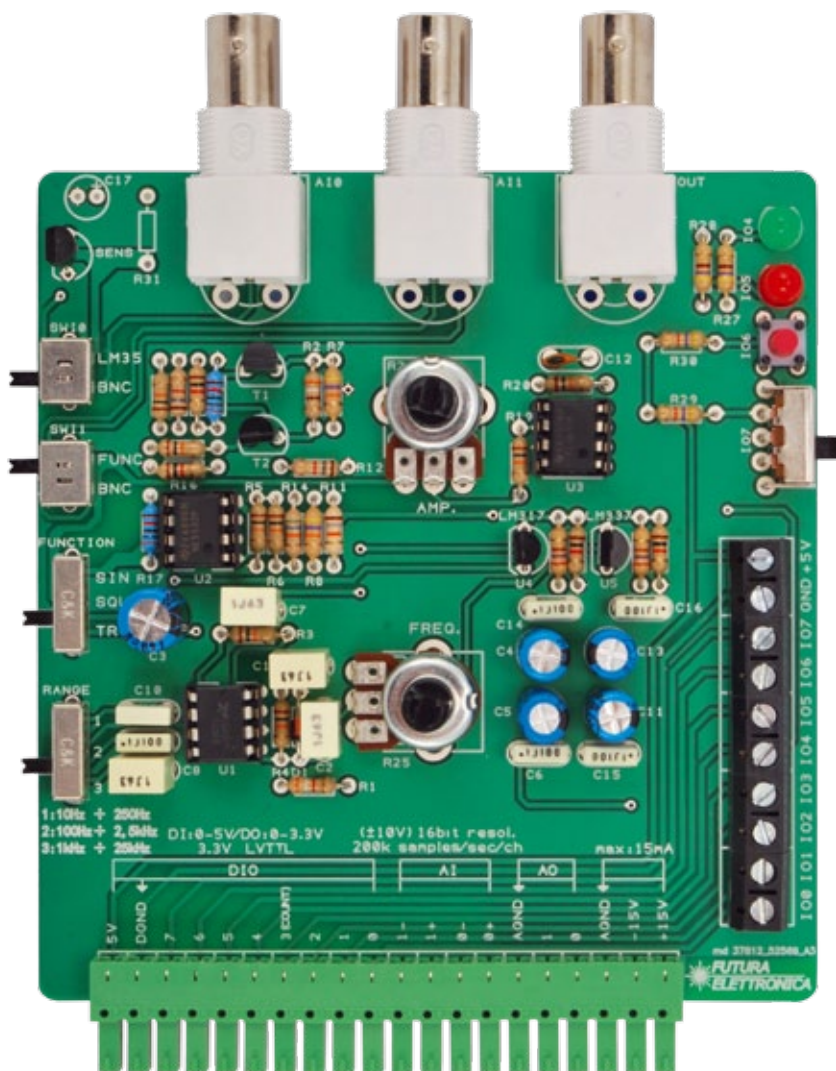
SWI1: Deviatore Doppio 2 vie (Mouser: 611-OS202011MA0QN1)

FUNCTION: Deviatore 3 vie (Mouser: 611-OS103011MA7QP1)

RANGE: Deviatore 3 vie (Mouser: 611-OS103011MA7QP1)

Varie:

- Zoccolo 4+4 (3 pz.)
- Morsetto 2 poli (5 pz.)
- Connettore BNC 90° da CS (3 pz.)
- Connettore per myDAQ
- Manopole per potenziometro (2 pz.)
- Circuito stampato S1057



APPLICAZIONI CON LABVIEW

Quanti di voi sono già in possesso del myDAQ, sicuramente conosceranno LabVIEW e i suoi tool. Per quanti invece non sono particolarmente esperti, diciamo che per utilizzare il nostro hardware bisogna innanzitutto innestare la scheda in myDAQ, quindi collegare quest'ultimo al computer tramite un cavo USB. Avviato il PC, bisogna accedere all'elenco dei programmi installati (la procedura dipende dal sistema operativo), cercare "National Instruments", selezionare "NI ELVISmx for NI ELVIS

& NI myDAQ", poi NI ELVISmx Instrument Launcher che farà apparire la relativa finestra (Fig. 6). Per visualizzare le forme d'onda generate dalla nostra scheda, selezioniamo "Scope" (Fig. 7). L'oscilloscopio si avvierà con le impostazioni predefinite (Fig. 8). L'uscita del generatore di funzioni è collegata all'ingresso analogico AI 1 del myDAQ, quindi, nella scheda *Basic Settings*, nel riquadro *Channel 0 settings*, togliamo il segno di spunta dalla casella "AI 0" "Enabled" e avviamo la visualizzazione dei dati acquisiti da AI 1 (Fig. 9). Per una rappre-

sentazione più stabile delle forme d'onda, nel riquadro *Trigger* della stessa scheda *Basic Settings* apriamo il menu a tendina e scegliamo "Edge" invece di "Immediate" (Fig. 10) e iniziamo l'acquisizione facendo clic su "Run" (Fig. 11). Come in un vero oscilloscopio, possiamo selezionare i volt/div (ossia il guadagno dello stadio d'ingresso) e impostare a piacimento la base dei tempi, con una dinamica più ristretta rispetto ad uno strumento professionale ma sufficiente per i nostri requisiti di natura prettamente didattica. Anche NI myDAQ dimostra come la

Fig. 10

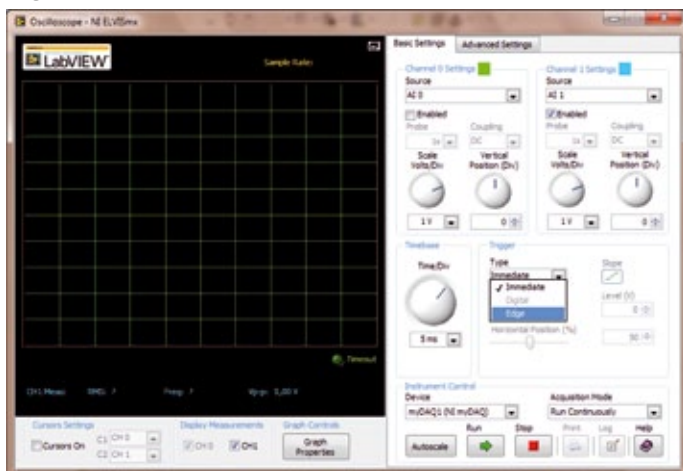


Fig. 11

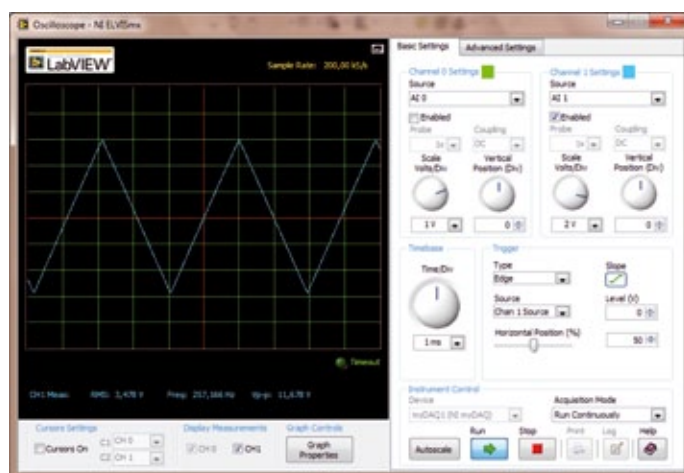
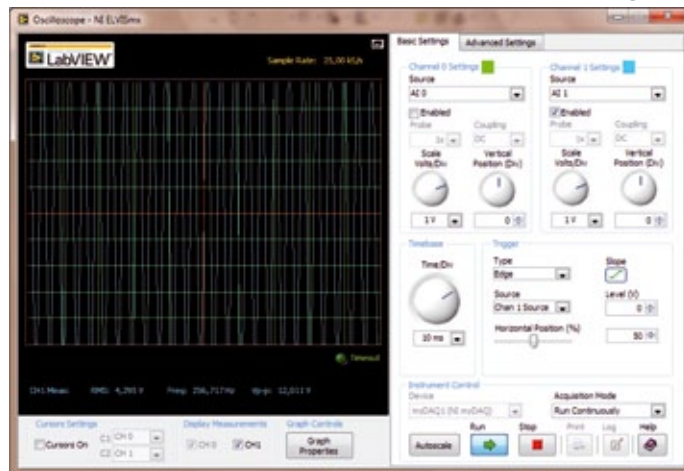


Fig. 12

Fig. 13

qualità della pseudo-sinusoide è veramente buona: la Fig. 12 e la Fig. 13 mostrano rispettivamente l'onda quadra e la triangolare. Come abbiamo visto, il pannello

"Instrument Launcher" contiene numerosi strumenti disponibili con myDAQ: vediamo insieme. Il primo a sinistra (DMM) è un multimetro digitale, in grado

di effettuare misure di tensione, corrente, resistenza, continuità e prova dei diodi (Fig. 14), tramite le relative icone, facilmente riconoscibili.

