

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

Elettronica In

www.elettronica.in.it

oltre l'elettronica



Dotiamo
Raspberry Pi di un
display TFT Touch Screen
sul quale visualizzare la console
di sistema, i film e le foto
preferite oppure comandare una
scheda relé... in punta di dita.

DISPLAY TOUCH PER RASPBERRY PI

..... di MARCO MAGAGNIN

Per evitare di utilizzare monitor HDMI dal costo ben superiore a quello di Raspberry Pi, negli articoli precedenti abbiamo sempre adottato la soluzione di collegarci al microcomputer in remoto, utilizzando strumenti come PuTTY e WinSCP. Questo approccio ci ha sempre vincolato ad utilizzare Raspberry Pi nella modalità cosiddetta "server". Di conseguenza abbiamo sempre presentato le applicazioni con interfaccia utente realizzate in modo da essere fruibili da browser web. Che fare, invece, quando vogliamo avere un'interfaccia in

stile tradizionale, fruibile direttamente dal desktop? Semplice, si collega un monitor, una tastiera, un mouse e si realizza una classica applicazione desktop utilizzando i linguaggi classici con le librerie per la realizzazione di interfacce grafiche. Già, ma in questo modo trasformiamo Raspberry Pi in un PC con attaccate tutte le periferiche tradizionali che lo rendono "ingombrante". La soluzione che vi presentiamo al posto di quella "classica" permette di realizzare interfacce grafiche utilizzando un grazioso monitor TFT da 2,8 pollici, a colori, ed in più

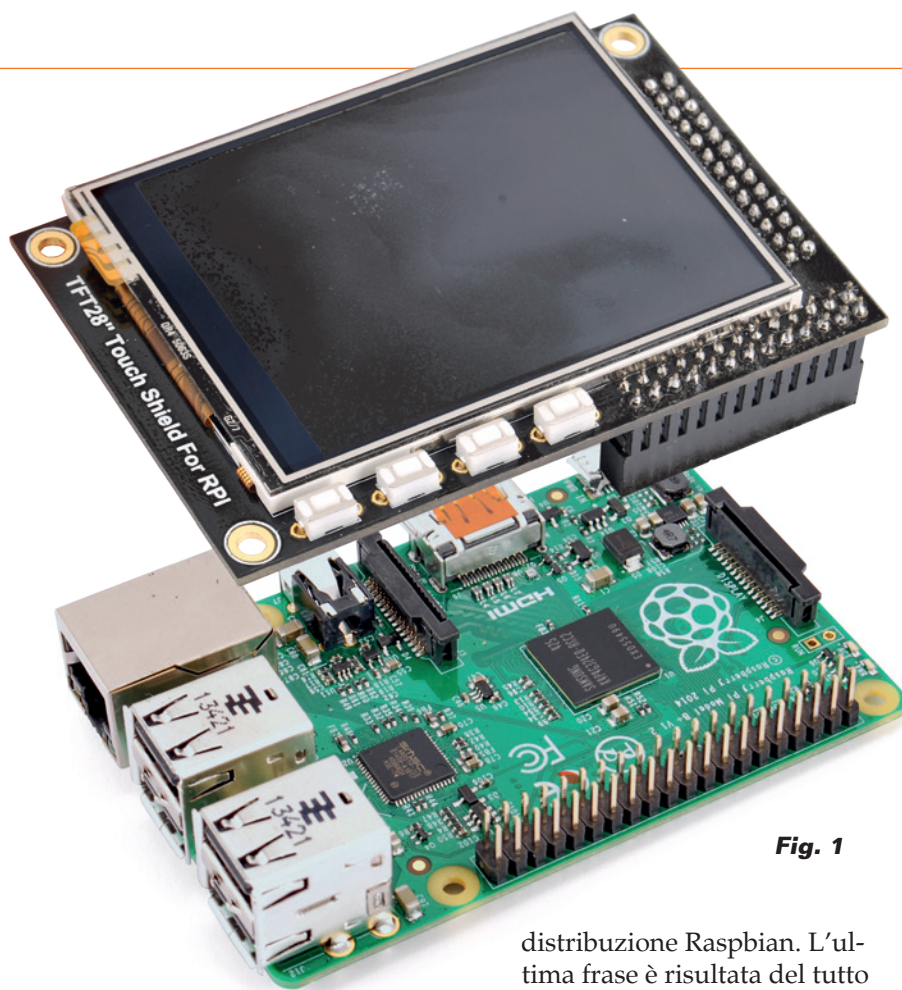


Fig. 1

con un'interfaccia touch di tipo resistivo. Niente monitor, niente tastiera e niente mouse. La scheda con il monitor ha più o meno le stesse dimensioni di Raspberry Pi, è già montata ed è compatibile con tutti i modelli del microcontrollore: A, B e B+. In questo articolo, abbiamo adottato il modello B+ che, per il progetto finale che vi proponiamo, mette a disposizione un maggior numero di pin di I/O rispetto alle versioni precedenti. Il monitor ha una risoluzione di 320 x 240 pixel, con una profondità di colore di 16 bit. Il mini display si collega a Raspberry Pi sul bus SPI e può essere utilizzato per reindirizzare la console, oppure come monitor X per visualizzare il desktop di Raspbian, foto, video e interfacce grafiche di applicazioni utente. Fin qui tutto bene. Il rovescio della medaglia è che il driver per la gestione del piccolo monitor non è incluso nel kernel della

distribuzione Raspbian. L'ultima frase è risultata del tutto incomprensibile? Per approfondire l'argomento vi rimandiamo al libro "Raspberry Pi – Il mio primo Linux embedded" per un viaggio di approfondimento sul microcomputer. Tra l'altro una delle Appendici è dedicata ad introdurre la struttura del sistema operativo GNU/Linux, kernel e driver inclusi. Ma torniamo al driver. Questo simpatico monitor ci permette di introdurre, in modo più approfondito di quanto abbiamo fatto fino ad ora, la realizzazione di un sistema GNU/Linux embedded. Nella pratica "professionale" le distribuzioni GNU/Linux destinate al mondo embedded sono le più scarse possibili, con inclusi unicamente i driver e le applicazioni essenziali per il funzionamento di base del microcomputer. Questo per minimizzare l'occupazione di memoria della distribuzione e massimizzare le prestazioni del sistema di destinazione. Questo approccio

permette di utilizzare chip di prestazioni ridotte, con dimensioni fisiche della soluzione finale il più possibile contenute e consumi anch'essi resi minimi per poter permettere soluzioni basate su alimentazioni a batterie. Il rovescio della medaglia consiste nel dover personalizzare il kernel della distribuzione con i driver delle periferiche che si vogliono utilizzare, come per esempio dongle WiFi, periferiche USB, display LCD e pannelli LED. Utilizzando Raspbian abbiamo sempre evitato di doverci occupare di questi problemi, in quanto, essendo un microcomputer progettato per la didattica, nella sua distribuzione Raspbian sono già inclusi una enorme quantità di driver che soddisfano in automatico (plug and play) la maggior parte delle esigenze. Questa volta però il driver per il display LCD TFT non è incluso. Fortunatamente la società Adafruit ha messo a disposizione un kernel di Raspbian già compilato che include il driver richiesto. Questo ci permette di evitare di dover personalizzare e compilare da noi il kernel di Raspbian. Stiamo comunque pensando di proporre sulla rivista una serie di articoli dedicati all'argomento, anche perché la presenza sul mercato di sempre nuove periferiche richiederà prima o poi di affrontare l'argomento con determinazione. Nella soluzione Adafruit si presentano infatti due problemi. Che fare quando dobbiamo adottare una seconda periferica che richiede l'inclusione nel kernel di Linux di un driver specifico? Anche se il produttore mette a disposizione una versione "compilata" del kernel per la sua periferica, se la adottassimo perderemmo il driver per la prima periferica. Non resta che configurarci e compilarci il

Fig. 2

kernel con l'aggiunta dei due, o più, driver aggiuntivi richiesti. Secondo problema. Adottare il kernel precompilato, personalizzato con l'aggiunta di un driver specifico, ci vincola alla versione di kernel utilizzato per la compilazione. In questa condizione non saremo in grado di adottare nuove distribuzioni che sfruttano versioni aggiornate del kernel, a meno che il produttore della periferica non fornisca a sua volta una nuova versione. A questo proposito ricordiamo che è stata appena rilasciata la release 3.16 del kernel di Linux, che presenta notevoli funzionalità aggiuntive per quanto riguarda la gestione dei chip con CPU ARM e relativi processori grafici. Segno che assisteremo ad una sempre maggior diffusione della nostra "filosofia". Ma torniamo al nostro display grafico visibile in **Fig. 1**, basato sul modulo LCD TFT ILI9340, che viene fornito già montato e dotato di connettore GPIO, che si installa su tutti i modelli di Raspberry Pi A, B e B+. La board include il controllore del display STMPE610 che comunica con Raspberry Pi utilizzando il bus SPI e quindi impegna i pin corrispondenti del connettore GPIO (SCK, MOSI, MISO, CE0, CE1) oltre ai pin GPIO25 e GPIO24. I quattro pulsanti aggiuntivi presenti sulla board sono collegati ai pin GPIO23, GPIO22, GPIO21, e GPIO18. Tutti gli altri pin sono inutilizzati e possono essere impiegati per altre esigenze. In questo caso è opportuno saldare un connettore sugli alloggiamenti previsti sul lato destro della scheda e utilizzare un cavo piatto

di prolunga. Fate attenzione a rispettare le corrispondenze dei pin. Il pin numero 1 del connettore GPIO è indicato sul retro della board con una piazzola di forma quadrata ed una piccola freccia bianca serigrafata.

PERSONALIZZAZIONE DELLA DISTRIBUZIONE RASPBIAN

Il processo che vi descriviamo per rendere operativo il display LCD TFT richiede di partire dalla distribuzione Raspbian. Noi siamo partiti dalla immagine rilasciata in data 9-settembre-2014 con release del kernel 3.12 (**Fig. 2**). Al momento è preferibile non installare il display sul connettore, lo faremo dopo aver installato il software di gestione. Per poter eseguire il processo di installazione è necessario che Raspberry Pi sia collegato alla rete locale in modo da poter "vedere" la rete internet pubblica. È inoltre necessario poter accedere a Raspberry Pi per utilizzare una finestra terminale ed una finestra con l'applicazione di gestione del file system. Come al solito, ma soprattutto in questo caso, preferiamo accedere da remoto via protocollo SSH ed utilizzare (in Windows) gli strumenti Putty e WinSCP. Per approfondimenti sulle modalità di connessione ed utilizzo di questi strumenti, vi rimandiamo al libro citato all'inizio dell'articolo. È ovviamente possibile utilizzare un collegamento alla console via cavo seriale.

Ci colleghiamo a Raspberry Pi con l'utente "root" ed iniziamo il processo di installazione con i soliti comandi di aggiornamento

della distribuzione:

```
apt-get update
apt-get upgrade
```

Andiamo ad una cartella di appoggio per il download dei file di installazione. Possiamo anche creare una cartella specifica per questo processo. I file sono disponibili come pacchetti .deb, che dovranno poi essere installati con il comando dpkg. Anche in questo caso vi rimandiamo al libro citato per approfondimenti. Possiamo utilizzare, per esempio la cartella "home". Ci posizioniamo su questa cartella con il comando:

```
cd /home
```

Ora scarichiamo tutti i pacchetti necessari alla "sostituzione" del kernel ed alla gestione del display:

```
wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-bin-adafruit.deb
wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-dev-adafruit.deb
wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-doc-adafruit.deb
wget http://adafruit-download.s3.amazonaws.com/libraspberrypi0-adafruit.deb
wget http://adafruit-download.s3.amazonaws.com/raspberrypi-bootloader-adafruit-20140917-1.deb
```



```
192.168.0.43 - PuTTY
root@raspberrypi:~# FRAMEBUFFER=/dev/fb1 startx

X.Org X Server 1.12.4
Release Date: 2012-08-27
X Protocol Version 11, Revision 0
Build Operating System: Linux 3.2.0-2-mx5 armv7l Debian
Current Operating System: Linux raspberrypi 3.12.26+ #7 PREEMPT Wed Sep 17 14:25:24 EDT 2014 armv6l
Kernel command line: dma.dmachans=0x7f35 bcm2708_fb.fbwidth=656 bcm2708_fb.fbheight=416 bcm2708.boardrev=0xc bcm2708.serial=0xa7bb39e2 smac95xx.macaddr=B8:27:EB:BB:39:E2 sdhci-bcm2708.emmc_clock_freq=250000000 vc_mem.mem_base=0x1ec00000 vc_mem.mem_size=0x20000000 dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
Build Date: 18 December 2013 05:20:52PM
xorg-server 2:1.12.4-6+deb7u2 (Julien Cristau <jcristau@debian.org>)
Current version of pixman: 0.33.1
  Before reporting problems, check http://wiki.x.org
  to make sure that you have the latest version.
Markers: (--) probed, (**) from config file, (==) default setting,
(++) from command line, (!!) notice, (II) informational,
(WW) warning, (EE) error, (NI) not implemented, (??) unknown.
(==) Log file: "/var/log/Xorg.0.log", Time: Wed Oct 15 09:12:15 2014
(==) Using config directory: "/etc/X11/xorg.conf.d"
(==) Using system config directory "/usr/share/X11/xorg.conf.d"
```

Fig. 3

Dovreste vedere apparire il desktop di Raspberry Pi sul piccolo display come visibile in Fig. 4.

Potete anche utilizzare la barra di menu e le applicazioni in modalità "touch".

Se tutto funziona correttamente il passo più importante è stato fatto. Ora facciamo in modo che tutto parta in automatico all'accensione di Raspberry Pi.

Chiudiamo il desktop grafico (in realtà il server X) dalla finestra di terminale premendo contemporaneamente i tasti <Control-X>. Apriamo da WinSCP il file:

Terminato il download, dalla stessa cartella, installiamo il tutto con il comando:

```
sudo dpkg -i -B *.deb
```

Questa operazione richiede un tempo abbastanza lungo, potete prendervi una pausa e rilassarvi un attimo.

Nel caso stiate utilizzando, come molto probabile, una versione di Raspbian posteriore a Settembre 2013, è necessario disattivare l'utilizzo del framebuffer X "accelerato", con il comando che rimuove il file di configurazione dell'acceleratore e lo salva nella cartella /home (non si sa mai):

```
mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf /home
```

Ora spegniamo Raspberry Pi con il comando tradizionale:

```
shutdown -h now
```

Togliamo l'alimentazione, montiamo il display sul connettore GPIO, verificando che non si verifichino contatti con qualche connettore o componente di Raspberry Pi e ridiamo tensione. Non preoccupatevi, sul piccolo display, per il momento, non

apparirà nulla. Ci ricollegiamo a Raspberry Pi con Putty e WinSCP. Per provare velocemente il funzionamento del display possiamo caricare i driver da linea di comando e lanciare il desktop di Raspberry Pi.

Digitiamo i seguenti comandi nell'ordine, come visibile in Fig. 3:

```
modprobe spi-bcm2708
modprobe fbtft_device
name=adafruitts rotate=90
export FRAMEBUFFER=/dev/fb1
startx
```

/etc/modules

ed aggiungiamo in fondo al file i due moduli da caricare all'accensione:

```
spi-bcm2708
fbtft_device
```

Salviamo e chiudiamo il file. Rispondiamo all'eventuale richiesta di password per perfezionare il salvataggio.

Un momento, non abbiamo ancora finito, dobbiamo aggiungere un file di configurazione

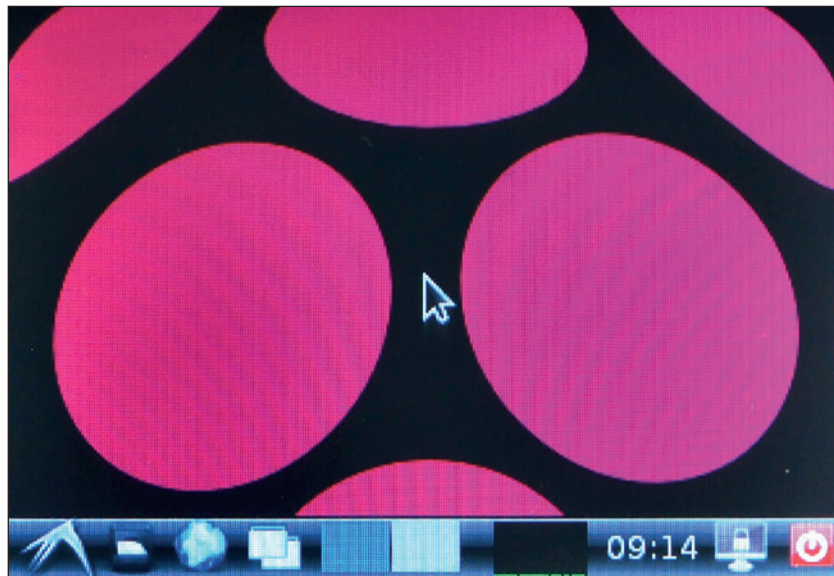
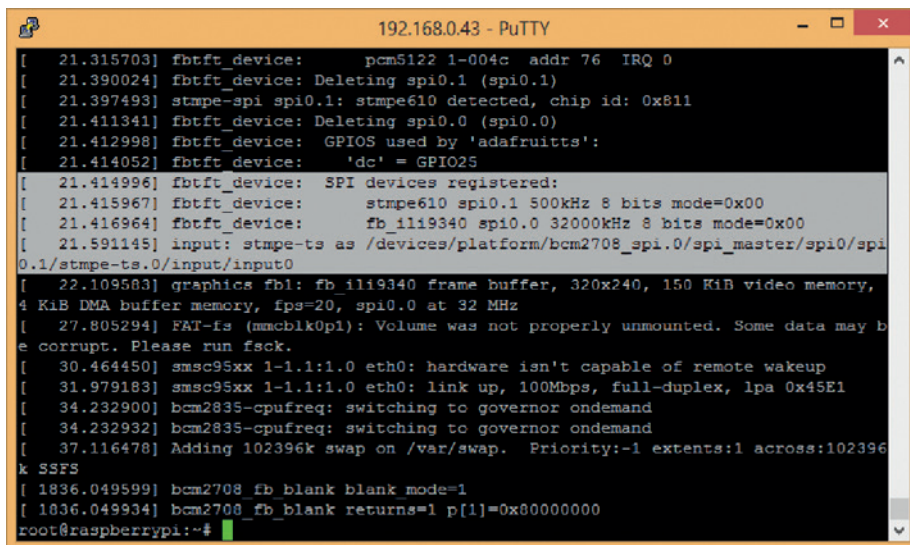


Fig. 4



per personalizzare l'utilizzo del display in termini di rotazione dell'immagine e frequenza di refresh. Apriamo il file:

`/etc/modprobe.d/adafruit.conf`

ed aggiungiamo le seguenti configurazioni:

```
options          fbtft_device
name=adafruitrt28 rotate=90
frequency=32000000
```

Il parametro "rotate" permette di ruotare l'immagine sul display di 0, 90, 180 o 270 gradi in modo da potersi adattare a tutte le situazioni di montaggio:

0 corrisponde al display in verticale, con il lato inferiore verso le scritte stampate sulla basetta;
90 corrisponde al display in orizzontale, con il lato inferiore verso i pulsanti;
180 corrisponde al display in verticale, con il lato superiore verso le scritte stampate sulla basetta;

270 corrisponde al display in orizzontale, con il lato superiore verso i pulsanti.

Il parametro "frequency" indica la velocità di refresh del display. Il valore "32000000" significa 32MHz e corrisponde ad un frame rate di circa 20FPS. Se il carico complessivo delle vostre applicazioni lo permette, potete provare ad abbassare la frequenza a 16 MHz (16000000). Fatto questo rendiamo effettive le modifiche con un comando di:

`reboot`

Dopo essersi ricollegati controlliamo i messaggi di console con il comando:

`dmesg`

facciamo particolare attenzione al riconoscimento del controllore STMPE610 ed alla impostazione della frequenza del display ILI9340 come visibile nella Fig. 5.

Fig. 5

Possiamo riverificare il funzionamento del tutto lanciando il desktop sul server x con il comando:

`FRAMEBUFFER=/dev/fb1 startx`

Per interrompere l'esecuzione del desktop premete <Control-X>. Tutto a posto? Non ancora, ora dobbiamo configurare e calibrare il touch screen in modo da poter funzionare sia con il desktop grafico che con le applicazioni che andremo a sviluppare.

Come prima operazione impostiamo una regola nella cartella udev che colleghi il touchscreen al corretto device di input. Può accadere che il touchscreen venga riconosciuto nella cartella `/dev/input` con un "event" diverso a seconda se sono presenti o meno tastiera, mouse o altri dispositivi di input. Creiamo il file

`95-stmpe.rules`

nella cartella

`/etc/udev/rules.d/`

ed inseriamo le seguenti configurazioni (Fig. 6):

```
SUBSYSTEM=="input",
ATTRS{name}=="stmpe-ts",
ENV{DEVNAME}=="*event*",
SYMLINK+="input/touchscreen"
```

Rimuoviamo e reinstalliamo il driver del touch screen con i comandi:

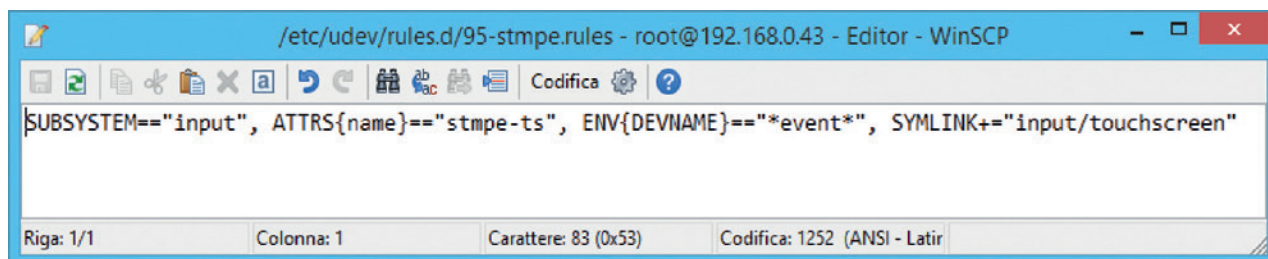


Fig. 6


```

192.168.0.43 - PuTTY
root@raspberrypi:~# ls -l /dev/input/touchscreen
lrwxrwxrwx 1 root root 6 Oct 20 08:55 /dev/input/touchscreen -> event0
root@raspberrypi:~#

```

Fig. 7

```

192.168.0.43 - PuTTY
Event: time 1413799520.383315, type 3 (EV_ABS), code 1 (ABS_Y), value 405
Event: time 1413799520.383315, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 167
Event: time 1413799520.383315, ----- SYN_REPORT -----
Event: time 1413799520.391004, type 3 (EV_ABS), code 0 (ABS_X), value 1689
Event: time 1413799520.391004, type 3 (EV_ABS), code 1 (ABS_Y), value 410
Event: time 1413799520.391004, ----- SYN_REPORT -----
Event: time 1413799520.398521, type 3 (EV_ABS), code 0 (ABS_X), value 1700
Event: time 1413799520.398521, type 3 (EV_ABS), code 1 (ABS_Y), value 413
Event: time 1413799520.398521, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 165
Event: time 1413799520.398521, ----- SYN_REPORT -----
Event: time 1413799520.406093, type 3 (EV_ABS), code 0 (ABS_X), value 1669
Event: time 1413799520.406093, type 3 (EV_ABS), code 1 (ABS_Y), value 406
Event: time 1413799520.406093, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 155
Event: time 1413799520.406093, ----- SYN_REPORT -----
Event: time 1413799520.434786, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 0
Event: time 1413799520.434786, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1413799520.434786, ----- SYN_REPORT -----

```

Fig. 8

```

192.168.0.43 - PuTTY
root@raspberrypi:/home# python pitft_touch_cal.py
USING ROTATION: 90

-----
CURRENT CONFIGURATION

Current /etc/pointercal configuration:
-30 -5902 22077792 4360 -105 -1038814 65536

Current /etc/X11/xorg.conf.d/99-calibration.conf configuration:
Section "InputClass"
    Identifier      "calibration"
    MatchProduct    "stmpe-ts"
    Option "Calibration" "3807 174 244 3872"
    Option "SwapAxes" "1"
EndSection

-----
NEW CONFIGURATION

New /etc/pointercal configuration:
-30 -5902 22077792 4360 -105 -1038814 65536

New /etc/X11/xorg.conf.d/99-calibration.conf configuration:
Section "InputClass"
    Identifier      "calibration"
    MatchProduct    "stmpe-ts"
    Option "Calibration" "3807 174 244 3872"
    Option "SwapAxes" "1"
EndSection

Update current configuration to new configuration? [y/N]: y
-----
Updated /etc/pointercal
Updated /etc/X11/xorg.conf.d/99-calibration.conf
root@raspberrypi:/home#

```

Fig. 9

```

rmmod stmpe_ts
modprobe stmpe_ts

```

A questo punto nella cartella /dev/input troveremo il device

“touchscreen” che punta al corretto device “event<x>”. Potete verificarlo con il comando (Fig. 7):

```
ls -l /dev/input/touchscreen
```

Vedremo la descrizione del device file con il collegamento (la freccetta) verso un device “event(x)”. Ora dobbiamo “calibrare” il nostro touch screen. Vi sono due modi per farlo, uno un po’ più “grossolano” ma automatico, l’altro più preciso ma più articolato e lungo da eseguire. Per le nostre esigenze ed un po’ perché siamo pigri utilizzeremo il metodo “automatico” utilizzando un programma Python realizzato e gentilmente messo a disposizione da Adafruit. Prima installiamo una libreria utile per eseguire i test sul touch screen.

```
apt-get install evtest tslib
libts-bin
```

Possiamo già fare una prova digitando:

```
evtest /dev/input/touchscreen
```

Se tocchiamo il display in vari punti vedremo gli eventi riconosciuti e le coordinate del “punto” toccato (Fig. 8). Per la calibrazione “grossolana” vera e propria scarichiamo il program-

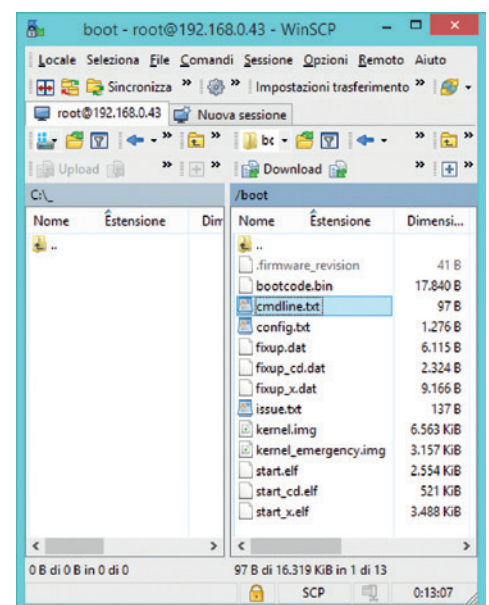


Fig. 10

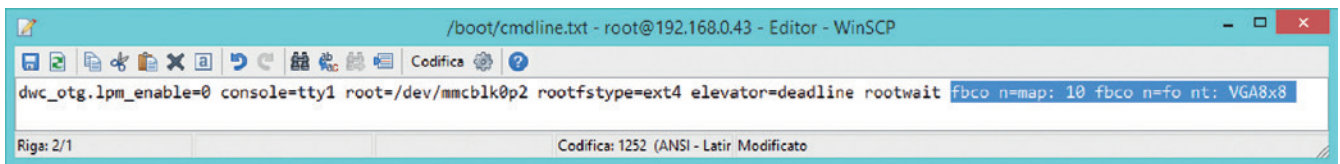


Fig. 11

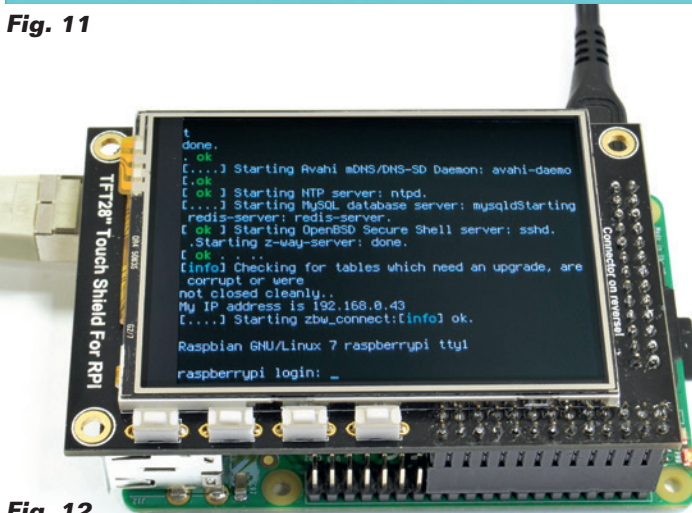


Fig. 12

ma Python dal sito di Adafruit e prepariamo la cartella per contenere il file di configurazione. Scarichiamo il programma con il comando:

```
wget https://github.com/adafruit/PiTFT_Extras/raw/master/pitft_touch_cal.py
```

Creiamo la cartella xorg.conf.d con il comando:

```
mkdir /etc/X11/xorg.conf.d/
```

ed eseguiamo il programma di calibrazione dalla stessa cartella dove lo abbiamo scaricato:

```
python pitft_touch_cal.py
```

Guardate l'output del programma, prima di aggiornare il file di configurazione il programma vi chiede l'autorizzazione, se siete soddisfatti dei risultati confermate con "Y" (Fig. 9).

REINDIRIZZARE LA CONSOLE SUL DISPLAY

Una operazione simpatica che si può impostare è di reindirizzare i messaggi di console sul display

invece che sull'uscita video standard. Anche se lo schermo è piccolo, scegliendo un font adatto si può ottenere un'area di testo di 20 righe per 40 caratteri.

Per ottenere questo risultato dobbiamo modificare il file di configurazione del kernel nella partizione di boot del file system.

Per chi volesse approfondire i dettagli rimandiamo sempre al libro citato all'inizio dell'articolo.

Aprirete il file (Fig. 10):

/boot/cmdline.txt

Lo scopo della modifica è reindirizzare la console dal frame buffer HDMI/TV standard /dev/fb0 al frame buffer del display TFT /dev/fb1.

Per farlo andate in fondo alle direttive del file, dopo la voce "rootwait" ed aggiungete le di-

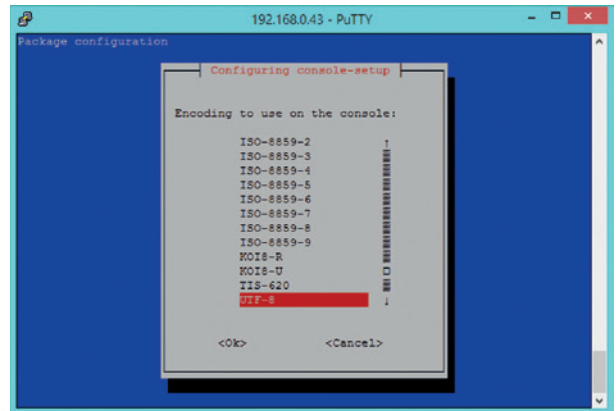


Fig. 13

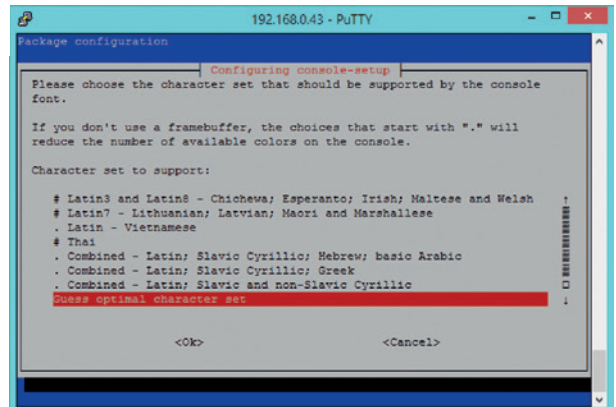


Fig. 14

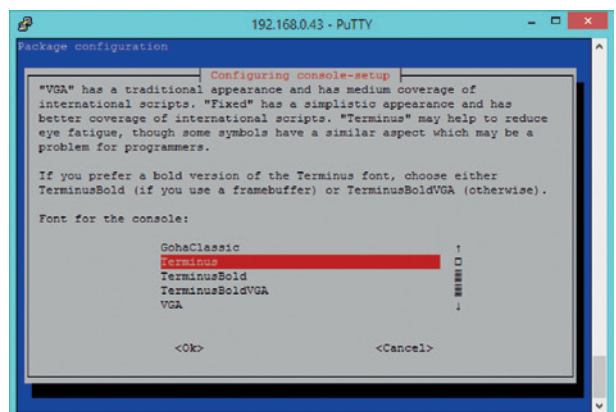


Fig. 15

chiarazioni (Fig. 11):

```
fbco n=map: 10 fbco n=fo nt: VGA8x8
```

Salvate il file e date un comando di reboot. Notate che, durante il

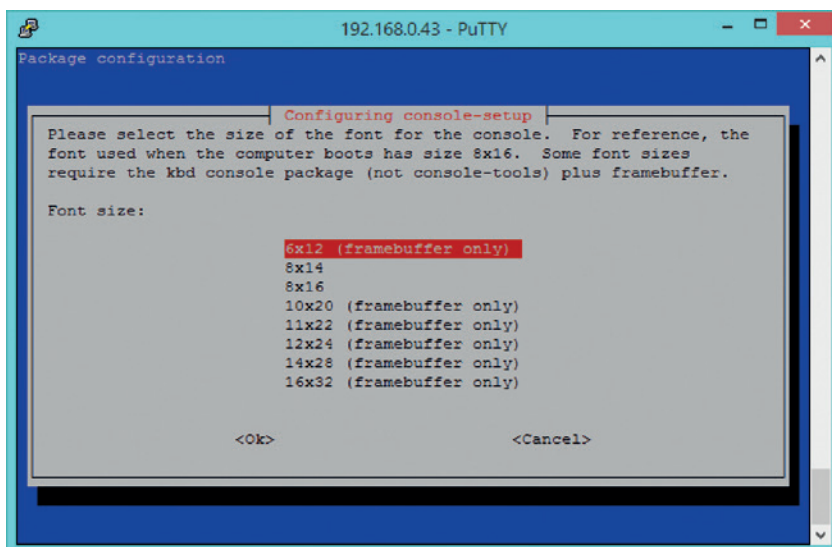


Fig. 16

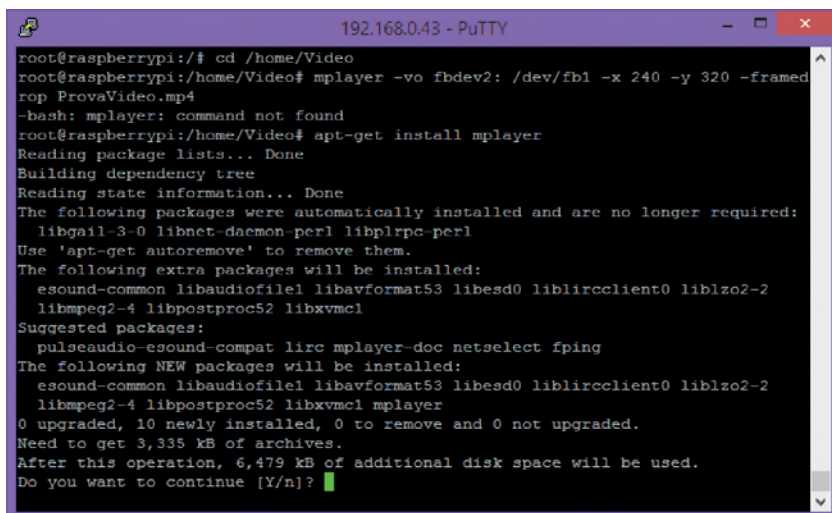


Fig. 17

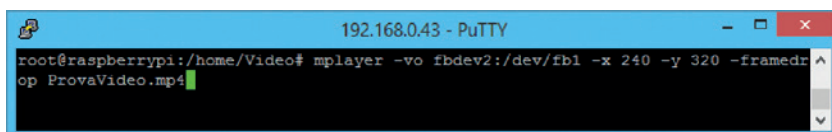


Fig. 18

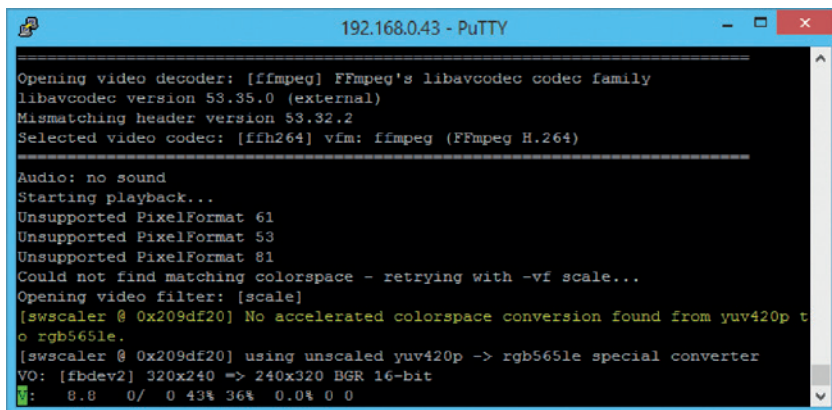


Fig. 19

caricamento iniziale, vi perderete lo splash screen multicolore iniziale ed una parte dei messaggi di console.

Ciò è assolutamente normale, in quanto, per poter lavorare, il driver deve essere caricato dal kernel ed attivato, il che avviene qualche tempo dopo l'inizio del processo di boot. Poi vedrete apparire i messaggi di console, come visibile in Fig. 12.

Per cambiare il font VGA8x8 in qualcosa di maggiormente leggibile come un 12x6 lanciamo il comando di riconfigurazione della console (sempre il solito libro per approfondire):

```
dpkg-reconfigure console-setup
```

seguite le figure da 13 a 16 per selezionare il font Terminus 6x12.

VISUALIZZARE FILE VIDEO

Si possono visualizzare sul nostro display file video con diverse estensioni, a patto che le dimensioni delle immagini siano di 320 x 240 pixel.

Per visualizzare un video utilizziamo il pacchetto player, se non è presente nel nostro sistema lo installiamo con il comando (Fig. 17):

```
apt-get install mplayer
```

Per vedere il video diamo il comando (Fig. 18, Fig. 19 e Fig. 20):

```
mplayer -vo fbdev2:/dev/fb1 -x 240 -y 320 -framedrop <file-video.ext>
```

per esempio "ProvaVideo.mp4". Se il video non è in formato 320 x 240 lo possiamo ridimensionare utilizzando lo strumento HandBrake, Open Source e disponibile per tutti i sistemi operativi.

Lo scarichiamo dall'indirizzo



Fig. 20

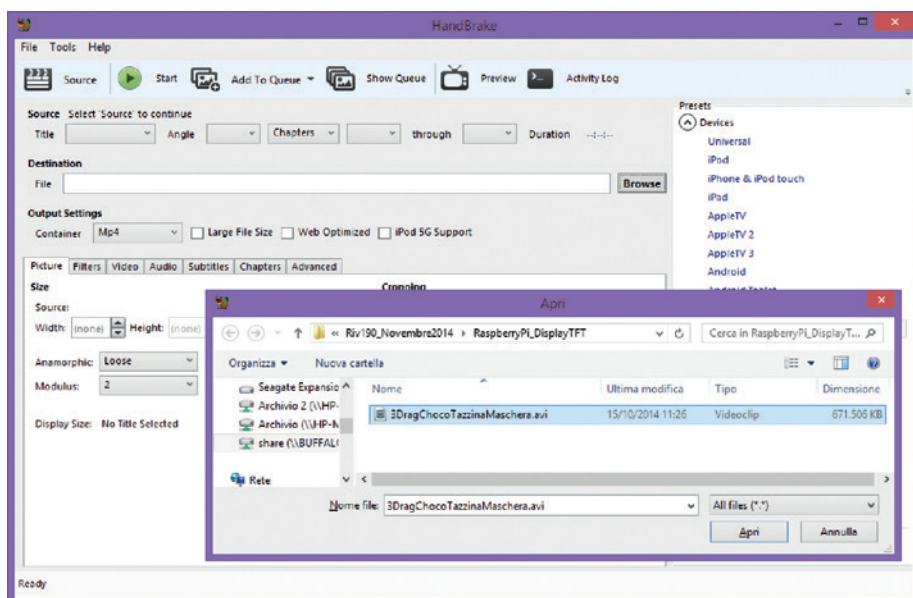


Fig. 21

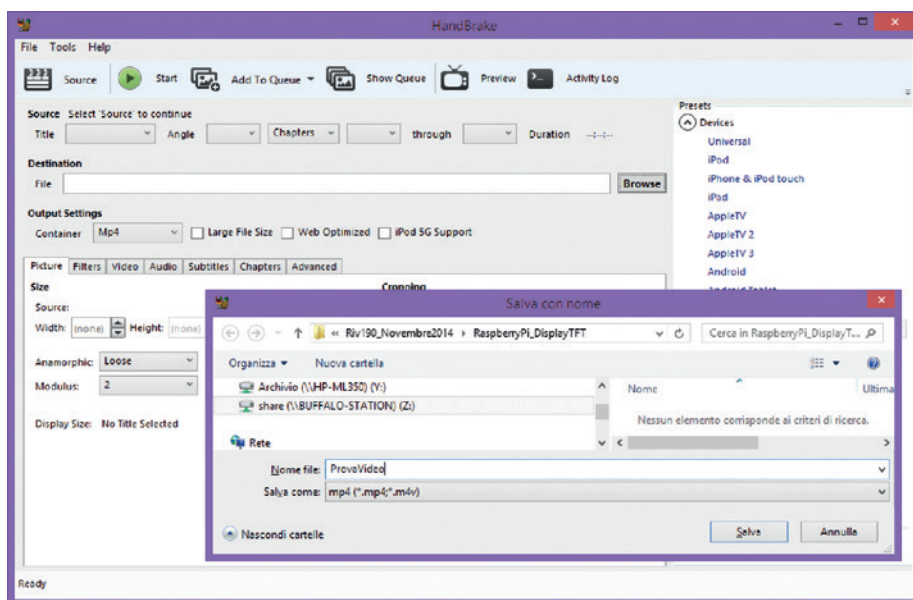


Fig. 22

<https://handbrake.fr/>.

Dopo averlo installato lo utilizziamo per ridimensionare il video di prova. Apriamo il file, preferibilmente con estensione .avi (Fig. 21),

La finestra viene precompilata con una quantità di informazioni relative al file caricato. Nel campo "Destination" inseriamo il percorso ed il nome del file che vogliamo ottenere (Fig. 22).

Nel pannello sottostante, in "Output Settings", nel campo "Container", selezioniamo "MP4", in "Width" impostiamo 320. In Anamorphic selezioniamo "Custom" ed in "Modulus" 2 (Fig. 23). Clicchiamo sul pulsante "Start" per iniziare la conversione (Fig. 24).

