

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

Elettronica In

www.elettronica.in.it

oltre l'elettronica

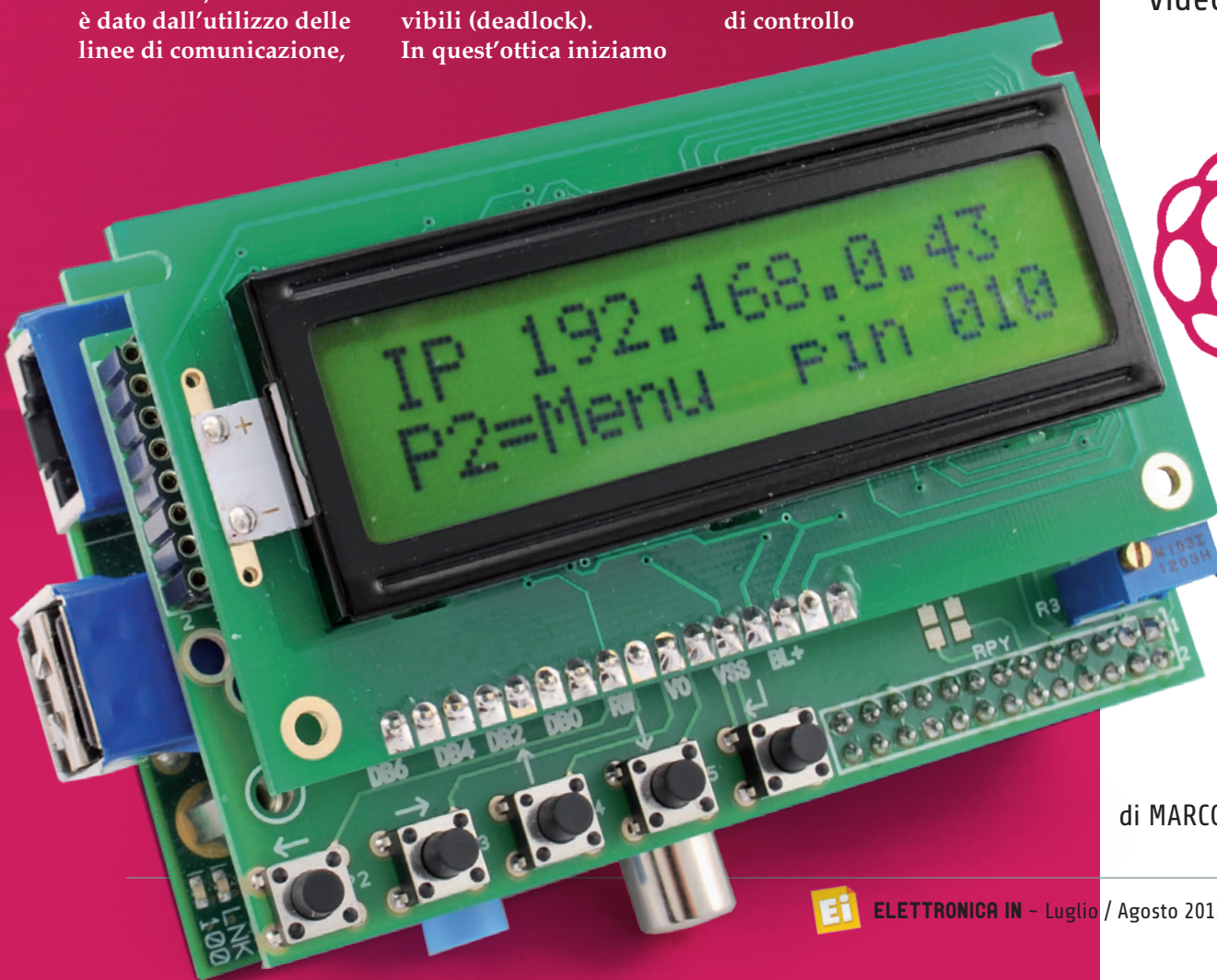
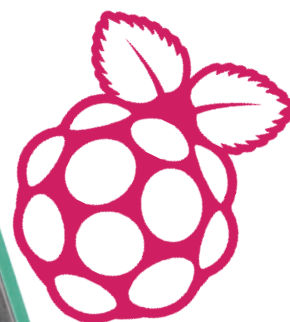
SHIELD DI ESPANSIONE LCD PER RASPBERRY PI