Fig. 14



Fig. 15



Fig. 16



Fig. 17

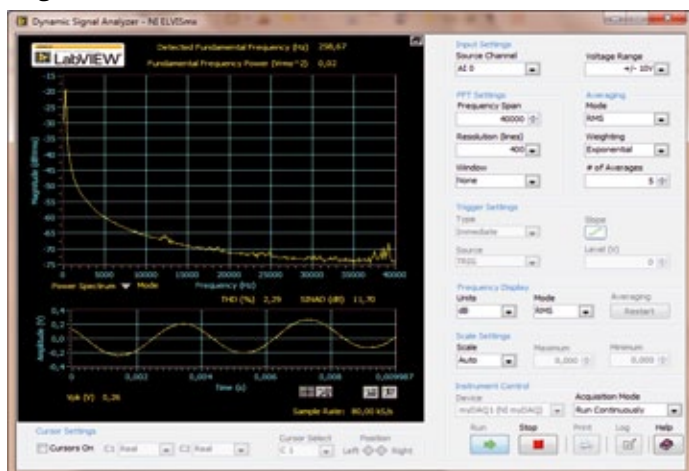


Fig. 18

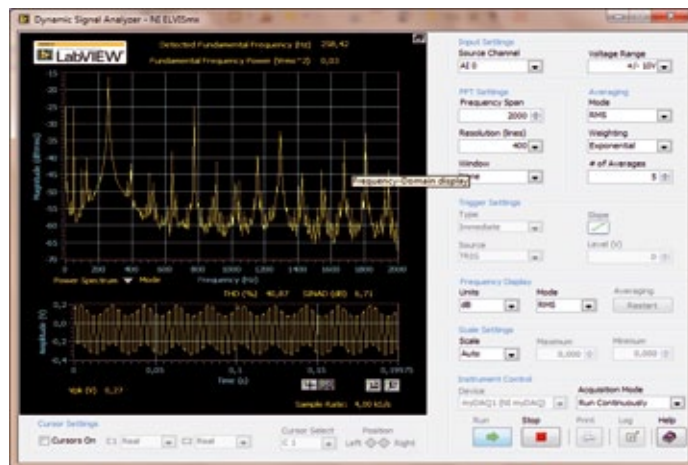
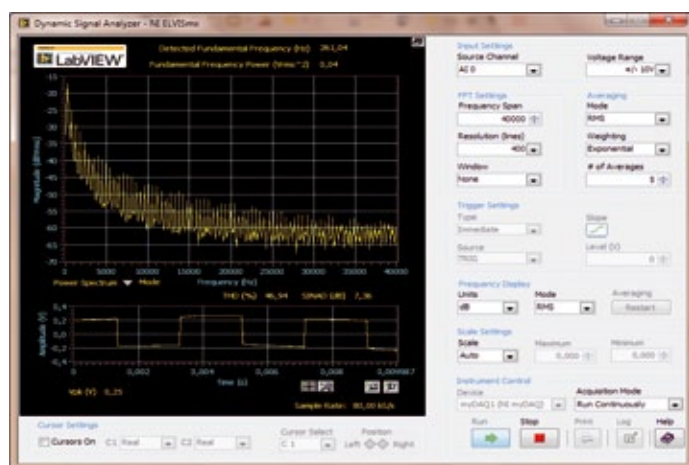


Fig. 19

Fig. 20

A destra di "Scope" c'è "FGEN", che è il generatore di funzioni del myDAQ (Fig. 15); anche in questo caso i comandi sono veramente intuitivi. Le capacità di visualizzazione non si fermano a quanto finora visto, infatti è possibile anche realizzare diagrammi di Bode tramite il relativo strumento con lo stesso nome (Fig. 16). Come visibile, tra i parametri più importanti selezionabili, ci sono il range di frequenza e la possibilità di scegliere tra una scala logaritmica o lineare per l'ampiezza. Di lato a questo strumento, c'è l'icona di "DSA" (Dynamic Signal Analyzer), che ci consente di visualizzare i segnali nel dominio della frequenza in modo da verificarne lo spettro (Fig. 17). Come possiamo vedere, lo spettro della pseudo-sinusoide

Fig. 21

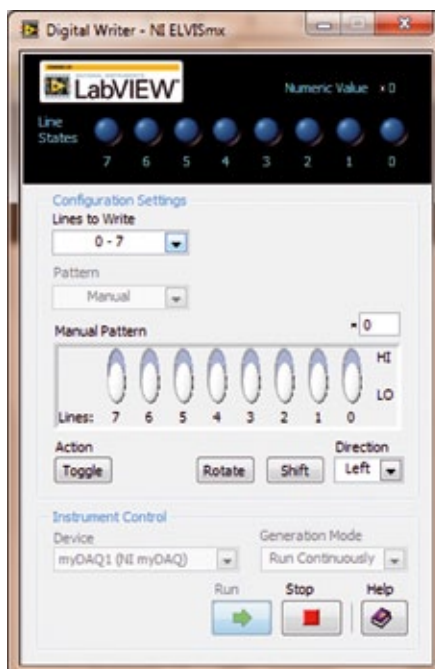
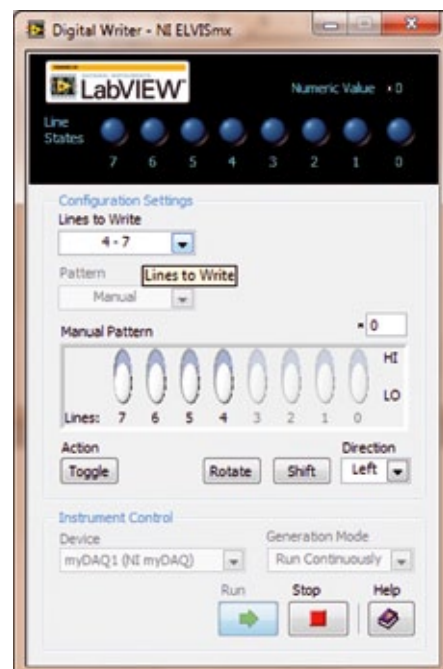


Fig. 22



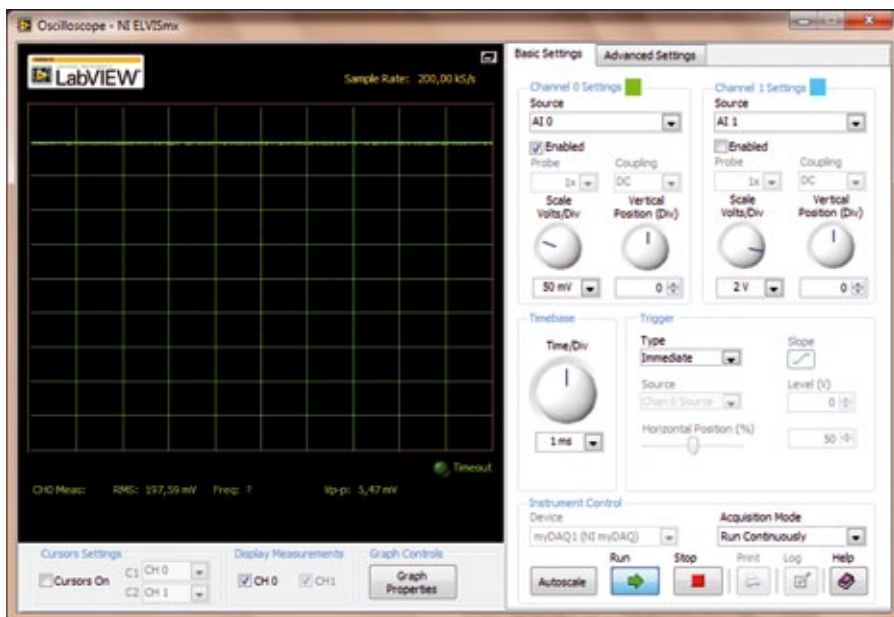


Fig. 23

è relativamente “puro”, come evidenziato anche dal dettaglio di **Fig. 18**, ottenuto restringendo il range di misura, tramite “Frequency Span”. La **Fig. 19** e la **Fig. 20** mostrano invece, nell’ordine, lo spettro dell’onda quadra e quello dell’onda triangolare. “ARB” è, invece, un generatore di funzioni arbitrarie, che permette di generare segnali non convenzionali con le caratteristiche desiderate dall’utente e quindi anche segnali misti. Oltre a tutti gli strumenti analogici visti, il myDAQ offre anche la possibilità di prendere confidenza con i segnali digitali: DigIn acquisisce e visualizza il livello logico degli ingressi digitali del