Al termine, se abbiamo eseguito la conversione in ambiente Windows, spostiamo il file su Raspberry Pi (Fig. 25) ed eseguiamo il comando descritto all'inizio del paragrafo per visualizzare il file.

VISUALIZZARE IMMAGINI

Un'applicazione simpatica è la visualizzazione di immagini. Per farlo possiamo utilizzare il pacchetto "fb" frame buffer image viewer. Lo installiamo con il solito comando:

```
apt-get install fb
```

Anche in questo caso le immagini devono avere formato 320 x 240 pixel. Per visualizzare un'immagine utilizziamo il comando (Fig. 26):

```
fb -T 2 -d /dev/fb1 -noverbose -a <nome-foto>.jpg
```

REGOLARE LA RETROILLUMINAZIONE

La retroilluminazione del display TFT è realizzata per mezzo di quattro LED che assorbono complessivamente circa 75 mA. Nel

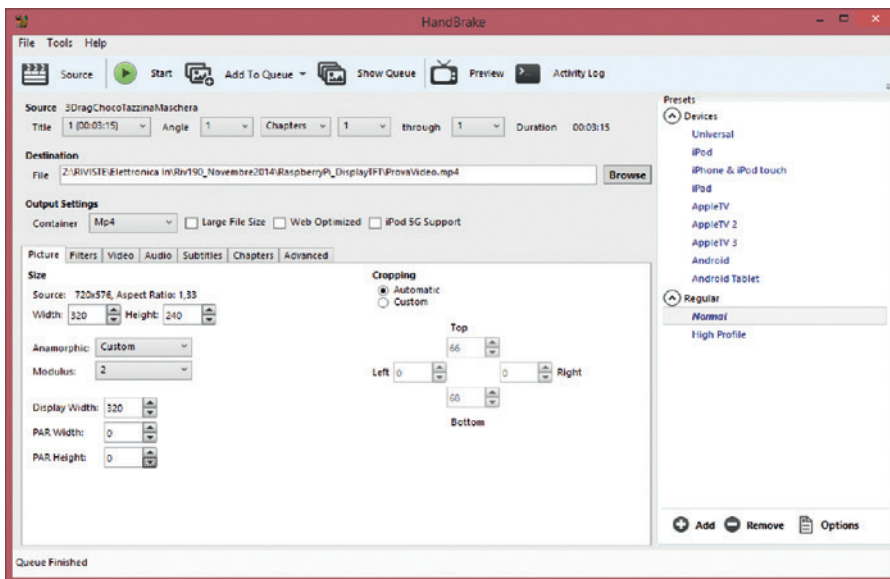


Fig. 23

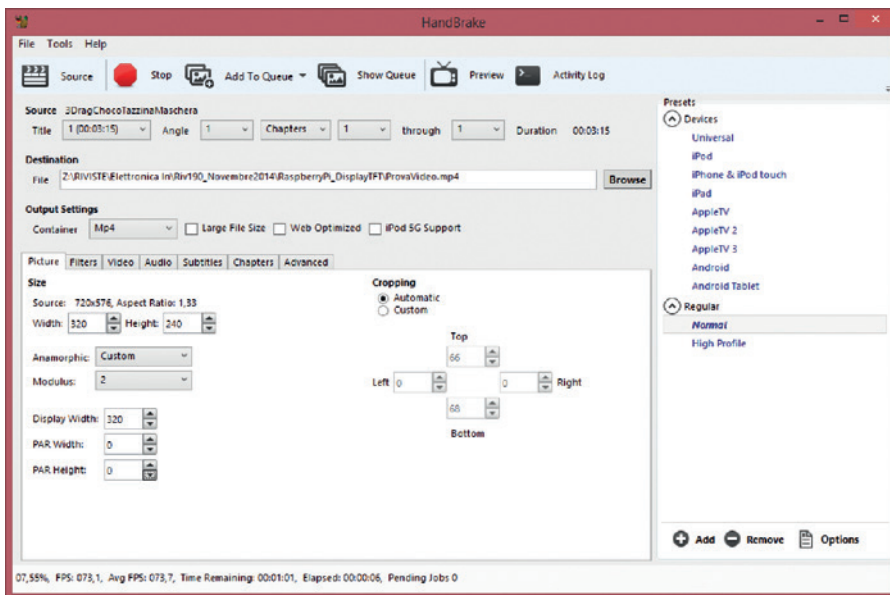


Fig. 24

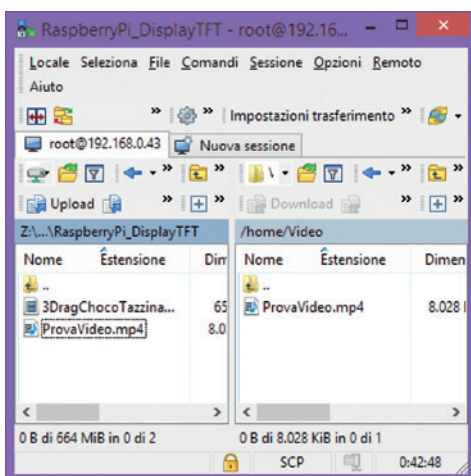


Fig. 25

caso di un utilizzo con alimentazione da batteria o se si vuole comunque spegnere la retroilluminazione il controllore del touch STMPE610 possiede due pin di I/O ed uno di questi è collegato al transistor che controlla la retroilluminazione. Questo pin è raggiungibile da linea di coman-

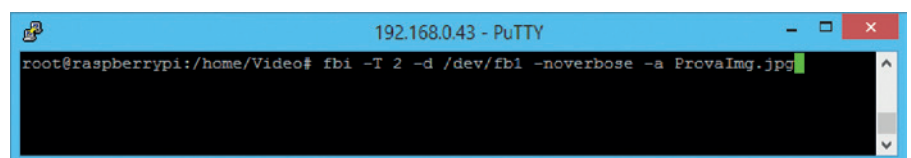


Fig. 26

do come un terminale GPIO di Raspberry Pi, riconoscibile come GPIO 252. Per accedere al GPIO è necessario esportare i suoi file di device con il comando:

```
echo 252 > /sys/class/gpio/export
```

Con il comando:

```
ls -l /sys/class/gpio
```

si verifica la effettiva esistenza del GPIO. Dichiariamo il pin di I/O come di output con il comando:

```
echo 'out' > /sys/class/gpio/gpio252/direction
```

Ora per accendere il display portiamo ad "1" il valore del pin con il comando:

```
echo '1' > /sys/class/gpio/gpio252/value
```

Per spegnere la retroilluminazione utilizziamo il comando:

```
echo '0' > /sys/class/gpio/gpio252/value
```

SCRIVIAMO UN'APPLICAZIONE

Cosa possiamo fare con un display così grazioso oltre a reindirizzare la console o l'interfaccia grafica del desktop? Possiamo scriverci le nostre applicazioni, in modo da comunicare con il GPIO e con eventuali altri sensori esterni. Le possibilità sono ovviamente infinite. Se però vogliamo mantenerci su qualcosa di estremamente semplice queste si ridu-

cono un po', senza tuttavia dover rinunciare a dei buoni risultati. Nel nostro caso ci manteniamo nel filone degli argomenti trattati nel libro citato all'inizio dell'articolo e nella ormai lunga serie di articoli presentati nella rivista e dedicati al mondo embedded ed a Raspberry Pi in particolare. Utilizzeremo il linguaggio Python per realizzare un'interfaccia grafica in grado di comandare le uscite di una scheda a quattro relè, in grado di commutare anche carichi alimentati con tensione di 220 Volt. Per interagire con il display grafico TFT sfrutteremo le funzionalità disponibili in pygame, la libreria del linguaggio Python destinata alla realizzazione di videogiochi ed applicazioni grafiche in genere. Nelle ultime distribuzioni di Raspbian la libreria pygame è già inclusa ed installata nella distribuzione. Se non lo fosse la installiamo semplicemente utilizzando il Package Manager di Python "pip" (Pip Installs Python), con il comando:

```
pip install pygame
```

Se non avessimo installato "pip" lo installiamo con il comando:

```
Apt-get install pip
```

Fatto? Proviamo il tutto utilizzando Python da linea di comando (rif. al solito libro). Verifichiamo che tutto funzioni digitando una dopo l'altra le seguenti istruzioni:

```
import pygame
import os
os.putenv('SDL_FBDEV', '/dev/fb1')
pygame.init()
lcd = pygame.display.set_mode((320, 240))
lcd.fill((255, 0, 0))
```

Listato 1

```
#!/usr/bin/python

# TFT_Rele.py.py

# Import librerie
import pygame
from pygame.locals import *
import os
from time import sleep
import RPi.GPIO as GPIO # Libreria Gestione GPIO

# Impostazione pin come output
GPIO.setmode(GPIO.BCM) # Utilizza numerazione GPIO
GPIO.setup(12, GPIO.OUT) # Set pin come OUTPUT
GPIO.setup(16, GPIO.OUT)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)

GPIO.output(12, False) # Porta i pin a off
GPIO.output(16, False)
GPIO.output(20, False)
GPIO.output(21, False)

# Definizione colori (R, G, B)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLACK = (0, 0, 0)

# Impostazione variabili di sistema per definizione device per
# display e touchscreen
os.putenv('SDL_FBDEV', '/dev/fb1')
os.putenv('SDL_MOUSEDRV', 'TSLIB')
os.putenv('SDL_MOUSEDEV', '/dev/input/touchscreen')

# Stato iniziale rele
pin12='off'
pin16='off'
pin20='off'
pin21='off'

# Costruisce la classe pygame
pygame.init()

# Toglie il puntatore del mouse dal display
pygame.mouse.set_visible(False)
lcd = pygame.display.set_mode((320, 240))

# Schermo nero
lcd.fill(BLACK)
pygame.display.update()

# Imposta font per le scritte
font_big = pygame.font.Font(None, 50)

# Dizionario Key=pin valore=coord. centro scritta
touch_buttons = {'12':(80,60), '16':(240,60), '20':(80,180), '21':(240,180)}

# Prepara la schermata iniziale
for k,v in touch_buttons.items():
    b = k + ' off' # Scritta con pin + off
    text_surface = font_big.render('%s'%b, True, RED) # Scritta rossa
    rect = text_surface.get_rect(center=v) # Centra la scritta
    lcd.blit(text_surface, rect) # Posiziona la scritta sul display
pygame.display.update()

# Ciclo principale del programma. Per sempre
while True:
    # Attesa evento dal display
    for event in pygame.event.get():
        if(event.type is MOUSEBUTTONDOWN): # Display toccato
```

(Continua)

Listato 1 (segue)

```
pos = pygame.mouse.get_pos()
print pos
elif(event.type is MOUSEBUTTONDOWN): # Via il dito dal display
    pos = pygame.mouse.get_pos() # Recupera posizione del dito
    print pos
    # Calcola in quale quarto dello schermo sta il dito
    x,y = pos
    if y < 120:
        if x < 160: # In alto a sinistra
            print 12
            v = touch_buttons.get('12') # Ritrova centro scritta
            if pin12 == 'off': # Se stato spento
                pin12 = 'on' # Cambia stato in on
                GPIO.output(12, True) # Livello pin alto
                b = '12 on' # Scritta on
                color = GREEN # Colore verde
            else:
                pin12 = 'off'
                GPIO.output(12, False)
                b = '12 off'
                color = RED
        else: # In alto a destra
            print 16
            v = touch_buttons.get('16')
            if pin16 == 'off':
                pin16 = 'on'
                GPIO.output(16, True)
                b = '16 on'
                color = GREEN
            else:
                pin16 = 'off'
                GPIO.output(16, False)
                b = '16 off'
                color = RED
    else:
        if x < 160: # In basso a sinistra
            print 20
            v = touch_buttons.get('20')
            if pin20 == 'off':
                pin20 = 'on'
                GPIO.output(20, True)
                b = '20 on'
                color = GREEN
            else:
                pin20 = 'off'
                GPIO.output(20, False)
                b = '20 off'
                color = RED
        else: # In basso a destra
            print 21
            v = touch_buttons.get('21')
            if pin21 == 'off':
                pin21 = 'on'
                GPIO.output(21, True)
                b = '21 on'
                color = GREEN
            else:
                pin21 = 'off'
                GPIO.output(21, False)
                b = '21 off'
                color = RED
    # Ridisegna il quarto di display interessato
    # e lo aggiorna
    text_surface = font_big.render('%s'%b, True, color)
    rect = text_surface.get_rect(center=v)
    lcd.fill(BLACK, rect) # Cancella la scritta precedente
    lcd.blit(text_surface, rect) # Posiziona nuova scritta
    pygame.display.update() # Aggiorna display
    sleep(0.3) # Un attimo di pausa
GPIO.cleanup()
```

```
pygame.display.update()
pygame.mouse.set_
visible(False)
lcd.fill((0,0,0))
pygame.display.update()
```

Tra le istruzioni importanti notiamo quella che imposta il device fb1 come framebuffer, l'init della libreria pygame. Successivamente incontriamo la definizione del display cui abbiamo dato nome "lcd" con dimensioni 320 x 240 pixel. Le altre istruzioni, come intuibile servono a cambiare colore allo sfondo dello schermo, utilizzando la solita codifica RGB. Concludiamo con un ultimo progetto che ci permette di azionare quattro relè con comandi touch sul display TFT. Utilizziamo la scheda 2846-RELAY4CH (reperibile presso Futura Elettronica) che porta a bordo quattro relè, è alimentata a 5 Volt ed è pilotabile con ingressi digitali diretti oppure optoisolati, che accettano tensioni da 1,5 a 5V e quindi è compatibile con i livelli di uscita del GPIO di Raspberry Pi. Scegliamo l'opzione diretta posizionando il jumper sui terminali COM e GND.

Ora colleghiamo la scheda a Raspberry Pi come nello schema di **Fig. 27**.

Collegiamo l'alimentazione e gli ingressi di pilotaggio dei relè a quattro pin del GPIO di Raspberry Pi. Noi abbiamo scelto i pin GPIO12, 16, 20 e 21, molto comodi da collegare sul GPIO della versione B+. Per l'alimentazione saldiamo una strip di connettori maschi sulla "replica" del GPIO sul modulo TFT e preleviamo l'alimentazione dai pin 2 (positivo) e 6 (massa) con dei cavetti femmina-femmina. Nel **Listato 1** vediamo il programma di pilotaggio dei relè. È sempre un programma

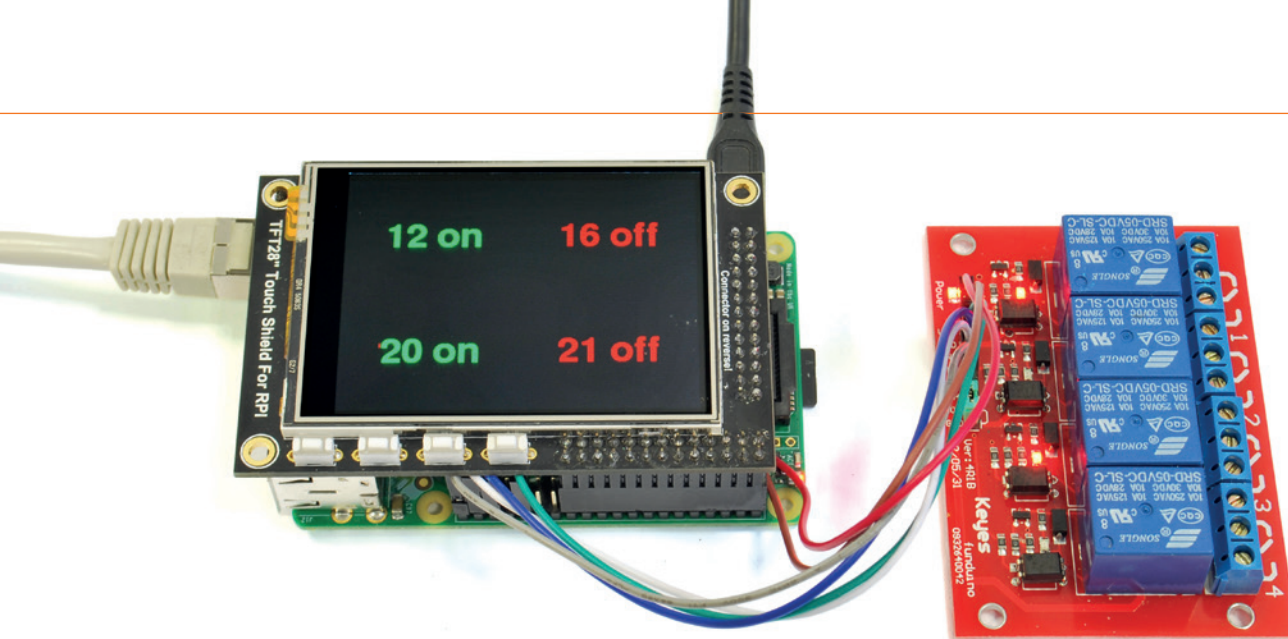


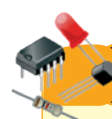
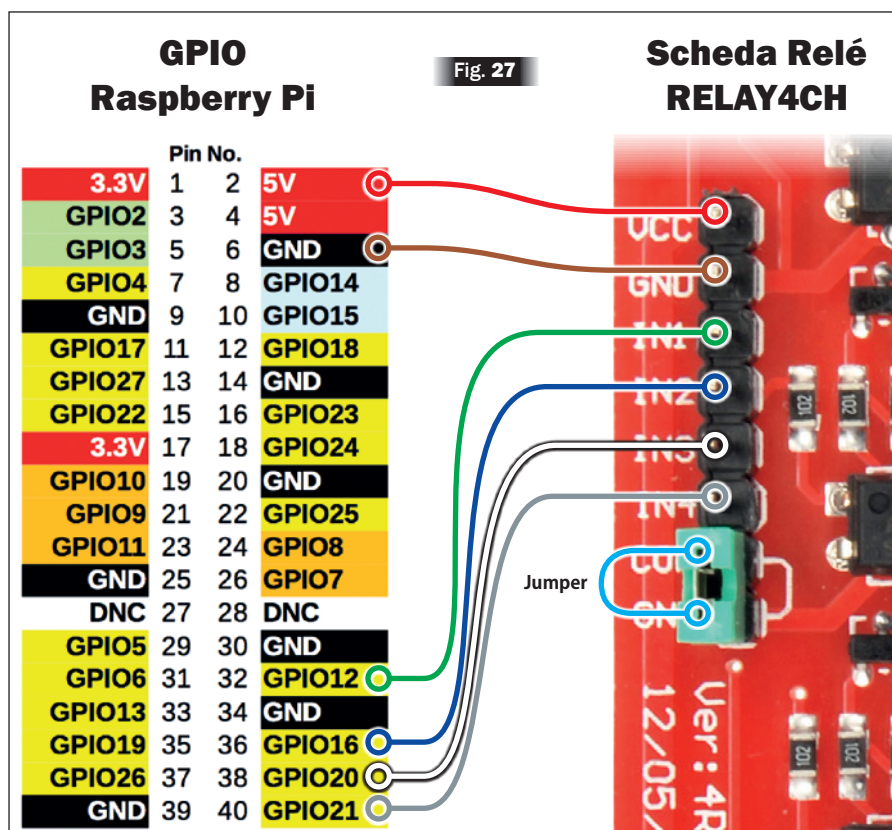
Fig. 28

Python. Per l'interfaccia grafica abbiamo utilizzato la libreria pygame, che vi abbiamo appena descritto. Il listato è ricco di commenti. La logica del programma consiste nella costruzione iniziale dell'interfaccia grafica. Il dizionario contiene le coordinate della posizione del centro dove vogliamo posizionare la scritta di ciascun

pin. Il ciclo "for" al di fuori del loop principale del programma prepara l'area del display con le scritte per i quattro pin. Al "nome" di ciascun pin viene aggiunta la dicitura "off". Il tutto viene colorato in rosso e posizionato in modo che il centro corrisponda alle coordinate contenute nel dizionario. Alla fine viene aggiornato fisi-

camente il display e le scritte diventano visibili. Una volta in esecuzione il programma, toccando una specifica scritta, il relè corrispondente cambia di stato, e la scritta indica lo stato corrente evidenziato dal colore associato, rosso con relè spento e verde con relè attivato, come visibile nella foto di Fig. 28.

Nei prossimi numeri vedremo come utilizzare il display con altri shield e con funzionalità arricchite, come la possibilità di azionare una periferica da display o da una pagina web. Per ora divertitevi a sperimentare.



per il MATERIALE

Il software presentato in questo articolo è scaricabile gratuitamente dal sito della rivista (www.elettronica.in.it). Le board utilizzate sono invece disponibili presso Futura Elettronica: Raspberry Pi B+ (Euro 36,00), TFT28TOUCHRASP (Euro 46,00) e scheda relè (cod. 2846-RELAY4CH, Euro 15,00) Tutti i prezzi si intendono IVA compresa.

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

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

Elettronica In

www.elettronica.in.it

oltre l'elettronica

TRASFORMIAMO RASPBERRY PI IN UN OSCILLOSCOPIO CON BitScope Micro

Trasformiamo Raspberry Pi in uno strumento di misura analogico/digitale con BitScope Micro, l'ultimo nato della generazione degli strumenti BitScope. Come dice il nome, la sua vocazione è la misura di segnali digitali. Il software è disponibile per le piattaforme GNU/Linux, Windows e Mac/OS X.

di MARCO MAGAGNIN

Il Bit Scope Micro, uno strumento di misura molto versatile e di qualità professionale, è un'ulteriore conferma della tendenza in atto che porta a mettere a disposizione degli sperimentatori e degli hobbysti dispositivi e strumenti sempre più professionali a costi sempre più abbordabili. In Fig. 1 è visibile lo strumento con la dotazione di cavetti inclusa nella confezione. Per quanto ci riguarda è la riprova che studiare ed imparare è sempre più alla portata di tutti e praticamente di tutte le tasche. BitScope Micro è il "piccolo" della famiglia di prodotti noti come "oscilloscopi USB" prodotti dalla

australiana BitScope Designs. Ci siamo focalizzati su questo specifico prodotto perché è stato pensato per funzionare anche con Raspberry Pi e con i micro PC con GNU/Linux in genere, oltre che con i PC normali, sia con GNU/Linux che con Windows e Mac/OS X.

Questa caratteristica non è sempre presente nei prodotti simili, in genere pensati unicamente per l'ambiente Microsoft. Possiamo dare una prima occhiata allo strumento analizzando la foto in Fig. 1. La quantità di cavetti disponibili, con relativo terminatore ci

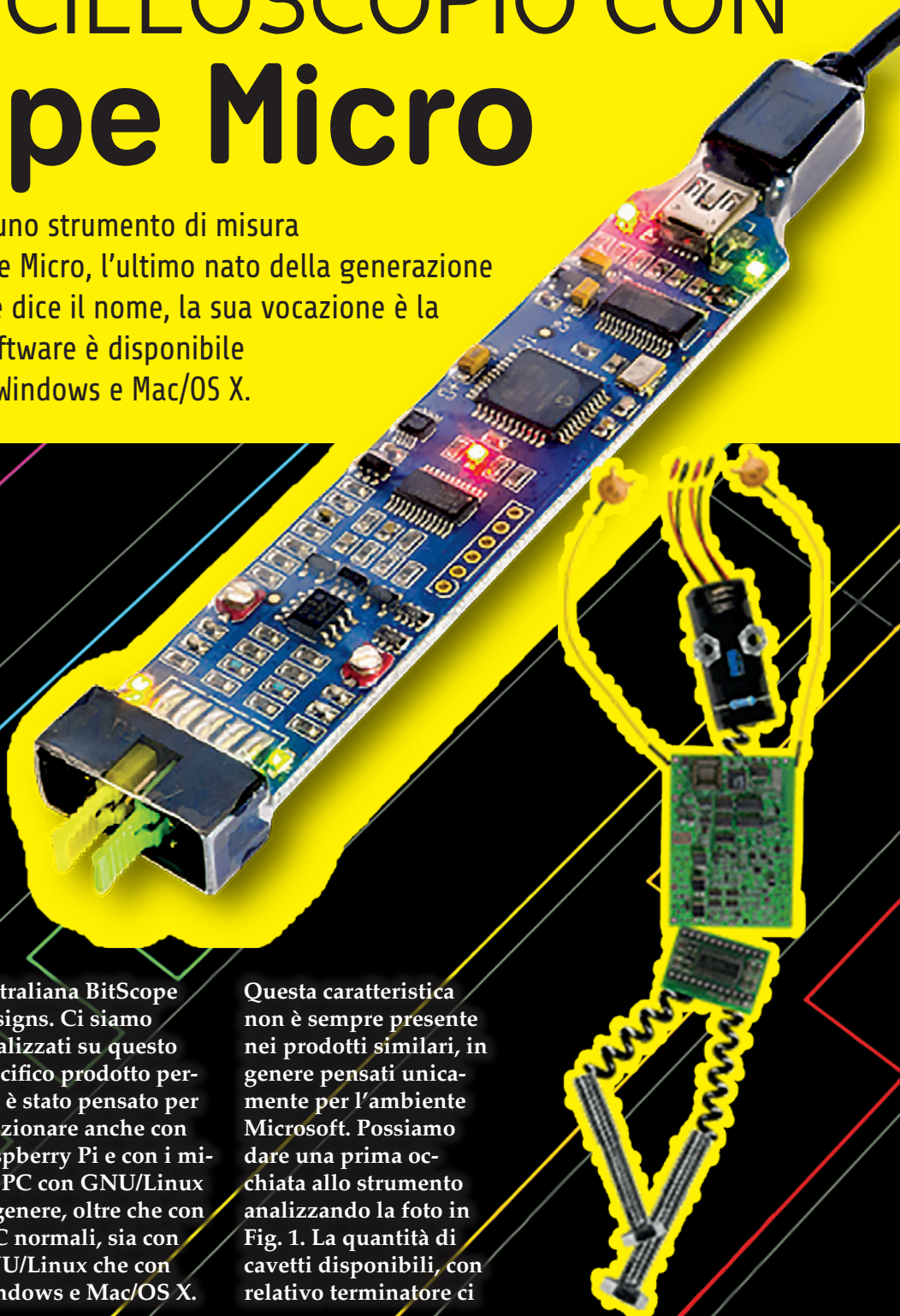
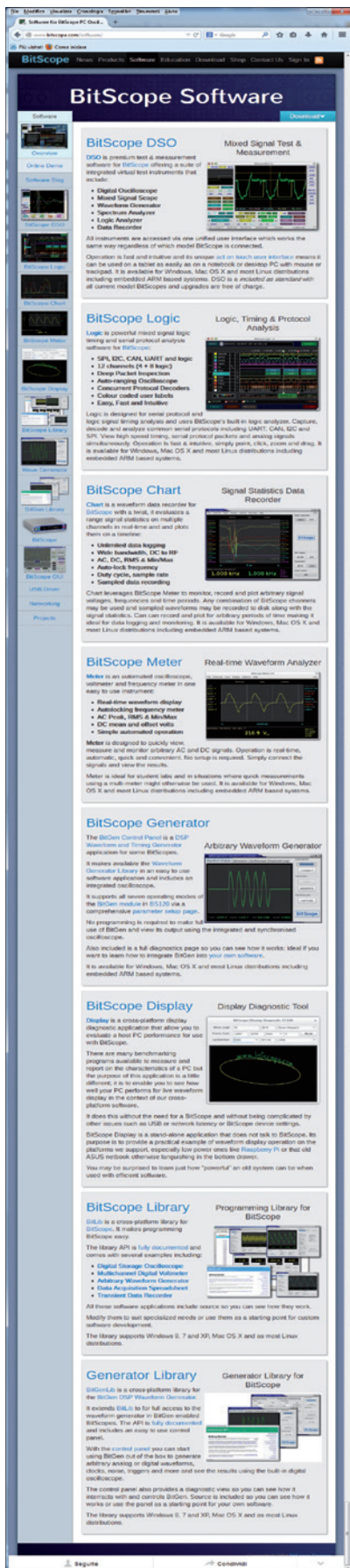


Fig. 2

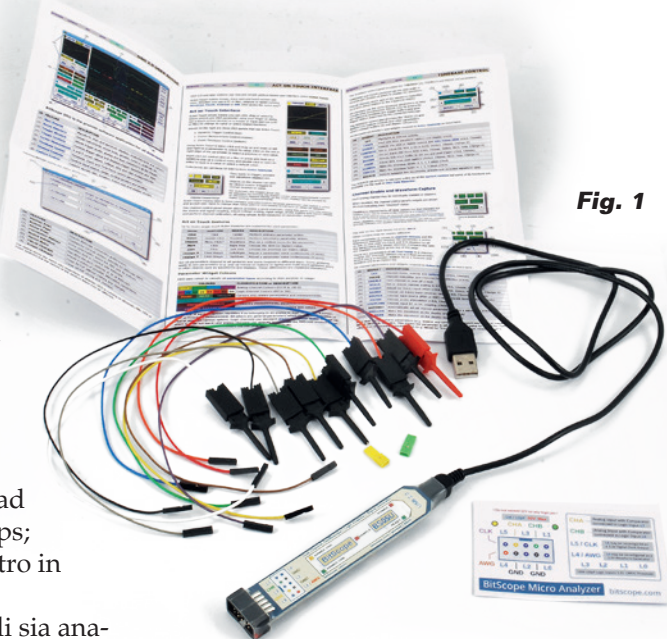


fanno pensare che lo strumento sia qualcosa di più di un semplice oscilloscopio. Infatti ci troviamo di fronte ad uno strumento con le seguenti caratteristiche:

- Oscilloscopio a due canali da 20 MHz di banda passante;
- Analizzatore logico ad otto canali da 40 Msps;
- Analizzatore di spettro in tempo reale;
- Generatore di segnali sia analogici che digitali;
- Tutti i segnali acquisiti possono essere registrati per essere visionati in un secondo tempo oppure per essere elaborati con strumenti informatici come un foglio di calcolo;
- Sono disponibili le API di accesso per utilizzare lo strumento all'interno delle proprie applicazioni;
- Possibilità di "remotizzare" su rete Internet la visualizzazione mediante un applicativo "server".

Tornando a Raspberry Pi, il BitScope Micro viene riconosciuto automaticamente non appena inserito nella presa USB. Se, come alcuni commenti sui blog fanno rilevare, il paragone tra il costo di una strumentazione tradizionale ed una strumentazione che prevede l'uso di un PC deve tenere conto, almeno in parte, anche del costo del PC, il BitScope Micro permette di contenere anche questa parte di costo. I costi contenuti, compatibilmente con le funzionalità di uno strumento di questo tipo, la compatibilità con i sistemi GNU/Linux embedded, la possibilità di remotizzare la gestione e la disponibilità di API di interfacciamento ne

Fig. 1



fanno anche una soluzione adatta ad essere installata permanentemente in sistemi di misura ed acquisizioni dati. Trattandosi di uno strumento di misura, più di tante descrizioni, vale una prova sul campo. Dunque installiamo il software sul nostro Raspberry Pi e poi colleghiamo ed utilizziamo lo strumento. Il sito di riferimento per il download del software e per la documentazione è raggiungibile all'indirizzo:

<http://my.bitscope.com>

Nel sito sono presenti diverse sezioni, una delle quali appositamente destinata all'utilizzo degli strumenti con Raspberry Pi. La guida per il software principale DSO (Digital Storage Oscilloscope) è reperibile all'indirizzo:

<http://www.bitscope.com/software/dso/guide/>

mentre la guida per la gestione degli analizzatori logici è disponibile all'indirizzo:

<http://www.bitscope.com/download/?i=182>

In Fig. 2 è visibile la gamma di

software disponibili per la gestione della famiglia di strumenti BitScope.

Iniziamo con l'installazione del software DSO. Per questo abbiamo bisogno di Raspberry Pi con collegati il monitor HDMI, il mouse e la tastiera. Il software è realizzato come un'applicazione desktop con interfaccia grafica e quindi ha poco senso una sua gestione da remoto, se non per la fase di installazione iniziale. Per la gestione in generale di Raspberry Pi e del sistema operativo Raspbian rimandiamo alla serie di articoli pubblicati sulla rivista ed al libro "Raspberry Pi – Il mio primo Linux embedded". Accendiamo Raspberry Pi, eseguiamo il login con l'utente "pi" e password "raspberrypi" e poi lanciamo l'applicazione desktop con il comando

```
startx
```

Può essere che il desktop si carichi automaticamente alla partenza, possiamo gestire questo comportamento utilizzando la funzionalità "Enable Boot to Desktop/Scratch" dello strumento "raspi-config", che lanciamo da una finestra di terminale con il comando

```
sudo raspi-config
```

se siete loggati come "root" non è necessario il comando "sudo".

Apriamo il browser di navigazione "Midori" (Fig. 3) ed andiamo all'indirizzo

<http://www.bitscope.com/pi>

Clicchiamo sul link "Download" nel rettangolo azzurro in alto a destra, sotto la scritta in grande "bit Scope Pi Oscilloscope" come indicato in Fig. 4.

Fig. 3



Fig. 4

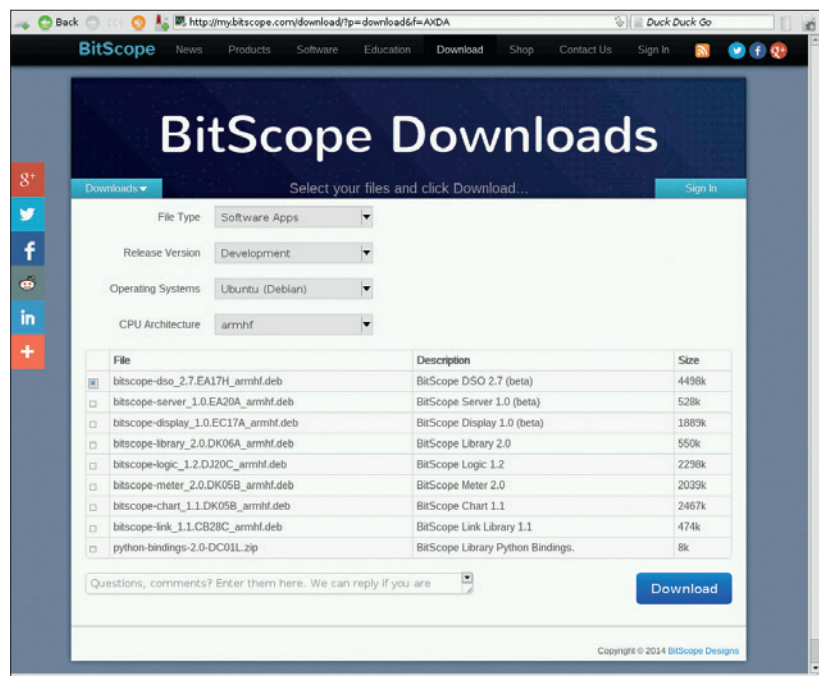


Fig. 5

Fig. 6

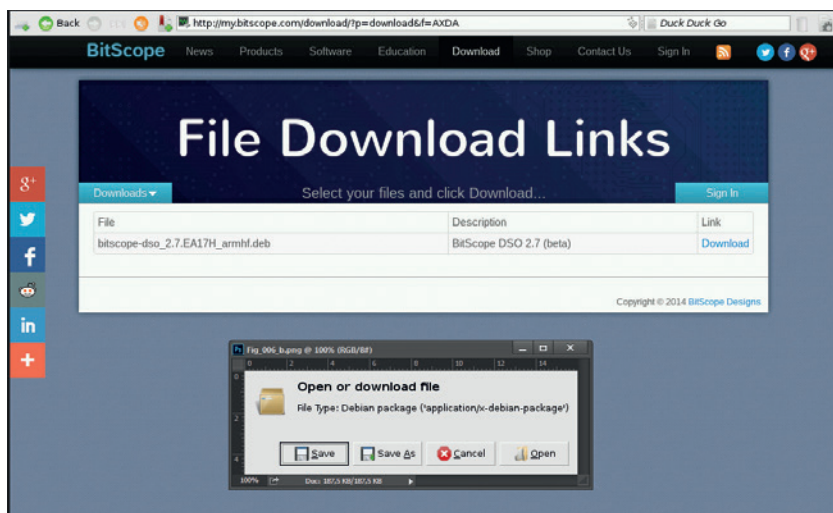


Fig. 7



Fig. 8

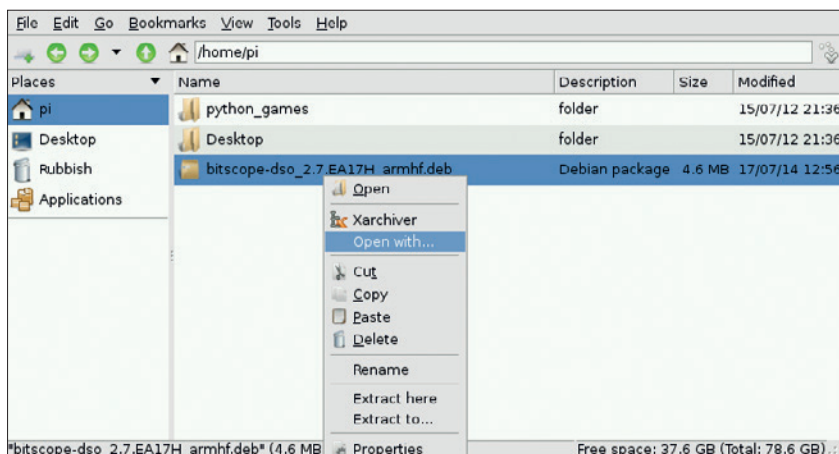
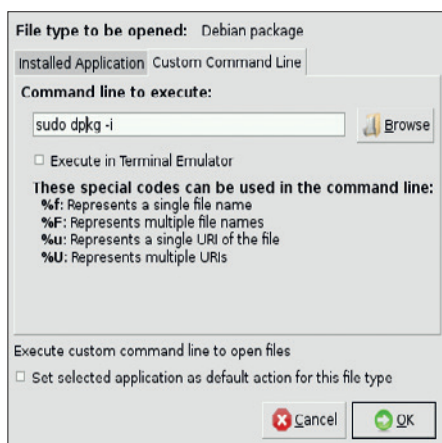


Fig. 9



Dalla pagina di download visibile in Fig. 5 selezioniamo il file

bitscope-dso_2.7.EA17H_armhf.deb

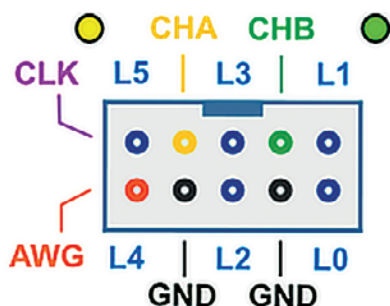
oppure più recente, se presente, e clicchiamo sul pulsante "Download". Nella pagina successiva (Fig. 6) clicchiamo sul link "Download" e nella finestra di dialogo scegliamo "Save". Il file verrà scaricato e

memorizzato nella cartella predefinita `/home/pi`. Una volta terminato il download, situazione che potremo controllare guardando la barra di progressione in basso a destra nella pagina, apriamo il file manager ed andiamo alla cartella `/home/pi`, che in genere è già quella visualizzata come predefinita (Fig. 7). Clicchiamo con il tasto destro del mouse sul file appena scaricato e nel menu di scelta rapida scegliamo l'opzione "Open with" (Fig. 8). Nella seconda tendina scegliamo l'opzione "custom command line", digitiamo (Fig. 9):

```
sudo dpkg -i
```

e premiamo il pulsante "OK". Non ci resta che aspettare pazientemente la fine dell'installazione. Alla fine dell'installazione, in genere, troveremo l'icona di lancio dell'applicazione DSO nella voce "Other" del menu "Start". Fermiamoci un attimo e prendiamo in mano il nostro piccolo strumento. Nella parte superiore un'etichetta riassume le funzionalità principali con la posizione dei LED che evidenziano le diverse modalità di funzionamento della scheda. Sulla sinistra è presente lo schema dei pin del connettore al quale vanno collegati i terminali di acquisizione e/o generazione dei segnali. Sono possibili molteplici combinazioni di acquisizione di segnali analogici, digitali, da bus i2c, spi, can, ecc. e/o generazione di segnali analogici o digitali. Nella confezione sono forniti 12 cavi dotati di terminali a pinza di diversi colori. A questo punto, per evitare continui inserimenti e disinserimenti di connettori abbiamo adottato lo schema colori dell'elettronica digitale, assegnando i colori in base alla posizione sul connettore dei segnali digitali, che sono riassunti

Fig. 10



nella Fig. 10.

Colleghiamo i connettori con i colori indicati, o come preferite, se volete adottare uno standard differente, e colleghiamo il cavo USB del Bit Scope ad una presa USB di Raspberry Pi. Come? Sono già utilizzate da mouse e tastiera? Per ora l'unica soluzione è di utilizzare un HUB USB, in questo caso autoalimentato. Fatto? Potete verificare che il Bit Scope è stato riconosciuto aprendo una finestra di terminale e digitando il comando:

`dmesg`

con il risultato visibile in Fig. 11. Ora lanciamo l'applicazione Bit Scope DSO (Fig. 12).

Clicchiamo sul pulsante "POWER" in alto a destra della finestra principale dell'applicazione. Si apre la finestra che ci permette di selezionare lo strumento che vogliamo gestire. Come è possibile vedere si possono utilizzare più strumenti contemporaneamente e collegati sia in "locale", via cavo USB, che in "remoto", tramite rete Internet, collegandosi all'indirizzo IP del server al quale è collegato il dispositivo. Selezioniamo il nostro Bit Scope Micro come indicato in Fig. 13 e premiamo il pulsante "OK". Si apre la finestra di gestione dello strumento visibile in Fig. 14. Ne

COLOURS	DESCRIPTION
CH-A CH-B CH-C CH-D	Analog Channel Colours (CH-A to CH-D)
D7 D6 D5 D4 D3 D2 D1 D0	Logic Channel Colours (D7 to D0)
POINT MARK ORIGIN	Cursors and related parameters and measurements.
TIME DOMAIN	Time measurements, parameters and values.
FREQUENCY DOMAIN	Frequency measurements, parameters and values.
GENERAL PARAMETERS	All other measurements, parameters and values.

approfittiamo per descrivere le diverse "sezioni" dello strumento, ciascuna indicata con un numero di riferimento. Nella parte principale dello schermo (rif. 1) sono presentati i risultati delle misurazioni in forma grafica. Le forme d'onda presentate dipendono dalla combinazione delle scelte selezionate

con i pulsanti sulla destra della pagina (rif. 2). In alto a sinistra è visibile la finestra con le impostazioni di trigger. Per il momento lo lasciamo impostato sul modo automatico predefinito. Nel manuale è indicato come modificare il comportamento e la sorgente del trigger utilizzando i controlli evidenziati come rif. 4. Il rif. 5

Fig. 11

```
192.168.0.43 - PuTTY
[ 268.151163] usb 1-1.2.4: SerialNumber: A4008HdF
[ 268.525058] usbcore: registered new interface driver usbserial
[ 268.526078] usbcore: registered new interface driver usbserial_generic
[ 268.526201] usbserial: USB Serial support registered for generic
[ 268.577240] usbcore: registered new interface driver ftdi_sio
[ 268.579893] usbserial: USB Serial support registered for FTDI USB Serial Device
[ 268.580618] ftdi_sio 1-1.2.4:1.0: FTDI USB Serial Device converter detected
[ 268.592931] usb 1-1.2.4: Detected FT232RL
[ 268.592969] usb 1-1.2.4: Number of endpoints 2
[ 268.592987] usb 1-1.2.4: Endpoint 1 MaxPacketSize 64
[ 268.593002] usb 1-1.2.4: Endpoint 2 MaxPacketSize 64
[ 268.593016] usb 1-1.2.4: Setting MaxPacketSize 64
[ 268.597420] usb 1-1.2.4: FTDI USB Serial Device converter now attached to ttyUSB0
root@raspberrypi:~#
```

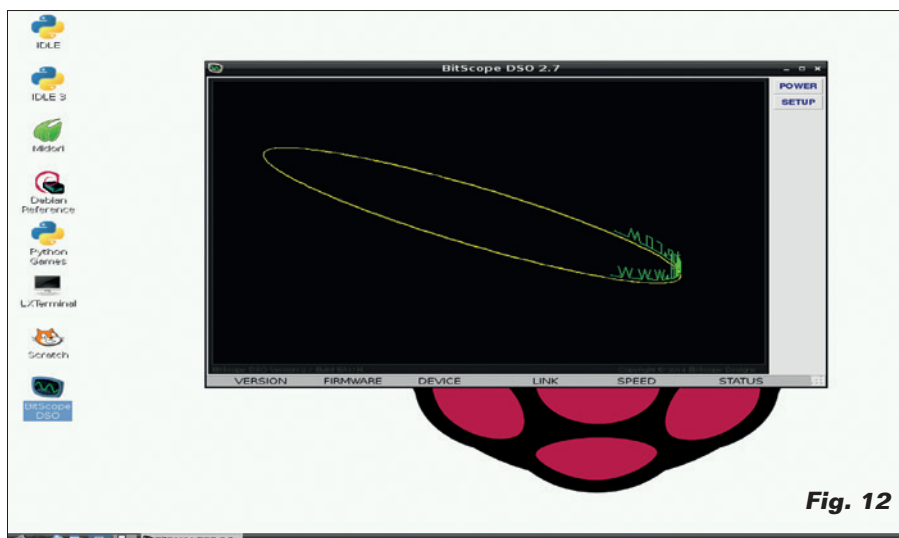


Fig. 12

Fig. 13

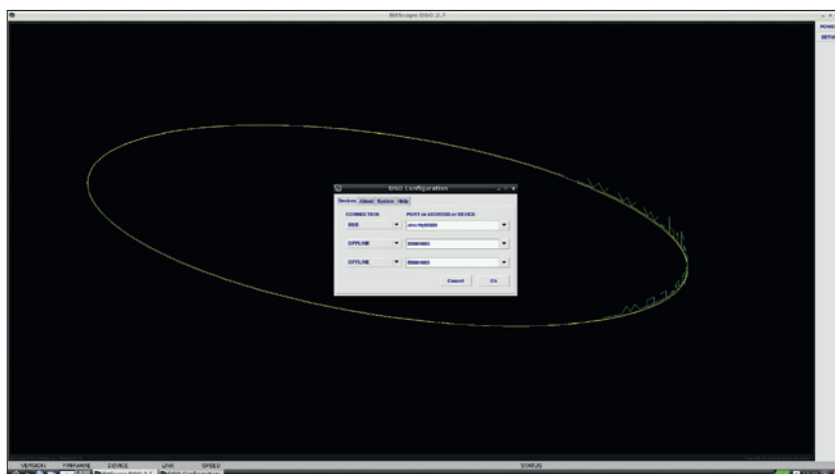
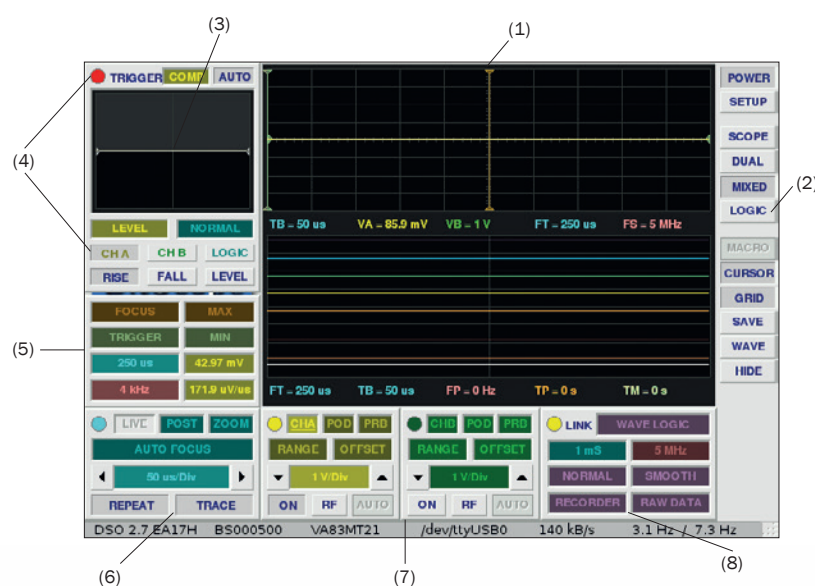


Fig. 14



indica il cursore per eseguire misurazioni accurate sui grafici presentati nella finestra principale. I controlli indicati dal rif. 6 permettono di impostare la base dei tempi in modo da ottimizzare la rappresentazione sul display delle forme d'onda acquisite, con maggiori o minori dettagli a seconda delle esigenze. A fianco, in basso (rif. 7) i controlli dei canali, attivi o meno sempre in base alle impostazioni dei pulsanti sulla destra, permette di impostare gli ingressi, le sorgenti dei segnali, l'intervallo di rappresentazione, la posizione verticale ed il fattore di scala. I controlli indicati dal rif. 8 permettono di configurare

la frequenza di acquisizione dei campioni, la durata, la frequenza di refresh video e la modalità di rappresentazione.

PRIMO TEST

Per verificare che tutto funzioni a dovere "attiviamo" il canale A dello strumento oscilloscopio, modifichiamo la base dei tempi, impostando il valore 5 msec/div cliccando sulla freccia a sinistra (o a destra) finché non vediamo visualizzato il valore richiesto. Impostiamo a 200 mV/div la scala delle tensioni. A questo punto impugnate il terminale del cavetto che fa capo al canale A, nel nostro caso quello di colore viola.

Spingete il pulsante a molla in modo da fare fuoriuscire il piccolo terminale metallico a pinza e toccatelo con un dito. Nessun timore, non c'è pericolo. Dovreste vedere una forma d'onda, più o meno sinusoidale, come quella mostrata in Fig. 15.

Ogni ciclo della forma d'onda copre quattro intervalli da cinque millisecondi l'uno, cioè $0,005 \times 4 = 0.02$ secondi. Dato che la frequenza è il reciproco dell'intervallo di tempo possiamo calcolare la frequenza di $1/0,02 = 50\text{Hz}$. La frequenza della rete elettrica, siccome qualunque conduttore attraversato da una corrente alternata si trasforma in un trasmettitore di onde radio e il nostro corpo funge da ricevente, e la forma d'onda ricevuta può essere evidenziata con un oscilloscopio, anche se un po' distorta. L'oscilloscopio non misura direttamente le frequenze, ma tensioni e intervalli di tempo. La frequenza deve essere derivata con il calcolo del reciproco. Nel numero 186 della rivista abbiamo presentato Redpitaya, lo strumento virtuale basato sul chip Zynq 7010, che sfrutta la componente FPGA per la misura e la generazione di segnali. BitScope può essere impiegato in modo analogo per misurare e generare frequenze fino a circa 20 MHz. Vogliamo invece sottolineare la peculiarità di questo strumento virtuale per quanto riguarda l'acquisizione e la generazione di segnali digitali, che ne fanno lo strumento ideale per lo sperimentatore evoluto che desidera realizzare progetti in campo embedded con periferiche elettroniche collegate e gestite da microcontrollori e/o microcomputer con sistema operativo. Siccome BitScope utilizza il bus USB per collegarsi ai dispositivi di visualizzazione può essere anche utilizzato con

smartphone e tablet, se dotati delle opportune interfacce. Utilizzando le funzionalità di analisi di segnali digitali possiamo impiegare per visualizzare anche i dati che transitano sui diversi bus che vogliamo utilizzare nei nostri progetti, come seriali, SPI, i2C, CAN, ecc. permettendo un debug delle nostre applicazioni molto più veloce, puntuale e ... professionale. Probabilmente, con i risparmi di tempo, già nei primi utilizzi ci si ripaga abbondantemente del costo del nostro piccolo ... aiutante.

SCHEMA ELETTRICO DEL CIRCUITO DI PROVA

Per sperimentare con il nostro BitScope abbiamo deciso di utilizzare un vecchio integrato contatore digitale, il CD4029, le cui caratteristiche sono descritte nel riquadro di riferimento dedicato. Utilizziamo il nostro strumento per rendere "visibile" il comportamento "logico" del contatore. Per fare questo abbiamo realizzato un semplice circuito che ci permette di "sperimentare" le funzionalità principali di BitScope. Il circuito è composto da due sezioni. La prima sezione è un contatore binario con uscita digitale mentre la seconda sezione è un convertitore digitale analogico con la tensione di uscita proporzionale ai valori binari presenti sui pin di uscita del contatore. Il tutto è visibile nello schema elettrico di Fig. 16. L'alimentazione è filtrata e stabilizzata con un condensatore elettrolitico C1 da 100µF ed uno (C2) da 10nF in parallelo all'alimentazione stessa. I pin "PRESET ENABLE" e "CARRY IN" dell'integrato sono collegati a massa in modo da forzare il conteggio a partire da zero, senza tenere conto della configurazione dei pin di PRESET J1..J4. Il pin

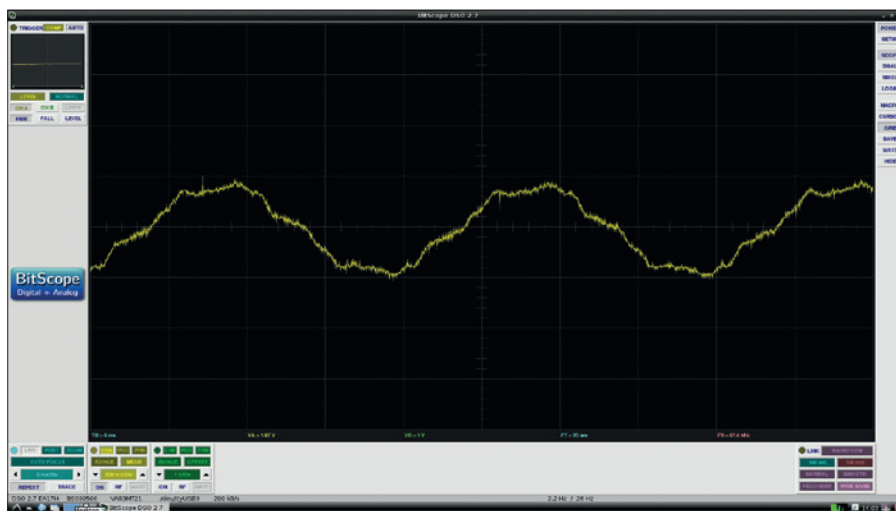


Fig. 15

"BINARY/DECADE" è collegato al positivo dell'alimentazione (livello alto) per forzare il contatore in avanti; per farlo contare a "ritroso" basta collegare il medesimo pin a massa (livello basso). La resistenza di pull-up R1 mantiene stabile il livello del pin di "CLOCK" durante i transienti. La seconda sezione del circuito è costituita dal convertitore digitale analogico a rete di resistenze a quattro bit. Il DAC è di tipo R-2R e lavora con qualsiasi valore di resistenze che rispettino il rapporto due a uno. Noi abbiamo scelto resistenze da 270

kohm e 560 kohm. Da tenere presente che più le resistenze scelte stanno tra di loro nel rapporto due a uno e più accurato sarà il funzionamento del DAC. Per una descrizione accurata del funzionamento del DAC di tipo R-2R vi rimandiamo all'articolo "Convertitore digitale analogico a rete di resistenze" in questo stesso numero della rivista. Ricordiamo solo che questa configurazione DAC non è in grado di produrre in uscita una scala di tensioni che arrivino ad eguagliare il livello delle linee di I/O digitale; la tensione massima che può essere

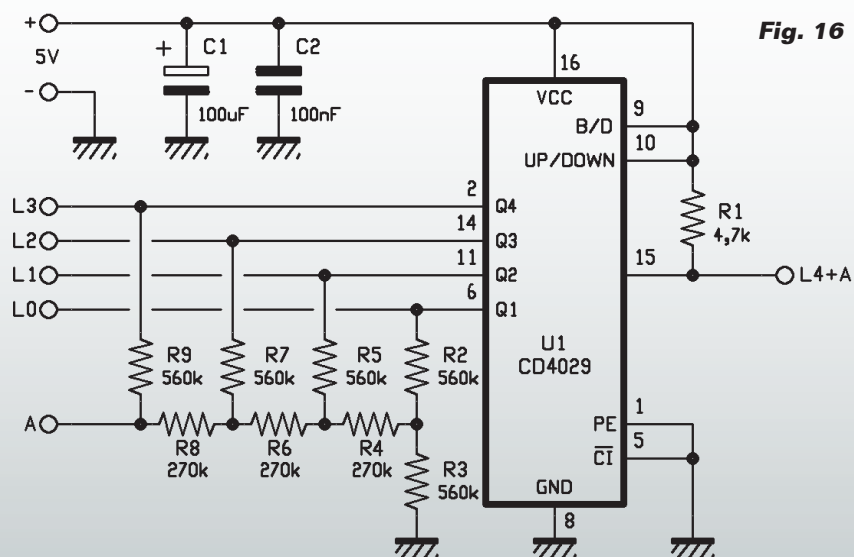


Fig. 16

L'integrato contatore CD4029

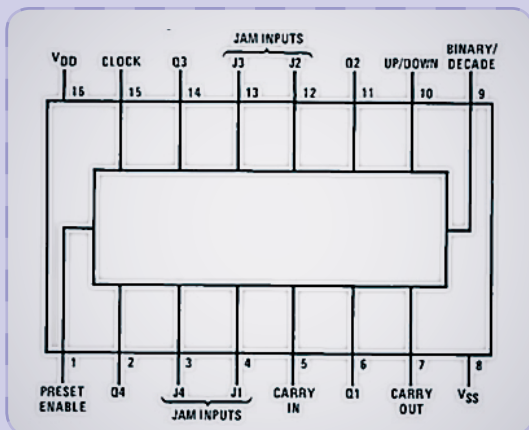


Fig. A

Scopo dell'integrato è di fornire in uscita il risultato in forma digitale del conteggio dei cambiamenti di livello del segnale di clock in ingresso. Il diagramma a blocchi di **Fig. A** mostra la disposizione dei segnali sui piedini dell'integrato. I terminali più importanti sono costituiti dal pin "CLOCK" e "UP/DOWN". Il primo è il pin dove viene applicato il segnale che fornisce il "ritmo" del conteggio

mentre il secondo permette di impostare il tipo di incremento, in somma o in sottrazione. L'uscita del conteggio è presentata sui pin Q1..Q4, in forma binaria o digitale a seconda dello stato del pin "BINARY/DECADE". La configurazione interna del contatore è costituita da una serie di gate logici configurati in modo da implementare un operatore aritmetico di addizione. Normalmente il

contatore incrementa di uno il valore della "word" di quattro bit in uscita ad ogni fronte di salita intercettato sul pin di "CLOCK". Se l'ingresso "UP/DOWN" è configurato con il livello basso, il contatore "conta" all'indietro, e sottrae un'unità ad un ciclo del clock. Dopo un intero ciclo di conteggio (0000-1111 o 1111-0000) il contatore riprende il conteggio dal valore di partenza. I segnali "CARRY OUT" e "CARRY IN" sono

utilizzati quando vengono impiegati diversi contatori in cascata. Gli integrati contatori possono essere collegati in cascata per realizzare contatori con numeri di cifre maggiori di quattro. Per fare questo si collega il terminale "CARRY OUT" al terminale "CARRY IN" dell'integrato successivo. Connettendo il "CARRY OUT" di un contatore al "CARRY IN" di un secondo contatore si realizza un contatore ad

Signal	Type	Descrizione
CLOCK	Mixed	Ingresso CLOCK del contatore.
UP/DOWN	Input	Conteggio Incrementale o Decrementale.
Q4..Q1	Output	Uscita binaria codificata su 4 bit.
J4..J1	Inputs	Ingresso binario del valore iniziale da cui partire per il conteggio, in formato binario su 4 bit.
BINARY/DECADE	Input	Modalità di conteggio
CARRY IN	Input	Pin di continuazione conteggio in ingresso di un contatore a monte.
CARRY OUT	Output	Pin di continuazione conteggio in uscita verso un contatore a valle.
PRESET ENABLE	INPUT	Abilita il conteggio a partire dai valori impostati in J4..J1

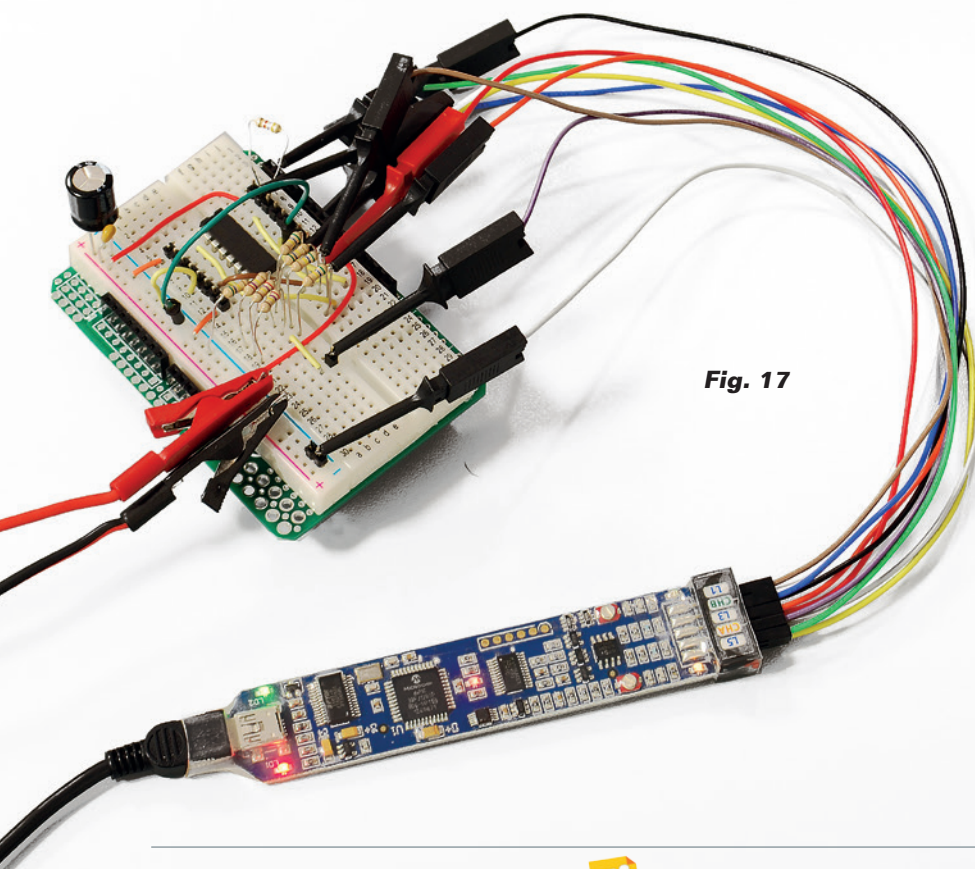


Fig. 17

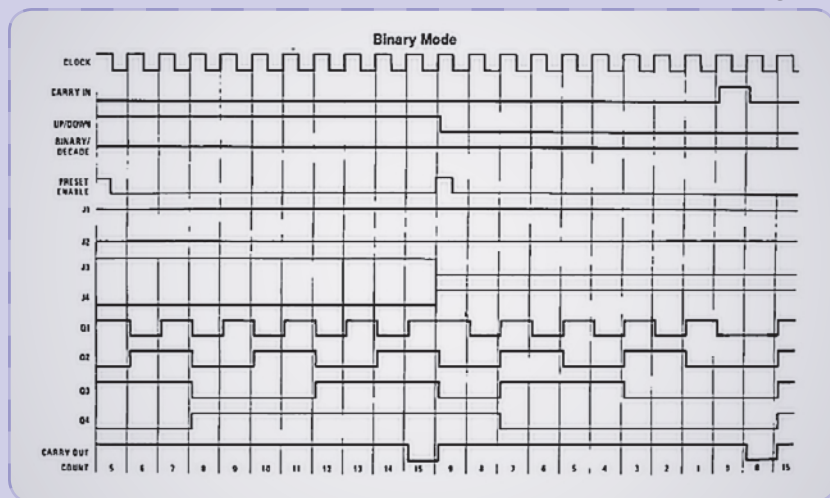
prodotta da questo tipo di DAC resistivo è dato dalla seguente formula, dove V_{DIG} è la tensione dell'ingresso digitale (**Fig. 16**):

$$V_{MAX} = (V_{DIG}) \frac{2^N - 1}{2^N}$$

Oltre alla tensione di fondo scala, l'altra importante caratteristica da considerare in un circuito DAC è la risoluzione, dove per risoluzione si intende la minima variazione di segnale d'uscita che il dispositivo può generare che è pari alla tensione del bit acceso meno significativo. Questa grandezza può essere calcolata con la formula:

$$RES = \frac{V_{DIG}}{2^N}$$

Nella formula la V_{DIG} è la tensio-

Fig. B

otto bit dove i pin Q4..Q1 del primo contatore rappresentano i quattro bit "low" e i pin Q4..Q1 del secondo i bit "high". Il segnale "CARRY" è generato ogni volta che il contatore raggiunge il suo massimo limite di conteggio e va in "overflow" per ricominciare il conteggio da capo. Il pin "PRESET ENABLE" ed i pin JAM "J4..J1"

permettono di preimpostare il conteggio con un valore di partenza predefinito, impostato in binario sui pin stessi. Il diagramma della logica dell'integrato visibile in **Fig. B** è il miglior punto di partenza per comprenderne il funzionamento. Per ogni ciclo di clock i quattro bit in uscita presentano la configurazione in

binario del "numero" raggiunto dal contatore. Essendo un contatore a quattro bit, una volta che tutti i livelli sono alti, a rappresentare il decimale 15, il contatore si azzerava e ricomincia a contare da capo. A seconda del livello dell'ingresso UP/DOWN il contatore funziona in modo incrementale o decrementale.

nel rispetto dello schema colori standard e rispettato anche dai cavetti forniti con BitScope:

L0 – Bianco – secondo lo standard dovrebbe essere nero, ma è stata preferita questa soluzione per evitare di confonderlo con le masse. Collegato a Q1
L1 – Marrone. Collegato a Q2;
L2 – Rosso. Collegato a Q3
L3 – Arancione. Collegato a Q4
L4 – Giallo. Utilizzato come uscita del generatore di segnali e collegato all'ingresso Clock dell'integrato
L5 – Verde
L6 – Blu
Canale A – Viola. Collegato anch'esso all'ingresso di clock per per visualizzare la forma d'onda prodotta dal generatore. GND (Grigio) del canale A collegato alla massa del nostro circuito.

ne di alimentazione delle linee digitali di I/O e N è il numero di bit previsti dal DAC, quattro nel nostro caso. Ciascuna uscita del contatore binario è collegata alla resistenza del corrispondente bit di ingresso: Q1 collegato a R2, Q2 collegato a R5, Q3 collegato a R7 e Q4 collegato a R9.

ANALISI DEL COMPORTAMENTO DIGITALE DEL CONTATORE CON BITSCOPE

Abbiamo inserito l'integrato su una piccola breadboard e realizzato il circuito visibile nella foto di **Fig. 17**. Abbiamo "esposto" i pin dell'integrato sulla breadboard utilizzando due strip di pin a passo 2,4mm in modo da poterli utilizzare agevolmente sia con i cavetti a doppio terminale che con i terminali a pinza dello strumento Bit Scope.

Collegiamo l'alimentazione ai piedini VDD e VSS utilizzando un alimentatore a 5V oppure prelevando la tensione dai pin 2 (+5V) e 2 (GND) di Raspberry Pi. I terminali di BitScope sono collegati nel seguente ordine,

Mettiamo in esecuzione il software DSO su Raspberry Pi e configuriamo la modalità di visualizzazione. Attiviamo le funzioni "MIXED" e "WAVE" premendo i rispettivi pulsanti sul pannello pulsanti di destra. Nel "WAVE CONTROL", in alto a sinistra, verifichiamo che

**Fig. 18**



Fig. 19



Fig. 20

“ENABLE” sia attivato, selezioniamo “STEP” come forma d’onda per generare un’onda quadra, impostiamo “FREQUENCY” a 4 kHz, lasciamo “SIMMETRY” a 50%, e “AMPLITUDE” a 3,3V (la massima tensione di uscita). Impostiamo “DC OFFSET” a 0V. In basso a destra impostiamo la base dei tempi a 500 ms/div. Nella sezione del canale A verifichiamo che il canale sia attivo ed impostiamo l’ampiezza verticale ad 1V/div. Ora guardiamo il display. Nella metà superiore vediamo la forma d’onda analogica in uscita dal generatore di segnali. Un’onda quadra di frequenza 4 kHz.

Nella parte bassa del display vediamo la sezione digitale. In alto la traccia del clock, che rispecchia l’onda quadra generata. In basso la traccia bianca del bit 0 (uscita Q1) del contatore binario. Partendo da zero al primo fronte di salita l’uscita Q1 assume il valore “0”, al secondo fronte assume il valore “1”. Le tracce delle uscite Q2 (marrone), Q3 (rossa) e Q4 (arancione) rimangono a livello basso. Al successivo fronte di salita Q1 torna basso e Q2 assume il valore “1” cioè “01” in binario che equivale a 2 in decimale. Al passo successivo Q2 rimane alta e anche Q1 torna alta a significare il valore

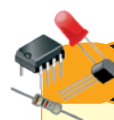
“11” (in decimale “3”). Così via fino a 15 (“1111”). Poi il contatore si azzerà ed il conteggio ricomincia da capo. Vediamo il risultato in Fig. 18.

Se portiamo a massa il pin UP/DOWN (livello basso) ad ogni inizio del ciclo di conteggio vedremo tutte le uscite al livello alto (valore “15”) per poi contare a ritroso fino a zero.

DAL DIGITALE ALL’ANALOGICO

Per visualizzare l’incremento o il decremento del conteggio spostiamo il cavetto che fa capo al canale A, di colore viola, dall’ingresso “CLOCK” all’uscita della rete di resistenze del convertitore DAC. A questo punto vediamo il risultato nella Fig. 19, dove la tensione sull’uscita del DAC sale per 15 “scalini” per poi riazzerarsi e ricominciare con un nuovo conteggio.

Se spostiamo il cavetto che collega il pin “UP/DOWN” a massa (fino ad ora è collegato al positivo della tensione), otterremo il conteggio a “ritroso” dal valore massimo, via via a scendere fino a zero, per poi ricominciare (Fig. 20).



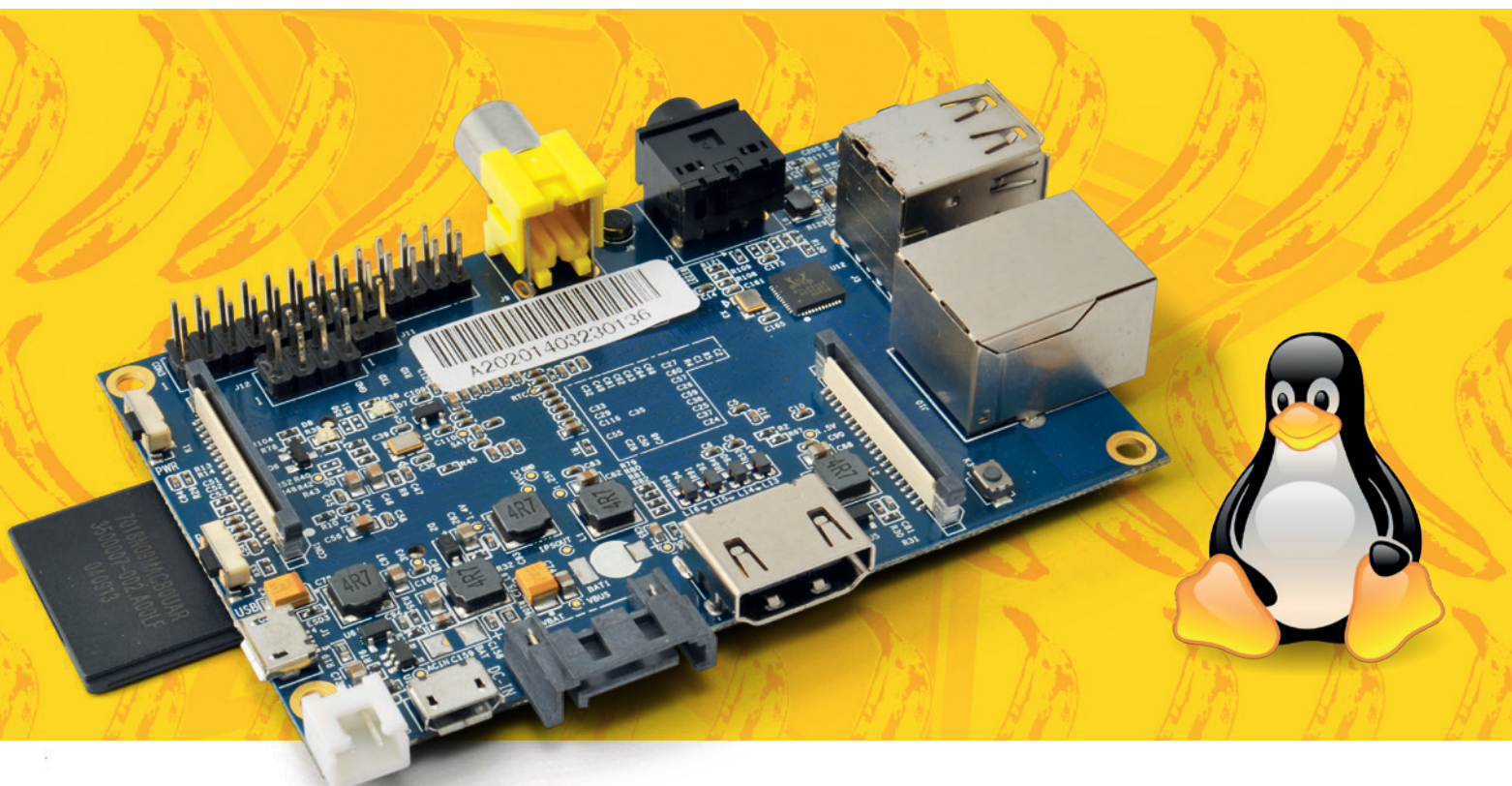
per il MATERIALE

Questo interessante dispositivo viene prodotto dalla società australiana BitScope Designs ed è distribuito nel nostro paese da Futura Elettronica (codice prodotto: BitScopeMicro) al prezzo di Euro 145,00 IVA compresa.

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



Un breve richiamo sugli aggiornamenti software rilasciati dal produttore di Banana Pi. Nuovi porting di sistemi operativi, una compatibilità sempre maggiore con Raspberry Pi e nuove configurazioni hardware ne fanno sempre più un'ottima soluzione per i nostri progetti.

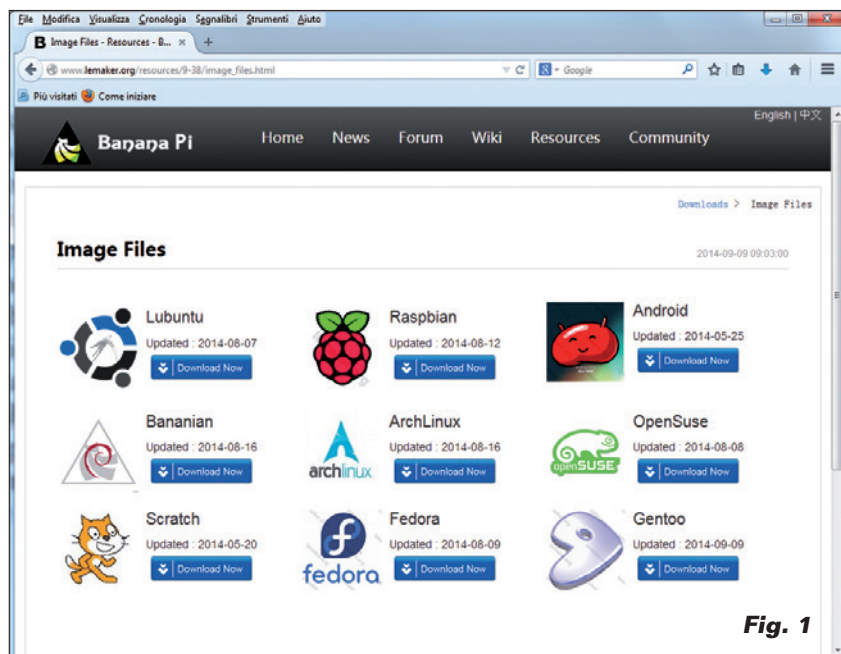


Banana Pi, UPGRADE SOFTWARE

di MARCO MAGAGNIN

Abbiamo descritto BananaPi nel numero 188 della rivista. In quell'articolo abbiamo sottolineato la riuscita della soluzione proposta da Lemaker, la società che produce il microcomputer, tanto da tenerla sempre in vista sulla nostra scrivania. L'unica lieve perplessità è stata sulla scelta di voler riprodurre anche la forma della concorrente Raspberry Pi. In ogni caso abbiamo iniziato

ad utilizzare Banana Pi in quei progetti dove Raspberry Pi iniziava a non fornire più prestazioni adeguate, soprattutto in applicazioni di raccolta dati da sensori ed elaborazioni di scenari e regole di auto-intervento. Grande aiuto ci è venuto dal poter collegare al microcomputer un disco fisso, direttamente al connettore SATA disponibile sul microcontrollore stesso. Nel frattempo la fonda-



non ha mai deluso, che vanta una comunità di sviluppo tra le più vaste del pianeta e che “gira” su una gamma di sistemi di elaborazione che spaziano dalla piccola “Arietta G25” di ACME Systems, alla Raspberry Pi e schede simili come Banana Pi, ai computer domestici, fino ai grandi server di aziende e provider fino ai più giganteschi supercomputer di università e centri di ricerca in tutto il mondo. Date queste premesse, ancora una volta non siamo stati deleusi. BananaPi ha presentato una nuova versione del suo sistema operativo basato su “Debian” e compatibile con Ra-

zione Raspberry presentava la versione B+ della loro scheda, che vi abbiamo presentato nello stesso numero 188 della rivista, dotata di un maggior numero di periferiche esterne (USB). Segno di un mondo in forte evoluzione che, se da un certo punto di vista può ingenerare confusione ed incertezza, dall'altro offre sempre nuove opportunità di sviluppo di progetti ed applicazioni sempre più evolute e con costi sempre più abbordabili. Per darci una certa garanzia di continuità e per minimizzare il rischio di veder vanificati gli sforzi di studio, progettazione ed apprendimento nel mondo embedded, come stiamo sostenendo ormai da lungo tempo, abbiamo fissato come elemento "invariante" della nostra attività, il fatto di "puntare" su prodotti embedded che supportassero GNU/Linux come sistema operativo e la distribuzione "Debian" in particolare. Difficile motivare con argomenti strettamente "tecnici" questa scelta. Potremmo indicare una propensione personale per un sistema operato che per decenni

spian di Raspberry Pi, insieme all'ampliamento dell'offerta con altri sistemi operativi, portati ex-novo od aggiornati. Per avere una panoramica dell'offerta basta andare all'indirizzo http://www.lemaker.org/resources/9-38/image_files.html per trovare la situazione aggiornata del software di sistema disponibile nella pagina di download visibile in Fig. 1.

Invece, per quanto riguarda l'evoluzione dell'hardware una possibile risposta la possiamo trovare all'indirizzo http://www.sinovoip.com.cn/ecp_view.asp?id=554 dove sono proposti microcomputer di "classe" BananaPi su schede in grado di ospitare a bordo il disco fisso, oltre ad una maggior dotazione di periferiche esterne (Fig. 2). Con queste schede possiamo realizzare dei veri e propri micro PC o sistemi dedicati come piccoli server NAS o di stampa oppure macchine di rete come router o gateway. Il tutto con le nuove versioni di sistema operativo che abbiamo citato.

La nostra attenzione va ovviamente alla nuova versione di Raspbian 3.1, la distribuzione Debian Wheezy che "gira" su Raspberry Pi. Come visibile in Fig. 3, in questa nuova versione sono disponibili alcuni pacchetti interessanti come l'IDE di Arduino e gli ambienti di sviluppo per il linguaggio di programmazione didattico Scratch. Quest'ultimo è disponibile sia nella versione "base" che con le estensioni in grado di interfacciare il GPIO.

Scratch è il linguaggio realizzato dall'MIT per introdurre la programmazione dei calcolatori elettronici nel modo più semplice possibile e con una modalità adatta a tutte le età, anche agli studenti delle scuole elementari

Microsoft rilascia una versione di Windows per Intel Galileo

Vi abbiamo presentato la scheda Galileo nel numero 181 come una scheda in tecnologia Intel x86 (in effetti è prodotta da Intel stessa) compatibile Arduino e con GNU/Linux a bordo, anche se in una versione decisamente preliminare. Dopo quasi un anno di silenzio, Microsoft annuncia una versione personalizzata del sistema operativo Windows, rivolta a tutti i possessori di una scheda Intel Galileo di prima generazione, aggiornata con il firmware 1.0.2. Sia la nuova release software di Windows, basata su Windows 8.1, sia la scheda Intel Galileo entrano a far parte del Windows Developer Program for IoT (Internet of Things), che Microsoft sta evidentemente spingendo, ampliando la possibilità di utilizzare la particolare release Windows ad un più ampio numero di possessori della scheda Galileo (sino

ad ora tale possibilità era limitata ad alcuni sviluppatori selezionati da Microsoft). Il programma Windows Developer Program for IoT, è finalizzato a sperimentare l'utilizzo di Windows con nuovi dispositivi Internet-connected e proseguirà anche in futuro con il supporto alle schede Intel Galileo di seconda generazione, già confermato dal portavoce della casa di Redmond. La versione di Windows per Galileo è compatibile con Arduino e Microsoft spera che alcuni dei prototipi realizzati dagli sviluppatori interessati possa tradursi in un prodotto commerciale. Per il momento rimaniamo sempre in attesa degli sviluppi.

e medie. In particolare stiamo pianificando di dedicare a questo linguaggio alcuni tutorial per introdurre al mondo della programmazione chi non ha mai affrontato questo argomento perchè intimorito dai linguaggi di programmazione tra-

dizionali. Data la natura della nostra rivista, come di consueto, sarà nostra cura dedicare spazio all'utilizzo di questo linguaggio per interfacciare moduli elettronici di "contatto" con il mondo reale ed anche di gestire la scheda Arduino Uno. In questa

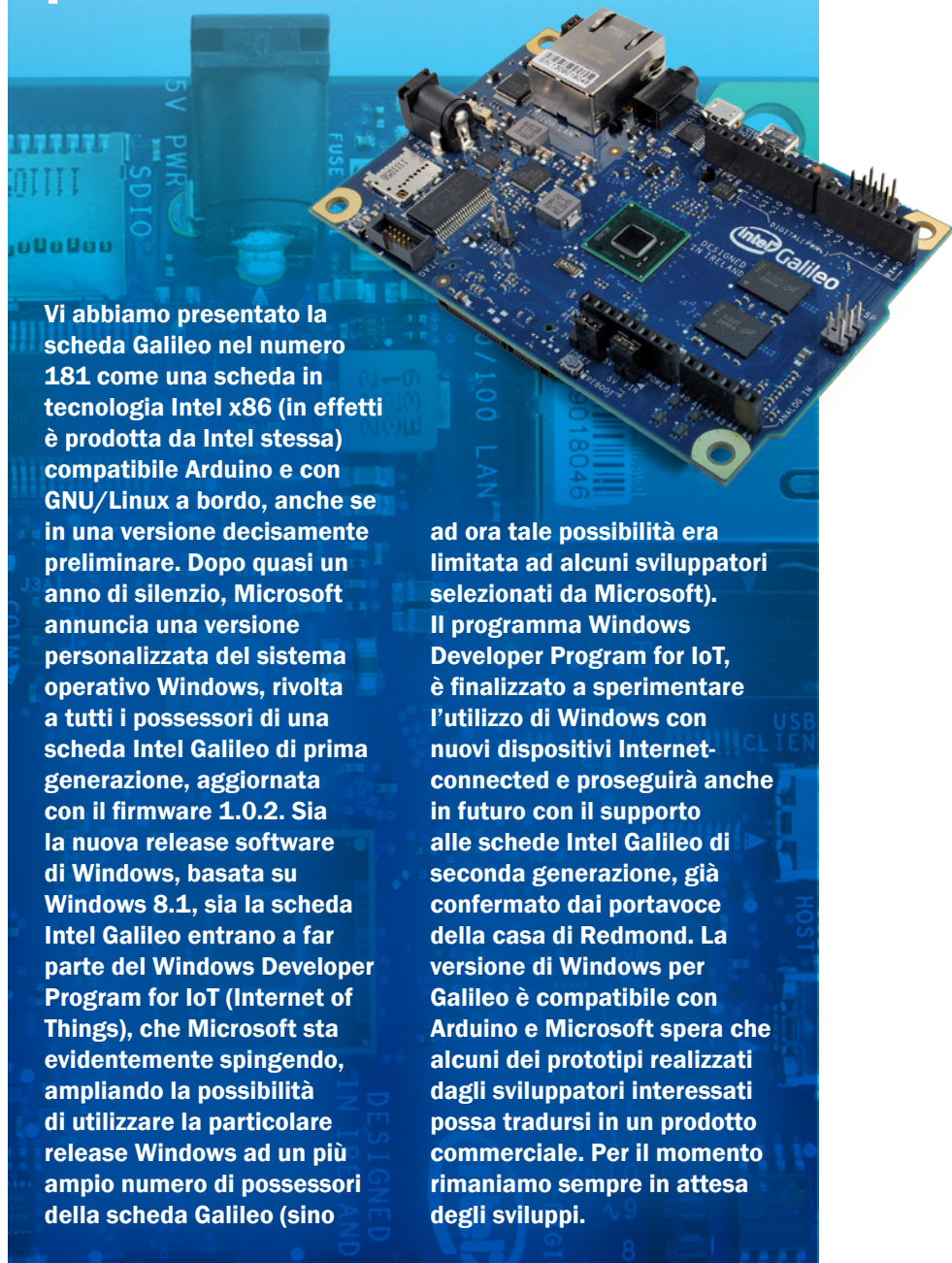


Fig. 4

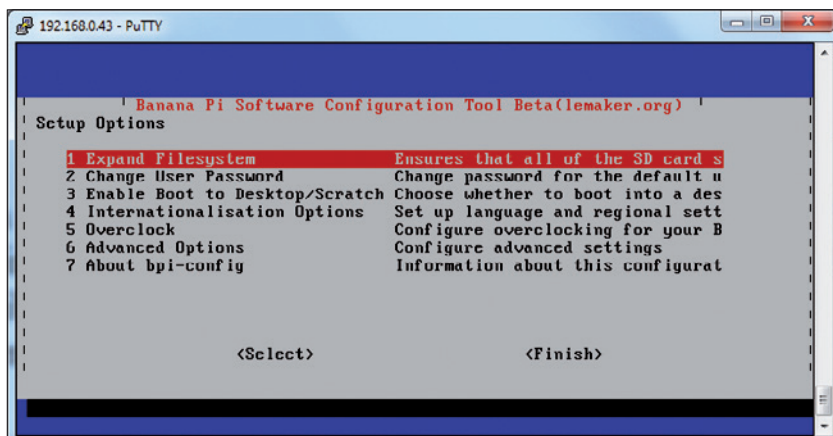
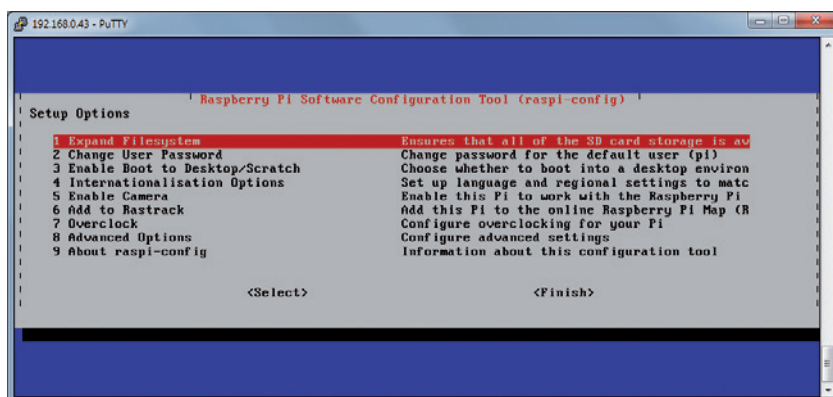


Fig. 5



nuova distribuzione sono stati risolti alcuni dei problemi che abbiamo evidenziato nell'articolo dedicato a Raspberry Pi. In particolare la possibilità di utilizzare fisicamente la stessa SD Card indifferente su Raspberry Pi e BananaPi, questa volta per davvero, ovviamente se non utilizziamo il disco fisso sul connettore SATA (Raspberry Pi non ha questa possibilità), ma con un case che converte SATA in USB potremmo "risolvere" il problema" configurando opportunamente la partizione di "boot" per Raspberry Pi. Al riguardo potete consultare il libro "Raspberry Pi - Il mio primo Linux embedded". Un'altra funzionalità che ci ha fatto molto piacere vedere "apparire" è la possibilità di utilizzare lo strumento di configurazione "raspi-config" anche su BananaPi. Lo abbiamo sempre ritenuto molto utile, soprattutto per chi

ha meno dimestichezza con l'utilizzo di utility di sistema come Gparted o i comandi di shell. Nelle Fig. 4 e Fig. 5 vediamo la finestra del menu principale dello strumento raspi-config lanciato su BananaPi e su Raspberry Pi. Notate che il menu, lanciato dallo stesso comando, viene personalizzato automaticamente a seconda della scheda sulla quale viene lanciato. Con raspi-config, per esempio, diventa estremamente semplice, dopo aver preparato la SD Card, espandere la partizione di "root" fino ad utilizzare tutto lo spazio disponibile, cosa che con le precedenti versioni di Raspbian per BananaPi richiedeva laboriose operazioni con gli strumenti di gestione delle partizioni. In Fig. 6 vediamo lo stato di occupazione spazio sulla SD Card appena creata.