myDAQ mentre invece DigOut permette di modificare i relativi livelli logici delle uscite digitali della scheda. Per impostazione predefinita sono selezionate le sole uscite comprese tra la 0 e la 3; agendo sul menu a tendina “Lines to Write” è possibile variare tale intervallo ed estenderlo anche a tutte le 8 uscite se necessario (**Fig. 21** e **Fig. 22**). Osservando lo schema elettrico della nostra scheda, notiamo che sulle uscite digitali 4 e 5 (IO4 e IO5) sono connessi due LED: agiamo quindi sui relativi controlli per accenderli. Come visibile, selezionando unicamente le uscite 4-7, il relativo pattern corrisponde al numero binario “11”, relativo alle uscite



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità ed il master del circuito stampato può essere scaricato gratuitamente dal sito della rivista (www.elettronicaonline.it). Questo progetto è anche disponibile in scatola di montaggio; il kit (cod. FT1057K, Euro 67,00 IVA compresa) comprende la basetta forata e serigrafata, tutti i componenti e le minuterie.

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

IO4 e IO5 a livello alto (LED IO4 e IO5 accesi), che convertito in esadecimale corrisponde a “3”, come correttamente indicato nel riquadro in alto a destra, appena sopra i controlli delle uscite “Manual Pattern”. Se invece definiamo l’intero intervallo di uscite digitali (“Lines to Write 0-7”), allora il pattern corrisponde alla cifra binaria 00110000, cioè 30 in base 16, come riportato dal relativo riquadro dell’interfaccia grafica.

Prima di concludere, ritorniamo un attimo allo schema elettrico ed al sensore di temperatura LM35 che può essere direttamente collegato ad AIO tramite il deviatore a due posizioni SWIO. Torniamo all’oscilloscopio, dopo aver impostato SWIO come detto. La caratteristica di trasduzione degli LM35 è lineare, pari a 10mV/°C (punto [4] dei Riferimenti); dopo aver correttamente impostato i parametri dell’oscilloscopio, tra cui la selezione di AI0 come sorgente, potremo leggere la tensione in uscita al sensore e quindi la temperatura misurata, nel caso specifico 19,7 °C (**Fig. 23**).

RIFERIMENTI:

- [1] <http://www.exar.com/communications/timing/voltage-controlled-oscillator-vco/xr2209>
- [2] <http://www.exar.com/common/content/document.ashx?id=179&languageid=1033>
- [3] <https://wiki.analog.com/university/courses/electronics/electronics-lab-12sg>
- [4] <http://www.ti.com/lit/gpn/lm35>

Concludiamo il nostro viaggio alla scoperta di Flowcode, l'innovativo sistema di sviluppo grafico per microcontrollori che consente di scrivere il codice facendo uso di oggetti grafici. In questa ottava ed ultima puntata spiegheremo come connettere all'ethernet sistemi embedded mediante protocollo TCP/IP.

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

Nella scorsa puntata abbiamo introdotto la periferica wireless che consente la comunicazione secondo lo standard ZigBee, ed illustrato la relativa implementazione in flowcode. Ora, a conclusione del corso, parleremo di come realizzare sistemi embedded connessi alla rete ethernet, in grado di gestire protocolli complessi come quello TCP/IP. Questo tema ci apre le porte alla gestione e alla realizzazione di sistemi domotici che oggi tendono ad essere sempre più connessi alla rete globale del world wide web. Inizieremo il nostro percorso parlando del protocollo TCP/IP e al termine della puntata realizzeremo in pochi semplici

passi un Web Server integrato in grado di inviare dati dinamici sulla rete.

Sviluppo di applicazioni Internet con Flowcode

Gestire sistemi integrati connessi alla rete Ethernet è molto complesso ed implica una profonda conoscenza di molteplici protocolli di comunicazione, ma Flowcode ci mette a disposizione i componenti necessari per risolvere questo problema in modo semplice e intuitivo. Mediante i componenti TCP/IP, UDP e WebServer possiamo realizzare praticamente qualsiasi applicazione che contempli una comunicazione ethernet senza costringerci ad approfondir-

ne tutte le problematiche. I componenti in libreria ci permettono di gestire l'integrato W5100, grazie al quale ci si può connettere direttamente alla Ethernet. Come punto di partenza iniziamo ad esaminare il protocollo TCP/IP.

Il protocollo TCP/IP

TCP/IP è l'acronimo di "Transmission Control Protocol / Internet Protocol"; il protocollo corrispondente racchiude tutte le specifiche necessarie per comunicare correttamente attraverso la rete Internet. È suddiviso in strati detti "Layer" (Fig. 1). Ciascuno dei layer dipende da quello che sta al di sotto e può comunicare solamente con quelli confinanti; ognuno ha un compito differente e i pacchetti, durante il transito verso gli strati inferiori, vengono sempre più incapsulati e, viceversa, man mano che risalgono vengono ripuliti dai dati di incapsulamento non più necessari. Questo tipo di architettura permette di poter trasferire in modo "sicuro" i dati tra un calcolatore ed un altro. Il protocollo IP si occupa della corretta ricezione/trasmisione di dati; attualmente esistono due versioni di IP: IPv4 e IPv6. La differenza principale tra i due è il numero di byte utilizzati per codificare in modo univoco gli indirizzi di rete. Il protocollo TCP è invece incaricato del riordino dei pacchetti e della ricomposizione del dato originale senza perdita di dati. Ogni computer è identificato da un indirizzo IP che lo identifica in modo univoco all'interno della rete internet. L'assegnazione di questo indirizzo può essere di tipo dinamico oppure statico; solitamente possiamo decidere se assegnarne uno di un tipo oppure di un altro perché, essendo collegati ad un ISP (Internet Service Provider), l'indirizzo effettivo visto dalla rete sarà quello dell'ISP e non quello scelto da noi. Quando vogliamo raggiungere un calcolatore collegato a Internet mediante un browser, digitiamo un nome, ad esempio *www.futuranet.it*, e questo, mediante il servizio di DNS, verrà trasformato nel corrispondente indirizzo IP utilizzabile con il protocollo TCP/IP. Questo processo permette così di utilizzare dei nomi che sono più facili da ricordare rispetto ad una sequenza di 4 o 6 numeri a tre cifre.

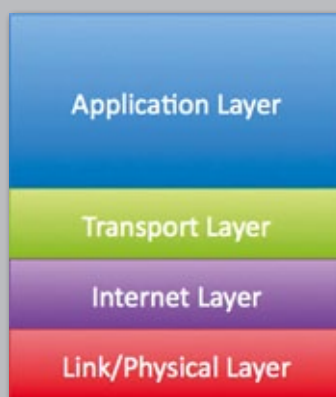


Fig. 1
Suddivisione
in Layer del
protocollo
TCP/IP.

Dato che l'indirizzo IP può cambiare in modo dinamico, è necessario basarsi su un qualcosa di fisso ed immutabile che possa identificare un sistema connesso sulla rete. Per questa ragione è utilizzato l'indirizzo MAC: esso è univoco per ogni scheda in grado di collegarsi alla rete ed è perciò fissato in fase di produzione della scheda ed assegnato in modo automatico a fine linea. Tornando al livello più alto, possiamo trovare un ulteriore livello di astrazione: le cosiddette "porte" di comunicazione. Ogni porta è adibita ad un particolare tipo di protocollo ma è possibile utilizzarle anche in modo differente; quella più nota e comunemente utilizzata è la 80, usata per la comunicazione *http*, ma ve ne sono altre come la 20,21 per l'FTP, la 53 per il DNS e molte altre ancora.

Con queste nozioni basilari possiamo ora accingerci ad esaminare il componente che Flowcode ci mette a disposizione per gestire il protocollo TCP/IP, UDP e MAC.

Il componente TCP/IP per Flowcode

Nella barra dei componenti nel menu "Periferiche" troviamo il componente TCP/IP che ci permette di gestire i protocolli TCP/IP, UDP e MAC. Per poterlo utilizzare è sufficiente inserirlo nel pannello e successivamente procedere alla configurazione. In Fig. 2 è mostrata l'unica pagina di configurazione delle proprietà del componente. Innanzitutto dobbiamo scegliere quale versione della scheda EB023 abbiamo a disposizione; troviamo facilmente questa indicazione sulla scheda che abbiamo acquistato (se stiamo realizzando una scheda custom, la deduciamo dal componente utilizzato). È ora necessario inserire le informazioni riguardanti la connessione di rete.

Tabella 1 - Macro utilizzabili con il componente TCP/IP

Nome macro	Descrizione
Initialize	Resetta ed inizializza la periferica Ethernet.
Create_MAC_Socket	Chiamando questa macro, viene creato un socket in grado di ricevere i dati trasmessi sulla rete ethernet. Impostando ad uno il valore del parametro: - Promiscuous mode per leggere tutto il traffico circolante sulla rete senza tener conto dell'indirizzo MAC; - Broadcast mode per leggere anche i pacchetti di tipo Broadcast; - Error mode per ricevere anche i pacchetti che contengono errori. Il valore di ritorno, se diverso da zero, indica che l'operazione di configurazione è andata a buon fine.
Create_UDP_Socket	La macro crea un socket per la ricetrasmisione di dati mediante il protocollo UDP. È necessario indicare come parametri: - Channel, ossia il canale scelto per la comunicazione (da 0 a 3); è possibile utilizzare contemporaneamente canali configurati in modo differente fin ad un massimo di quattro; - l'indirizzo della porta sorgente suddiviso in src_port_hi e src_port_lo. Il valore di ritorno, se diverso da zero, indica che l'operazione di configurazione è andata a buon fine.
Create_IP_Socket	Crea un socket per la ricetrasmisione di dati mediante il protocollo IP. È necessario indicare il canale scelto (channel) e il protocollo che si intende realizzare mediante il set dei due parametri interni alla macro. Il valore di ritorno, se diverso da zero, indica che l'operazione di configurazione è andata a buon fine.
Create_TCP_Socket	Crea un socket di tipo TCP. Necessita delle stesse configurazioni utilizzate con la macro Create_UDP_Socket.
Set_Destination	È utilizzata con socket di tipo IP e UDP e serve per impostare l'indirizzo IP e la porta verso il quale si vogliono trasmettere i dati. Richiede come parametri il canale, l'indirizzo IP e la porta di destinazione. Per connessioni di tipo IP non è necessario indicare la porta.
Set_My_IP	Questa macro è utilizzata per modificare, durante l'esecuzione, l'indirizzo IP del dispositivo; non è necessaria se si desidera mantenere l'indirizzo IP inserito nella finestra di configurazione del componente.
GetSocketStatus	La macro restituisce lo stato attuale del socket. I valori possibili sono diciannove, ma quelli principali sono i seguenti: - 0 (SOCK_CLOSED) = il socket è chiuso (non sono né trasmessi né ricevuti dati); - 6 (SOCK_ESTABLISHED) = il socket è pronto e la comunicazione è attiva in modalità passiva; - 7 (SOCK_CLOSE_WAIT) = il socket è in fase di chiusura; - 14 (SOCK_INIT) = il socket è in fase di inizializzazione.
TCP_Listen	Imposta il canale scelto (indicato come parametro) in modalità passiva in modo che possa essere pronto a ricevere dei dati.
TCP_Connect	Imposta il canale scelto (indicato come parametro) in modalità attiva in modo che possa essere pronto per trasmettere dei dati sul canale di comunicazione. Necessita come parametri il valore del canale, l'indirizzo IP e la porta di destinazione. Se il valore restituito è differente da zero vuol dire che la comunicazione è stata stabilita con successo e che possono quindi essere trasmessi dei dati.
TCP_Close	Chiude la connessione TCP mediante un'operazione attiva e porta il socket in attesa di terminazione.
Tx_start	Avvia la trasmissione di dati sul canale specificato dai parametri ma, per funzionare correttamente necessita delle macro successive (Tx_sendbyte, Tx_sendmymac, Tx_sendmymac e Tx_end). La sequenza corretta inizia con un comando di Tx_start, prosegue con sequenze di Tx_sendbyte, Tx_sendmymac, Tx_sendmymac nell'ordine prescelto (può anche solo essere utilizzata una sola macro delle tre) e termina con Tx_end per effettuare l'invio dei dati sul canale di comunicazione.

(segue)

In base alla tipologia di rete impostiamo l'indirizzo IP, che deve essere di tipo statico, e la Subnet Mask. Il gateway Address deve essere impostato uguale all'indirizzo del "router" o del modem con il quale si accede a Internet, almeno se vogliamo inviare pacchetti al di fuori della rete LAN; qualora questo indirizzo fosse sbagliato, non sarebbe possibile comunicare con computer non direttamente collegati a Intranet. Il settaggio dell'indirizzo MAC (Hardware Address) è la parte più delicata: tale valore dovrebbe essere univoco sull'intera rete Internet. Per fare delle prove all'interno della rete LAN possiamo impostarlo ad un valore qualsiasi: basta che differisca da quello di tutte le altre schede di rete wireless ed ethernet affacciate su di essa. Per utilizzarlo su Internet bisogna essere certi di assegnare un indirizzo che non corrisponda ad alcuna periferica ethernet affacciata al momento su Internet nel

mondo; il modo migliore per fare ciò è prendere l'indirizzo di una scheda di rete attualmente non utilizzata, così si è certi che il relativo IP non sia attivo.

Ulteriore passo è la configurazione delle connessioni; per poter funzionare correttamente è necessario impostare i pin sul quale vengono mappati il piedino di interrupt ed il c.s. Questa configurazione varierà in base a come colleghiamo la scheda alla nostra piattaforma oppure su come realizziamo il PCB in caso di progetti Custom. Passiamo ora ad esaminare le macro che Flowcode ci mette a disposizione per gestire le comunicazioni su rete ethernet. Nella **Tabella 1** sono descritte le principali. Queste macro permettono di gestire in modo molto semplice comunicazioni sulla rete ethernet utilizzando socket a differenti livelli: MAC, UDP e TCP/IP.