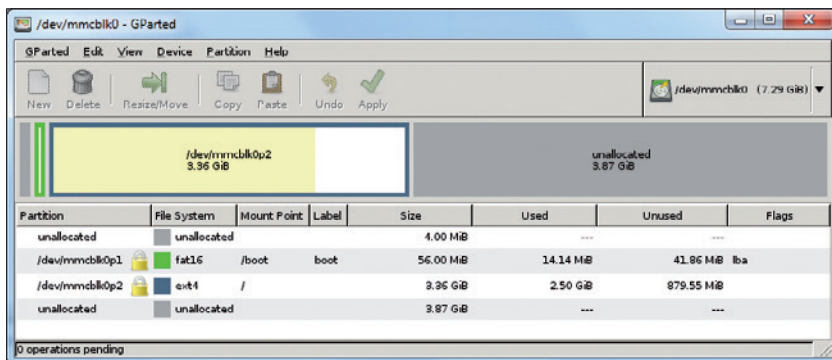


Fig. 6

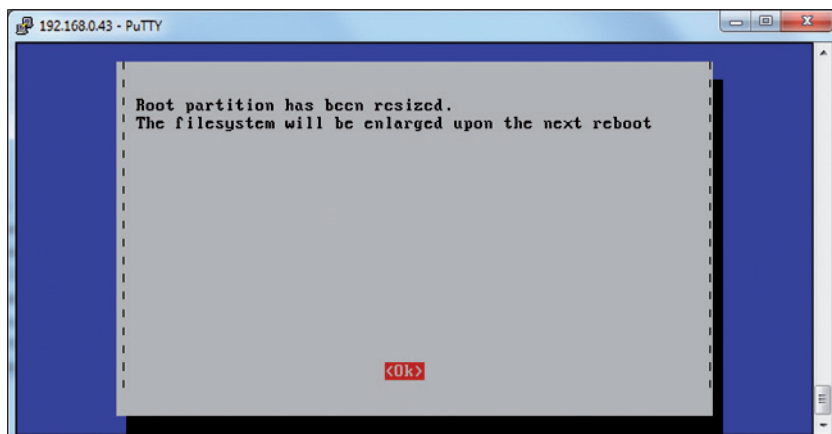


Fig. 7

Nelle Fig. 7 ed Fig. 8 vediamo i semplici passaggi per utilizzare tutto lo spazio presente sulla SD Card dopo aver selezionato nello strumento raspi-config la voce di menu "1. Expand Filesystem". Nella Fig. 9 vediamo il risultato dell'operazione dopo il "reboot" della scheda.

Tornano ad essere semplici anche le altre configurazioni disponibili nel menu di raspi-config, come la suddivisione della memoria RAM tra video e CPU o le abilitazioni della video camera e del server SSH. Un'aggiunta comoda è la possibilità di tenere sotto controllo la temperatura della CPU. Basta andare alla cartella:

```
/sys/devices/platform/sunxi-i2c.0/i2c-0/0-0034
```

e digitare il comando:

```
cat temp1_input
```

In Fig. 10 abbiamo misurato una temperatura della CPU di 49 gradi. Nella distribuzione è inclusa in modo predefinito la libreria wiringPi nella sua ultima versione beta, che permette l'utilizzo dei pin di GPIO sia come I/O che configurati come bus I2C, SPI, oppure come uscite PWM. Questa libreria funziona anche con Raspberry Pi. Ricordiamo ancora che l'utente predefinito in questa distribuzione è:

bananapi (nella precedente versione 3.0 l'utente era "pi") con password:

```
bananapi
```

Unica nota dolente, se volete sperimentare la nuova versione e avete già installato la versione precedente, eventualmente



Fig. 8

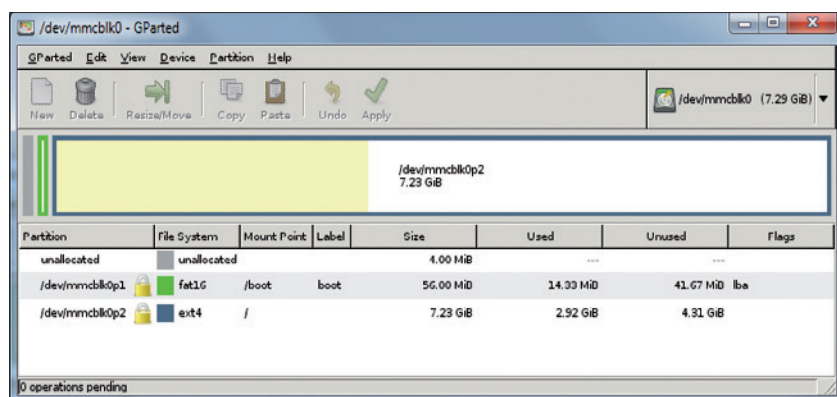


Fig. 9

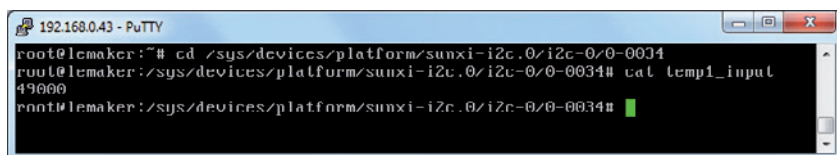
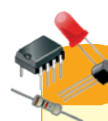


Fig. 10

utilizzando un disco fisso, è data dal fatto che non è possibile procedere con un'operazione di upgrade, ma è necessario scaricare la nuova distribuzione completa e creare la SD card con il metodo descritto tante volte, ed in particolare nel libro "Raspberry Pi - Il mio primo Linux embedded". Per collegare un disco fisso al connettore SATA e trasferire su quest'ultimo la partizione di root rimandiamo all'articolo di presentazione di Banana Pi nel numero 188. Intanto ci aspettiamo ulteriori sviluppi sulle schede dotate di SoC Allwinner. Ce lo conferma il fatto che mentre stiamo scrivendo queste note, Lemaker

presenta Banana Pro, evoluzione di Banana Pi in stile Raspberry Pi versione B+. Vi terremo informati.



per il MATERIALE

La board Banana Pi può essere acquistata in Italia presso Futura Elettronica al prezzo di Euro 75,00 IVA compresa (cod. 7300-BANA-NAPI).

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

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

Elettronica In

www.elettronica.in.it

oltre l'elettronica



Dotiamo
Raspberry Pi di un
display TFT Touch Screen
sul quale visualizzare la console
di sistema, i film e le foto
preferite oppure comandare una
scheda relé... in punta di dita.

DISPLAY TOUCH PER RASPBERRY PI

..... di MARCO MAGAGNIN

Per evitare di utilizzare monitor HDMI dal costo ben superiore a quello di Raspberry Pi, negli articoli precedenti abbiamo sempre adottato la soluzione di collegarci al microcomputer in remoto, utilizzando strumenti come PuTTY e WinSCP. Questo approccio ci ha sempre vincolato ad utilizzare Raspberry Pi nella modalità cosiddetta "server". Di conseguenza abbiamo sempre presentato le applicazioni con interfaccia utente realizzate in modo da essere fruibili da browser web. Che fare, invece, quando vogliamo avere un'interfaccia in

stile tradizionale, fruibile direttamente dal desktop? Semplice, si collega un monitor, una tastiera, un mouse e si realizza una classica applicazione desktop utilizzando i linguaggi classici con le librerie per la realizzazione di interfacce grafiche. Già, ma in questo modo trasformiamo Raspberry Pi in un PC con attaccate tutte le periferiche tradizionali che lo rendono "ingombrante". La soluzione che vi presentiamo al posto di quella "classica" permette di realizzare interfacce grafiche utilizzando un grazioso monitor TFT da 2,8 pollici, a colori, ed in più

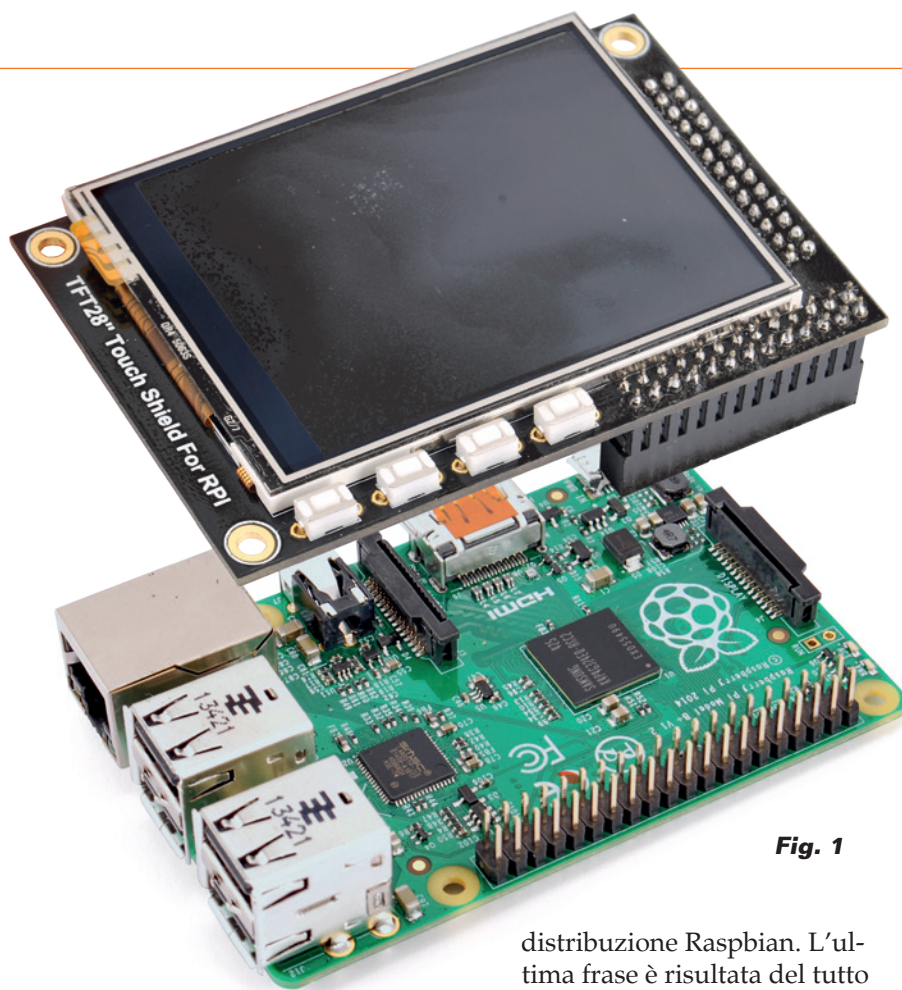


Fig. 1

con un'interfaccia touch di tipo resistivo. Niente monitor, niente tastiera e niente mouse. La scheda con il monitor ha più o meno le stesse dimensioni di Raspberry Pi, è già montata ed è compatibile con tutti i modelli del microcontrollore: A, B e B+. In questo articolo, abbiamo adottato il modello B+ che, per il progetto finale che vi proponiamo, mette a disposizione un maggior numero di pin di I/O rispetto alle versioni precedenti. Il monitor ha una risoluzione di 320 x 240 pixel, con una profondità di colore di 16 bit. Il mini display si collega a Raspberry Pi sul bus SPI e può essere utilizzato per reindirizzare la console, oppure come monitor X per visualizzare il desktop di Raspbian, foto, video e interfacce grafiche di applicazioni utente. Fin qui tutto bene. Il rovescio della medaglia è che il driver per la gestione del piccolo monitor non è incluso nel kernel della

distribuzione Raspbian. L'ultima frase è risultata del tutto incomprensibile? Per approfondire l'argomento vi rimandiamo al libro "Raspberry Pi – Il mio primo Linux embedded" per un viaggio di approfondimento sul microcomputer. Tra l'altro una delle Appendici è dedicata ad introdurre la struttura del sistema operativo GNU/Linux, kernel e driver inclusi. Ma torniamo al driver. Questo simpatico monitor ci permette di introdurre, in modo più approfondito di quanto abbiamo fatto fino ad ora, la realizzazione di un sistema GNU/Linux embedded. Nella pratica "professionale" le distribuzioni GNU/Linux destinate al mondo embedded sono le più scarse possibili, con inclusi unicamente i driver e le applicazioni essenziali per il funzionamento di base del microcomputer. Questo per minimizzare l'occupazione di memoria della distribuzione e massimizzare le prestazioni del sistema di destinazione. Questo approccio

permette di utilizzare chip di prestazioni ridotte, con dimensioni fisiche della soluzione finale il più possibile contenute e consumi anch'essi resi minimi per poter permettere soluzioni basate su alimentazioni a batterie. Il rovescio della medaglia consiste nel dover personalizzare il kernel della distribuzione con i driver delle periferiche che si vogliono utilizzare, come per esempio dongle WiFi, periferiche USB, display LCD e pannelli LED. Utilizzando Raspbian abbiamo sempre evitato di doverci occupare di questi problemi, in quanto, essendo un microcomputer progettato per la didattica, nella sua distribuzione Raspbian sono già inclusi una enorme quantità di driver che soddisfano in automatico (plug and play) la maggior parte delle esigenze. Questa volta però il driver per il display LCD TFT non è incluso. Fortunatamente la società Adafruit ha messo a disposizione un kernel di Raspbian già compilato che include il driver richiesto. Questo ci permette di evitare di dover personalizzare e compilare da noi il kernel di Raspbian. Stiamo comunque pensando di proporre sulla rivista una serie di articoli dedicati all'argomento, anche perché la presenza sul mercato di sempre nuove periferiche richiederà prima o poi di affrontare l'argomento con determinazione. Nella soluzione Adafruit si presentano infatti due problemi. Che fare quando dobbiamo adottare una seconda periferica che richiede l'inclusione nel kernel di Linux di un driver specifico? Anche se il produttore mette a disposizione una versione "compilata" del kernel per la sua periferica, se la adottassimo perderemmo il driver per la prima periferica. Non resta che configurarci e compilarci il

Fig. 2

kernel con l'aggiunta dei due, o più, driver aggiuntivi richiesti. Secondo problema. Adottare il kernel precompilato, personalizzato con l'aggiunta di un driver specifico, ci vincola alla versione di kernel utilizzato per la compilazione. In questa condizione non saremo in grado di adottare nuove distribuzioni che sfruttano versioni aggiornate del kernel, a meno che il produttore della periferica non fornisca a sua volta una nuova versione. A questo proposito ricordiamo che è stata appena rilasciata la release 3.16 del kernel di Linux, che presenta notevoli funzionalità aggiuntive per quanto riguarda la gestione dei chip con CPU ARM e relativi processori grafici. Segno che assisteremo ad una sempre maggior diffusione della nostra "filosofia". Ma torniamo al nostro display grafico visibile in Fig. 1, basato sul modulo LCD TFT ILI9340, che viene fornito già montato e dotato di connettore GPIO, che si installa su tutti i modelli di Raspberry Pi A, B e B+. La board include il controllore del display STMPE610 che comunica con Raspberry Pi utilizzando il bus SPI e quindi impegna i pin corrispondenti del connettore GPIO (SCK, MOSI, MISO, CE0, CE1) oltre ai pin GPIO25 e GPIO24. I quattro pulsanti aggiuntivi presenti sulla board sono collegati ai pin GPIO23, GPIO22, GPIO21, e GPIO18. Tutti gli altri pin sono inutilizzati e possono essere impiegati per altre esigenze. In questo caso è opportuno saldare un connettore sugli alloggiamenti previsti sul lato destro della scheda e utilizzare un cavo piatto

di prolunga. Fate attenzione a rispettare le corrispondenze dei pin. Il pin numero 1 del connettore GPIO è indicato sul retro della board con una piazzola di forma quadrata ed una piccola freccia bianca serigrafata.

PERSONALIZZAZIONE DELLA DISTRIBUZIONE RASPBIAN

Il processo che vi descriviamo per rendere operativo il display LCD TFT richiede di partire dalla distribuzione Raspbian. Noi siamo partiti dalla immagine rilasciata in data 9-settembre-2014 con release del kernel 3.12 (Fig. 2). Al momento è preferibile non installare il display sul connettore, lo faremo dopo aver installato il software di gestione. Per poter eseguire il processo di installazione è necessario che Raspberry Pi sia collegato alla rete locale in modo da poter "vedere" la rete internet pubblica. È inoltre necessario poter accedere a Raspberry Pi per utilizzare una finestra terminale ed una finestra con l'applicazione di gestione del file system. Come al solito, ma soprattutto in questo caso, preferiamo accedere da remoto via protocollo SSH ed utilizzare (in Windows) gli strumenti Putty e WinSCP. Per approfondimenti sulle modalità di connessione ed utilizzo di questi strumenti, vi rimandiamo al libro citato all'inizio dell'articolo. È ovviamente possibile utilizzare un collegamento alla console via cavo seriale.

Ci colleghiamo a Raspberry Pi con l'utente "root" ed iniziamo il processo di installazione con i soliti comandi di aggiornamento

della distribuzione:

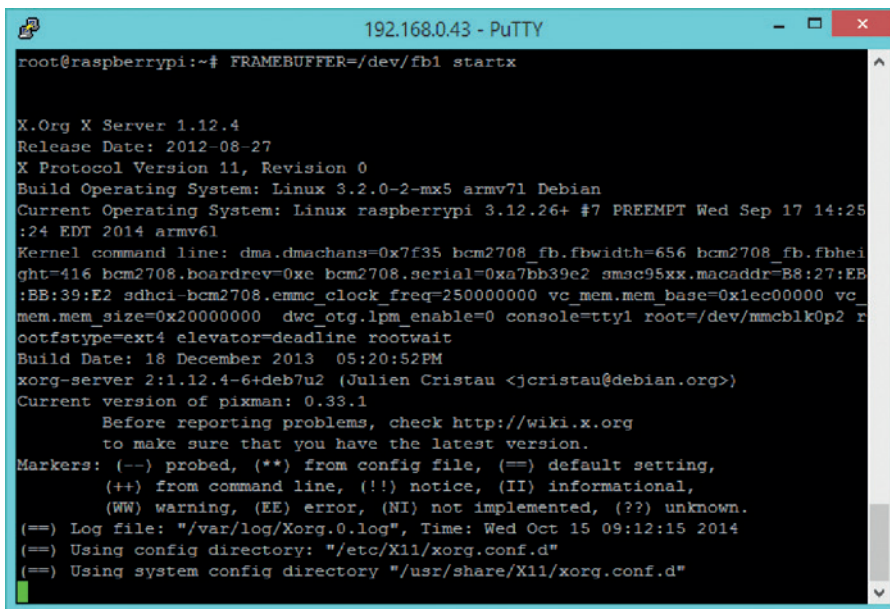
```
apt-get update
apt-get upgrade
```

Andiamo ad una cartella di appoggio per il download dei file di installazione. Possiamo anche creare una cartella specifica per questo processo. I file sono disponibili come pacchetti .deb, che dovranno poi essere installati con il comando dpkg. Anche in questo caso vi rimandiamo al libro citato per approfondimenti. Possiamo utilizzare, per esempio la cartella "home". Ci posizioniamo su questa cartella con il comando:

```
cd /home
```

Ora scarichiamo tutti i pacchetti necessari alla "sostituzione" del kernel ed alla gestione del display:

```
wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-bin-adafruit.deb
wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-dev-adafruit.deb
wget http://adafruit-download.s3.amazonaws.com/libraspberrypi-doc-adafruit.deb
wget http://adafruit-download.s3.amazonaws.com/libraspberrypi0-adafruit.deb
wget http://adafruit-download.s3.amazonaws.com/raspberrypi-bootloader-adafruit-20140917-1.deb
```

```
root@raspberrypi:~# FRAMEBUFFER=/dev/fb1 startx

X.Org X Server 1.12.4
Release Date: 2012-08-27
X Protocol Version 11, Revision 0
Build Operating System: Linux 3.2.0-2-mx5 armv7l Debian
Current Operating System: Linux raspberrypi 3.12.26+ #7 PREEMPT Wed Sep 17 14:25:24 EDT 2014 armv6l
Kernel command line: dma.dmachans=0x7f35 bcm2708_fb.fbwidth=656 bcm2708_fb.fbheight=416 bcm2708.boardrev=0xc bcm2708.serial=0xa7bb39e2 smac95xx.macaddr=B8:27:EB:BB:39:E2 sdhci-bcm2708.emmc_clock_freq=250000000 vc_mem.mem_base=0x1ec00000 vc_mem.mem_size=0x20000000 dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait
Build Date: 18 December 2013 05:20:52PM
xorg-server 2:1.12.4-6+deb7u2 (Julien Cristau <jcristau@debian.org>)
Current version of pixman: 0.33.1
  Before reporting problems, check http://wiki.x.org
  to make sure that you have the latest version.
Markers: (--) probed, (**) from config file, (==) default setting,
(++) from command line, (!!) notice, (II) informational,
(WW) warning, (EE) error, (NI) not implemented, (??) unknown.
(==) Log file: "/var/log/Xorg.0.log", Time: Wed Oct 15 09:12:15 2014
(==) Using config directory: "/etc/X11/xorg.conf.d"
(==) Using system config directory "/usr/share/X11/xorg.conf.d"
```

Fig. 3

Dovreste vedere apparire il desktop di Raspberry Pi sul piccolo display come visibile in Fig. 4.

Potete anche utilizzare la barra di menu e le applicazioni in modalità “touch”.

Se tutto funziona correttamente il passo più importante è stato fatto. Ora facciamo in modo che tutto parta in automatico all’accensione di Raspberry Pi.

Chiudiamo il desktop grafico (in realtà il server X) dalla finestra di terminale premendo contemporaneamente i tasti <Control-X>. Apriamo da WinSCP il file:

Terminato il download, dalla stessa cartella, installiamo il tutto con il comando:

```
sudo dpkg -i -B *.deb
```

Questa operazione richiede un tempo abbastanza lungo, potete prendervi una pausa e rilassarvi un attimo.

Nel caso stiate utilizzando, come molto probabile, una versione di Raspbian posteriore a Settembre 2013, è necessario disattivare l’utilizzo del framebuffer X “accelerato”, con il comando che rimuove il file di configurazione dell’acceleratore e lo salva nella cartella /home (non si sa mai):

```
mv /usr/share/X11/xorg.conf.d/99-fbturbo.conf /home
```

Ora spegniamo Raspberry Pi con il comando tradizionale:

```
shutdown -h now
```

Togliamo l’alimentazione, montiamo il display sul connettore GPIO, verificando che non si verifichino contatti con qualche connettore o componente di Raspberry Pi e ridiamo tensione. Non preoccupatevi, sul piccolo display, per il momento, non

apparirà nulla. Ci ricollegiamo a Raspberry Pi con Putty e WinSCP. Per provare velocemente il funzionamento del display possiamo caricare i driver da linea di comando e lanciare il desktop di Raspberry Pi.

Digitiamo i seguenti comandi nell’ordine, come visibile in Fig. 3:

```
modprobe spi-bcm2708
modprobe fbtft_device
name=adafruitts rotate=90
export FRAMEBUFFER=/dev/fb1
startx
```

/etc/modules

ed aggiungiamo in fondo al file i due moduli da caricare all’accensione:

```
spi-bcm2708
fbtft_device
```

Salviamo e chiudiamo il file. Rispondiamo all’eventuale richiesta di password per perfezionare il salvataggio.

Un momento, non abbiamo ancora finito, dobbiamo aggiungere un file di configurazione

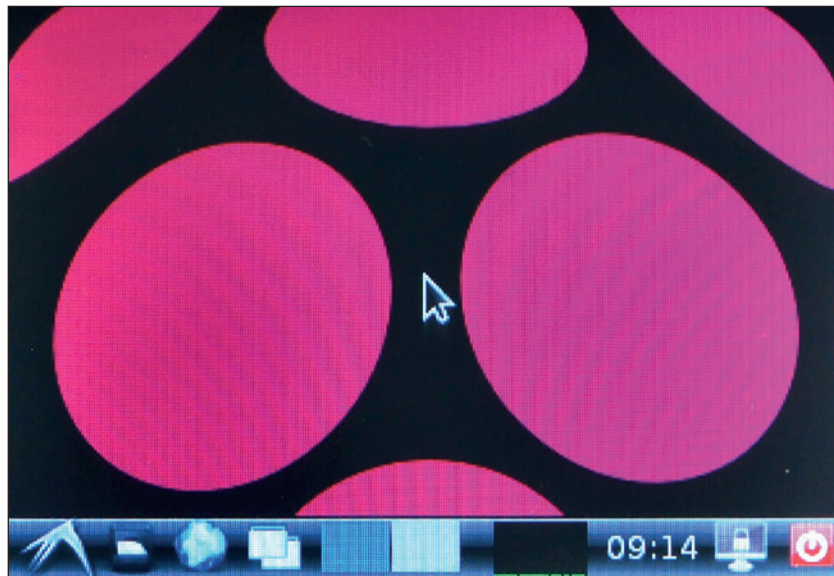
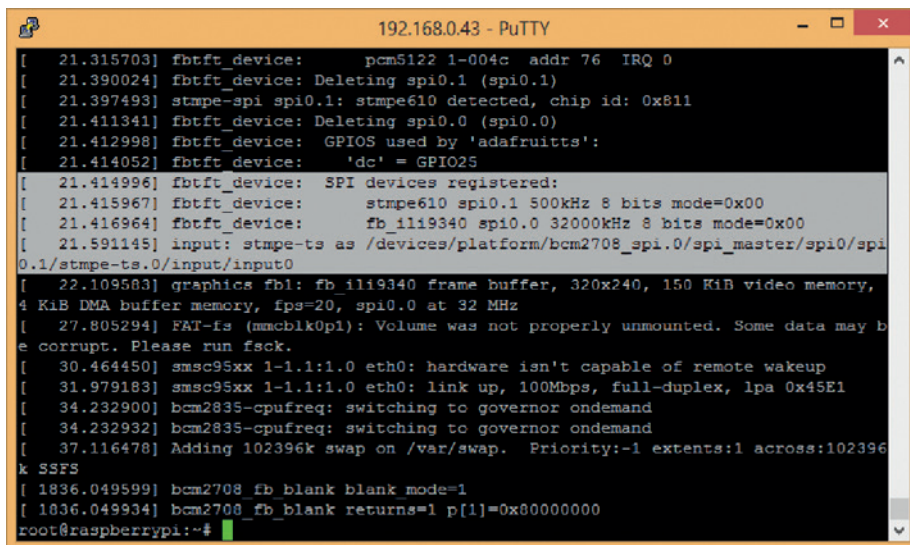


Fig. 4



per personalizzare l'utilizzo del display in termini di rotazione dell'immagine e frequenza di refresh. Apriamo il file:

`/etc/modprobe.d/adafruit.conf`

ed aggiungiamo le seguenti configurazioni:

```
options          fbttft_device
name=adafruitrt28 rotate=90
frequency=32000000
```

Il parametro "rotate" permette di ruotare l'immagine sul display di 0, 90, 180 o 270 gradi in modo da potersi adattare a tutte le situazioni di montaggio:

0 corrisponde al display in verticale, con il lato inferiore verso le scritte stampate sulla basetta;
90 corrisponde al display in orizzontale, con il lato inferiore verso i pulsanti;
180 corrisponde al display in verticale, con il lato superiore verso le scritte stampate sulla basetta;

270 corrisponde al display in orizzontale, con il lato superiore verso i pulsanti.

Il parametro "frequency" indica la velocità di refresh del display. Il valore "32000000" significa 32MHz e corrisponde ad un frame rate di circa 20FPS.

Se il carico complessivo delle vostre applicazioni lo permette, potete provare ad abbassare la frequenza a 16 MHz (16000000). Fatto questo rendiamo effettive le modifiche con un comando di:

`reboot`

Dopo essersi ricollegati controlliamo i messaggi di console con il comando:

`dmesg`

facciamo particolare attenzione al riconoscimento del controllore STMPE610 ed alla impostazione della frequenza del display ILI9340 come visibile nella Fig. 5.

Fig. 5

Possiamo riverificare il funzionamento del tutto lanciando il desktop sul server x con il comando:

`FRAMEBUFFER=/dev/fb1 startx`

Per interrompere l'esecuzione del desktop premete <Control-X>. Tutto a posto? Non ancora, ora dobbiamo configurare e calibrare il touch screen in modo da poter funzionare sia con il desktop grafico che con le applicazioni che andremo a sviluppare.

Come prima operazione impostiamo una regola nella cartella udev che colleghi il touchscreen al corretto device di input. Può accadere che il touchscreen venga riconosciuto nella cartella `/dev/input` con un "event" diverso a seconda se sono presenti o meno tastiera, mouse o altri dispositivi di input. Creiamo il file

`95-stmpe.rules`

nella cartella

`/etc/udev/rules.d/`

ed inseriamo le seguenti configurazioni (Fig. 6):

```
SUBSYSTEM=="input",
ATTRS{name}=="stmpe-ts",
ENV{DEVNAME}=="*event*",
SYMLINK+="input/touchscreen"
```

Rimuoviamo e reinstalliamo il driver del touch screen con i comandi:

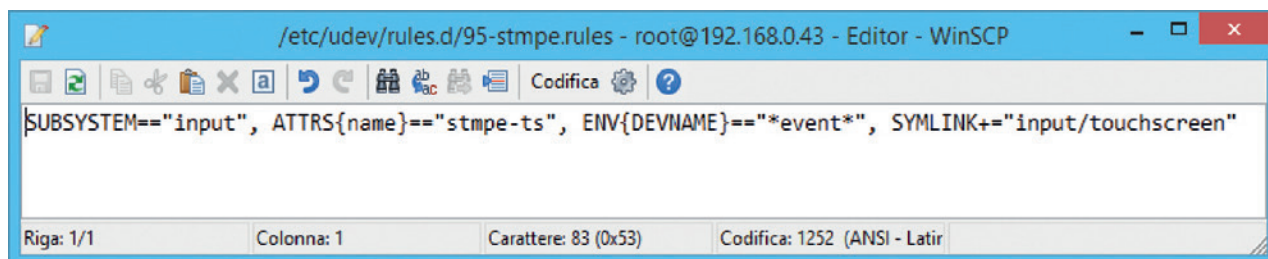


Fig. 6


```

192.168.0.43 - PuTTY
root@raspberrypi:~# ls -l /dev/input/touchscreen
lrwxrwxrwx 1 root root 6 Oct 20 08:55 /dev/input/touchscreen -> event0
root@raspberrypi:~#

```

Fig. 7

```

192.168.0.43 - PuTTY
Event: time 1413799520.383315, type 3 (EV_ABS), code 1 (ABS_Y), value 405
Event: time 1413799520.383315, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 167
Event: time 1413799520.383315, ----- SYN_REPORT -----
Event: time 1413799520.391004, type 3 (EV_ABS), code 0 (ABS_X), value 1689
Event: time 1413799520.391004, type 3 (EV_ABS), code 1 (ABS_Y), value 410
Event: time 1413799520.391004, ----- SYN_REPORT -----
Event: time 1413799520.398521, type 3 (EV_ABS), code 0 (ABS_X), value 1700
Event: time 1413799520.398521, type 3 (EV_ABS), code 1 (ABS_Y), value 413
Event: time 1413799520.398521, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 165
Event: time 1413799520.398521, ----- SYN_REPORT -----
Event: time 1413799520.406093, type 3 (EV_ABS), code 0 (ABS_X), value 1669
Event: time 1413799520.406093, type 3 (EV_ABS), code 1 (ABS_Y), value 406
Event: time 1413799520.406093, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 155
Event: time 1413799520.406093, ----- SYN_REPORT -----
Event: time 1413799520.434786, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 0
Event: time 1413799520.434786, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 0
Event: time 1413799520.434786, ----- SYN_REPORT -----

```

Fig. 8

```

192.168.0.43 - PuTTY
root@raspberrypi:/home# python pitft_touch_cal.py
USING ROTATION: 90

-----
CURRENT CONFIGURATION

Current /etc/pointercal configuration:
-30 -5902 22077792 4360 -105 -1038814 65536

Current /etc/X11/xorg.conf.d/99-calibration.conf configuration:
Section "InputClass"
    Identifier      "calibration"
    MatchProduct    "stampe-ts"
    Option "Calibration" "3807 174 244 3872"
    Option "SwapAxes" "1"
EndSection

-----
NEW CONFIGURATION

New /etc/pointercal configuration:
-30 -5902 22077792 4360 -105 -1038814 65536

New /etc/X11/xorg.conf.d/99-calibration.conf configuration:
Section "InputClass"
    Identifier      "calibration"
    MatchProduct    "stampe-ts"
    Option "Calibration" "3807 174 244 3872"
    Option "SwapAxes" "1"
EndSection

Update current configuration to new configuration? [y/N]: y
-----
Updated /etc/pointercal
Updated /etc/X11/xorg.conf.d/99-calibration.conf
root@raspberrypi:/home#

```

Fig. 9

```

rmmod stampe_ts
modprobe stampe_ts

```

A questo punto nella cartella /dev/input troveremo il device

“touchscreen” che punta al corretto device “event<x>”. Potete verificarlo con il comando (Fig. 7):

```
ls -l /dev/input/touchscreen
```

Vedremo la descrizione del device file con il collegamento (la freccetta) verso un device “event(x)”. Ora dobbiamo “calibrare” il nostro touch screen. Vi sono due modi per farlo, uno un po’ più “grossolano” ma automatico, l’altro più preciso ma più articolato e lungo da eseguire. Per le nostre esigenze ed un po’ perché siamo pigri utilizzeremo il metodo “automatico” utilizzando un programma Python realizzato e gentilmente messo a disposizione da Adafruit. Prima installiamo una libreria utile per eseguire i test sul touch screen.

```
apt-get install evtest tslib
libts-bin
```

Possiamo già fare una prova digitando:

```
evtest /dev/input/touchscreen
```

Se tocchiamo il display in vari punti vedremo gli eventi riconosciuti e le coordinate del “punto” toccato (Fig. 8). Per la calibrazione “grossolana” vera e propria scarichiamo il program-

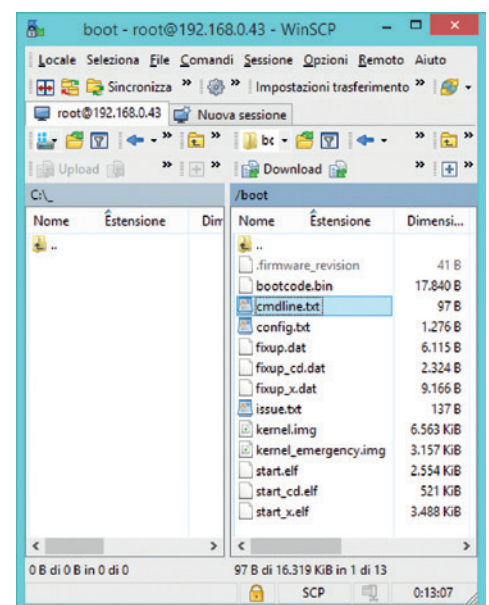


Fig. 10

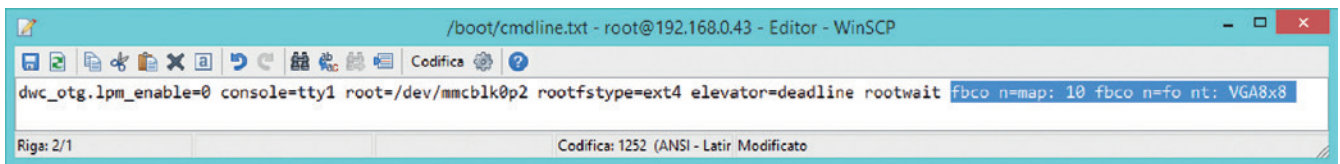


Fig. 11

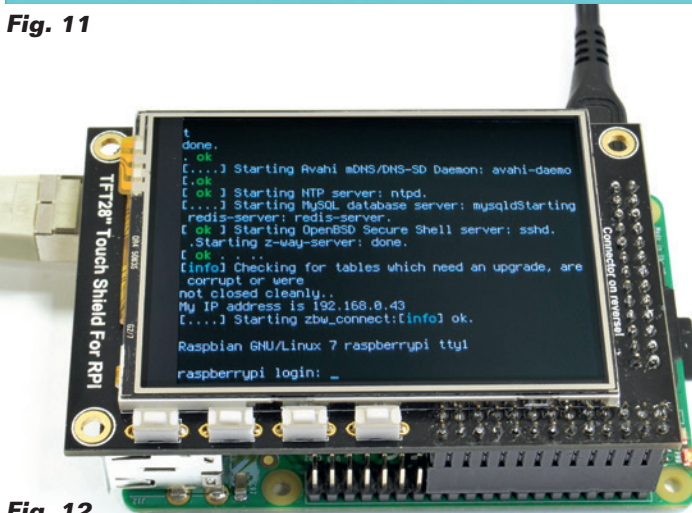


Fig. 12

ma Python dal sito di Adafruit e prepariamo la cartella per contenere il file di configurazione. Scarichiamo il programma con il comando:

```
wget https://github.com/adafruit/PiTFT_Extras/raw/master/pitft_touch_cal.py
```

Creiamo la cartella xorg.conf.d con il comando:

```
mkdir /etc/X11/xorg.conf.d/
```

ed eseguiamo il programma di calibrazione dalla stessa cartella dove lo abbiamo scaricato:

```
python pitft_touch_cal.py
```

Guardate l'output del programma, prima di aggiornare il file di configurazione il programma vi chiede l'autorizzazione, se siete soddisfatti dei risultati confermate con "Y" (Fig. 9).

REINDIRIZZARE LA CONSOLE SUL DISPLAY

Una operazione simpatica che si può impostare è di reindirizzare i messaggi di console sul display

invece che sull'uscita video standard. Anche se lo schermo è piccolo, scegliendo un font adatto si può ottenere un'area di testo di 20 righe per 40 caratteri.

Per ottenere questo risultato dobbiamo modificare il file di configurazione del kernel nella partizione di boot del file system.

Per chi volesse approfondire i dettagli rimandiamo sempre al libro citato all'inizio dell'articolo.

Aprirete il file (Fig. 10):

/boot/cmdline.txt

Lo scopo della modifica è reindirizzare la console dal frame buffer HDMI/TV standard /dev/fb0 al frame buffer del display TFT /dev/fb1.