Il componente TCP/IP è quello che ci permette di gestire i principali protocol-

Tabella 1 - Macro utilizzabili con il componente TCP/IP (seguito)

Nome macro	Descrizione
Tx_sendbyte	Invia i dati al buffer di trasmissione per il canale specifico. Oltre al canale deve essere indicato il parametro di dato. Come dato possono anche essere utilizzate stringhe.
Tx_sendmymac	Invia nel buffer di trasmissione, del canale selezionato, l'indirizzo MAC attuale della scheda.
Tx_sendmyip	Invia nel buffer di trasmissione, del canale selezionato, l'indirizzo IP impostato nella finestra di configurazione della scheda.
Tx_end	È la macro che permette di inviare fisicamente i dati sulla linea di comunicazione.
Rx_data_available	Controlla se vi sono dati disponibili nel buffer di ricezione del canale. Il valore di ritorno, se differente da zero, indica che vi sono dati disponibili. Questa macro è anche utilizzata per inizializzare il buffer per la ricezione dei dati qualora non lo fosse già. In pratica azzerà l'indice con il quale viene scandito il buffer con la macro Rx_data_size .
Rx_data_size	Macro utilizzata per sapere quanti byte sono disponibili all'interno del buffer di ricezione.
Rx_readheader	Macro utilizzata per leggere l' header dei dati ricevuti. Richiede come parametri di ingresso il canale di comunicazione e l'indice idx del byte che si vuole leggere. In base al tipo di comunicazione utilizzata sul canale scelto si possono avere le informazioni seguenti. - UDP mode • Idx 0 - 1: Dimensione del pacchetto (MSB idx=0); • Idx 2 - 5: Indirizzo IP sorgente (MSB idx=2); • Idx 6 - 7: indirizzo della porta (MSB idx=7). - IP mode • Header size = 6; • Idx 0 - 1: Dimensione del pacchetto (MSB idx=0); • Idx 2 - 5: Indirizzo IP sorgente (MSB idx=2). - MAC mode • Header size = 3; • Idx 0 - 1: Dimensione del pacchetto (MSB idx=0); • Idx 2: Stato del pacchetto ricevuto: ■ Bit 7 = 1 indica che il MAC di destinazione equivale all'indirizzo della scheda; ■ Bit 5 = 1 indica che l'indirizzo è di tipo broadcast; ■ Bit 4 = 1 indica che il pacchetto contiene degli errori.
Rx_readbyte	Legge dal buffer di ricezione del canale scelto un byte che viene restituito come valore di ritorno.
Rx_skipbytes	Permette di "saltare" dei dati dal buffer di ricezione.
Rx_match2bytes Rx_match4bytes Rx_match6bytes	Legge 2/4/6 byte dal buffer di ricezione e li confronta con quelle indicate nei parametri con la quale viene chiamata. Restituisce un valore differente da zero se i valori coincidono.
Rx_match_mymac	Legge 6 byte dal buffer di ricezione e li confronta con l'indirizzo MAC con il quale è stata configurata la scheda. Restituisce un valore differente da zero se i valori coincidono.
Rx_match_myip	Legge 4 byte dal buffer di ricezione e li confronta con l'indirizzo IP con il quale è stata configurata la scheda. Restituisce un valore differente da zero se i valori coincidono.

li di comunicazione su rete ethernet, tuttavia per realizzare un Web Server in modo ancora più semplice abbiamo a disposizione un componente specifico; tale componente, a patto di sottostare ad alcune limitazioni, ci permette di realizzare il Web Server in modo ancora più semplice e intuitivo.

Il componente Webserver

La maniera più immediata per poter comunicare con sistemi remoti, anche a migliaia di chilometri di distanza, è quello di utilizzare semplici pagine web, perché necessitano unicamente che entrambi i sistemi siano connessi alla rete Internet. L'intero sistema è veramente economico ed è molto intuitivo da utilizzare. Realizzare un Web Server su un calcolatore è abbastanza semplice, ma su una piattaforma embedded con potenza di calcolo limitata diventa assai più complesso. Per queste ragioni, Flowcode ci viene in soccorso fornendoci un componente specifico che troviamo nella barra dei componenti nel menu "Periferiche" con il nome *Webserver*. Nelle puntate precedenti abbiamo visto come periferiche o sistemi molto complessi possano essere realizzati in modo molto semplice grazie all'utilizzo del programma; è così anche in questo caso. Come



Fig. 2 - Finestra delle proprietà del componente TCP/IP.

Tabella 2 - Macro utilizzabili con il componente WebServer

Nome macro	Descrizione
Initialize	Resetta ed inizializza la periferica Ethernet. Questa macro deve essere chiamata prima di ogni qualsiasi altra macro per la gestione del Web Server.
CreateServerSocket	La macro crea un socket per la ricetrasmisione di dati su Web Server. È necessario indicare come parametri: - Channel il canale scelto per la comunicazione (da 0 a 3); è possibile utilizzare contemporaneamente canali configurati in modo differente fin ad un massimo di quattro; - l'indirizzo della porta dal quale trasmettere e ricevere i dati suddiviso in src_port_hi e src_port_lo . Il valore di ritorno, se diverso da zero, indica che l'operazione di configurazione è andata a buon fine.
CheckSocketActivity	Richiede in ingresso il canale sul quale si vuole effettuare il controllo di attività. Deve periodicamente essere richiamata in modo da permettere la gestione delle richieste entranti.

vediamo in Fig. 3, la finestra di configurazione del componente è semplice e intuitiva: notiamo che è esattamente identica al componente TCP/IP e quindi può essere configurata allo stesso modo. Ciò è dovuto al fatto che il componente webserver non

remoto. Il codice deve essere compatibile con lo standard HTML, ma ciò non vieta di inserire comandi JavaScript o altri linguaggi interpretati dal browser remoto. La caratteristica fondamentale del componente *Webserver* è la possibilità, che offre, di inserire nel codice HTML il valore dinamico di alcune variabili; in questo modo è possibile monitorare da remoto le variabili volute (al massimo quattro). Allo scopo basta fare riferimento, all'interno del codice HTML, al nome della variabile, inserendolo tra %. I nomi delle variabili utilizzabili sono quindi B001, B002, B003 e B004. Vi sono alcune limitazioni sulla tipologia delle variabili scelte perché non possono essere di tipo stringa o virgola mobile. Nell'esempio pratico utilizzeremo questo semplice stratagemma allo scopo di realizzare una centralina per il monitoraggio della temperatura.

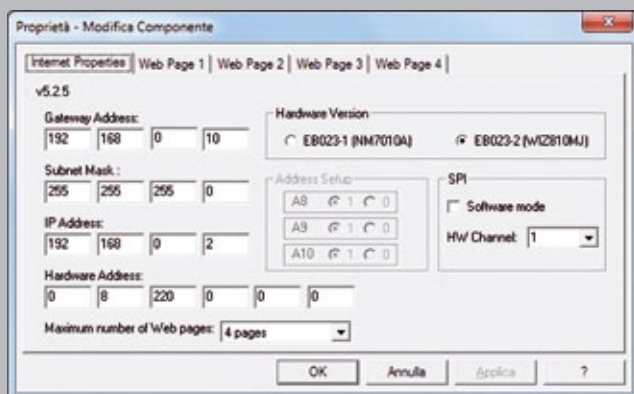


Fig. 3 - Proprietà del componente Webserver.

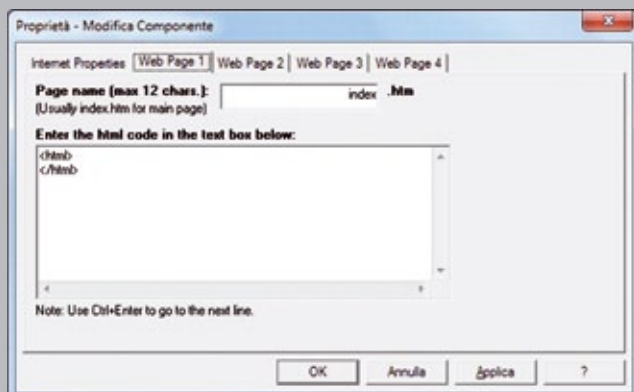


Fig. 4 - Proprietà del componente Webserver per la descrizione delle pagine HTML.

è altro che un gestore preconfigurato di pagine web basato su TCP/IP. Le differenze risiedono nei successivi tab di configurazione, mostrati in Fig. 4. In alto dobbiamo inserire il nome delle pagine, mentre nello spazio centrale è necessario editare il codice che intendiamo visualizzare sulla pagina richiesta da

Progetto Pratico: Centralina per il monitoraggio della temperatura mediante Web Server

Il Web Server integrato che realizzeremo ci permetterà di monitorare da remoto, tramite una connessione a Internet, le ultime quattro temperature misurate dal nostro sistema.

Per realizzare il progetto abbiamo bisogno dei seguenti componenti:

- EB006 "USB PICmicromultiprogrammer-board" (PIC18F4685 e quarzo 19,6608 MHz);
- EB023 "Internet board";
- EB016 "Prototype board";
- sensore di temperatura LM35.

Collegiamo le schede come mostrato in Fig. 5 posizionando la scheda EB023 sulla porta C e la scheda di prototipazione EB016 sulle porte A e B. Alimentiamo il sensore, come indicato nella seconda

Listato 1

```
<!DOCTYPE html>
<HTML>
<HEAD><title>Esempio web-server</title>

<script language=JavaScript type="text/javascript">

var Temperature = new Array();
Temperature[0] = %BO01% - 128;
Temperature[1] = %BO02% - 128;
Temperature[2] = %BO03% - 128;
Temperature[3] = %BO04% - 128;

function Temp(x) {
    if (x/20<=1) return Temperature[0];
    else if (x/20<=2) return Temperature[1];
    else if (x/20<=3) return Temperature[2];
    else return Temperature[3];
}

function funGraph (ctx,axes,func,color,thick) {
    var xx, yy, dx=1, x0=axes.x0, y0=axes.y0, scale=axes.scale;
    var iMax = Math.round((ctx.canvas.width-x0)/dx);
    var iMin = axes.doNegativeX ? Math.round(-x0/dx) : 0;
    ctx.beginPath();
    ctx.lineWidth = thick;
    ctx.strokeStyle = color;

    for (var i=iMin;i<=iMax;i++) {
        xx = dx*i;
        yy = scale*func(xx/scale);
        if (i==iMin) ctx.moveTo(x0+xx,y0-yy);
        else ctx.lineTo(x0+xx,y0-yy);
    }
    ctx.stroke();
}

// Disegna gli assi
function showAxes(ctx,axes) {
    var x0=axes.x0,w=ctx.canvas.width; scale=axes.scale;
    var y0=axes.y0,h=ctx.canvas.height;
    var xmin = axes.doNegativeX ? 0 : x0;
    var hline = 0;

    ctx.beginPath();
    ctx.strokeStyle = "rgb(128,128,128)";
    ctx.moveTo(xmin,y0); ctx.lineTo(w,y0); // X axis
    ctx.moveTo(x0,0); ctx.lineTo(x0,h); // Y axis

    ctx.stroke();
    ctx.beginPath();
    ctx.lineWidth = 1;
    ctx.strokeStyle = "rgb(128,128,128)";
    for (var i=-20;i<=50;i= i + 10) {
        yy = i*scale;
        ctx.moveTo(xmin-5,y0-yy); ctx.lineTo(w,y0-yy);
        ctx.font="15px Georgia";
        ctx.strokeText(i + "°C",xmin-50,y0-yy+5);
    }
}
```

(segue)

puntata per la realizzazione del termometro LCD, e colleghiamo l'uscita alla porta A piedino numero 2 in modo da stimolare il canale analogico An2 del microcontrollore. Configuriamo il progetto in modo da ottenere una frequenza di clock del PIC pari a 19.660.800 Hz, impostando i parametri come mostrato in Fig. 6. Ora che l'hardware è pronto possiamo

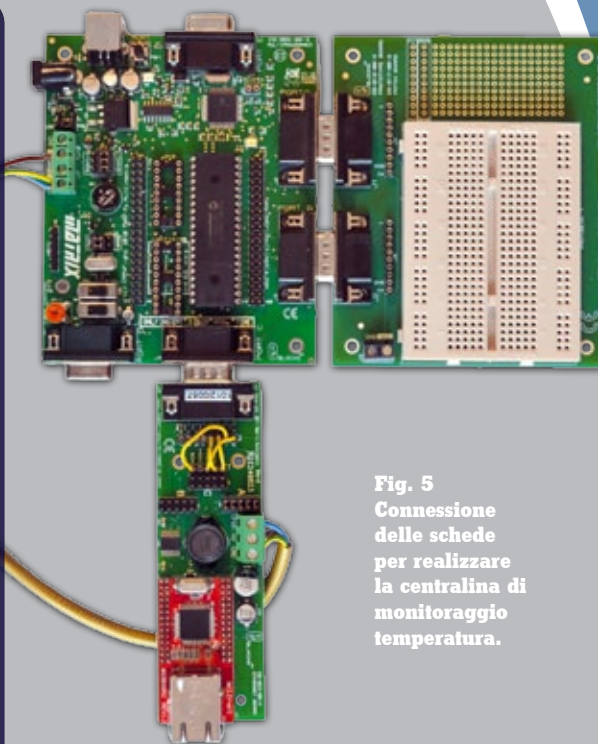


Fig. 5
Connessione
delle schede
per realizzare
la centralina di
monitoraggio
temperatura.

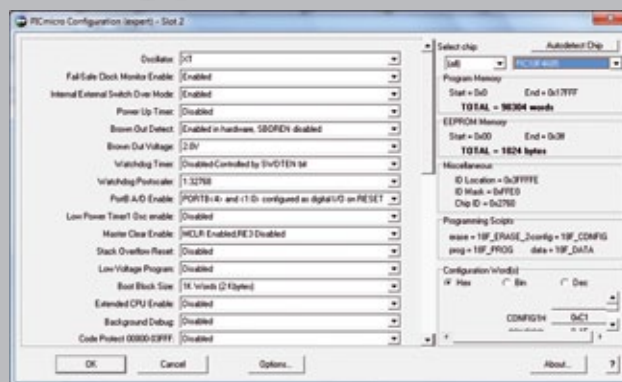


Fig. 6 - Configurazione del componente.

Tabella 3 - Dati di configurazione componente WebServer

GatewayAddress	192.168.0.10
SubnetMask	255.255.255.0
IP Address	192.168.0.5
HardwareAddress	0:8:220:0:0:0
Maximum number of WebPages	1

dedicarci alla stesura del diagramma di controllo. Inseriamo quindi dalla barra dei componenti il componente Webserver e il componente ADC configurato in modo che legga l'ingresso An2 collegato al sensore di temperatura LM35. Passiamo ora ad impostare il componente Webserver: indichiamo i seguenti valori all'interno dei parametri (Tabella 3). Il codice html che verrà visualizzato sulla

Listato 1 (seguito)

```

    }
    ctx.stroke();

    ctx.beginPath();
    ctx.lineWidth = 3;
    ctx.strokeStyle = "rgb(80,80,80)";
    yy = 0;
    ctx.moveTo(xmin,y0-yy); ctx.lineTo(w,y0-yy);
    ctx.stroke();
}

function draw() {
    var canvas = document.getElementById("canvas");
    if (null==canvas || !canvas.getContext) return;

    // Imposta l'area di disegno
    var axes={}, ctx=canvas.getContext("2d");
    axes.x0 = 50; // .5 + .5*canvas.width; // x0 pixels from left to x=0
    axes.y0 = 12/16*canvas.height; // .5 + .5*canvas.height; // y0 pixels from top to y=0
    axes.scale = 7; // 7 pixels from x=0 to x=1
    axes.doNegativeX = false

    showAxes(ctx,axes);
    funGraph(ctx,axes,Temp,"rgb(66,44,255)",4);
}

function clear(ctx) {
    ctx.fillStyle = "rgb(255,255,255)";
    ctx.fillRect(0, 0, 500, 500);
}
</script>

</HEAD>
<body onload="draw()">
<script>
</script>
<p><center>
<font face="Verdana" size="+2">
Temperature misurate<br><br>
<canvas id="canvas" width="600" height="400"></canvas>
<br><font face="Verdana" size="-2">Web Server by <br>
<a href="http://www.elettronica.in.it/">ElettronicaIN</a></font>
</center><p>
</body></HTML>

```

pagina deve essere indicato nel tab *Web page 1* con il titolo *index.htm*. Il testo del codice html che ci permette di visualizzare le ultime quattro temperature e plottarne il grafico è contenuto nel **Listato 1**; per poterlo utilizzare ci basterà inserirlo (mediante copia/incolla) all'interno della finestra di configurazione della pagina.

Dato che abbiamo da svolgere due task principali, possiamo dividere il nostro flusso di programma in due macro. La prima, il main in **Fig. 8**, si occuperà di inizializzare il sistema e controllare le richieste del socket al quale è collegato il webserver, mentre la seconda macro in **Fig. 7**, chiamata dall'interrupt del timer, si occuperà della lettura periodica del sensore di temperatura e dell'aggiornamento dell'array di temperature da visualizzare in remoto. Le variabili utilizzate sono quelle descritte qui di seguito.