Per farlo andate in fondo alle direttive del file, dopo la voce "rootwait" ed aggiungete le di-

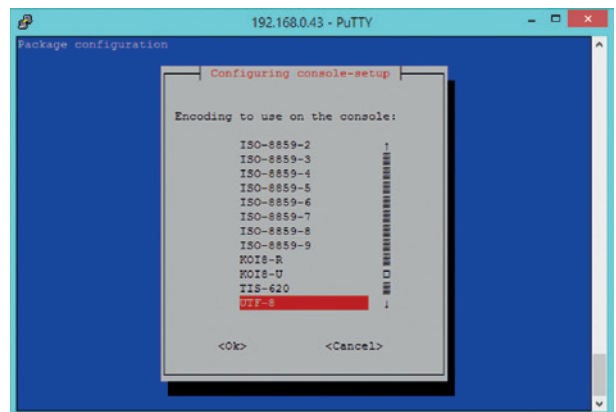


Fig. 13

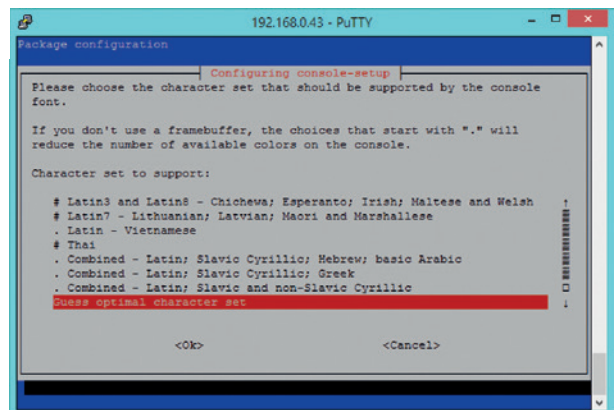


Fig. 14

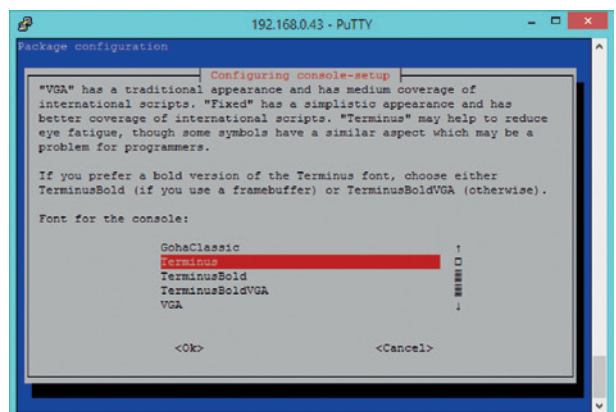


Fig. 15

chiarazioni (Fig. 11):

```
fbco n=map: 10 fbco n=fo nt: VGA8x8
```

Salvate il file e date un comando di reboot. Notate che, durante il

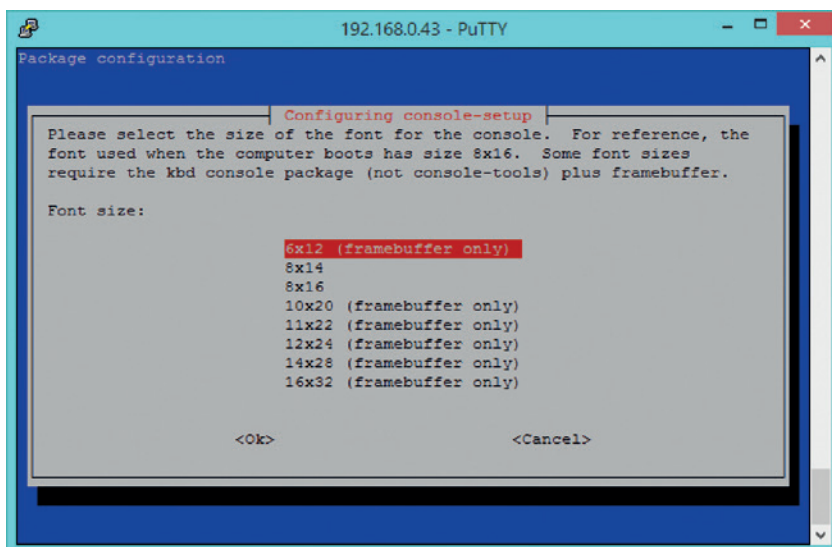


Fig. 16

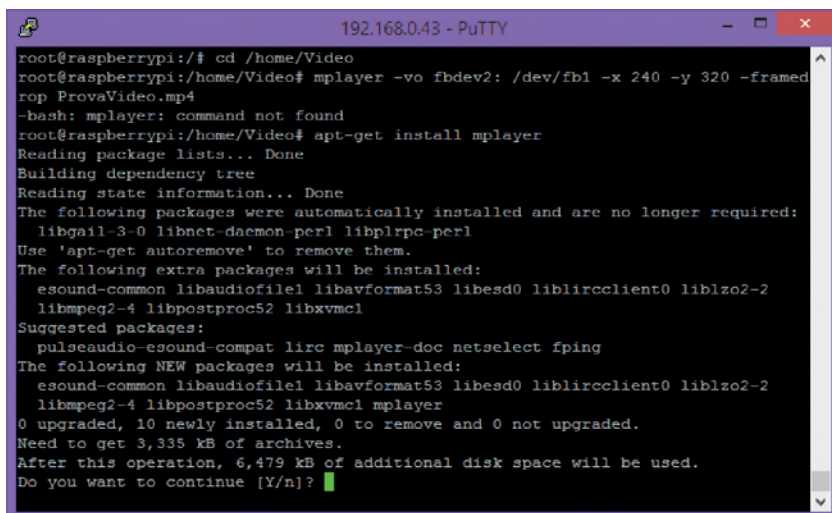


Fig. 17

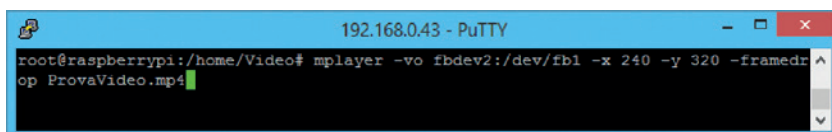


Fig. 18

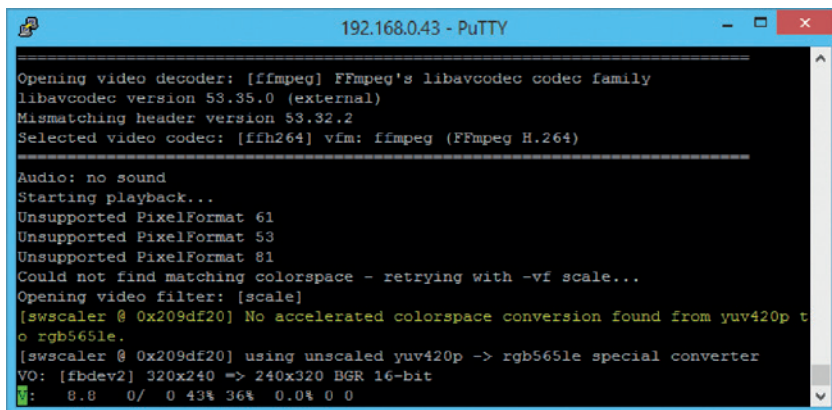


Fig. 19

caricamento iniziale, vi perderete lo splash screen multicolore iniziale ed una parte dei messaggi di console.

Ciò è assolutamente normale, in quanto, per poter lavorare, il driver deve essere caricato dal kernel ed attivato, il che avviene qualche tempo dopo l'inizio del processo di boot. Poi vedrete apparire i messaggi di console, come visibile in Fig. 12.

Per cambiare il font VGA8x8 in qualcosa di maggiormente leggibile come un 12x6 lanciamo il comando di riconfigurazione della console (sempre il solito libro per approfondire):

```
dpkg-reconfigure console-setup
```

seguite le figure da 13 a 16 per selezionare il font Terminus 6x12.

VISUALIZZARE FILE VIDEO

Si possono visualizzare sul nostro display file video con diverse estensioni, a patto che le dimensioni delle immagini siano di 320 x 240 pixel.

Per visualizzare un video utilizziamo il pacchetto player, se non è presente nel nostro sistema lo installiamo con il comando (Fig. 17):

```
apt-get install mplayer
```

Per vedere il video diamo il comando (Fig. 18, Fig. 19 e Fig. 20):

```
mplayer -vo fbdev2:/dev/fb1 -x 240 -y 320 -framedrop <file-video.ext>
```

per esempio "ProvaVideo.mp4". Se il video non è in formato 320 x 240 lo possiamo ridimensionare utilizzando lo strumento HandBrake, Open Source e disponibile per tutti i sistemi operativi.

Lo scarichiamo dall'indirizzo



Fig. 20

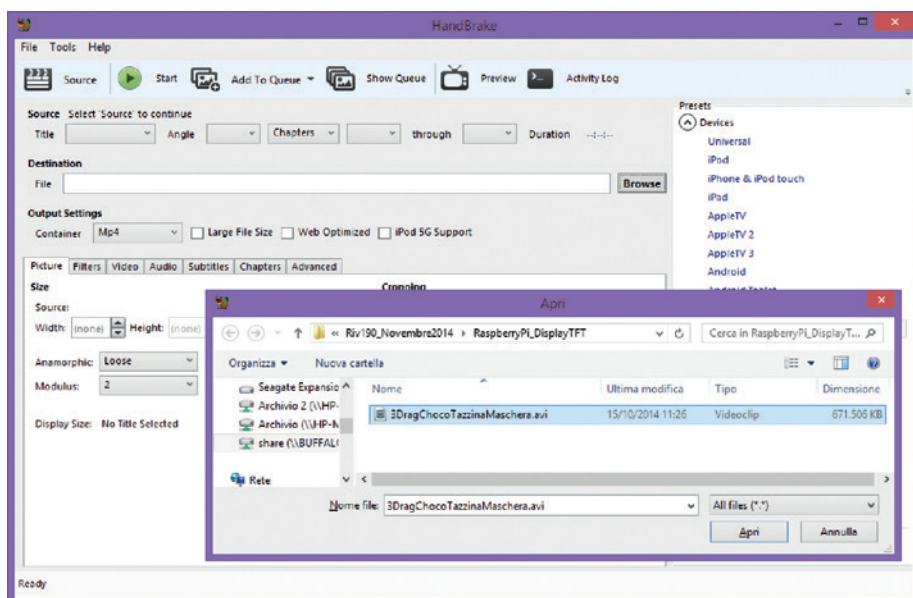


Fig. 21

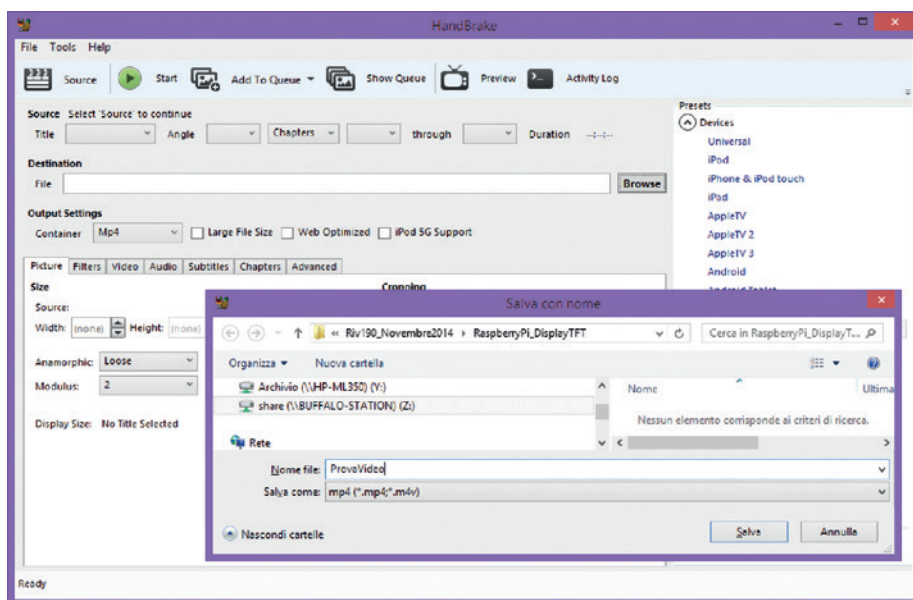


Fig. 22

<https://handbrake.fr/>.

Dopo averlo installato lo utilizziamo per ridimensionare il video di prova. Apriamo il file, preferibilmente con estensione .avi (Fig. 21),

La finestra viene precompilata con una quantità di informazioni relative al file caricato. Nel campo "Destination" inseriamo il percorso ed il nome del file che vogliamo ottenere (Fig. 22).

Nel pannello sottostante, in "Output Settings", nel campo "Container", selezioniamo "MP4", in "Width" impostiamo 320. In Anamorphic selezioniamo "Custom" ed in "Modulus" 2 (Fig. 23). Clicchiamo sul pulsante "Start" per iniziare la conversione (Fig. 24).

Al termine, se abbiamo eseguito la conversione in ambiente Windows, spostiamo il file su Raspberry Pi (Fig. 25) ed eseguiamo il comando descritto all'inizio del paragrafo per visualizzare il file.

VISUALIZZARE IMMAGINI

Un'applicazione simpatica è la visualizzazione di immagini. Per farlo possiamo utilizzare il pacchetto "fb" frame buffer image viewer. Lo installiamo con il solito comando:

```
apt-get install fb
```

Anche in questo caso le immagini devono avere formato 320 x 240 pixel. Per visualizzare un'immagine utilizziamo il comando (Fig. 26):

```
fb -T 2 -d /dev/fb1 -noverbose -a <nome-foto>.jpg
```

REGOLARE LA RETROILLUMINAZIONE

La retroilluminazione del display TFT è realizzata per mezzo di quattro LED che assorbono complessivamente circa 75 mA. Nel

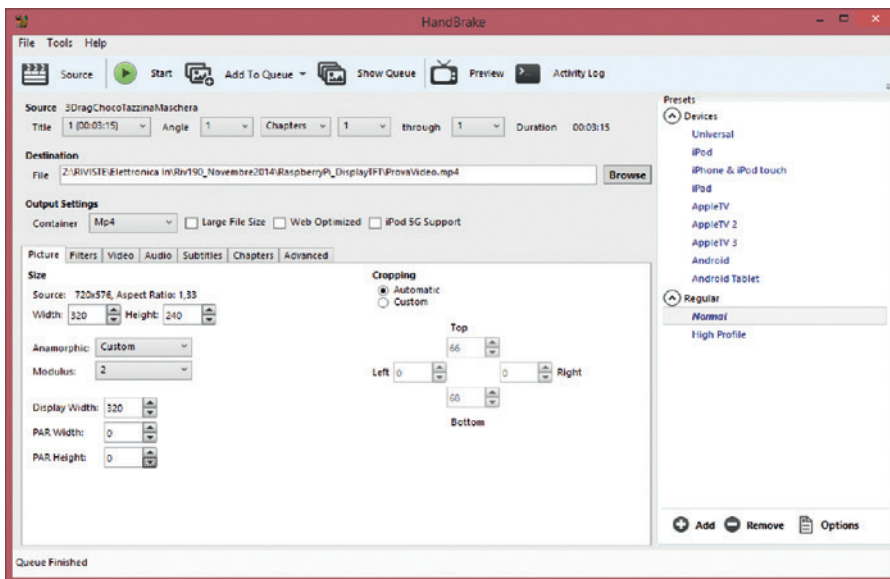


Fig. 23

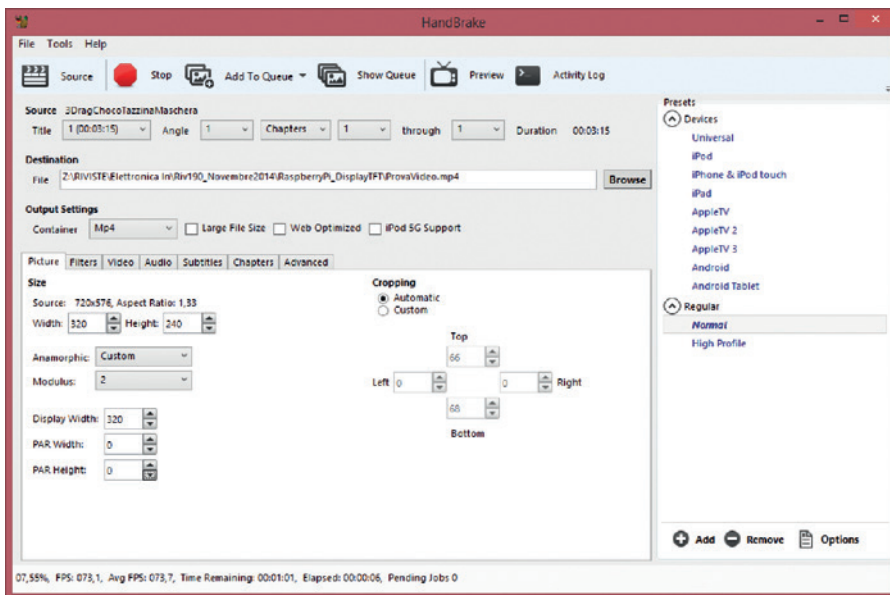


Fig. 24

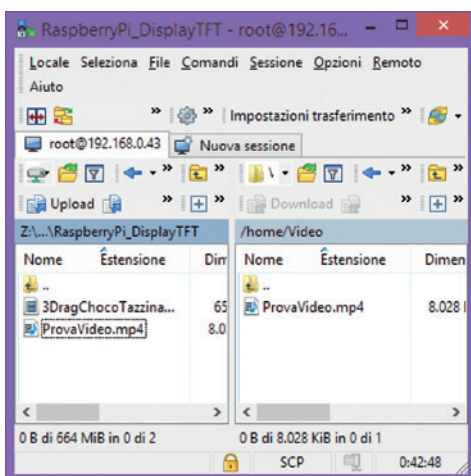


Fig. 25

caso di un utilizzo con alimentazione da batteria o se si vuole comunque spegnere la retroilluminazione il controllore del touch STMPE610 possiede due pin di I/O ed uno di questi è collegato al transistor che controlla la retroilluminazione. Questo pin è raggiungibile da linea di coman-

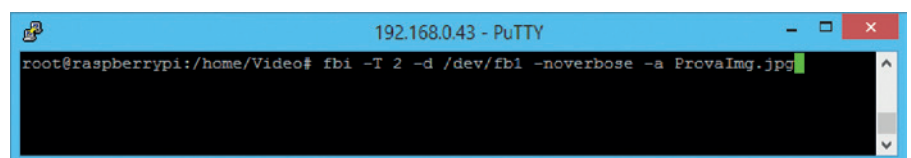


Fig. 26

do come un terminale GPIO di Raspberry Pi, riconoscibile come GPIO 252. Per accedere al GPIO è necessario esportare i suoi file di device con il comando:

```
echo 252 > /sys/class/gpio/export
```

Con il comando:

```
ls -l /sys/class/gpio
```

si verifica la effettiva esistenza del GPIO. Dichiariamo il pin di I/O come di output con il comando:

```
echo 'out' > /sys/class/gpio/gpio252/direction
```

Ora per accendere il display portiamo ad "1" il valore del pin con il comando:

```
echo '1' > /sys/class/gpio/gpio252/value
```

Per spegnere la retroilluminazione utilizziamo il comando:

```
echo '0' > /sys/class/gpio/gpio252/value
```

SCRIVIAMO UN'APPLICAZIONE

Cosa possiamo fare con un display così grazioso oltre a reindirizzare la console o l'interfaccia grafica del desktop? Possiamo scriverci le nostre applicazioni, in modo da comunicare con il GPIO e con eventuali altri sensori esterni. Le possibilità sono ovviamente infinite. Se però vogliamo mantenerci su qualcosa di estremamente semplice queste si ridu-

cono un po', senza tuttavia dover rinunciare a dei buoni risultati. Nel nostro caso ci manteniamo nel filone degli argomenti trattati nel libro citato all'inizio dell'articolo e nella ormai lunga serie di articoli presentati nella rivista e dedicati al mondo embedded ed a Raspberry Pi in particolare. Utilizzeremo il linguaggio Python per realizzare un'interfaccia grafica in grado di comandare le uscite di una scheda a quattro relè, in grado di commutare anche carichi alimentati con tensione di 220 Volt. Per interagire con il display grafico TFT sfrutteremo le funzionalità disponibili in pygame, la libreria del linguaggio Python destinata alla realizzazione di videogiochi ed applicazioni grafiche in genere. Nelle ultime distribuzioni di Raspbian la libreria pygame è già inclusa ed installata nella distribuzione. Se non lo fosse la installiamo semplicemente utilizzando il Package Manager di Python "pip" (Pip Installs Python), con il comando:

```
pip install pygame
```

Se non avessimo installato "pip" lo installiamo con il comando:

```
Apt-get install pip
```

Fatto? Proviamo il tutto utilizzando Python da linea di comando (rif. al solito libro). Verifichiamo che tutto funzioni digitando una dopo l'altra le seguenti istruzioni:

```
import pygame
import os
os.putenv('SDL_FBDEV', '/dev/fb1')
pygame.init()
lcd = pygame.display.set_mode((320, 240))
lcd.fill((255, 0, 0))
```

Listato 1

```
#!/usr/bin/python
# TFT_Rele.py.py

# Import librerie
import pygame
from pygame.locals import *
import os
from time import sleep
import RPi.GPIO as GPIO # Libreria Gestione GPIO

# Impostazione pin come output
GPIO.setmode(GPIO.BCM) # Utilizza numerazione GPIO
GPIO.setup(12, GPIO.OUT) # Set pin come OUTPUT
GPIO.setup(16, GPIO.OUT)
GPIO.setup(20, GPIO.OUT)
GPIO.setup(21, GPIO.OUT)

GPIO.output(12, False) # Porta i pin a off
GPIO.output(16, False)
GPIO.output(20, False)
GPIO.output(21, False)

# Definizione colori (R, G, B)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLACK = (0, 0, 0)

# Impostazione variabili di sistema per definizione device per
# display e touchscreen
os.putenv('SDL_FBDEV', '/dev/fb1')
os.putenv('SDL_MOUSEDRV', 'TSLIB')
os.putenv('SDL_MOUSEDEV', '/dev/input/touchscreen')

# Stato iniziale rele
pin12='off'
pin16='off'
pin20='off'
pin21='off'

# Costruisce la classe pygame
pygame.init()

# Toglie il puntatore del mouse dal display
pygame.mouse.set_visible(False)
lcd = pygame.display.set_mode((320, 240))

# Schermo nero
lcd.fill(BLACK)
pygame.display.update()

# Imposta font per le scritte
font_big = pygame.font.Font(None, 50)

# Dizionario Key=pin valore=coord. centro scritta
touch_buttons = {'12':(80,60), '16':(240,60), '20':(80,180), '21':(240,180)}

# Prepara la schermata iniziale
for k,v in touch_buttons.items():
    b = k + ' off' # Scritta con pin + off
    text_surface = font_big.render('%s'%b, True, RED) # Scritta rossa
    rect = text_surface.get_rect(center=v) # Centra la scritta
    lcd.blit(text_surface, rect) # Posiziona la scritta sul display
pygame.display.update()

# Ciclo principale del programma. Per sempre
while True:
    # Attesa evento dal display
    for event in pygame.event.get():
        if(event.type is MOUSEBUTTONDOWN): # Display toccato
```

(Continua)

Listato 1 (segue)

```
pos = pygame.mouse.get_pos()
print pos
elif(event.type is MOUSEBUTTONDOWN): # Via il dito dal display
    pos = pygame.mouse.get_pos() # Recupera posizione del dito
    print pos
    # Calcola in quale quarto dello schermo sta il dito
    x,y = pos
    if y < 120:
        if x < 160: # In alto a sinistra
            print 12
            v = touch_buttons.get('12') # Ritrova centro scritta
            if pin12 == 'off': # Se stato spento
                pin12 = 'on' # Cambia stato in on
                GPIO.output(12, True) # Livello pin alto
                b = '12 on' # Scritta on
                color = GREEN # Colore verde
            else:
                pin12 = 'off'
                GPIO.output(12, False)
                b = '12 off'
                color = RED
        else: # In alto a destra
            print 16
            v = touch_buttons.get('16')
            if pin16 == 'off':
                pin16 = 'on'
                GPIO.output(16, True)
                b = '16 on'
                color = GREEN
            else:
                pin16 = 'off'
                GPIO.output(16, False)
                b = '16 off'
                color = RED
    else:
        if x < 160: # In basso a sinistra
            print 20
            v = touch_buttons.get('20')
            if pin20 == 'off':
                pin20 = 'on'
                GPIO.output(20, True)
                b = '20 on'
                color = GREEN
            else:
                pin20 = 'off'
                GPIO.output(20, False)
                b = '20 off'
                color = RED
        else: # In basso a destra
            print 21
            v = touch_buttons.get('21')
            if pin21 == 'off':
                pin21 = 'on'
                GPIO.output(21, True)
                b = '21 on'
                color = GREEN
            else:
                pin21 = 'off'
                GPIO.output(21, False)
                b = '21 off'
                color = RED
    # Ridisegna il quarto di display interessato
    # e lo aggiorna
    text_surface = font_big.render('%s'%b, True, color)
    rect = text_surface.get_rect(center=v)
    lcd.fill(BLACK, rect) # Cancella la scritta precedente
    lcd.blit(text_surface, rect) # Posiziona nuova scritta
    pygame.display.update() # Aggiorna display
    sleep(0.3) # Un attimo di pausa
GPIO.cleanup()
```

```
pygame.display.update()
pygame.mouse.set_
visible(False)
lcd.fill((0,0,0))
pygame.display.update()
```

Tra le istruzioni importanti notiamo quella che imposta il device fb1 come framebuffer, l'init della libreria pygame. Successivamente incontriamo la definizione del display cui abbiamo dato nome "lcd" con dimensioni 320 x 240 pixel. Le altre istruzioni, come intuibile servono a cambiare colore allo sfondo dello schermo, utilizzando la solita codifica RGB. Concludiamo con un ultimo progetto che ci permette di azionare quattro relè con comandi touch sul display TFT. Utilizziamo la scheda 2846-RELAY4CH (reperibile presso Futura Elettronica) che porta a bordo quattro relè, è alimentata a 5 Volt ed è pilotabile con ingressi digitali diretti oppure optoisolati, che accettano tensioni da 1,5 a 5V e quindi è compatibile con i livelli di uscita del GPIO di Raspberry Pi. Scegliamo l'opzione diretta posizionando il jumper sui terminali COM e GND.

Ora colleghiamo la scheda a Raspberry Pi come nello schema di **Fig. 27**.

Colleghiamo l'alimentazione e gli ingressi di pilotaggio dei relè a quattro pin del GPIO di Raspberry Pi. Noi abbiamo scelto i pin GPIO12, 16, 20 e 21, molto comodi da collegare sul GPIO della versione B+. Per l'alimentazione saldiamo una strip di connettori maschi sulla "replica" del GPIO sul modulo TFT e preleviamo l'alimentazione dai pin 2 (positivo) e 6 (massa) con dei cavetti femmina-femmina. Nel **Listato 1** vediamo il programma di pilotaggio dei relè. È sempre un programma

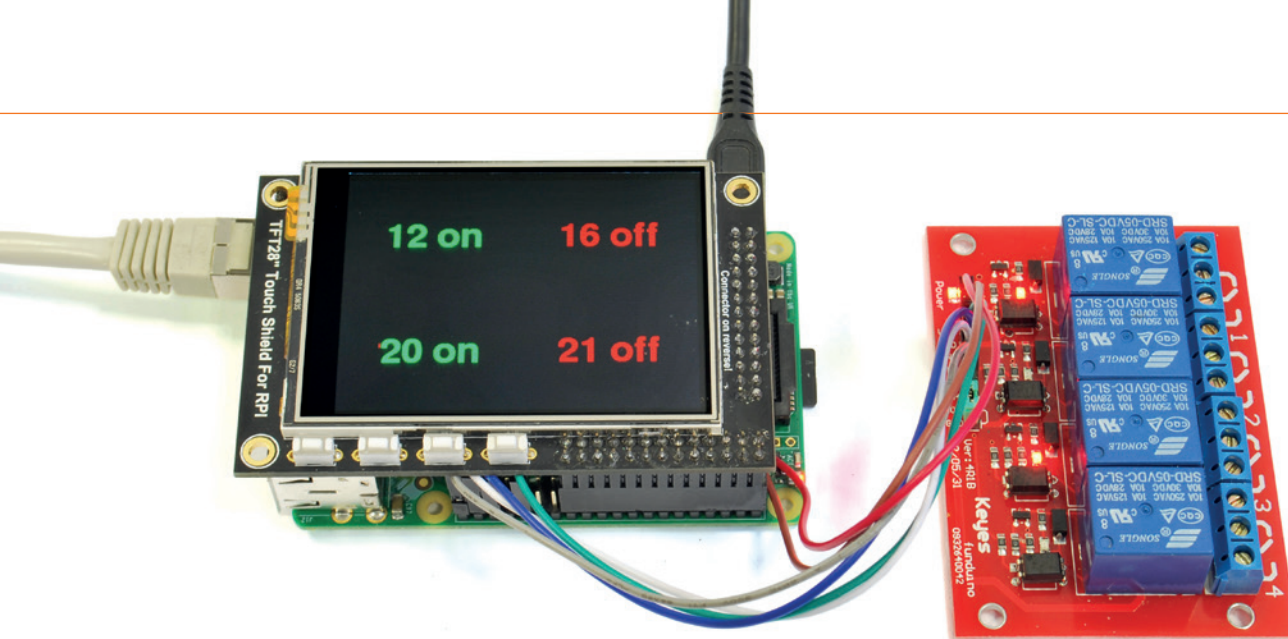


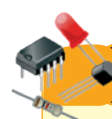
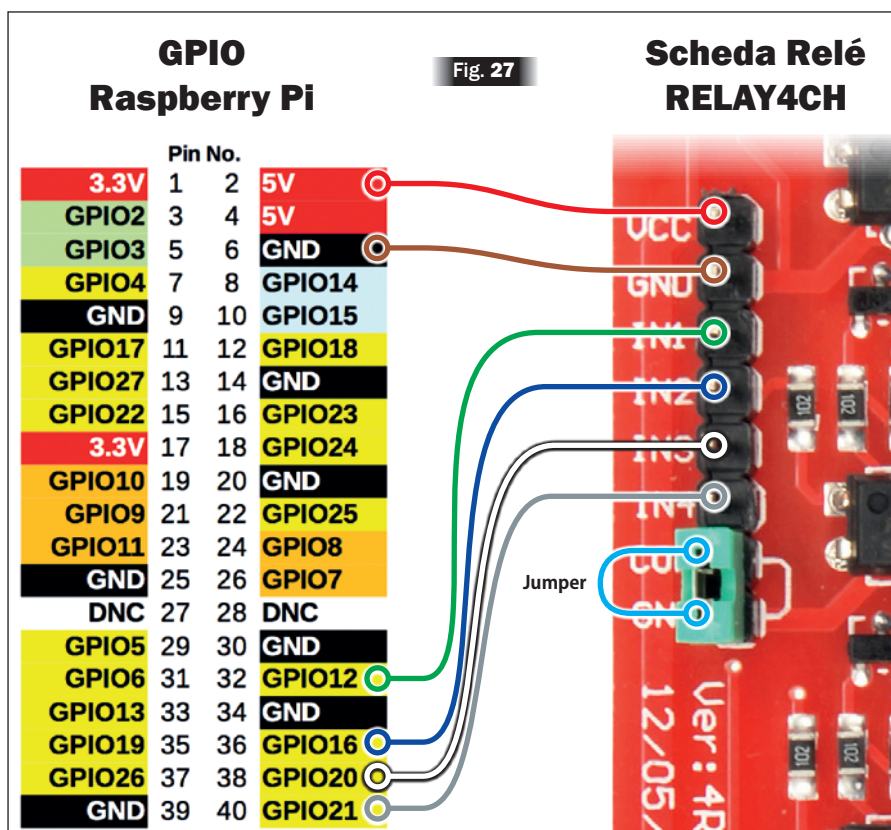
Fig. 28

Python. Per l'interfaccia grafica abbiamo utilizzato la libreria pygame, che vi abbiamo appena descritto. Il listato è ricco di commenti. La logica del programma consiste nella costruzione iniziale dell'interfaccia grafica. Il dizionario contiene le coordinate della posizione del centro dove vogliamo posizionare la scritta di ciascun

pin. Il ciclo "for" al di fuori del loop principale del programma prepara l'area del display con le scritte per i quattro pin. Al "nome" di ciascun pin viene aggiunta la dicitura "off". Il tutto viene colorato in rosso e posizionato in modo che il centro corrisponda alle coordinate contenute nel dizionario. Alla fine viene aggiornato fisi-

camente il display e le scritte diventano visibili. Una volta in esecuzione il programma, toccando una specifica scritta, il relè corrispondente cambia di stato, e la scritta indica lo stato corrente evidenziato dal colore associato, rosso con relè spento e verde con relè attivato, come visibile nella foto di Fig. 28.

Nei prossimi numeri vedremo come utilizzare il display con altri shield e con funzionalità arricchite, come la possibilità di azionare una periferica da display o da una pagina web. Per ora divertitevi a sperimentare.



per il MATERIALE

Il software presentato in questo articolo è scaricabile gratuitamente dal sito della rivista (www.elettronica.in.it). Le board utilizzate sono invece disponibili presso Futura Elettronica: Raspberry Pi B+ (Euro 36,00), TFT28TOUCHRASP (Euro 46,00) e scheda relè (cod. 2846-RELAY4CH, Euro 15,00) Tutti i prezzi si intendono IVA compresa.

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

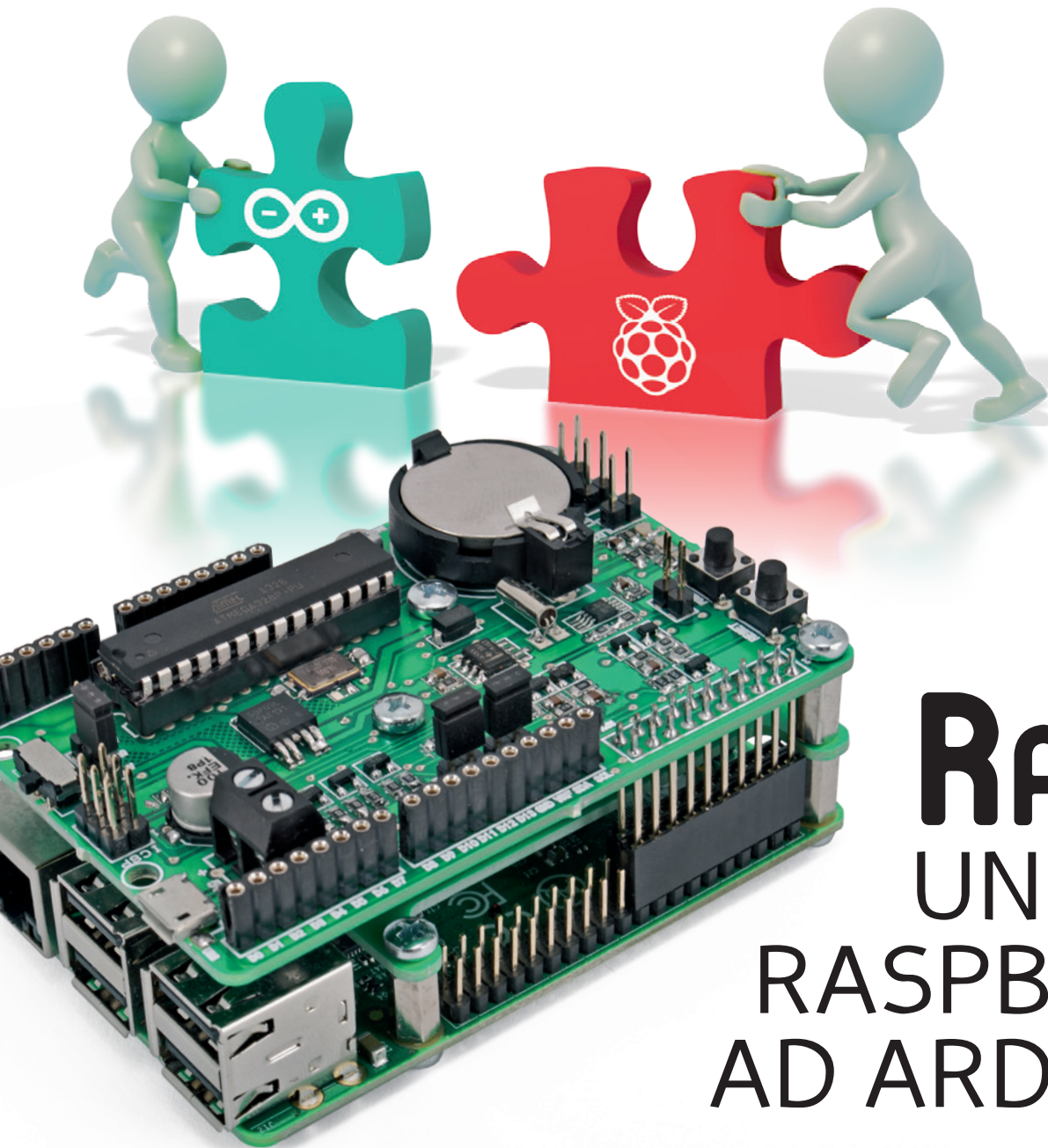
Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

Elettronica In

www.elettronica.in.it

oltre l'elettronica

Bridge tra il mondo Raspberry Pi e Arduino, consente di sfruttare la dotazione hardware e la potenza di calcolo della prima con gli shield e le applicazioni della seconda.



RANDA UNIAMO RASPBERRY AD ARDUINO

di DANIELE DENARO

I mondi di Raspberry e Arduino tendono sempre più a convergere per offrire agli utenti un ponte fra le enormi potenzialità della piattaforma Linux embedded e la quantità di shield e sketch disponibili per Arduino. Raspberry e Arduino sono ormai diventate le assolute star nel firmamento degli hobbisti digitali e non solo: Raspberry è un completo sistema multitasking con un professionale sistema operativo Linux, mentre Arduino è monotask e, per così dire, "nature", ma ha una naturale predisposizione all'input/output e, soprattutto, ha

un praticissimo sistema di sviluppo. L'IDE (Integrated Development Environment) di Arduino, gira su qualunque PC e permette di creare programmi (chiamati sketch) e scaricarli automaticamente sul processore.

Inoltre Arduino può contare su una grandissima serie di schede aggiuntive, sviluppate, dall'organizzazione Arduino e da terze parti, per soddisfare qualunque necessità di interfaccia verso il mondo esterno, sia in input che in output.

Il modo migliore di far collaborare i due mondi è

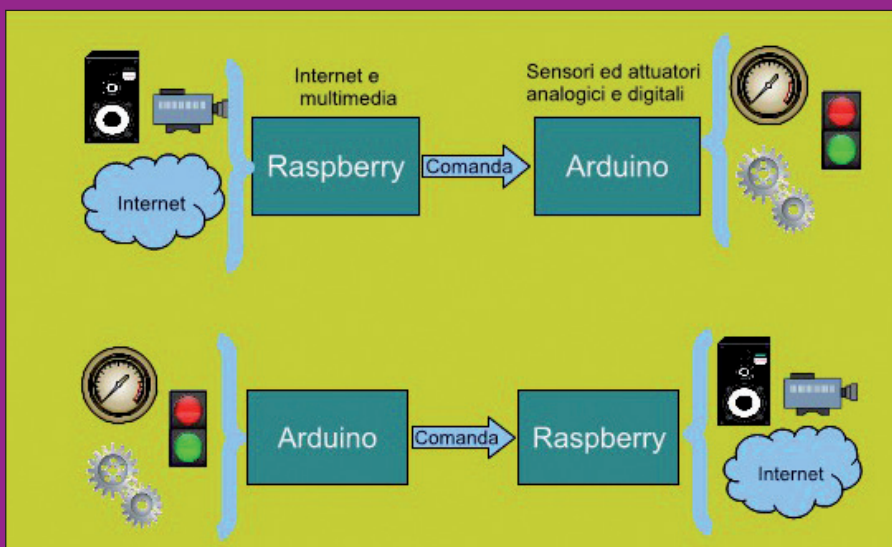


Fig. 1 - Possibili utilizzi.

sfruttare le specifiche caratteristiche di entrambi secondo varie configurazioni. Per esempio:

- Raspberry Pi potrebbe utilizzare Arduino come periferica configurabile, il cui programma è scelto di volta in volta, sulla base di una libreria di sketch predisposti, e viene caricato automaticamente su Arduino;
- al contrario, Arduino potrebbe funzionare come controllore di Raspberry Pi, ovvero potrebbe lanciare dei comandi e dialogare con essa; in pratica, Raspberry diventerebbe un'appendice sofisticata di Arduino, permettendo a quest'ultimo l'accesso ad ambienti complessi come la rete, oppure complesse elaborazioni di dati o -infine- l'accesso alla multimedialità.

Il collegamento hardware può essere implementato facilmente tramite il collegamento USB, o tramite una connessione attraverso la porta seriale, se l'hardware è completamente integrato. Ed è proprio questo che abbiamo realizzato: una scheda per Raspberry con processore Atmel basata su ATmega 328

senza USB, ma con tutto il restante hardware di Arduino Uno, compresi, ovviamente, i connettori per il montaggio di shield. Il dialogo fra Raspberry ed Arduino avviene tramite la porta seriale che ambedue possiedono. Questa scheda, che abbiamo chiamato RandA (Raspberry and Arduino), è pronta per interfacciarsi con la nuova Raspberry Pi Plus, pur essendo compatibile con la Raspberry Pi tradizionale.

Poiché su Raspberry Pi si sente la mancanza di un RTC (Real Time Clock) e di un pulsante di accensione e spegnimento programmato, abbiamo pensato di inserire entrambi questi elementi sulla nostra scheda RandA. Infatti, anche su Raspberry Pi Plus, l'accensione avviene collegando l'alimentazione attraverso il connettore micro USB o tramite un interruttore a monte fornito dall'utente, mentre per lo spegnimento è necessario lanciare il comando di "shutdown" e successivamente scollegare l'alimentazione. Per risolvere questa macchinosità abbiamo previsto sulla scheda anche i due connettori di alimentazione a 5 volt, un ulteriore connettore micro USB

ed un morsetto a vite. Questi connettori alimentano Raspberry Pi e Arduino tramite un circuito comandato da un pulsante. Inoltre, un software installato su Raspberry Pi rileva l'azione del pulsante e lancia lo shutdown e il successivo spegnimento dell'alimentazione. Il pulsante, quindi, comanda sia l'accensione che lo spegnimento di Raspberry Pi. Un connettore a due pin permette di replicare il pulsante all'esterno di un eventuale contenitore.

Ma c'è di più: è possibile accendere Raspberry Pi e Arduino ad orari programmati; infatti abbiamo inserito un RTC con sveglia programmabile. Questo RTC, oltre a fornire un orologio a Raspberry Pi anche quando non è connesso ad Internet, è anche collegato al circuito di accensione. L'orologio è gestito da Raspberry Pi tramite la connessione I²C e per il suo utilizzo

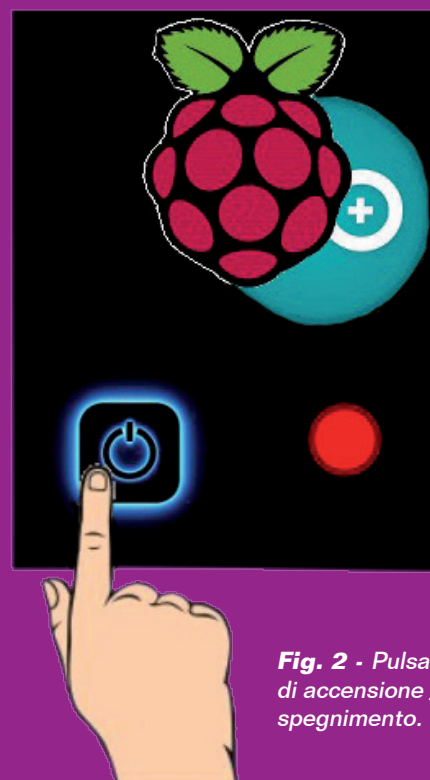


Fig. 2 - Pulsante di accensione / spegnimento.

abbiamo predisposto alcuni comandi Linux; è anche possibile gestire l'orologio via web, come vedremo nei prossimi articoli di approfondimento.