- **ContatoreTimer** di tipo Grande+ inizializzato a -1 in modo da forzare la lettura della temperatura all'accensione del sistema.
- **ADC_Temp** di tipo Virgola mobile per poter leggere la tensione del sensore LM35.
- **Temperatura_1 / Temperatura_4** di tipo intero per la memorizzazione delle temperature.
- **B001 / B004** di tipo intero per la visualizzazione sulla pagina HTML dei valori di temperatura.

Sono state inoltre utilizzate due costanti globali per la conversione degli impulsi del timer in valori di tempo:

- **IMPULSI_OGNI_SECONDO** vale 75 perché la frequenza di interrupt del timer0 è prefissata a 75 Hz (questo valore è dovuto al fatto che il prescaler del

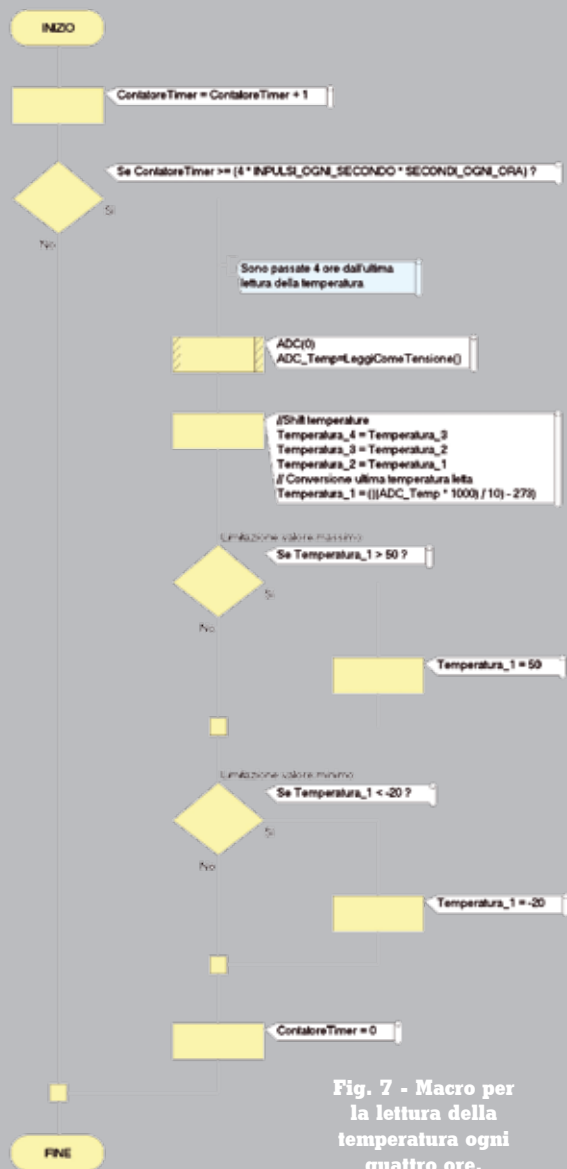


Fig. 7 - Macro per la lettura della temperatura ogni quattro ore.

timer è fissato a 256 e l'oscillatore è configurato per lavorare a 19.660.800 Hz;
- **SECONDI_OGNI_ORA** vale 3600 ed è la costante che permette di sapere quanti secondi vi sono per ogni ora.

Per poter effettuare un semplice test, possiamo collegare il sistema mediante il cavo ethernet al router con il quale ci connettiamo in rete e dal browser del nostro computer e digitare l'indirizzo del Web Server, che in questo caso è 192.168.0.5. La pagina che si aprirà è mostrata in Fig. 9. Permettere l'accesso del web server dall'esterno della nostra rete locale è più complicato, perché implica l'ottenimento di un IP fisso oppure poter avere a disposizione un modem in grado di utilizzare servizi di DNS dinamico. Un servizio gra-

tuito, con alcune limitazioni, è fornito ad esempio da www.dyndns.it. Basterà quindi configurare il modem in modo da comunicare correttamente con uno di questi server, scegliere l'acronimo con il quale verrà identificato il nostro sistema embedded ed il gioco è fatto: abbiamo reso accessibili, da qualsiasi parte del mondo, i dati forniti dalla nostra scheda embedded.

Utilizzare MPLAB per integrare il codice generato da Flowcode

Utilizzare Flowcode per realizzare progetti sempre più complessi porta ad un incremento della difficoltà del debug del codice generato soprattutto se vi sono blocchi con all'interno del codice C. Il Debugger classico di Flowcode ci permette di monitorare le variabili e gli stati del sistema ma non è possibile, in alcun modo,

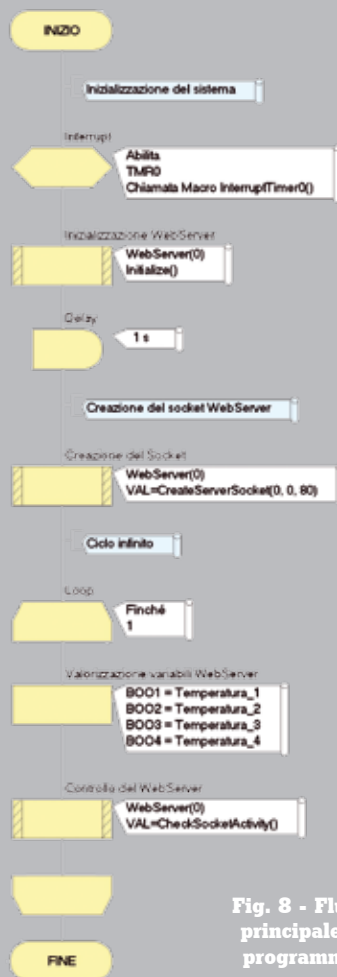


Fig. 8 - Flusso principale di programma.

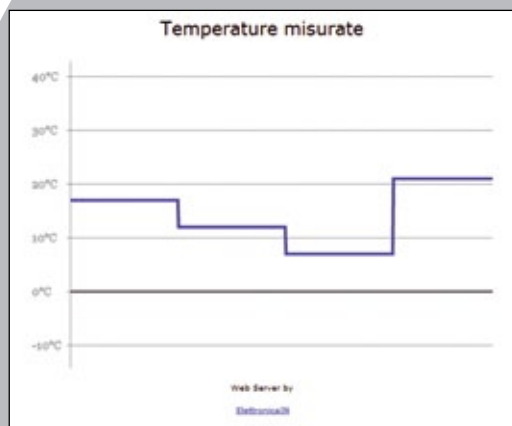


Fig. 9 - Pagina inviata dal Web Server.

visualizzare direttamente i registri del microcontrollore. Per queste operazioni, che potrebbero risultare complesse per un principiante, ma che sono essenziali per un utilizzatore esperto, possiamo avvalerci di MPLAB e del compilatore C BoostC. Sono entrambi scaricabili da Internet in versione free rispettivamente all'indirizzo www.microchip.com e www.sourceboost.com.

Per integrare i due sistemi è necessario prestare molta attenzione ai passi con il quale viene installato il compilatore: per prima cosa occorre installare, se non già presente, MPLAB IDE, quindi il compilatore BoostC, seguendo attentamente la procedura. Al passo in cui viene visualizzata l'immagine in Fig. 11 è necessario scegliere la cartella dove è stato installato MPLAB IDE e fare clic sul pulsante "Integrate"; solo in questo modo sarà possibile scegliere dall'IDE il compilatore BoostC.

Al termine dell'installazione è necessario aprire MPLAB IDE e impostare, dal menu *Project->Set Language Tool Locations*, correttamente le informazioni sul compilatore, come indicato in Fig. 12 e Fig. 13.