Infine Arduino può accendere Raspberry Pi; infatti, tramite un deviatore, è possibile disabilitare il circuito di accensione/spengimento per Arduino, che si trova così ad essere sempre alimentato. Poiché RandA contiene un Arduino basico, senza regolatore di tensione e senza circuito USB, il suo consumo può essere bassissimo, a tal punto che sarebbe pensabile alimentarlo con batterie e pannelli solari.

A questo punto manca solo un collegamento fra Arduino ed il pulsante di accensione, cosa che abbiamo realizzato tramite un pin di output.

In questo modo Arduino può funzionare come controllore di Raspberry Pi ed accenderlo nel

caso si verifichi una qualche condizione rilevata dai suoi pin digitali o analogici. Dopo l'accensione, Arduino può passare la palla a Raspberry Pi ed eseguire i suoi eventuali comandi, oppure può continuare a gestire l'evento di sua iniziativa attivando dei programmi su Raspberry Pi.

COME DIALOGANO RASPBERRY PI E ARDUINO

Come si è detto, il collegamento di base tra i due mondi avviene tramite la linea seriale di cui sono predisposti sia Arduino che Raspberry Pi. Il link fisico è realizzato tramite il connettore di interfaccia di Raspberry Pi, con cui RandA è collegato ma solo per le prime 20 posizioni. La porta seriale è utilizzata sia per la programmazione di Ar-

duino, sia per il dialogo tra lui e Raspberry Pi. Inoltre, tramite un GPIO di Raspberry, è gestito il Reset di Arduino.

La programmazione di Arduino è realizzata tramite uno script Linux di interfaccia che attiva il reset in modo opportuno; questo perché in Arduino standard, l'uploading degli sketch è preceduto da un segnale di reset realizzato in automatico dal circuito che gestisce l'USB, che qui manca.

La programmazione di Arduino può avvenire in locale tramite l'ambiente IDE installato su Raspberry Pi e adattato da noi; oppure in remoto, utilizzando un IDE per PC, modificato per gestire un Arduino collegato in rete simulando una porta remota.

Infine, come vedremo nei prossimi articoli, è possibile caricare sketch (compilati) su Arduino tramite interfaccia web.

Se Raspberry Pi vuole utilizzare Arduino può inviargli delle direttive o dei dati tramite programmi che utilizzano la porta seriale di Raspberry Pi e di cui esistono esempi predisposti; esempi sia in C che in Java.

Se invece è Arduino a gestire il sistema, allora bisogna che su Raspberry Pi sia presente un processo in "background" pronto a rilevare l'arrivo di comandi. Per ottenere ciò, siamo stati



Fig. 3 - RandA prevede l'accensione programmata di Raspberry Pi.

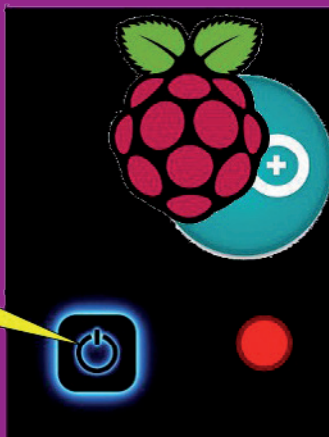
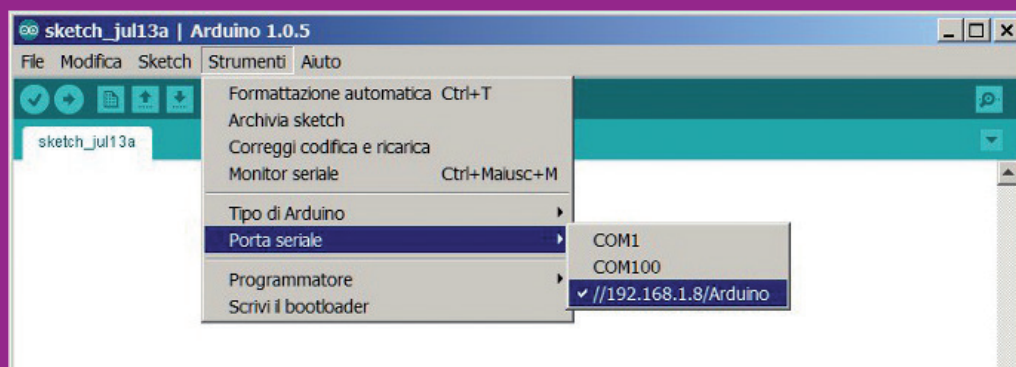
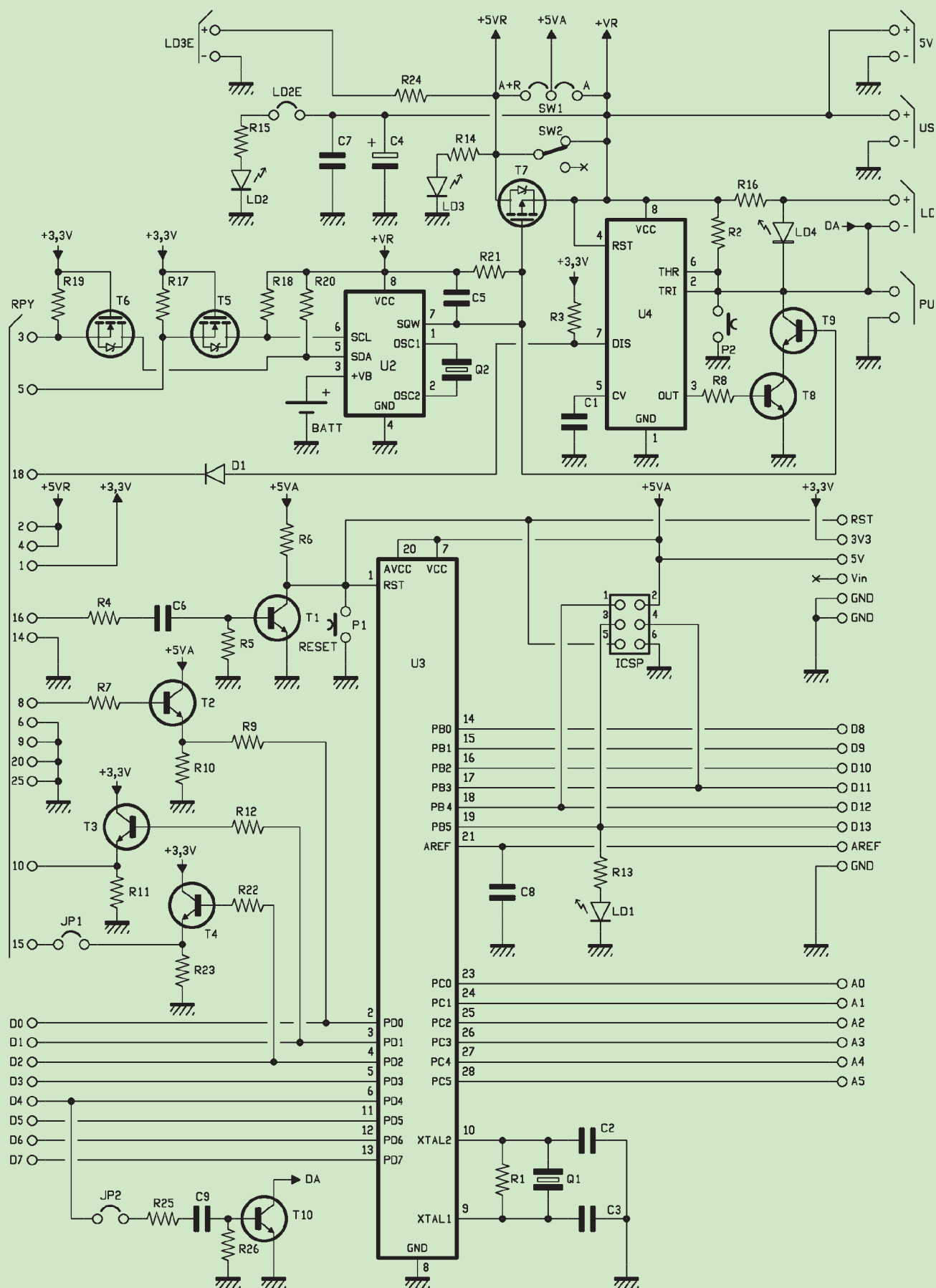


Fig. 4 - IDE con porta remota.





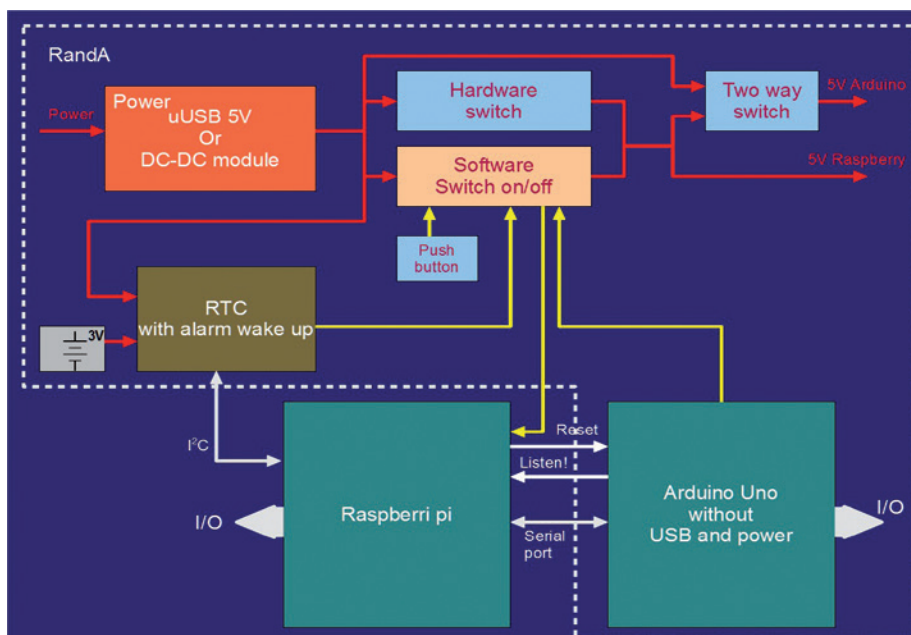


Fig. 5 - Schema a blocchi di RandA.

costretti ad utilizzare un ulteriore GPIO di Raspberry Pi ed un pin di Arduino per risvegliare l'interesse verso questo processo; ciò, a causa dell'unica linea seriale di comunicazione.

In ogni caso abbiamo predisposto una libreria Arduino per:

- l'invio di comandi Linux e la ricezione di un'eventuale risposta;
- l'apertura di un file su Raspberry Pi e la lettura e/o scrittura su di esso.

Inoltre sono stati predisposti comandi (script e programmi) Linux per gestire l'orologio, la sveglia e l'invio di e-mail.

SCHEMA ELETTRICO

Diamo ora uno sguardo alla struttura generale della scheda RandA. Come si è detto, la comunicazione tra Raspberry e RTC (DS1339) avviene tramite collegamento I²C, mentre il collegamento con il circuito del pulsante è effettuato tramite il pin GPIO24 (in input). Arduino, poi, è collegato a Raspberry Pi tramite il suo pin D2 (in output) ed il pin GPIO22 (in input), oltre che tramite il suo pin di reset (in input)

ed il pin GPIO23 (in output). Con il primo collegamento, Arduino può richiamare l'attenzione di Raspberry Pi tramite un processo ascoltatore attivo su di esso, per fare in modo che Raspberry Pi si metta in ascolto sulla porta seriale ed interagisca con Arduino. Il secondo collegamento è reso necessario dall'eliminazione del circuito USB, che realizza una gestione automatica del RESET associandolo al segnale DTR prodotto dalla simulazione della porta seriale. Infine Arduino, per poter accendere Raspberry Pi quando viene fatto funzionare da controller, è connesso in modo opzionale tramite il pin D4 (programmato in output con pull-up) al pulsante.

Lo schema generale a blocchi è rappresentato in Fig. 5. Vediamo adesso nei dettagli le sezioni che compongono la scheda RandA.

MODULO RTC

L'orologio è realizzato da un DS1339 che dialoga con Raspberry Pi tramite un bus I²C; per la precisione, il DS1339 viene visto all'indirizzo 0xFFD... (slave address) del protocollo I²C. Ri-

cordiamo che nel protocollo I²C, l'ultimo bit dell'indirizzo varia in funzione della modalità (lettura o scrittura), quindi 0xFFD0 o 0xFFD1.

Le due linee SCL e SDA del bus I²C interfacciano Raspberry Pi tramite due adattatori di tensione bidirezionali realizzati mediante i MOSFET T6 e T5; infatti il DS1339 è alimentato a 5 volt, mentre la scheda Raspberry Pi, come si sa, è alimentata a 3,3 volt. Questo pratico sistema di adattatore di tensione bidirezionale è stato già utilizzato in altri shield e funziona in questo modo, con riferimento alla linea SDA: quando il contatto 3 del connettore di interfaccia di Raspberry Pi si trova a livello logico basso (0 volt) il source del MOSFET T6 (mantenuto a riposo a 3,3V dal resistore di pull-up R19) va a potenziale di massa e il drain trascina più o meno allo stesso potenziale l'SDA dell'RTC. Quando invece il contatto 3 si trova a 1 logico (3,3V) il MOSFET è interdetto perché il potenziale di source eguaglia quello di gate; in queste condizioni l'SDA dell'RTC viene tenuto a +V_R dal resistore di pull-up R20. Il funzionamento dello stadio relativo a T5 è analogo. Il DS1339 contiene un orologio (data ed ora) ed un paio di sveglie programmabili, di cui solo una viene utilizzata dal nostro software. Il DS1339 lo abbiamo alimentato con la tensione di 5 volt; ma nel caso questa venisse a mancare, è presente un batteria a bottone di 3 volt che mantiene il funzionamento in stand-by. Il circuito di intervento della batteria in tampone è compreso nel DS1339.

La sveglia può essere programmata sulla base dei soli minuti trascorsi o dell'ora e dei minuti trascorsi, ovvero a seconda del giorno (del mese o della settimana-

na), dell'ora e dei minuti. Una volta programmata la sveglia, bisogna attivarla; quando ciò avviene, l'uscita 7 di U2 (DS1339) viene posta a valore alto (5V) e quindi interdice il MOSFET T7 (a canale P), che a sua volta blocca l'alimentazione. Quando la sveglia scatta, porta a livello logico basso (0V) l'uscita 7 di U2 e quindi attiva il MOSFET T7, che fornisce l'alimentazione. Purtroppo, quando la sveglia viene disabilitata, non viene posta a valore basso costante, ma ad una condizione di generatore di onda quadra con differenti frequenze programmabili. Questo ci ha costretto ad inserire un filtro passabasso sul contatto di U2, in modo che con la massima frequenza di 32.765 kHz questa uscita desse un segnale corrispondente al livello basso costante e quindi mantenesse abilitato il T7. In sostanza, quando Raspberry viene accesa disabilita la sveglia, mantenendo l'alimentazione, poi fa un reset della sveglia ponendola ad un valore irraggiungibile (tutti zeri). In questo modo, un successivo spegnimento del sistema sarà definitivo e per riaccendere il sistema stesso occorrerà procedere manualmente, a meno che, durante il funzionamento, la sveglia non sia stata di nuovo programmata. Infatti lo spegnimento del sistema avviene abilitando la sveglia, operazione che ha come effetto quello di mandare in stato di off il MOSFET T7. A questo punto lo si potrà riaccendere manualmente aggirando la sveglia tramite il circuito on-off, oppure aspettando lo scadere del tempo, se la sveglia è stata impostata. La gestione software dell'orologio è ottenuta tramite la libreria "wiringPi" installata su Raspberry; questa libreria open source, oltre alla gestione dei pin GPIO

di Raspberry Pi, implementa anche il driver per il bus I²C e la libreria C++ per utilizzarlo. È stata installata anche la libreria "pi4j" che permette di pilotare i pin GPIO e il bus I²C con il linguaggio Java. Ma degli aspetti software e dei programmi predisposti parleremo in prossimi articoli dedicati.

BLOCCO ALIMENTAZIONE E ACCENSIONE/SPEGNIMENTO

L'alimentazione a 5 V tramite connettore microUSB può essere fornita da qualunque alimentatore ormai standardizzato per cellulari e smartphone. Se si vuole che Raspberry Pi e/o Arduino alimentino periferiche con buon assorbimento, bisogna però orientarsi su alimentatori capaci di erogare da 2 a 3 amper di corrente. In caso contrario, sono sufficienti gli alimentatori standard da 1A.

Inoltre si può alimentare il sistema con 5 V anche tramite un connettore a morsetto (5VIN), che sulla RandA è connessa in parallelo con + e - del connettore microUSB, utile, ad esempio, nel caso si debba associare un gruppo di continuità o prendere alimentazione da un dispositivo che non ha l'uscita USB.

Un interruttore generale (SW1) può dare tensione, a prescindere dalla sveglia o da altri switch software, nel caso si voglia alimentare il sistema in modo costante. L'alimentazione non riguarda solo Raspberry Pi, ma anche Arduino e gli eventuali shield ad esso aggiunti. Si è scelta, però, la strada di separare le due alimentazioni, in modo da permettere ad Arduino, facoltativamente, di essere sempre alimentato. Questa è un'opzione che si può attivare utilizzando un deviatore (SW2) che separa a monte le due alimentazioni.

Poiché su RandA la struttura Arduino è priva del classico regolatore lineare di tensione e del previsto circuito di interfaccia TTL/USB, la nostra scheda presenta un consumo veramente basso e potrebbe essere usata come controllore di Raspberry, accendendola e spegnendola. L'accensione e lo spegnimento possono essere ottenuti sia facendole programmare la sveglia, sia direttamente tramite il suo pin D4 collegato al pulsante di accensione. L'accensione potrebbe, quindi, essere soggetta ad altri eventi oltre al tempo, come una condizione digitale o analogica sui pin di Arduino.

Oppure, infine, Arduino può essere collegato alla stessa alimentazione di Raspberry Pi ed essere soggetto alle stesse condizioni di accensione o spegnimento.

Una delle principali caratteristiche di RandA è il circuito elettronico che permette il controllo software dell'accensione e spegnimento. Di questo circuito fanno parte: il MOSFET a canale P (a bassissima resistenza di on) T7, che agisce da interruttore, la sveglia del RTC e U4 (un timer 555), che funziona da flip-flop. Il T7 è capace di sopportare 3A e più di corrente e viene mandato in conduzione quando il suo gate è a livello basso (basta che la tensione su di esso non superi 2V), mentre è interdetto quando è a livello alto (5V).

Della sveglia abbiamo già parlato, quindi concentriamoci su U4, che serve a gestire l'alimentazione tramite pulsante (P1), remotizzabile mediante il connettore PUSH, al quale può essere collegato un altro pulsante, magari fissato ad un contenitore.

Anche i LED LD3 e LD4 sono replicabili collegando dei diodi luminosi esterni sui connettori corrispondenti. Il LED LD3 (ros-

so) segnala la presenza di alimentazione, mentre LD4 (giallo) è attivo solo durante il periodo di startup del sistema Raspberry Pi e si spegne quando questo ha completato la sua inizializzazione ed è pronto ad operare. Quando il sistema è spento e viene premuto il pulsante, il trigger del timer 555 fa scattare il flip-flop e porta l'uscita "out" a livello alto; tale condizione logica porta in conduzione T8, il quale, attraversando la giunzione di base di T9 come fosse un diodo, porta a livello basso il gate di T7 e lo fa condurre. Per riattivare il pulsante e renderlo predisposto per lo spegnimento, il flip-flop va resettato: di questo si occupa T9. Infatti se Raspberry Pi, tramite un comando allo startup, disattiva la sveglia, porta a livello basso il gate di T7 per altra via; allora T9 non conduce più e porta a livello alto il piedino Threshold (6) del 555, resettando il flip-flop. A questo punto una ulteriore pressione del pulsante attiverà di nuovo il flip-flop che attiverà anche il discharge del 555 che, questa volta, essendo Raspberry Pi accesa, potrà essere intercettato da un suo processo attivo dedicato a questo ed attento al pin GPIO24, collegato al Discharge (7) del 555. Il processo in background (demone) lancia il comando di shutdown che porterà allo spegnimento. Il pulsante funziona, quindi, sia per l'accensione, sia per lo spegnimento. Un LED giallo è collegato all'attività di accensione per segnalare lo "startup in progress", ma appena il sistema si è avviato, si spegne automaticamente, segnalando che Raspberry Pi è pronta ad accettare il login. Infatti il LED è alimentato attraverso la corrente di collettore del transistor T9. Nel caso sia collegato all'alimentazione in modo indipendente, tramite lo

Arriva il mini PC basato su Raspberry Pi



Kano, il mini PC in kit basato su Raspberry Pi, proposto su Kickstarter, grazie al quale ha raccolto ben 1,52 milioni di dollari, da ottobre è disponibile al pubblico dopo aver superato la fase di disponibilità limitata per il backers. Il kit comprende il mini PC composto da una Raspberry Pi, un'apposita tastiera Bluetooth con touchpad integrato, un case per Raspberry Pi e una scheda SD classe 10 con a bordo Kano OS, il sistema operativo Linux-like basato su Debian. Si

tratta di un sistema operativo mirato all'apprendimento per i più giovani e i meno esperti, che dispone di tool creati ad hoc. Oltre al software per l'apprendimento, Kano OS integra alcuni giochi, sempre graditi al grosso del pubblico, specialmente a quello più giovane. Il kit con relativo manuale di istruzioni e cavi è disponibile per l'acquisto al prezzo di 149.99 euro, spese di spedizione escluse; chi volesse acquistarlo potrà ordinarlo dal sito del progetto, <http://get.kano.me>.

switch SW2, Arduino può accendere o spegnere direttamente Raspberry Pi portando a livello alto il pin D4 che attiva T10 il quale a sua volta si trova in parallelo al pulsante.

MODULO ARDUINO

Come abbiamo già detto, su RandA è presente una versione di Arduino Uno privata del suo sistema di alimentazione e dell'interfaccia USB. La sua programmazione deve, perciò, avvenire tramite Raspberry Pi o tramite rete.

Su Raspberry Pi è stato installato l'IDE di Arduino per Linux, tramite il quale è possibile programmare Arduino e fare il caricamento (uploading) dello sketch compilato. L'IDE può essere lanciato da una console Linux remota con protocollo SSH, ma trattandosi di un ambiente grafico, occorre che la console possa gestire anche un server Xwindow. In caso contrario occorre collegarsi con tastiera e schermo a Raspberry Pi. Ma esiste un'altra possibilità per programmare Arduino con un

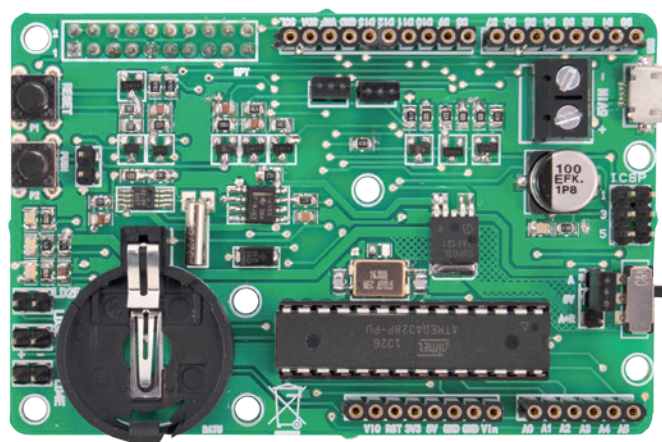
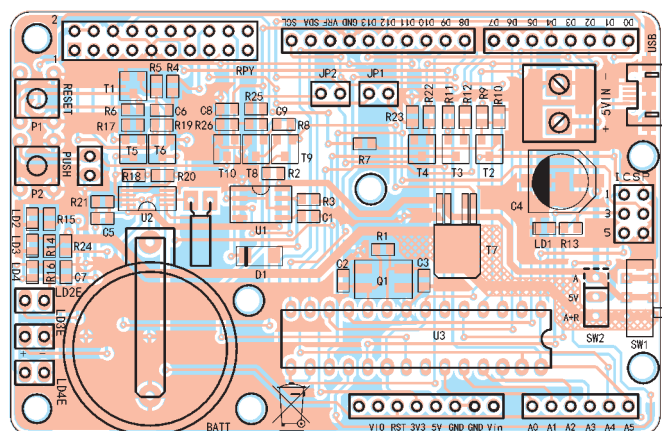
Elenco Componenti:

R1: 1 Mohm (0805)
 R2, R3, R6, R8: 10 kohm (0805)
 R4, R7: 4,7 kohm (0805)
 R5: 47 kohm (0805)
 R9: 1 kohm (0805)
 R10, R11: 4,7 kohm (0805)
 R12: 1 kohm (0805)
 R13÷R16: 470 ohm (0805)
 R17÷R21: 10 kohm (0805)
 R22: 1 kohm (0805)
 R23: 4,7 kohm (0805)
 R24: 470 ohm (0805)
 R25: 4,7 kohm (0805)
 R26: 10 kohm (0805)
 C1: 47 nF ceramico (0805)
 C2, C3: 15 pF ceramico (0805)
 C4: 100 µF 25 VL elettrolitico (E)
 C5÷C9: 100 nF ceramico (0805)
 U1: NE555D
 U2: DS1339U-33+
 U3: ATMEGA328P
 D1, D2: GF1M
 T1÷T4: BC817
 T5, T6: BSS138
 T7: SPD50P03L G
 T8÷T10: BC817

LD1: LED verde (0805)
 LD2: LED rosso (0805)
 LD3: LED verde (0805)
 LD4: LED giallo (0805)
 Q1: Quarzo 16 MHz (C7S)
 Q2: Quarzo 32.768 kHz
 P1, P2: Microswitch
 SW1: Deviatore slitta 90°
 USB: Connettore micro-USB

Varie:

- Morsetto 2 poli
- Zoccolo 14+14
- Jumper (4 pz.)
- Strip maschio 2 poli (6 pz.)
- Strip maschio 3 poli (3 pz.)
- Strip femmina 6 poli (1 pz.)
- Strip femmina 8 poli (2 pz.)
- Strip femmina 10 poli (1 pz.)
- Strip maschio/femmina 10 poli (2 pz.)
- Porta batteria CR2032
- Batteria CR2032
- Torretta M/F 10mm (2 pz.)
- Torretta F/F 10mm (7 pz.)
- Vite 3 MA 6 mm (14 pz.)
- Circuito stampato S1106



accesso remoto: utilizzare l'IDE Arduino per PC, da noi modificato per gestire una porta seriale remota. Questo IDE, lanciato su PC permette di vedere eventuali Arduino remotizzati ed appoggiati su Raspberry Pi; per ottenere questo effetto è stato necessario anche installare su Raspberry un software che dialoga con l'IDE e gestisce la porta seriale collegata con Arduino facendola vedere come una ulteriore porta accessibile dall'IDE aperta sul PC collegato in rete locale. Poiché RandA utilizza il collegamento seriale diretto con Raspberry Pi, è necessario avere la possibilità di attivare il reset tramite un pin di Raspberry Pi; ciò perché nell'Arduino standard il reset software è realizzato dallo stesso

circuito di interfaccia USB che ora manca. Il circuito di interfaccia USB che simula un collegamento seriale trasforma un segnale DTR in un impulso di reset. Noi possiamo ottenere lo stesso risultato, su RandA, utilizzando il pin GPIO23 di Raspberry Pi. Anche nel caso dei collegamenti tra Raspberry Pi e Arduino abbiamo lo stesso problema di adattamento di tensione che avevamo incontrato con U2 (RTC). T1, T2, T3, T4 e T10 realizzano questa separazione tra i livelli a 5 volt di Arduino e quelli a 3,3 volt di Raspberry. In particolare, T1 collega GPIO23 con il RESET di Arduino, mentre T2 e T3 interfacciano le linee seriali TX (D1) e RX(D0). T4, invece, permette ad Arduino di richiedere l'attenzione di

Raspberry Pi mediante il pin D2, collegato al GPIO22; infatti in Raspberry Pi è installato un processo in background che rileva il cambiamento di stato su questo pin e attiva un ulteriore processo che interagisce tramite seriale con qualunque sketch di Arduino che utilizzi una libreria predisposta a mandare comandi Linux eseguiti dal processo. In questo modo Arduino può comandare Raspberry Pi. Viceversa Raspberry Pi, tramite comandi, può installare su Arduino sketch compilati e memorizzati sul suo "file system". Un ulteriore collegamento opzionale (tramite ponticello) collega il pulsante di on/off con il pin D4 di Arduino, con l'interposizione di T10, per permettere ad esso di

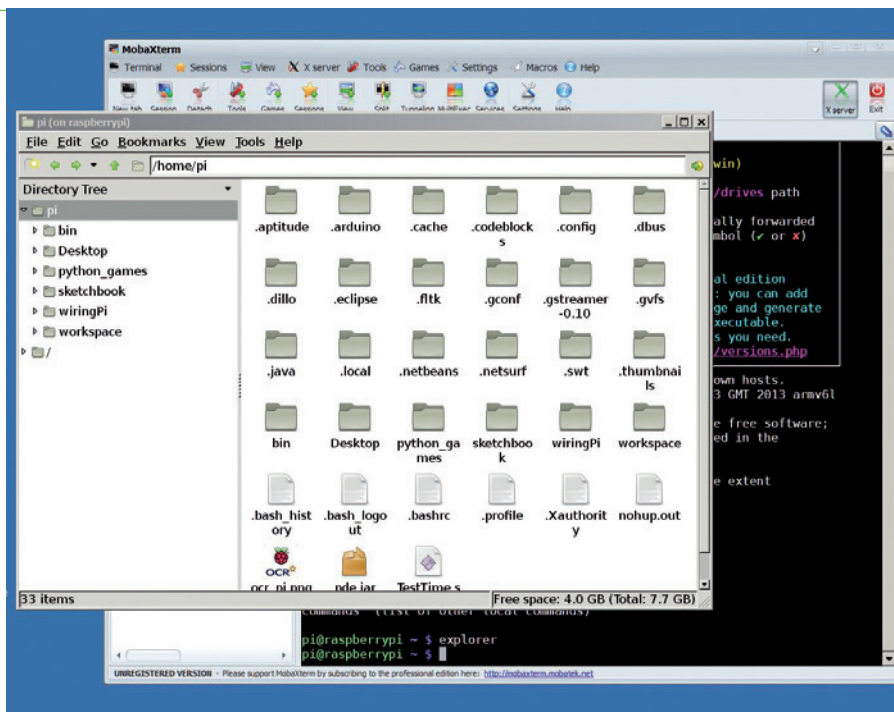


Fig. 7 - Sessione MobaXterm.

accendere e spegnere Raspberry Pi. Sia T1 che T10 lavorano ad impulsi perchè sono portati in saturazione tramite condensatori. Il resto di RandA è in standard Arduino, con tutti i connettori, il quarzo, il pulsante di reset ecc.

RASPBERRY PI

RandA si sovrappone a Raspberry Pi Plus mediante una struttura intermedia necessaria per ancorare la scheda, in quanto non è stato possibile utilizzare tutti e quattro i fori di Raspberry Pi a causa della necessità di rispettare sia lo standard dei connettori di espansione di Arduino sia il connettore di collegamento di Raspberry Pi, e soprattutto per la volontà di rispettare le dimensioni di Raspberry Pi Plus. In questo modo RandA e Raspberry Pi+ formano un sistema veramente compatto, mantenendo la possibilità di utilizzare gli shield di Arduino.

REALIZZAZIONE PRATICA

RandA è realizzata su circuito stampato a doppia faccia, praticamente con soli componenti SMD; una volta ottenuta la ba-

setta è perciò necessario dotarsi di un saldatore a punta molto fine, da 20 W, di fluxante in gel e filo di lega saldante da 0,5 mm di diametro. Nell'eseguire il montaggio date la precedenza ai componenti più piccoli, spalmando del fluxante sulle piazzole degli elementi che hanno i terminali molto vicini tra loro, per

evitare che lo stagno li unisca. Il quarzo va saldato lateralmente, perché ha i contatti ad L che partono dal lato e vanno sotto, dopo aver spalmato del fluxante sulle piazzole corrispondenti. Per l'ATmega 328 è richiesto uno zoccolo DIP tradizionale e per ospitare gli shield, gli stessi strip femmina caratteristici di Arduino Uno (devono avere i pin corti, in quanto non devono sporgere troppo dal lato inferiore dello stampato). Non dimenticate i jumper a passo 2,54 mm. terminate le saldature, bisogna inserire la pila a bottone nel portapila e il microcontrollore ATmega 328, già programmato con l'apposito bootloader Arduino, nel proprio zoccolo.

SVILUPPARE I PROGRAMMI

Arduino può essere programmato sia attraverso un IDE installato su computer, sia con un IDE locale (in ambiente Linux). Ma come si fa a utilizzare l'IDE su Raspberry Pi e, più in generale, come si può accedere all'ambiente Linux in modo pratico?

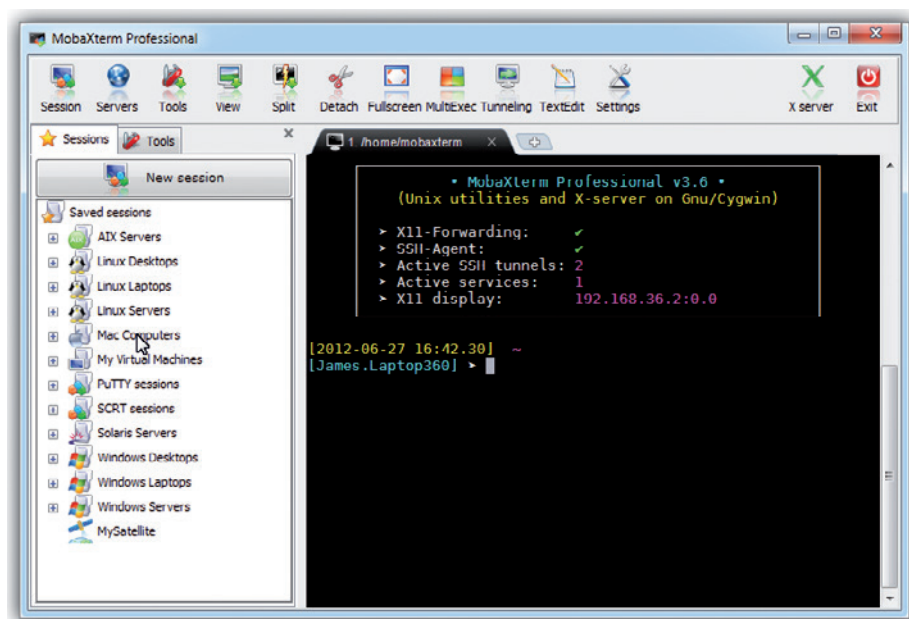


Fig. 6 - Schermata principale del software Mobaxterm.

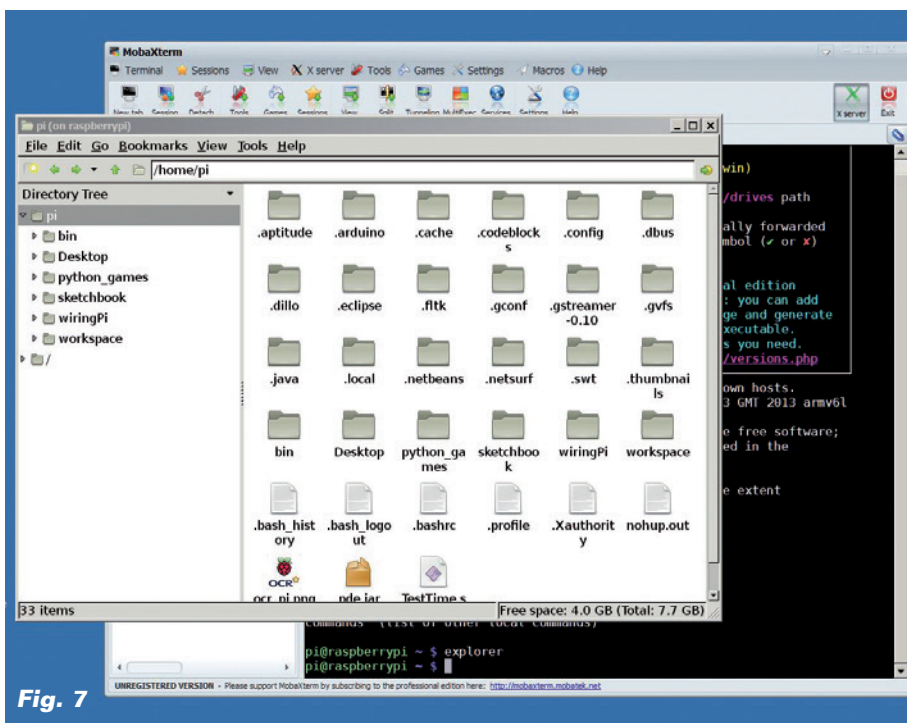


Fig. 7

Sicuramente Raspberry Pi può essere gestita collegando tastiera, mouse sull'USB e un monitor al connettore HDMI di cui è provvista, ma l'utilizzo usuale di Raspberry Pi è tramite rete locale, attraverso la porta ethernet o tramite una chiavetta WiFi. In questo caso l'accesso avviene con protocollo SSH, automaticamente abilitato nelle ultime distribuzioni del sistema operativo Raspbian. Ma invece dell'utilizzo del programma PuTTY, installabile su tutti i computer e molto conosciuto, vi suggeriamo il software "MobaXterm", più potente ed ugualmente gratuito (Fig. 6). Tale software permette di realizzare una connessione SSH come quella implementabile con PuTTY, ma in più dispone di un Xserver automaticamente gestito. Questo significa che le interfacce grafiche di Raspbian vengono correttamente visualizzate sul computer remoto, ovvero che è possibile gestire Raspberry Pi come se fossero collegati tastiera, mouse e video. Benché sia possibile lanciare il completo desktop, è preferibile, per ragioni di per-

formance, utilizzare l'interfaccia a console come base dei comandi ed attivare i programmi grafici quando servono.

Per esempio si possono lanciare:

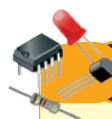
- comando di file management (come Explorer di Windows);
- editor grafico;
- ambienti di sviluppo per C (come Codeblocks), per Java, o l'IDE Arduino.

Le finestre XWINDOW appaiono sul computer remoto automaticamente grazie al Xserver incorporato in MobaXterm. In realtà MobaXterm permette di aprire anche una sessione FTP ed altro ancora. A questo punto non rimane che decidere che programma sviluppare ed utilizzare l'ambiente opportuno: editor per script Linux, IDE C, Java, Python ecc. per i programmi e l'IDE Arduino locale o remoto per Arduino. In ogni caso, distribuiremo software da installare che includerà sia i programmi di utilizzo di RandA, sia gli ambienti di sviluppo, le librerie per la gestione dei GPIO di Raspberry e diversi esempi. Infine, stiamo preparando un Web Server (con

Tomcat) che permetterà una gestione remota di RandA tramite Browser, per caricare su Arduino uno sketch compilato e memorizzato in una cartella di Raspberry Pi, gestire orologio e sveglia, aprire una console Arduino o spegnere il sistema. Sarà anche possibile richiedere una microSD pronta ad operare, cioè completa della distribuzione Raspbian e del software aggiuntivo per RandA.

CONCLUSIONI

RandA non è solo Arduino montato su Raspberry, ma vuol essere un sistema integrato dotato di gestione dell'alimentazione e di RTC programmabile. In questo modo si rendono facilmente implementabili applicazioni che accendono e spengono il sistema in base a condizioni esterne o alla temporizzazione. Con l'aggiunta di una batteria in tampone, magari sostenuta da un pannello solare, e di una chiavetta WiFi o di un modulo con modem GSM, è possibile realizzare un completo e potente sistema di controllo e gestione remoto. ■



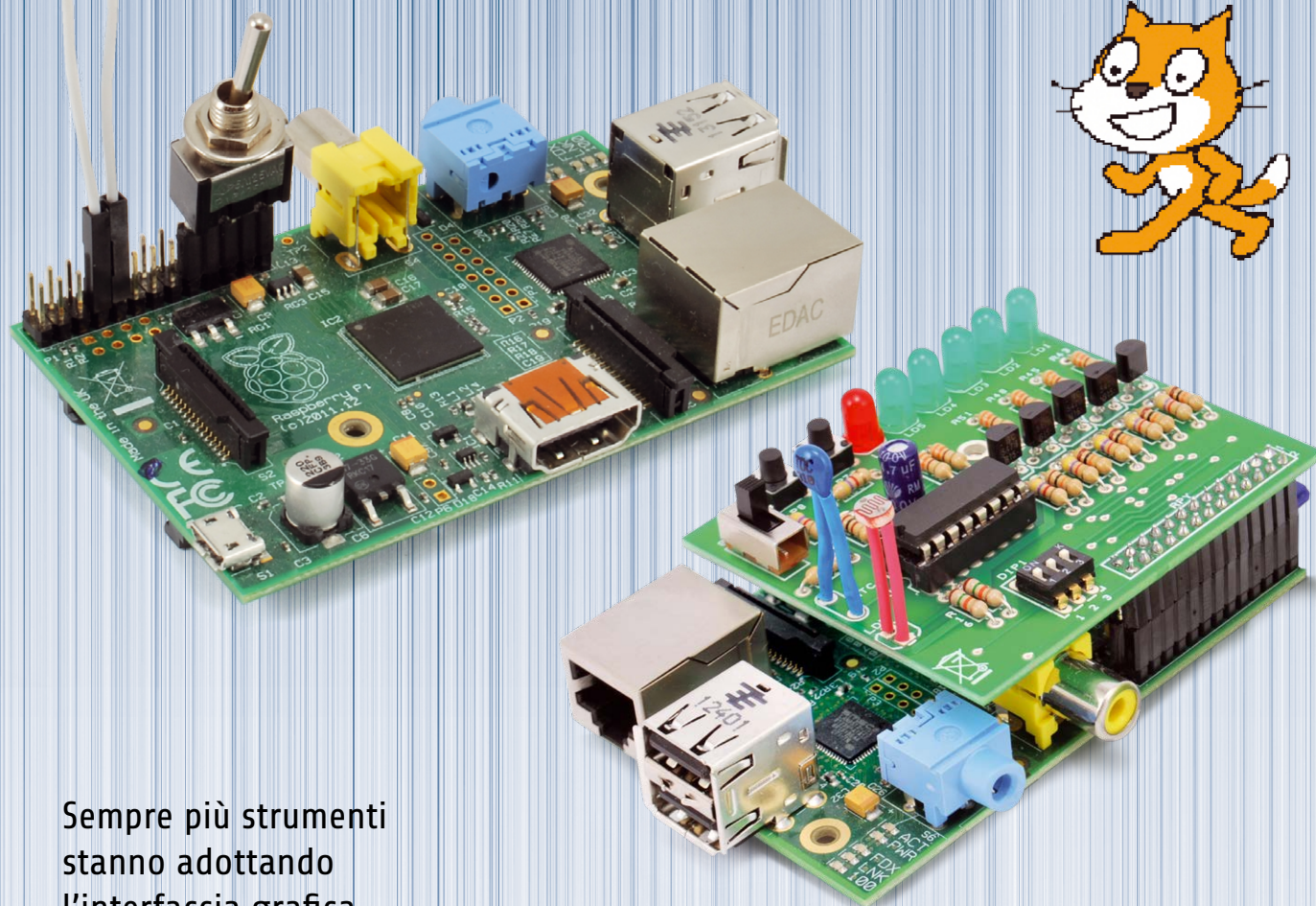
per il MATERIALE

Questo progetto può essere facilmente realizzato da chiunque abbia un minimo di esperienza nel montaggio manuale di componenti SMD e disponga dell'attrezzatura necessaria. La board RandA è comunque disponibile già montata e collaudata (comprese le minuterie) al prezzo di 39,00 Euro IVA compresa.

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

SCRATCH E RASPBERRY PI

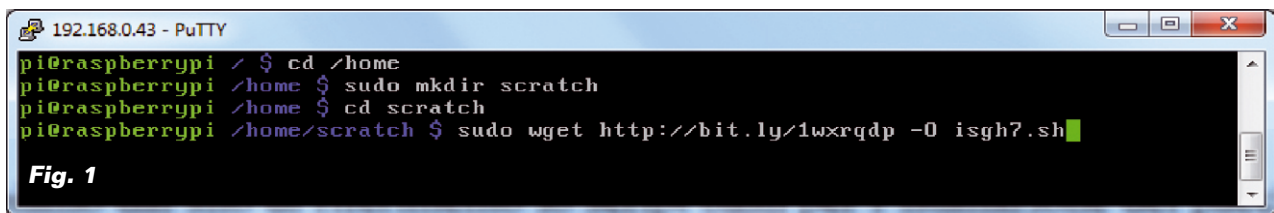
..... di MARCO MAGAGNIN



Sempre più strumenti stanno adottando l'interfaccia grafica di Scratch. Uno di questi, ScratchGPIO7, ci permette di utilizzare il linguaggio grafico di Scratch per gestire il GPIO di Raspberry Pi. Terza parte.

Quando abbiamo deciso di proporre questa serie di articoli, con l'obiettivo di facilitare l'apprendimento della programmazione dei calcolatori elettronici per mezzo di uno strumento didattico dall'aspetto ludico ma dal contenuto professionale, non pensavamo alla reale portata di

questo approccio. Pur essendo disponibile da diverso tempo, questa tipologia di strumenti ha conosciuto negli ultimi mesi una diffusione inaspettata. Oltre alla versione di Scratch tradizionale, quella che vi abbiamo presentato fino ad ora, sono apparse nuove soluzioni che ricalcano la stessa



```
192.168.0.43 - PuTTY
pi@raspberrypi ~ $ cd /home
pi@raspberrypi /home $ sudo mkdir scratch
pi@raspberrypi /home $ cd scratch
pi@raspberrypi /home/scratch $ sudo wget http://bit.ly/1wxrqdp -O isgh7.sh
```

Fig. 1

filosofia, soprattutto per quanto riguarda l'impostazione grafica delle applicazioni. Si va dagli ambienti di sviluppo totalmente utilizzabili all'interno di un browser web, come la versione 2.0 di Scratch o l'ambiente Snap (messo a disposizione dall'Università di Berkeley), a quelli dedicati a specifici hardware, fino alle soluzioni, per così dire, proprietarie.

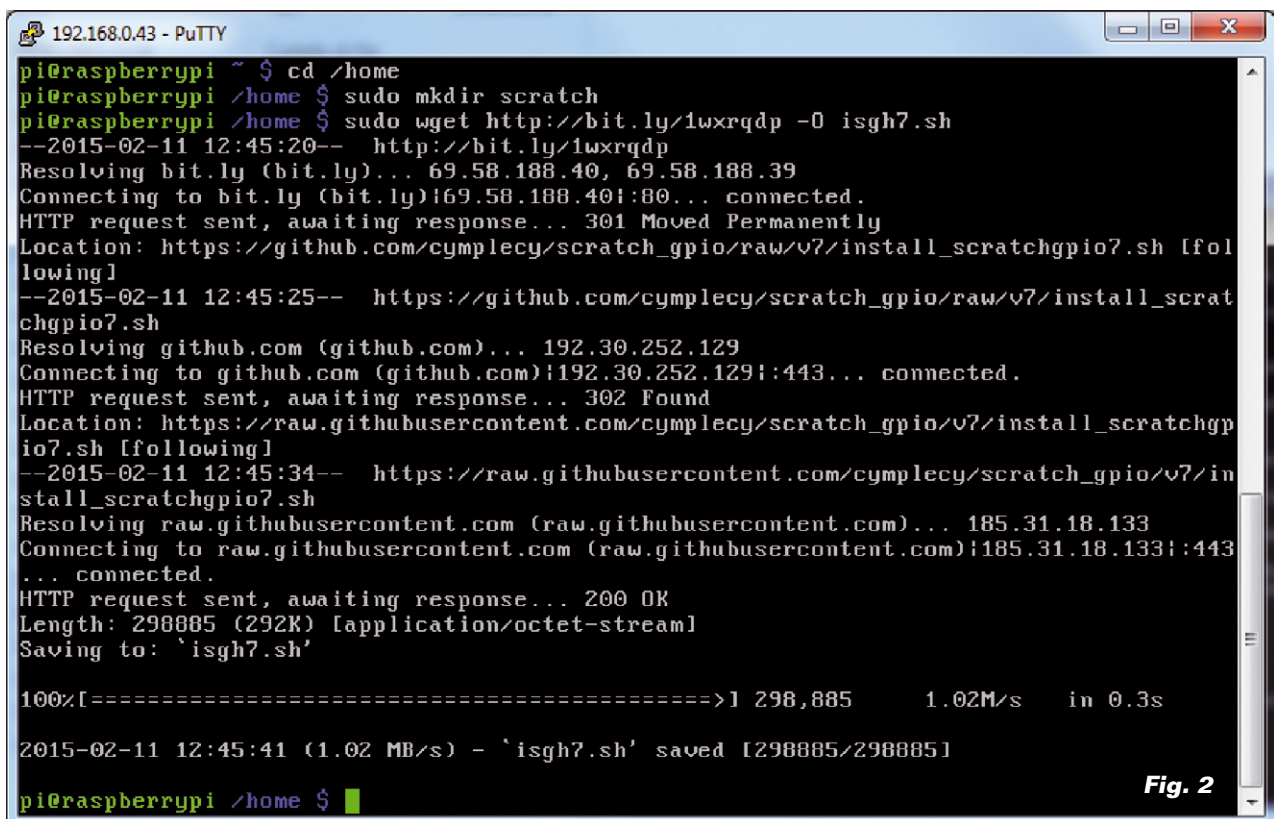
Nel caso delle applicazioni destinate ad uno specifico hardware citiamo ScratchGPIO7 che include le "istruzioni" che permettono di interfacciare il GPIO di Raspberry Pi e che, ovviamente, "gira" su Raspberry Pi; allo stesso modo

S4A che sta per "Scratch For Arduino", permette di scrivere gli sketch per Arduino utilizzando elementi simili a quelli utilizzati dal linguaggio grafico Scratch. Tra le soluzioni "proprietarie" citiamo Wyliodrin, un ambiente che permette di utilizzare i sensori di diversi produttori anche da remoto, in modo da poter realizzare una sorta di sistema di automazione distribuito e gestibile in remoto.

La soluzione prevede un utilizzo gratuito fino ad un numero limitato di schede e di applicazioni, per poi proporre diverse tipologie di abbonamento, in base alle esigenze degli utilizzatori. Ma torniamo alla implemen-

tazione di Scratch che permette di interfacciare il GPIO di Raspberry Pi, ovvero il pacchetto ScratchGPIO7. L'ultimo numero rappresenta la versione del software ed incrementa molto velocemente, probabilmente al tempo dell'uscita dell'articolo potrebbe già essere disponibile una nuova versione.

Procediamo con l'installazione. Partiamo da una immagine fresca del sistema operativo Raspbian su Raspberry Pi. ScratchGPIO7 richiede di essere lanciato dall'interfaccia grafica di Raspberry Pi, quindi dobbiamo "allestire" il nostro Raspberry Pi in "stile" desktop, dotato di monitor HDMI, tastiera e mou-



```
192.168.0.43 - PuTTY
pi@raspberrypi ~ $ cd /home
pi@raspberrypi /home $ sudo mkdir scratch
pi@raspberrypi /home $ sudo wget http://bit.ly/1wxrqdp -O isgh7.sh
--2015-02-11 12:45:20-- http://bit.ly/1wxrqdp
Resolving bit.ly (bit.ly)... 69.58.188.40, 69.58.188.39
Connecting to bit.ly (bit.ly):69.58.188.40:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://github.com/cymplecy/scratch_gpio/raw/v7/install_scratchgpio7.sh [following]
--2015-02-11 12:45:25-- https://github.com/cymplecy/scratch_gpio/raw/v7/install_scratchgpio7.sh
Resolving github.com (github.com)... 192.30.252.129
Connecting to github.com (github.com):192.30.252.129:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://raw.githubusercontent.com/cymplecy/scratch_gpio/v7/install_scratchgpio7.sh [following]
--2015-02-11 12:45:34-- https://raw.githubusercontent.com/cymplecy/scratch_gpio/v7/install_scratchgpio7.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.31.18.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com):185.31.18.133:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 298885 (292K) [application/octet-stream]
Saving to: 'isgh7.sh'

100%[=====>] 298,885      1.02M/s   in 0.3s

2015-02-11 12:45:41 (1.02 MB/s) - 'isgh7.sh' saved [298885/298885]
pi@raspberrypi /home $
```

Fig. 2

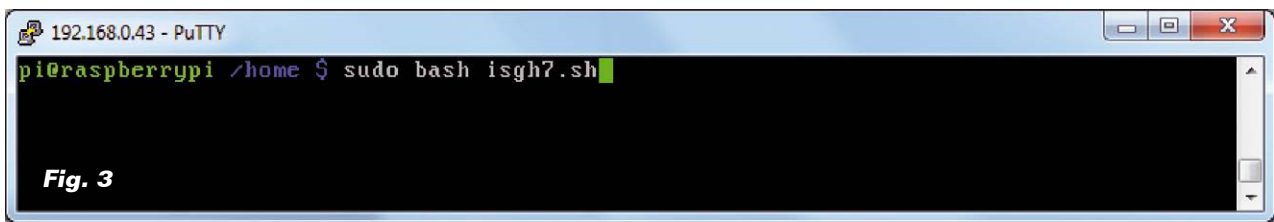


Fig. 3

se. Una possibile alternativa è utilizzare, in ambiente Windows, lo strumento di accesso remoto MobaXterm, come indicato nell'articolo dedicato alla scheda RandA pubblicato nel numero 190 della rivista. Ovviamente, come è di prassi nei sistemi GNU/Linux, Raspberry Pi deve essere collegato alla rete internet, per poter scaricare i pacchetti necessari all'installazione. Apriamo un terminale, in locale od in remoto, ci logghiamo con l'utente "pi" (password "raspberry") ed aggiorniamo il sistema operativo con i comandi consueti, preceduti dal comando "sudo" in quanto non siamo utente "root":

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get dist-upgrade
```

Per l'utilizzo di base di Raspberry Pi e del sistema operativo Raspbian, distribuzione di GNU/Linux basata su Debian, rimaniamo agli argomenti contenuti nel libro "Raspberry Pi – Il mio primo Linux embedded". Terminato l'aggiornamento, che può richiedere un riavvio del sistema con il comando:

```
sudo reboot
```

possiamo procedere all'installazione del pacchetto ScratchGPIO7. Il sito di riferimento, in inglese, per il pacchetto ScratchGPIO7 è "<http://simplepi.net/scratchgpio/scratch-raspberrypi-gpio/>". Per prepararci all'installazione entriamo nella cartella /home e creiamo al suo interno una cartella di nome "scratch", dove

scaricheremo lo script di installazione, come visibile in Fig. 1, con i comandi:

```
cd /home
sudo mkdir scratch
cd scratch
```

Scarichiamo dal sito di riferimento lo script bash per l'installazione del pacchetto con il comando (Fig. 2):

```
sudo wget http://bit.ly/1wxrqdp
-O isgh7.sh
```

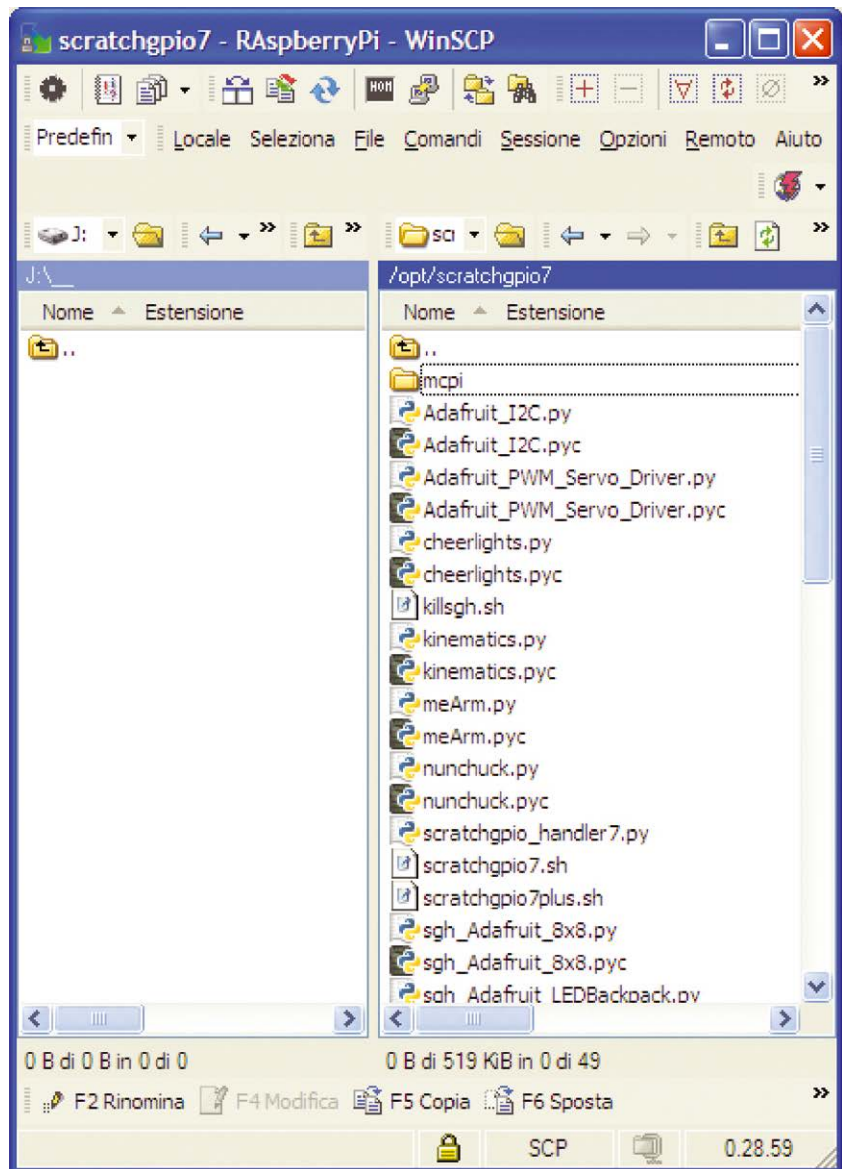
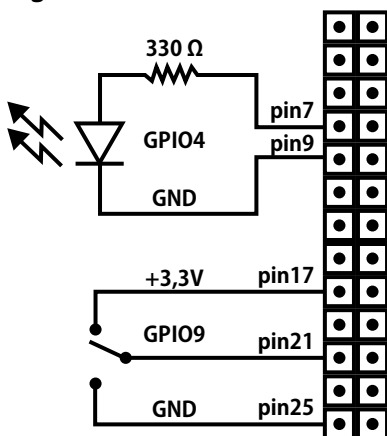


Fig. 4

Fig. 5



Una volta scaricato correttamente lo script di installazione lo possiamo eseguire con il comando (Fig. 3):

```
sudo bash isgh7.sh
```

Una volta terminato il processo di installazione troveremo tutti i file dell'applicazione installati nella cartella `/opt/scratchgpio7` come visibile in Fig. 4, dove possiamo notare che l'estensione di Scratch per Raspberry Pi è costituita da una gerarchia di programmi scritti in linguaggio Python. Sul desktop di Raspberry Pi troveremo due icone aggiuntive. Un'icona di nome ScratchGPIO7 che permette di fare esperimenti con semplici circuiti ed un'icona di nome ScratchGPIO7 Plus che contiene le configurazioni predefinite per poter interfacciare numerose schede add-on disponibili in commercio. Visto che parliamo di icone e di entrate nel menu di "start", ne approfitteremo per approfondire uno dei principali cambiamenti avvenuti nell'interfaccia grafica Lxde (Lightweight X11 Desktop Environment) di Raspberry Pi, con il rilascio della versione di Raspian datata 22-12-2014. Il rilascio successivo, 31-1-2015 è finalizzato al supporto della nuova versione 2 di Raspberry Pi, con il nuovo processore Cortex A7 quadricore ed un GB di memoria RAM, che

la rende del tutto equivalente ad un piccolo PC desktop. A questa nuova versione di Raspberry Pi abbiamo dedicato un articolo di presentazione in questo stesso numero della rivista. Per inciso, tenete presente che al momento ScratchGPIO7 non "gira" sul nuovo Raspberry Pi 2 a causa della necessità di aggiornare alcune librerie dal quale dipende. Al momento dell'uscita della rivista il problema potrebbe essere già risolto. Tornando all'interfaccia grafica, la nuova versione prevede la barra delle applicazioni in alto sullo schermo, con il menu start in alto a sinistra. Questa soluzione è adottata da numerose distribuzioni GNU/Linux e sembra essere ispirata dalla somiglianza con il modo di utilizzare i libri tradizionali, che si scorrono dall'alto in basso e da sinistra a destra, almeno nella cultura occidentale, ed anche per uniformarsi alle applicazioni mobili, che hanno la barra delle notifiche in alto. La "scrivania" viene proposta vuota, con il solo "cestino" visibile in alto a sinistra. Tutte le altre icone possono essere posizionate a piacere sulla scrivania a partire dal menu start. Una volta individuata un'applicazione, è possibile portarne l'icona sulla scrivania, semplicemente, cliccandovi sopra col tasto

destro del mouse e selezionando l'opzione "Add to desktop". Per spiegare il funzionamento delle nuove istruzioni presenti nel pacchetto ScratchGPIO7 usiamo il "solito" metodo del LED e dell'interruttore. Dunque realizziamo il piccolo circuito, lo schema del quale è visibile in Fig. 5.

Noi lo abbiamo realizzato semplicemente saldando la resistenza direttamente sul terminale positivo del LED e "allungando" i due reofori con due spezzoni di cavetto terminati con un connettore femmina ciascuno. Lo stesso per l'interruttore, nel nostro caso abbiamo trovato un modello con i terminali distanti 5 mm tra loro. Abbiamo saldato sui terminali uno spezzone di strip femmina a cinque poli, in modo da lasciare liberi i poli pari. Il componente così realizzato trova perfetta corrispondenza con i pin +3.3V, GPIO9, e GND di Raspberry Pi, corrispondenti rispettivamente ai pin fisici 17, 21 e 25. Collegheremo il terminale di massa del LED al pin fisico 9 ed il positivo al pin fisico 7, che corrisponde all'uscita GPIO4, anch'essi "vicini" tra di loro, come si può notare consultando lo schema del GPIO in Fig. 6. La foto dell'insieme è visibile in Fig. 7.

Per collegare i due componenti così realizzati è raccomanda-

			P1			
<50mA	3V3		1	2	5V	
BCM GPIO00/02	SDA0/1	8	3	4	5V	
BCM GPIO01/03	SCL0/1	9	5	6	GND	
BCM GPIO04		7	7	8	15 TX	BCM GPIO14
	GND		9	10	16 RX	BCM GPIO15
BCM GPIO17		0	11	12	1 PWM0	BCM GPIO18
BCM GPIO21/27		2	13	14	GND	
BCM GPIO22		3	15	16	4	BCM GPIO23
<50mA	3v3		17	18	5	BCM GPIO24
BCM GPIO10	SPIMOSI	12	19	20	GND	
BCM GPIO9	SPIMOSO	13	21	22	6	BCM GPIO25
BCM GPIO11	SPI SCLK	14	23	24	10 SPI CE0 N	BCM GPIO08
	GND		25	26	11 SPI CE1 N	BCM GPIO07

Fig. 6

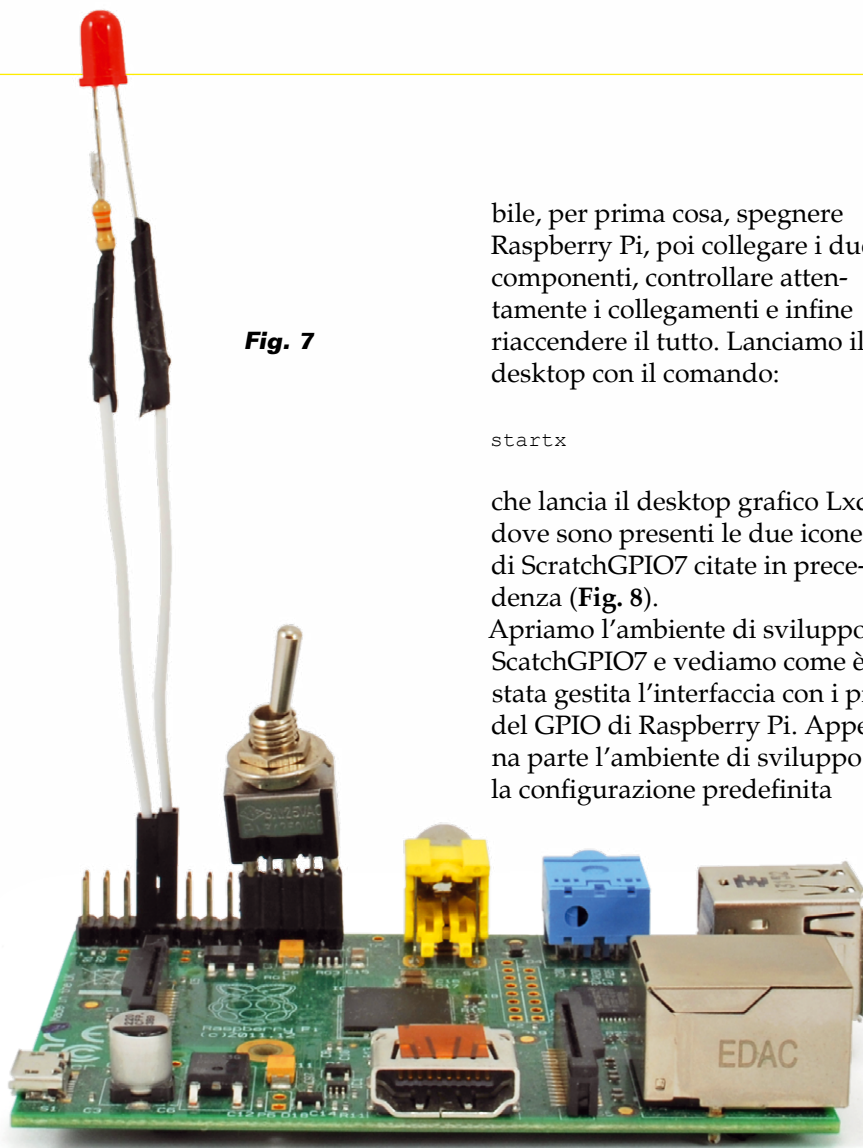


Fig. 7

bile, per prima cosa, spegnere Raspberry Pi, poi collegare i due componenti, controllare attentamente i collegamenti e infine riaccendere il tutto. Lanciamo il desktop con il comando:

```
startx
```

che lancia il desktop grafico Lxde dove sono presenti le due icone di ScratchGPIO7 citate in precedenza (**Fig. 8**).

Apriamo l'ambiente di sviluppo ScratchGPIO7 e vediamo come è stata gestita l'interfaccia con i pin del GPIO di Raspberry Pi. Appena parte l'ambiente di sviluppo la configurazione predefinita

prevede alcuni pin configurati come input ed altri come output. In particolare sono impostati come input i pin 22, 7, 3, 5, 24, 26, 19, 21, 23, 8 e 10 e come output i pin 11, 12, 13, 15, 16 e 18. Tenete presente che la numerazione utilizzata in ScratchGPIO7 fa corrispondere i pin alla loro posizione fisica sul connettore GPIO e non al nome di "sistema" del pin. Potete personalizzare la lingua nella quale sono presentate le istruzioni di ScratchGPIO7 cliccando sull'icona in alto a sinistra, a forma di piccolo "mondo" e scegliendo la lingua desiderata, come visibile in **Fig. 9**. Una delle istruzioni che permettono di interagire con i pin è la "invia a tutti" ("broadcast") visibile in **Fig. 10**. "Invia a tutti", in Scratch, è l'istruzione utilizzata per inviare messaggi ad altri script o, come in questo caso, a periferiche di sistema. Nello spazio "messaggio" dell'istruzione possiamo impostare lo stato dei pin inserendo "messaggi" costruiti con una sintassi particolare. Per portare

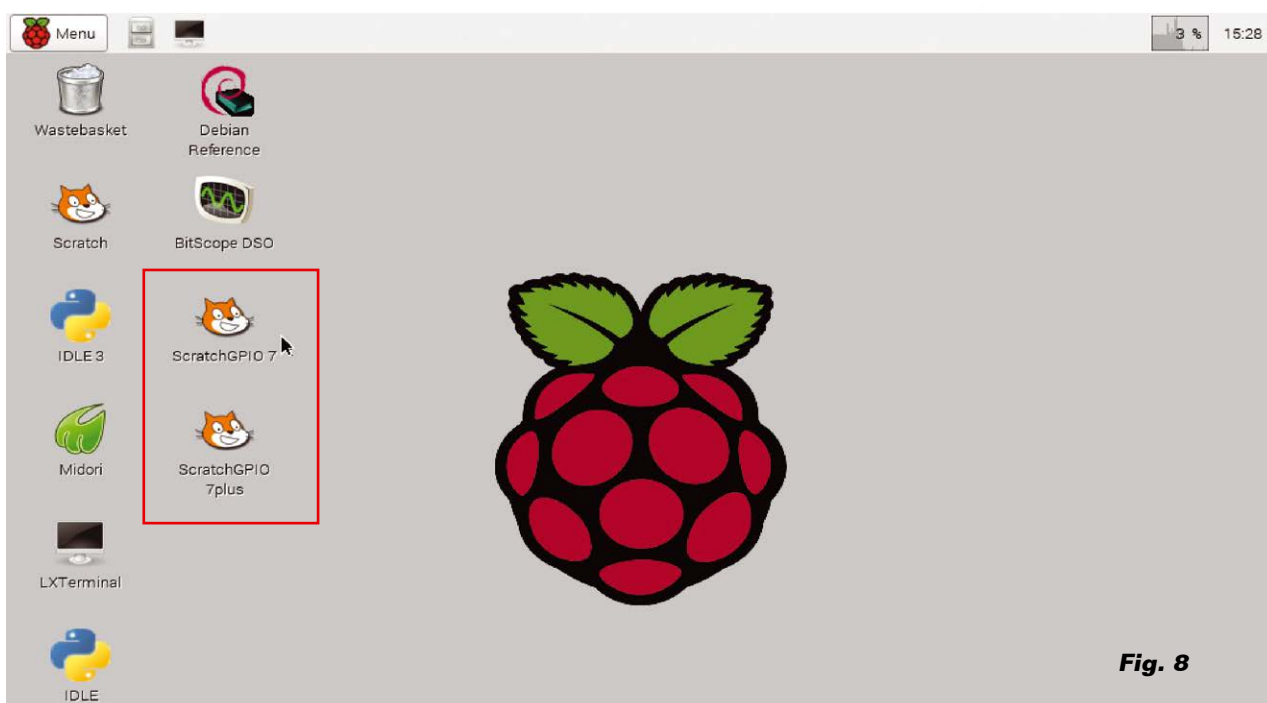


Fig. 8



Fig. 9

desiderata. È possibile anche alzare o abbassare i livelli di tutti i pin contemporaneamente con i messaggi “allon” e “alloff”. Nello stesso messaggio possono essere concatenati più “ordini” come “pin7on pin5off”.

In questo modo possiamo scrivere il nostro primo programma Scratch in grado di spegnere ed accendere un LED, il “listato” del quale è visibile in **Fig. 11**.

Sono validi anche i messaggi “pin7on” e “pin7off”, come visibile nel “listato” di **Fig. 12**.

Con questo metodo potremmo riprodurre realmente il semaforo presentato nel primo articolo di questa serie, utilizzando tre LED, uno rosso, uno giallo ed uno verde. Piccola digressione. Un altro modo di interfacciare i pin è di creare delle variabili, nominarle in modo standard con i nomi dei pin come “pin26” o “pin19”. In questo modo avremo a disposizione (**Fig. 13**):

- una variabile utilizzabile nel campo “variabile” delle istruzioni che lo prevedono;
- le istruzioni predefinite per l'impostazione dei valori da assegnare alle variabili;
- la possibilità di rendere visibile la variabile sul “palcoscenico”.

Un piccolo accorgimento ma importante per evitare di perdere tempo a capire le cause di quello che sembra un malfunzionamento. Scratch, almeno fino alla versione attuale, “ricorda” lo stato di una variabile ed imposta un nuovo valore solo se questo è diverso da quello corrente. Quando si “clicca” sulla bandierina verde, Scratch invia solo i valori delle variabili che sono stati modificati rispetto alle esecuzioni precedenti, in pratica, le variabili che non hanno cambiato valore non vengono inviate e lo stato dei pin

può apparire incongruente con ciò che ci aspettiamo. Un modo per “girare attorno” a questo problema è quello di impostare le variabili all'inizio del programma, prima del primo ciclo “loop” a valori simbolici, anche di tipo non corrispondente ai dati attesi dalla variabile,



Fig. 10



Fig. 11



Fig. 12



Fig. 13

il pin 7 a livello alto utilizziamo il messaggio “pin7high” mentre per portarlo a livello basso utilizziamo il messaggio “pin7low”. Non esiste controllo formale per i messaggi, quindi è necessario prestare la massima attenzione alla loro scrittura. In caso di errore, durante la composizione dello script, appare un messaggio di errore generico e, una volta lanciata l'esecuzione, il contorno dello script appare rosso invece che bianco. In questi casi è molto utile la funzione di “aiuto”, che si attiva “cliccando” sull'istruzione

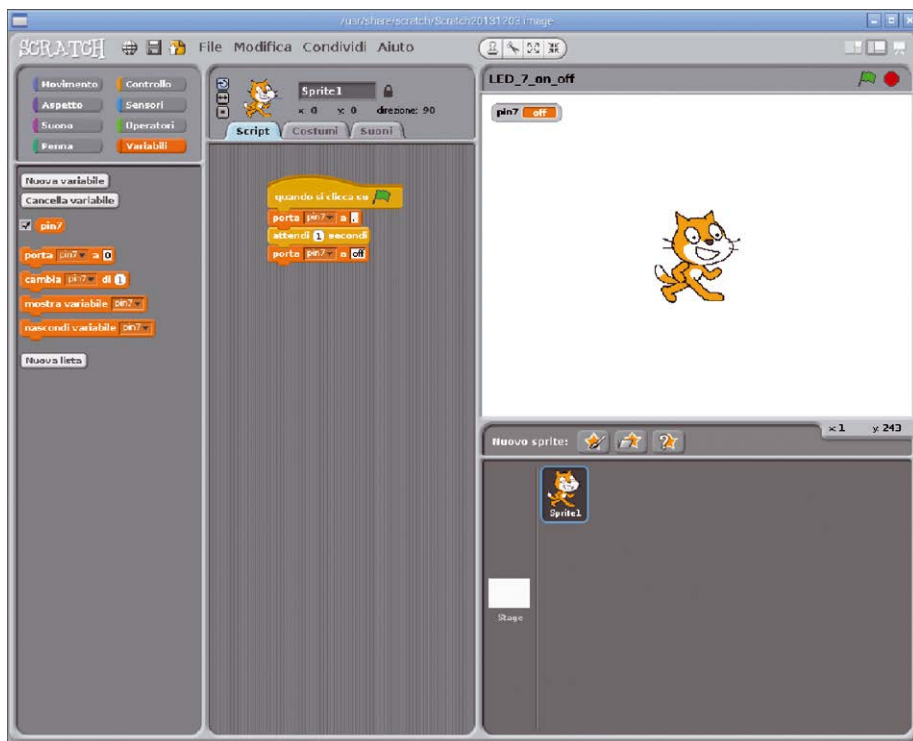


Fig. 14

dell'istruzione è di "intercettare" il cambiamento di stato, "0" quando il deviatore è collegato a massa e "1" quando è collegato alla tensione di +3,3V.

Se per le vostre applicazioni avete bisogno di un numero di pin di output superiore a quello messo a disposizione nella configurazione predefinita, è sufficiente per esempio, eseguire un'istruzione "invia a tutti" ("broadcast") sul pin 10, impostato in modo predefinito come "input", per esempio "invia a tutti pin10on", per trasformarlo automaticamente in un pin di output. Il pacchetto ScratchGPIO7 comprende la possibilità di riconoscere e gestire un

per esempio con ".". Nel ciclo normale, poi, impostiamo i valori desiderati, come nell'esempio di **Fig. 14**.

Tornando al programma di accensione e spegnimento del LED, clicchiamo sulla bandierina per far partire l'esecuzione del programma. Inizialmente abbiamo collegato il terminale centrale del nostro deviatore al pin 21 e i terminali estremi rispettivamente a massa ed al pin +3,3V di Raspberry Pi (pin 25 e 17). Con questa configurazione possiamo realizzare un piccolo programma per verificare in pratica l'utilizzo delle istruzioni di gestione dei pin in ingresso. Andiamo nella sezione "Sensore" ed ispezioniamo le opzioni disponibili nell'istruzione "valore del sensore". Si può notare che sono indicati i pin 22, 7, 3, 5, 24, 26, 19, 21, 23, 8 e 10, ovvero quelli predefiniti come pin di ingresso, come mostrato in **Fig. 15**.

Creiamo un programma come quello visibile in **Fig. 17** dove l'istruzione centrale è quella indicata con la freccia. Scopo

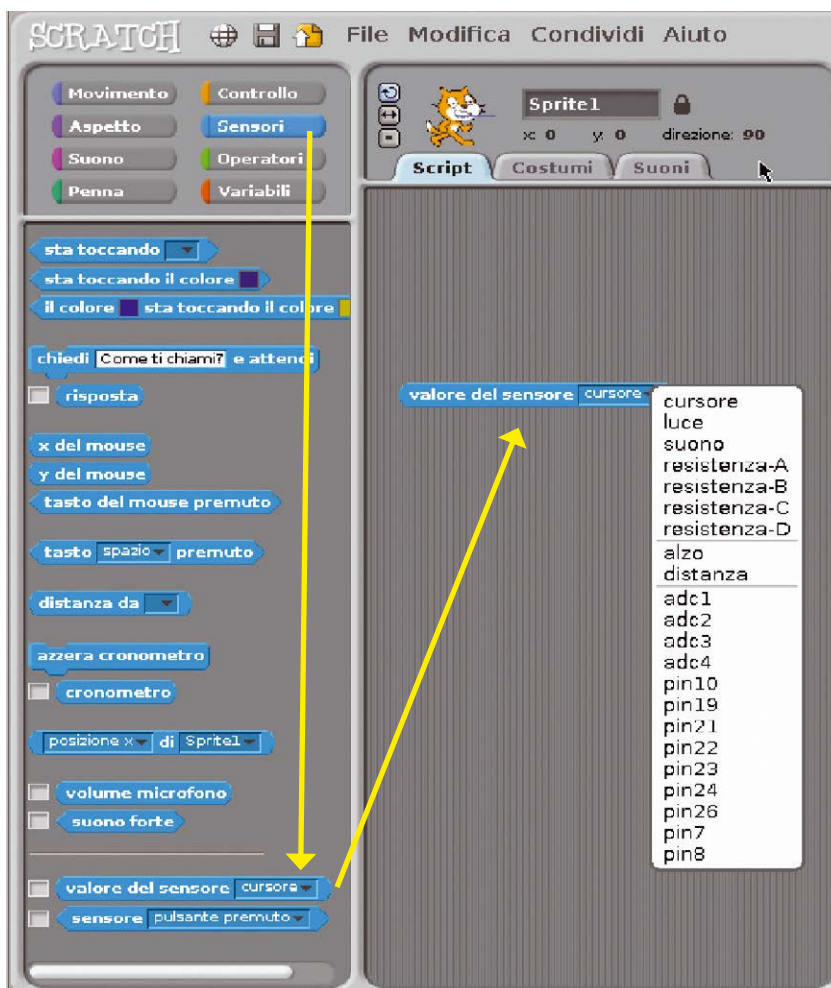
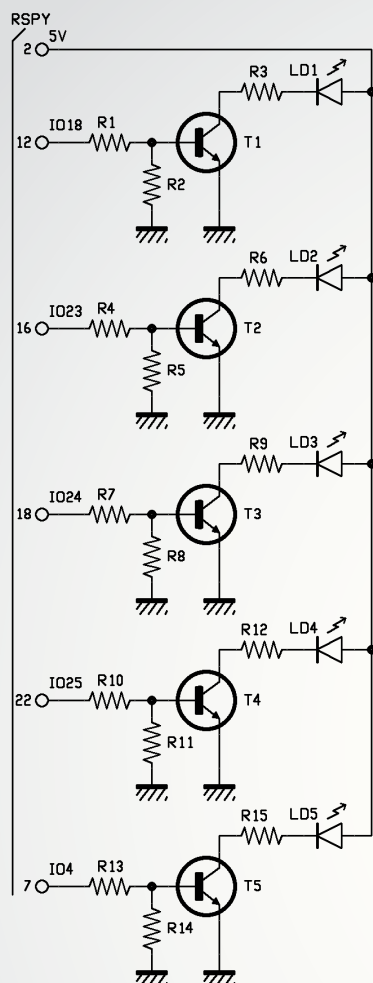
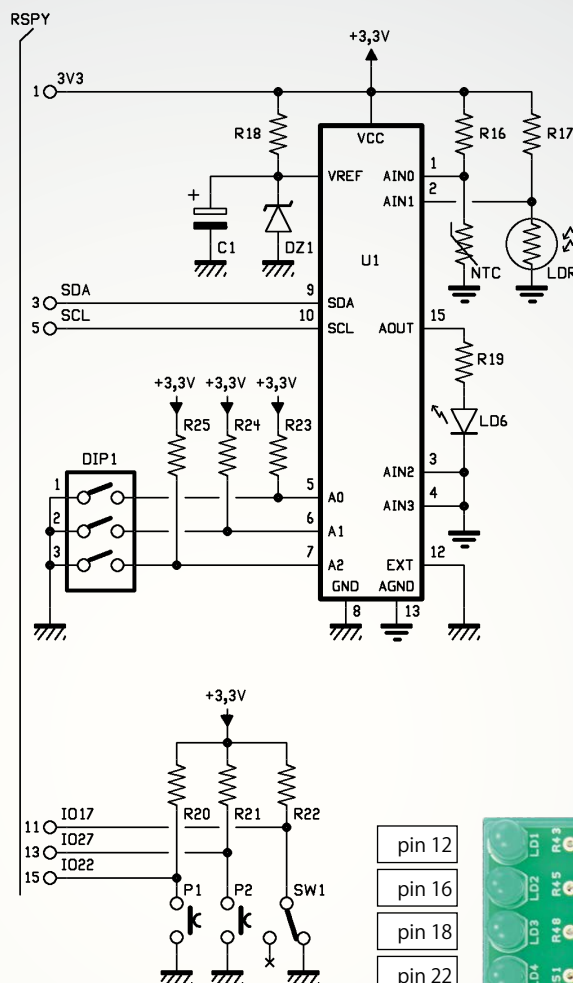


Fig. 15



Elenco Componenti:

R1, R4, R7, R10, R13, R20÷R25: 4,7 kohm
 R2, R5, R8, R11, R14: 10 kohm
 R3, R6, R9, R12, R15, R19: 330 ohm
 R16, R17: 15 kohm
 R18: 100 ohm
 C1: 4,7 μ F 100 VL elettrolitico
 DZ1: Zener 3,3V 400mW
 T1÷T5: BC547
 LD1÷LD5: LED 5 mm verde
 LD6: LED 5 mm rosso
 U1: PCF8591



P1, P2: Microswitch
 SW1: Deviatore slitta verticale
 DIP1: Dip-Switch 3 vie
 NTC: NTC 10 kohm
 LDR: Fotoresistenza 2-20 kohm

Varie:
 - Zoccolo 8+8
 - Strip Femmina 13x2 vie per Raspberry Pi
 - Distanziale M/F 16mm
 - Vite 8 mm 3 MA (2 pz.)
 - Circuito stampato

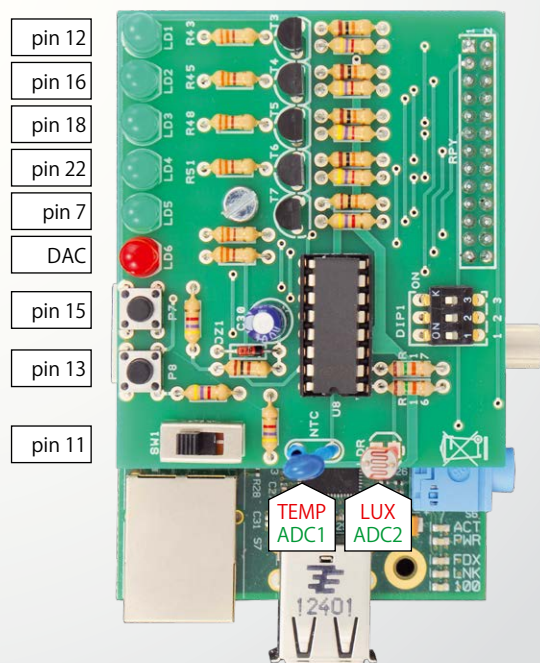


Fig. 16

certo numero di shield disponibili sul mercato. Una di queste permette a Raspberry Pi di acquisire dati analogici interfacciando l'integrato PCF8591 per mezzo del bus I2C, esattamente come avviene per la scheda FT1060, di

corredo al libro "Raspberry Pi – Il mio primo Linux embedded". L'integrato PCF8591, come visibile nello schema di Fig. 16 dispone di quattro ingressi analogico digitali a otto bit ed una uscita DAC, sempre ad otto bit.

All'ingresso ADC1 dello shield è collegato un termistore (NTC nello schema) che possiamo utilizzare per misurare la temperatura ambiente. All'ingresso ADC2 è collegata una fotoresistenza (LDR nello schema) che possiamo



Fig. 17

utilizzare per misurare la luminosità ambiente. Infine all'uscita DAC è collegato un LED rosso (LD6) che potremo far illuminare con un'intensità proporzionale al valore digitale impostato sull'uscita DAC stessa. Nel libro citato potete approfondire le caratteristiche dell'integrato PCF8591. Per provare questa funzionalità, spegniamo Raspberry Pi con il comando:

```
sudo shutdown -h now
```

Smontiamo il LED ed il deviatore e posizioniamo la scheda FT1060 sul connettore GPIO di Raspberry Pi, facendo bene

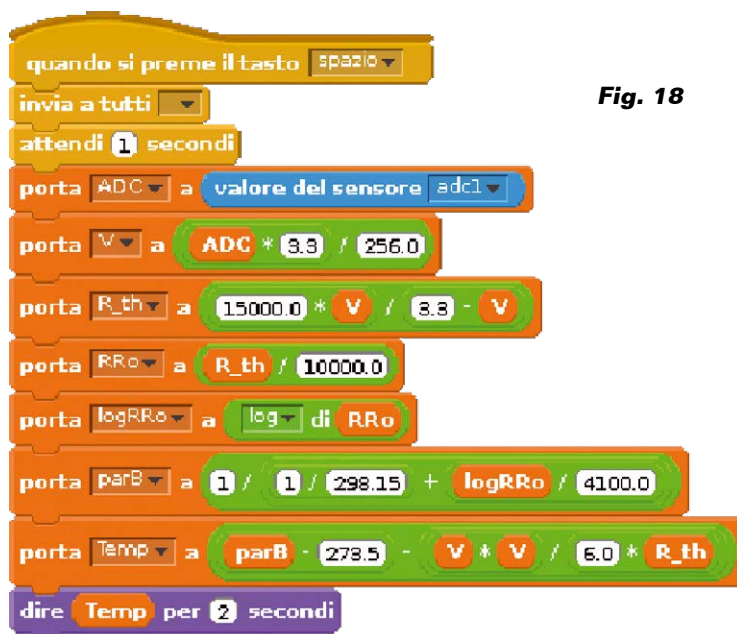


Fig. 18

attenzione a far corrispondere i pin dello shield con quelli del microcomputer. Dopo aver controllato tutto ancora una volta riaccendiamo Raspberry Pi ed attendiamo la

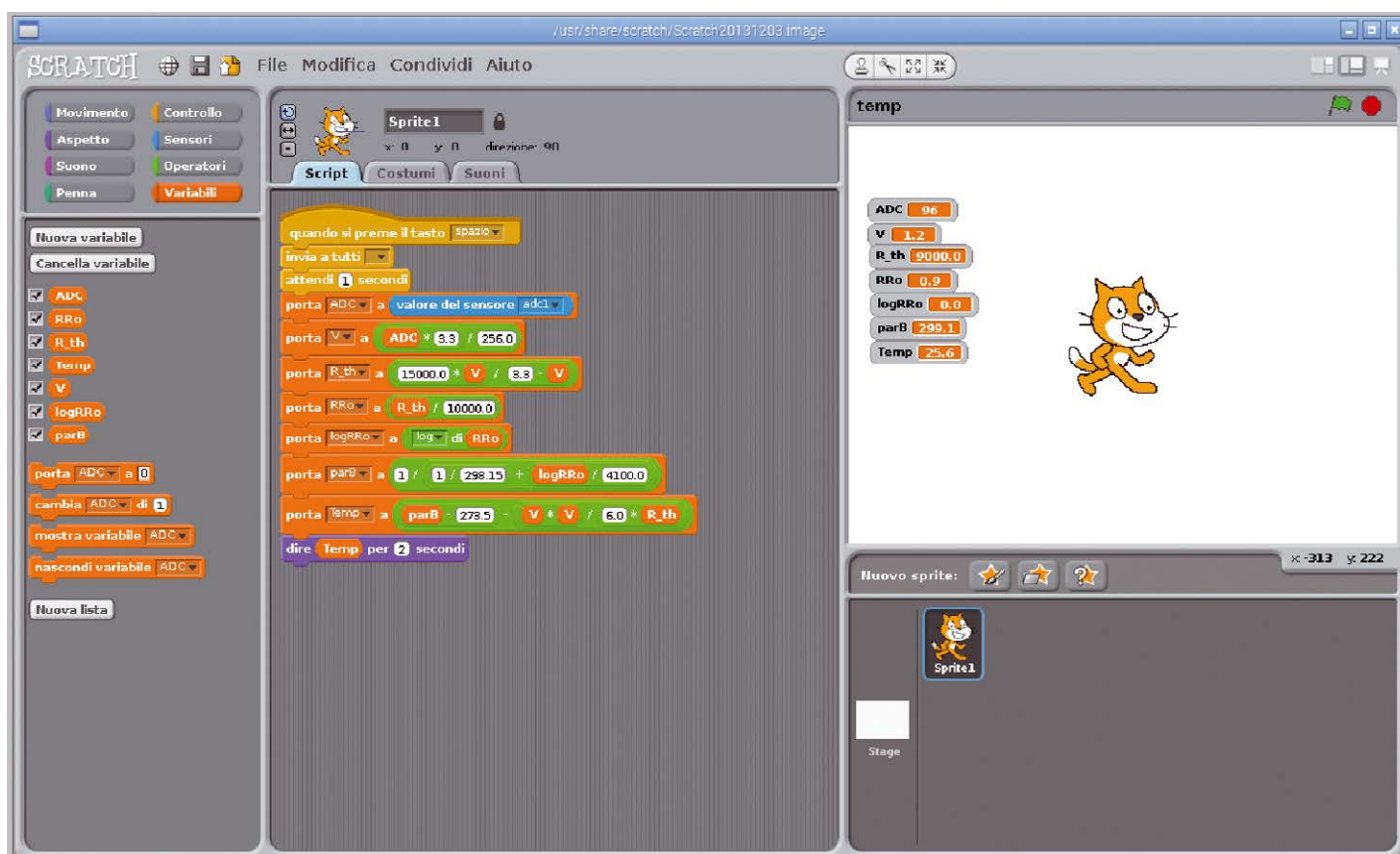
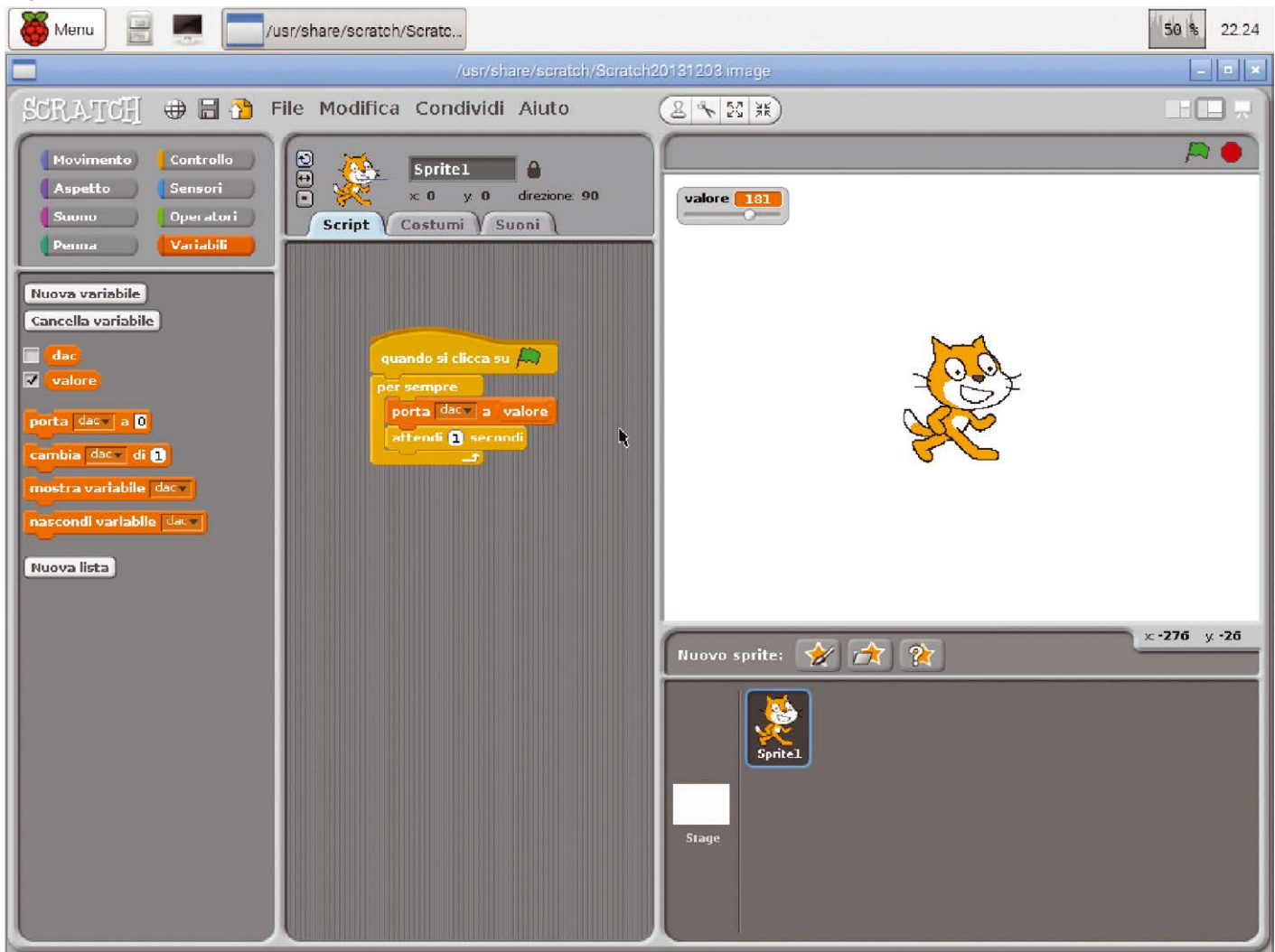


Fig. 19

Fig. 20

fine del processo di boot. Ci connettiamo con l'utente generico "pi" e password "raspberrypi", lanciamo l'interfaccia grafica con il comando "startx" ed il programma ScratchGPIO7. Prima però verifichiamo se abbiamo attivato il driver di gestione del bus I2C.

**Fig. 21**

Se così non fosse, e stiamo utilizzando le versioni di Raspbian più recenti, apriamo una finestra di terminale e lanciamo il configuratore raspi-config. Selezioniamo la voce di menu "I2C" e nelle pagine di configurazione successive rispondiamo "yes" a tutti i passaggi. Se ci viene chiesto di riavviare il microcomputer rispondiamo affermativamente e attendiamo il riavvio. Nel nostro caso è stato necessario attivare il bus I2C anche nel modo tradizionale, commentando l'entrata "# blacklist i2c-bcm2708" nel file /etc/modprobe.d/raspi-blacklist.conf, dare il comando modprobe i2c-dev e installare il pacchetto

to "i2c-tools". Altro riavvio per poi riportarci nell'applicazione ScratchGPIO7. Ora proviamo a leggere il valore del sensore di temperatura collegato all'ingresso ADC1 dell'integrato PCF8591. Per eseguire la lettura utilizziamo l'istruzione "invia a tutti ADC1". Per acquisire il risultato utilizziamo la "variabile" "valore del sensore adc1", come potete vedere nel programma di Fig. 18 e nella vista d'insieme di Fig. 19, dove abbiamo anche riprodotto in Scratch le operazioni matematiche per calcolare la temperatura a partire dal valore ADC, con il metodo del "parametro B" descritto numerose volte nella

rivista e nel libro citato all'inizio. Per impostare l'uscita DAC con un valore digitale da convertire in una tensione analogica proporzionale al valore impostato si utilizza l'istruzione "porta dac a <valore>" dove <valore> è un numero compreso tra "0" e "255". Il programma visibile in **Fig. 21** permette di impostare la luminosità del LED rosso proporzionalmente alla posizione del cursore (**Fig. 20**).

Per i nostri esperimenti, della scheda FT1060, con riferimento allo schema di Fig. 16, possiamo utilizzare, oltre ai convertitori ADC e DAC, anche i seguenti pin:

- Come Uscite possiamo utilizzare i pin 12 (GPIO18), pin 16 (GPIO23), pin 18 (GPIO24), pin 22 (GPIO25) e pin 7 (GPIO4). Ciascuna di queste uscite è provvista di un LED pilotato da un transistor configurato ad emettitore comune per evitare di sovraccaricare i pin del GPIO.
- In ingresso abbiamo a disposizione due pulsanti collegati ai pin 13 (GPIO27) e pin 15 (GPIO22). Un deviatore è collegato al pin 11 (GPIO17). Prima di utilizzare questi pin ricordiamo di impostarli come pin di ingresso con istruzioni

del tipo "invia a tutti Config11In".

In ScratchGPIO7 l'istruzione "invert" permette di invertire la logica di comportamento di un pin, per esempio riconoscere il pin come On quando il livello di tensione è basso e viceversa. L'istruzione "invia a tutti Config11In" permette di impostare in modo esplicito il pin 11 da output ad input. L'istruzione "invia a tutti GetIP" permette di "leggere" l'indirizzo IP di Raspberry Pi, che può poi essere evidenziato con l'istruzione "dire" e la variabile "valore del sensore ipaddress" come visibile in **Fig. 22**. ■

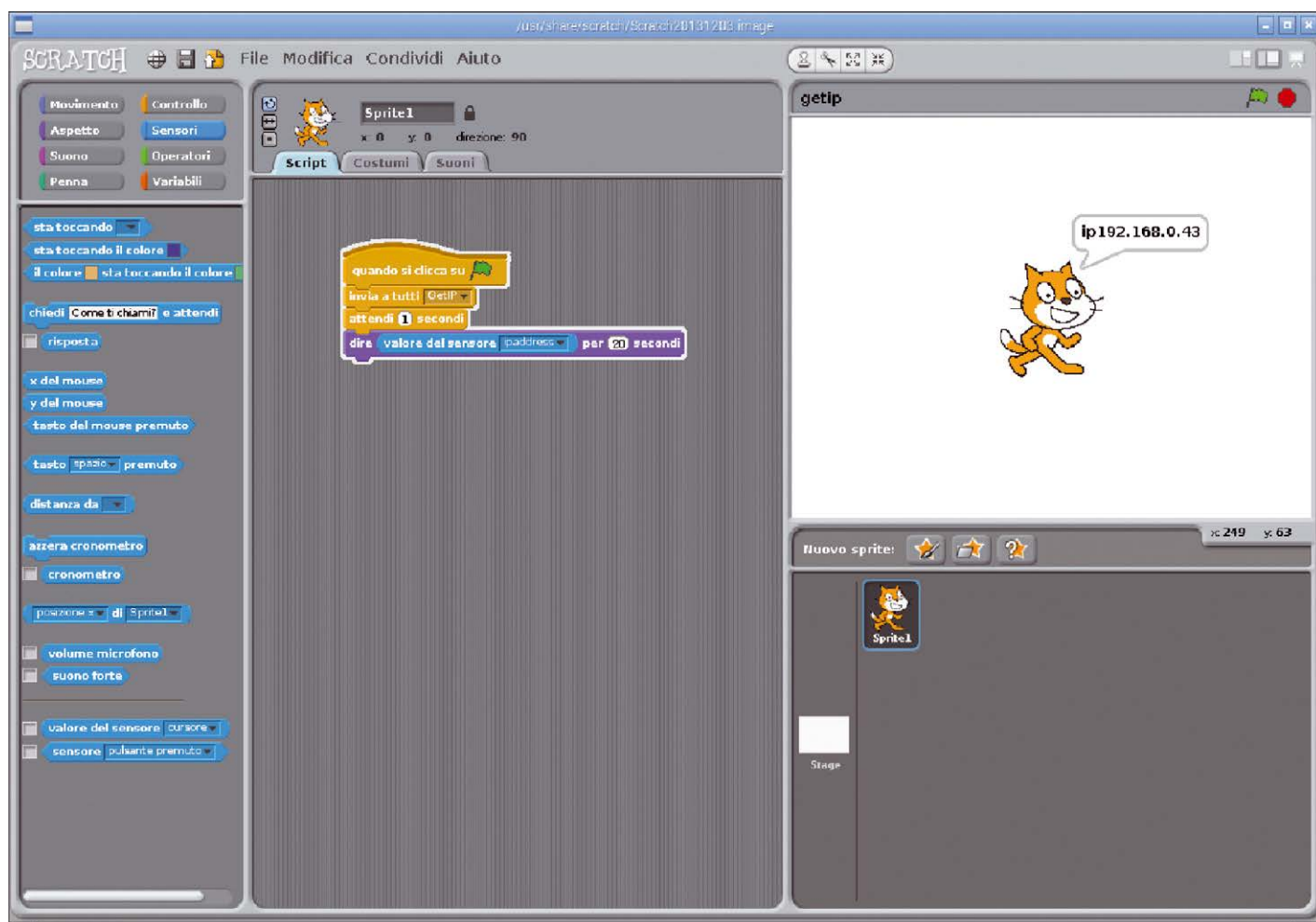


Fig. 22

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

Elettronica In

www.elettronica.in.it

oltre l'elettronica

UTILIZZIAMO RANDA

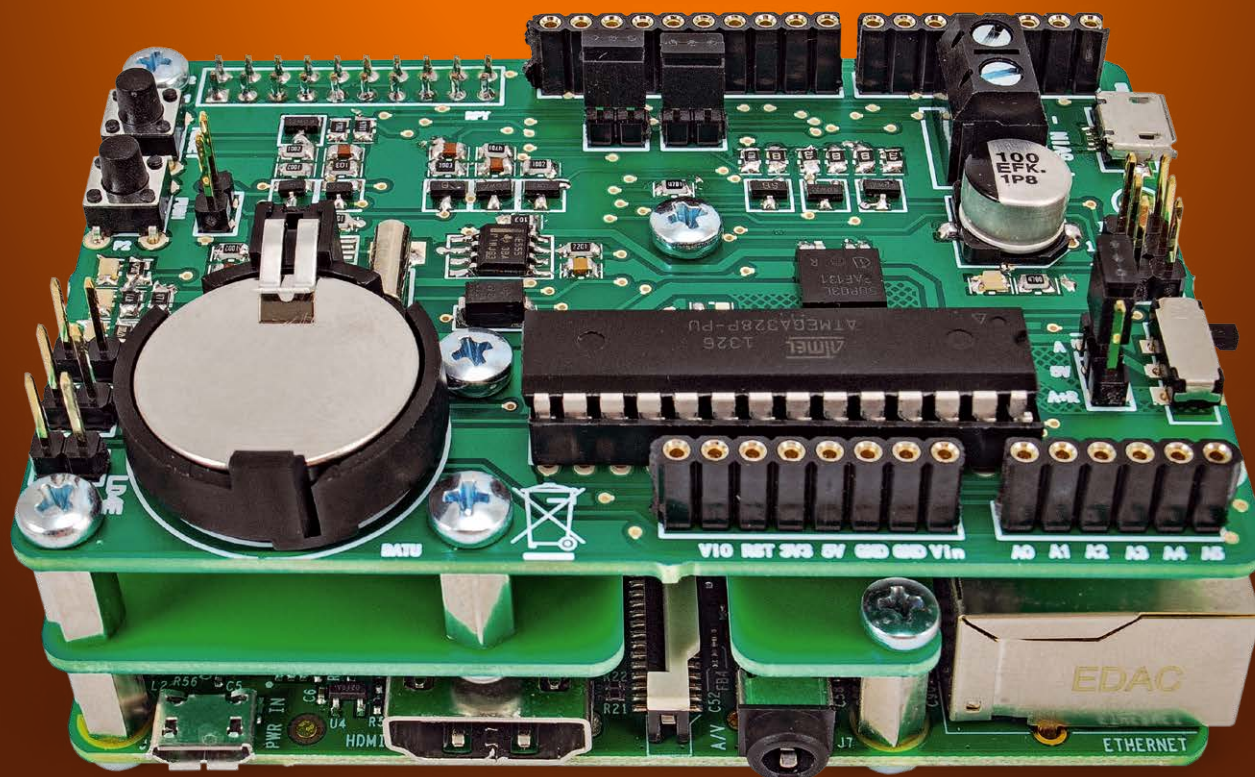
di DANIELE DENARO

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

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

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

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



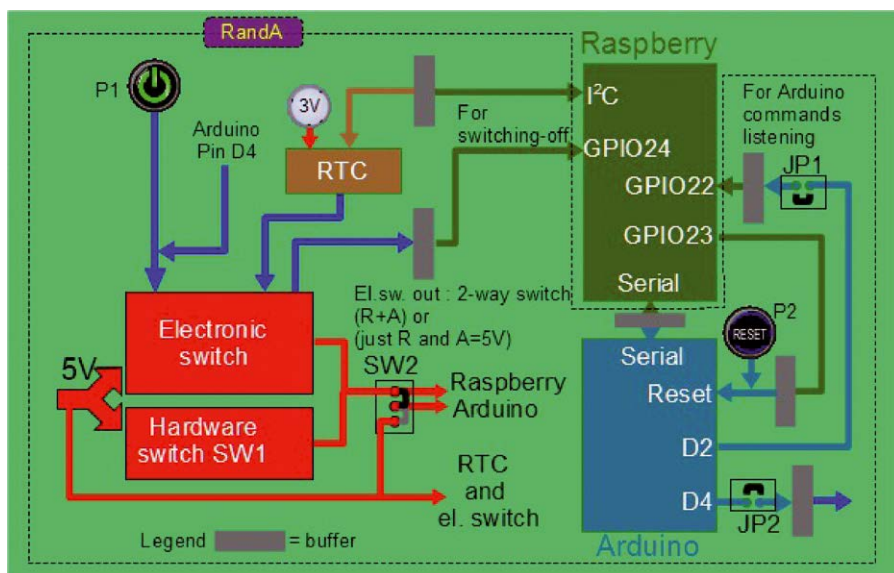


Fig. 1a - Schema a blocchi di RandA.

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

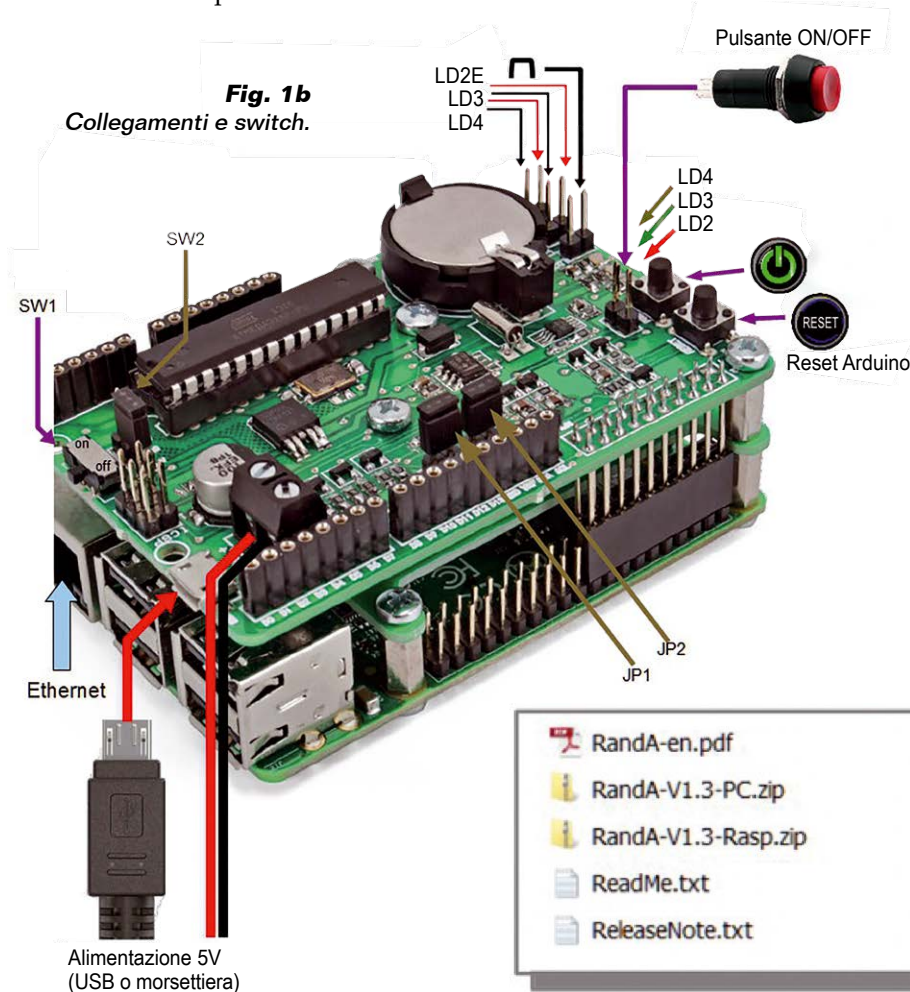


Fig. 1b

Collegamenti e switch.

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

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

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

A questo punto dovete operare su Raspberry Pi. Poiché le



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

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

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

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

```
./RandAinstall.sh
```

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

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

Però, dopo l'installazione di Ar-

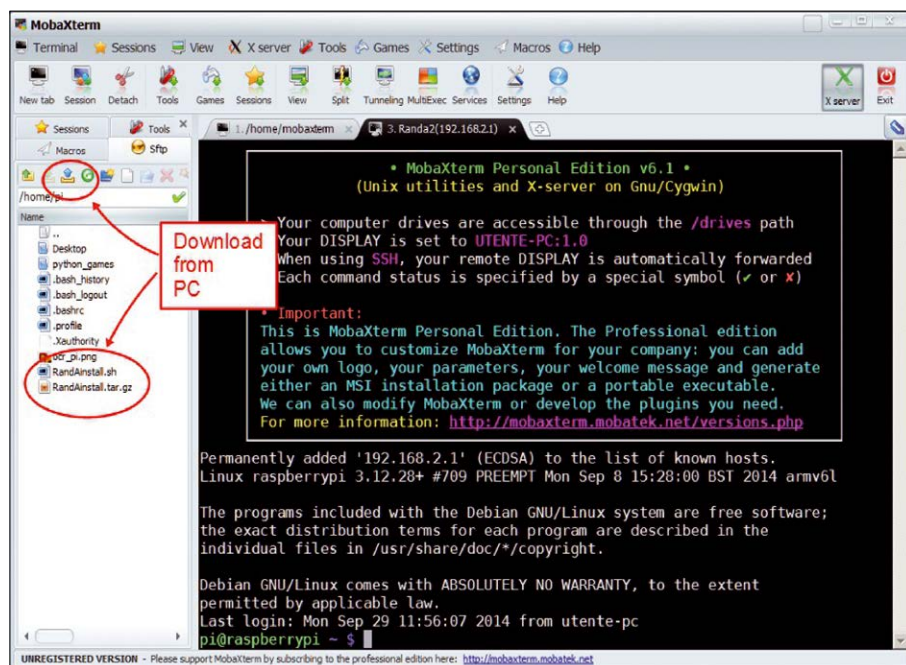


Fig. 3 - Console MobaXterm.

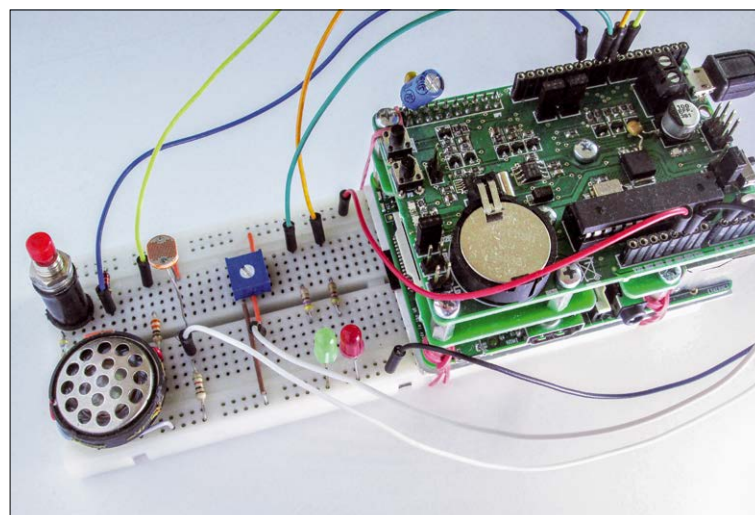
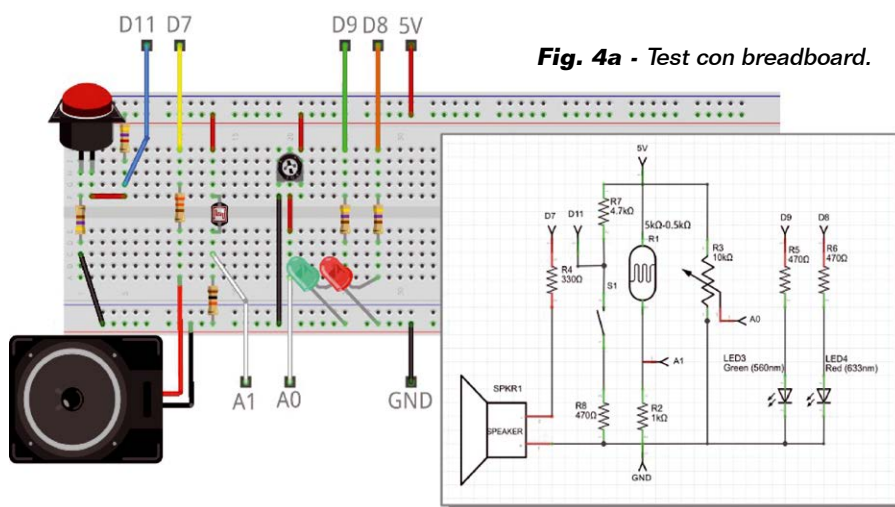


Fig. 4b
Foto
breadboard.

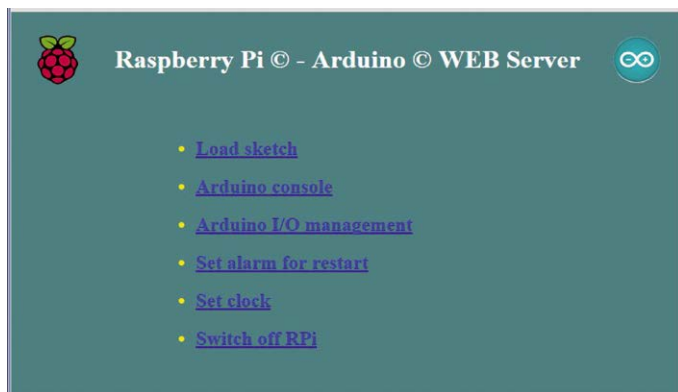


Fig. 5 - Menu generale di Arduino Web Server.

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

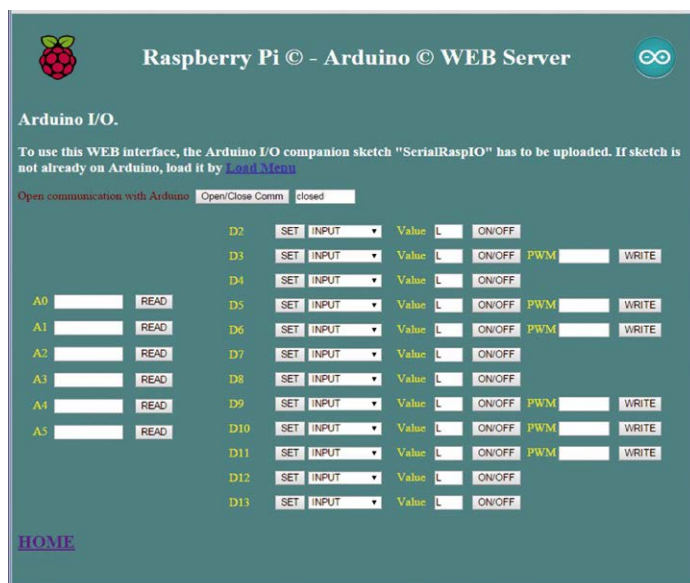
TEST DEL SISTEMA

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

```
"sudo reboot"
```

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

Fig. 6
Schermata I/O.



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

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

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

```
ArduIO -wrd 8 0
```

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

```
ArduIO -wra 9 50
```

Leggiamo il valore della fotore-

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

```
ArduIO -rda 1
```

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

```
ArduIO -pou 7 600
```

Per bloccare l'onda, impartiamo:

```
ArduIO -pou 7 0
```

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

```
ArduIO -pin 11 0
```

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

```
ArduIO -pui 11 1
```

Se usate il pulsante, ricordate che

Fig. 7 - Load sketch.

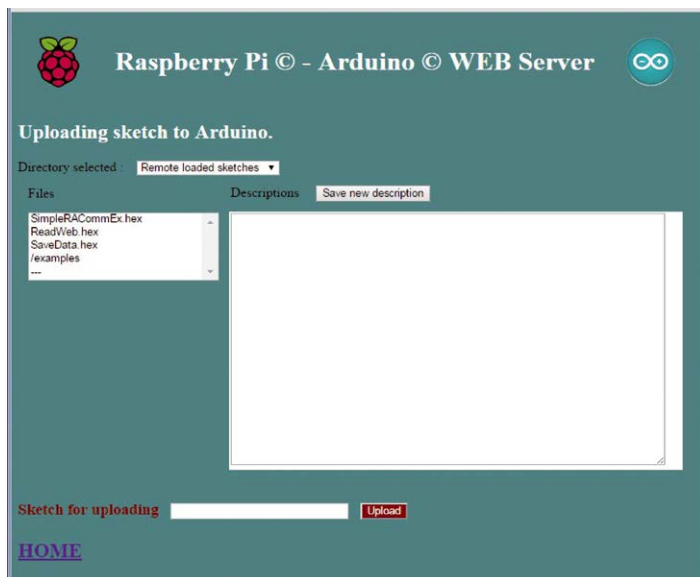
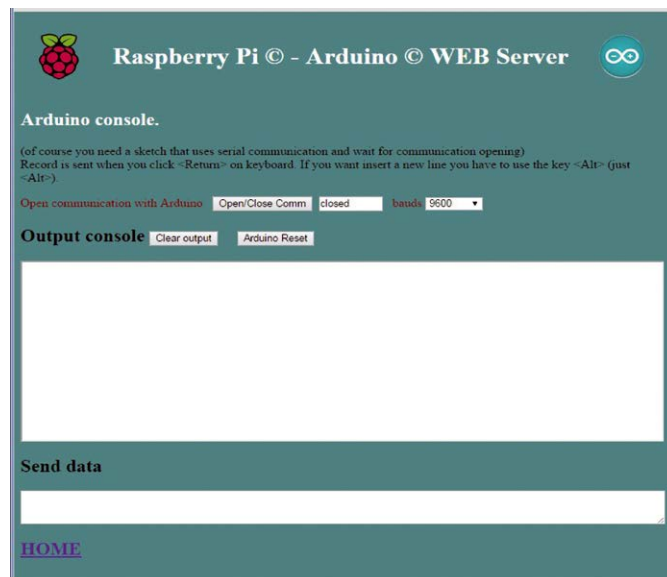


Fig. 8 - Arduino console.



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

```
ArduIO -rdd 11
```

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

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

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

WEB SERVER

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

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

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

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

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

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

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

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

Randa DIVENTA UN ARDUINO POTENZIATO

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

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

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

Per fare una prova potete utilizzare il classico Blink oppure un

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

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

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

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

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

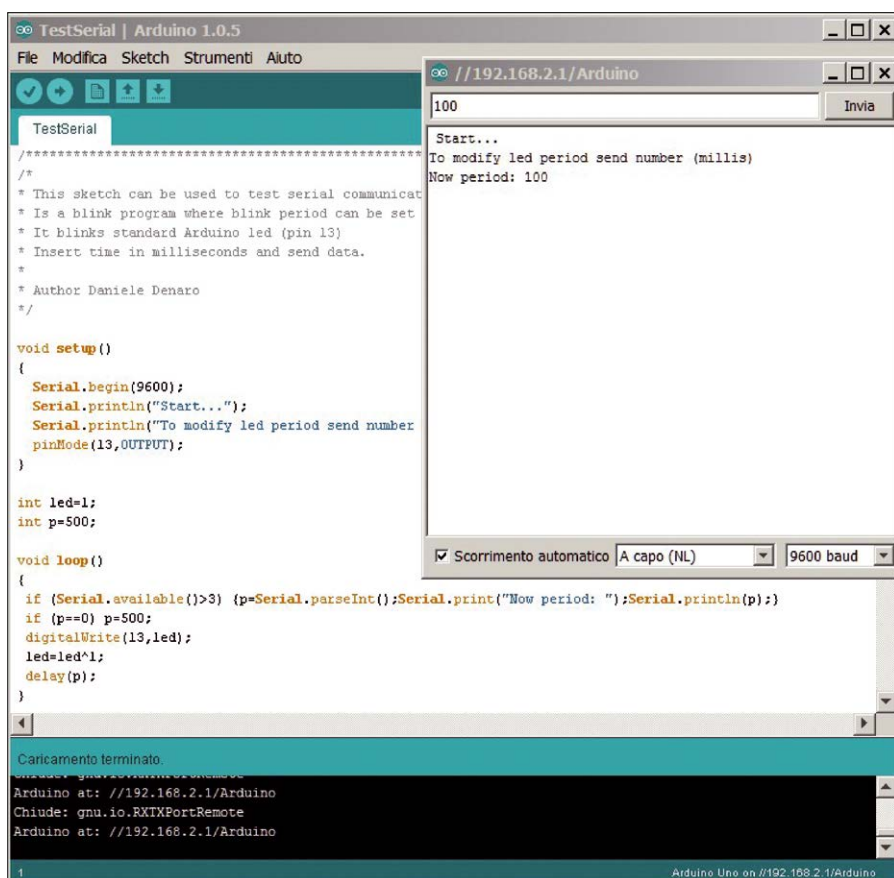


Fig. 9 - Lo sketch TestSerial da IDE Arduino.

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

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

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

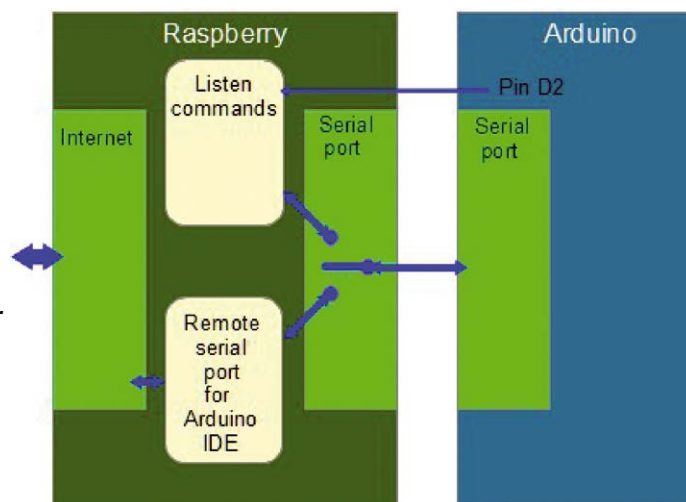
Infine nella cartella esempi c'è uno sketch che utilizza la modalità "Arduino sempre acceso". In questo sketch, il valore su un pin analogico decide se accendere Raspberry Pi -se non fosse già acceso- per poi scrivere su un file di log l'evento e spegnere di nuovo Raspberry Pi nel caso lo avesse trovato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

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

PROGRAMMIAMO RandA

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

Fig. 10
Porta seriale
al servizio di
programmi diversi.



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

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

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

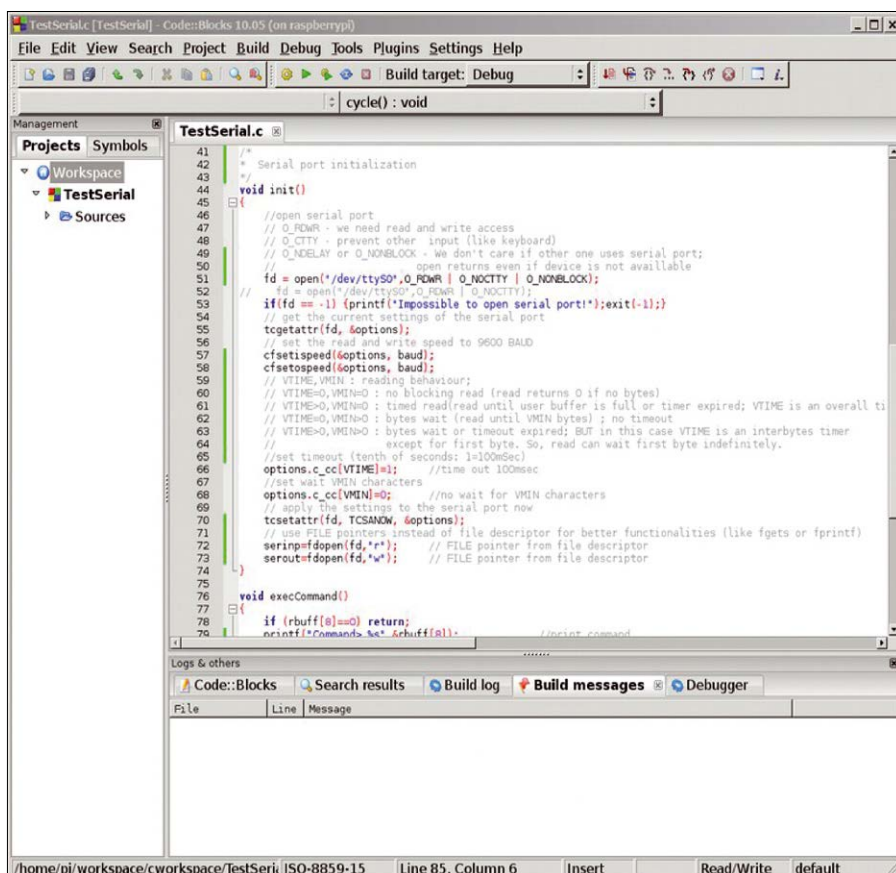
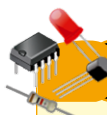


Fig. 11 - Codeblocks.



per il MATERIALE

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

Il materiale va richiesto a:

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

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

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

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

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

Naturalmente codeblocks, come tutti gli ambienti di programma-

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

RM ELETTRONICA

Forniture per hobbisti, scuole e industria

Distributore autorizzato:

**FUTURA
ELETTRONICA**

**ELSE
Kit**

**NUOVA
ELETTRONICA**

Il tuo punto
di riferimento
per l'elettronica
a ROMA

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