Impostare anche gli altri indirizzi per il "BoostC Compiler for PIC18" per poterlo utilizzare anche su microcontrollori della famiglia 18. In questo caso, bisogna scegliere il *boostc_pic18.exe*. Utilizzando la versione gratuita di BoostC si possono compilare solo progetti che non superino i limiti di memoria RAM e ROM, mentre con la versione PRO, acquistabile dal produttore, non vi sono limitazioni di alcun tipo. Prendiamo come esempio il diagramma che abbiamo realizzato nella puntata 4 per pilotare dei relé tramite RS232. Apriamo MPLAB IDE e dal menu scegliamo il

Il linguaggio **HTML**

Il linguaggio HTML, acronimo di **H**yper**T**ext **M**arkup **L**anguage, è il linguaggio base con le quali vengono realizzate le pagine web. È basato sull'utilizzo di Tag che permettono di formattare il testo nel modo desiderato. Non può essere considerato un linguaggio di programmazione perché non possono essere definite alcune variabili e cicli di controllo ma unicamente un linguaggio di formattazione testuale. Per ovviare a questa limitazione è però possibile inserire all'interno della pagina dei contenuti esterni come immagini, video o script in altri linguaggio di programmazione come ad esempio il JavaScript. I tag sono racchiusi tra parentesi angolate ed ogni volta che un tag viene aperto deve poi anche essere chiuso inserendo prima del nome la barra. Ad esempio, per inserire il testo "Testo formattato" in grassetto dovremo scrivere:

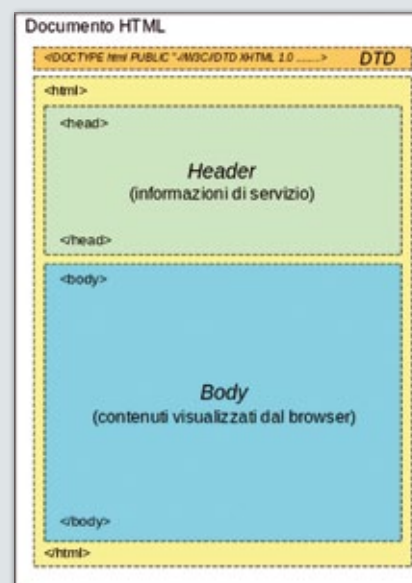


Fig. 10
Pagina
inviata dal
Web Server.

```
<b>Testo Formattato</b>
```

La struttura base di un file Html è raffigurata in **Figura 10**. Partendo dalla testa del file, si ha una descrizione del documento, alcune informazioni di servizio ed il corpo del documento nel quale viene formattato l'intero contenuto della pagina.

Figura 10 - Struttura base di un file HTML

Per vedere alcuni esempi di codice HTML basta che apriamo una pagina web e, nel caso il browser sia Firefox, cliccare con il tasto destro sull'area centrale e scegliere dal menu "Visualizza sorgente pagina".

Project->Project Wizard per creare un nuovo progetto; scegliamo il microcontrollore che avevamo scelto in Flowcode (il PIC18F4580), impostiamo come compilatore il BoostC per i pic della famiglia 18 ed aggiungiamo il file Rs232.c generato da Flowcode.

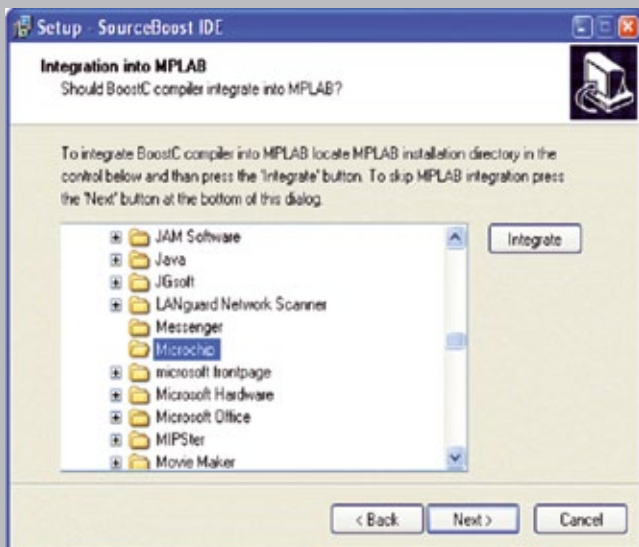


Fig. 11 - Indicazione cartella Microchip MPLAB.

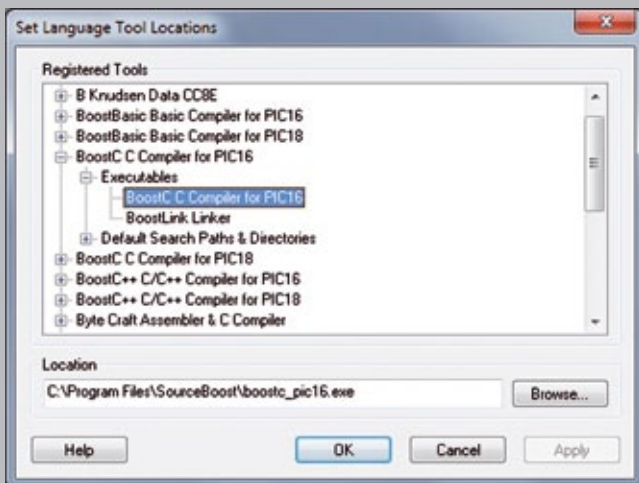


Fig. 12 - Configurazione eseguibile compilatore BoostC per PIC16.

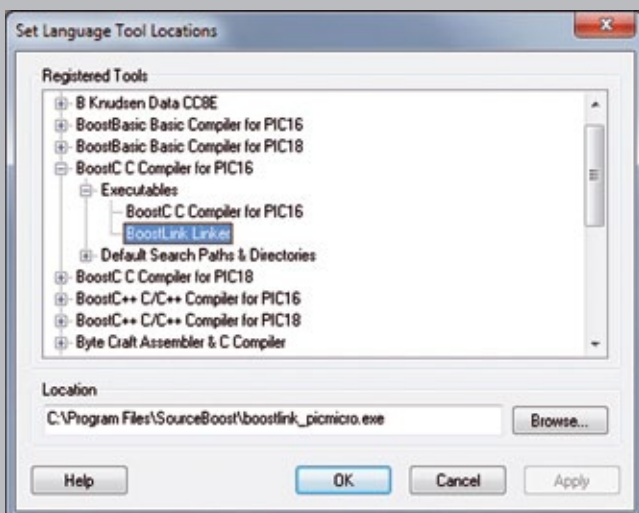


Fig. 13 - Configurazione eseguibile linker BoostC per PIC16.

Per poter completare il progetto dobbiamo ancora aggiungere le librerie che utilizziamo all'interno di Flowcode: allo scopo entriamo nella cartella del compilatore, che nel nostro caso è in "C:\Program Files\SourceBoost", apriamo la cartella Lib e scegliamo i file .lib che sono necessari per questo progetto. In questo caso sono *float.pic18.lib* e *libc.pic18.lib*, ma in progetti più complessi potrebbero esserne necessari altri per supportare tutte le funzioni richieste dal codice. Con il Drag&Drop portiamoli all'interno di MPLAB nell'albero di progetto, come indicato in Fig. 14. Con la scorciatoia da tastiera CTRL+F10, avviamo la compilazione e se abbiamo svolto tutte le operazioni correttamente l'esito sarà positivo.

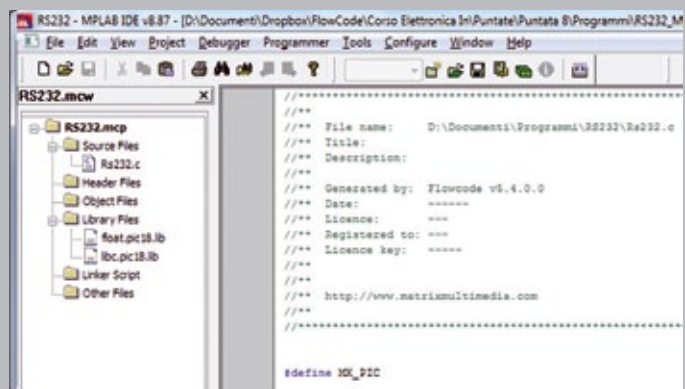


Fig. 14 - Configurazione del progetto di debug.

Ora che il codice è compilato non ci resta che scegliere il debugger dal menu "Debugger", programmare il dispositivo ed eseguire i test in MPLAB come se avessimo scritto a mano il codice. Qualora vengano trovati errori o imperfezioni è consigliabile effettuare le dovute correzioni su Flowcode, e non direttamente sul codice, in modo che, ad una successiva generazione di codice, vengano automaticamente corretti gli errori e non si debba ritoccare a mano il codice. Questo semplice stratagemma ci permette di utilizzare tutte le potenzialità di Flowcode anche in progetti molto complessi, avendo però a disposizione, per la risoluzione dei problemi, uno strumento completo e dettagliato come MPLAB.