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

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

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

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



di MARCO MAGAGNIN

L'integrato MCP23017

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

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

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

Registro d'indirizzamento

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

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

Registri di gestione

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

Fig. B

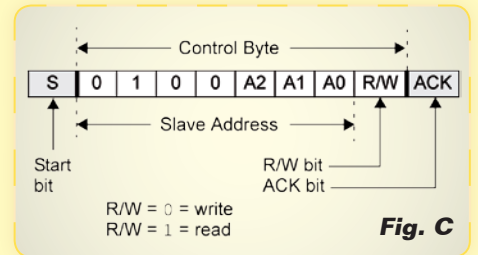
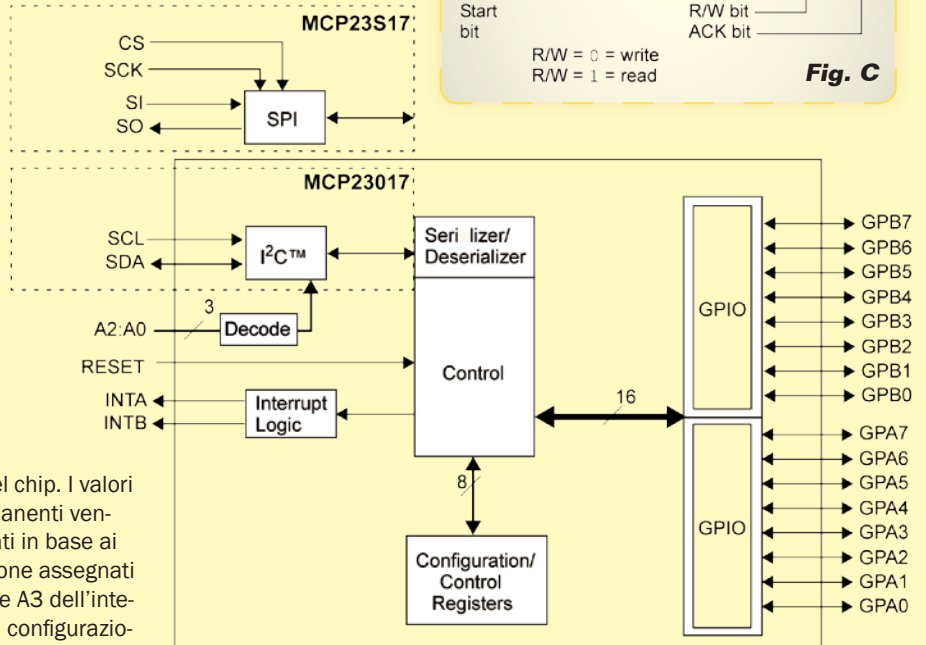


Fig. C

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

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

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

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

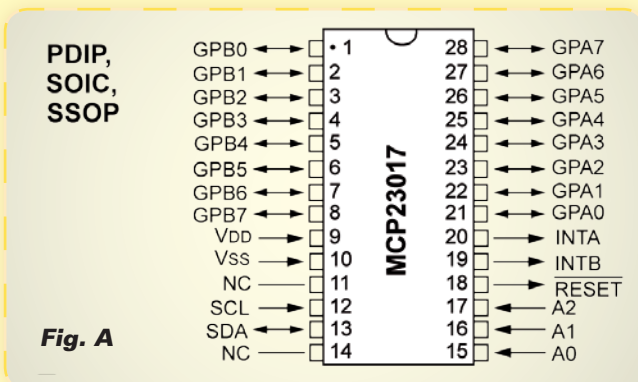


Fig. A

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

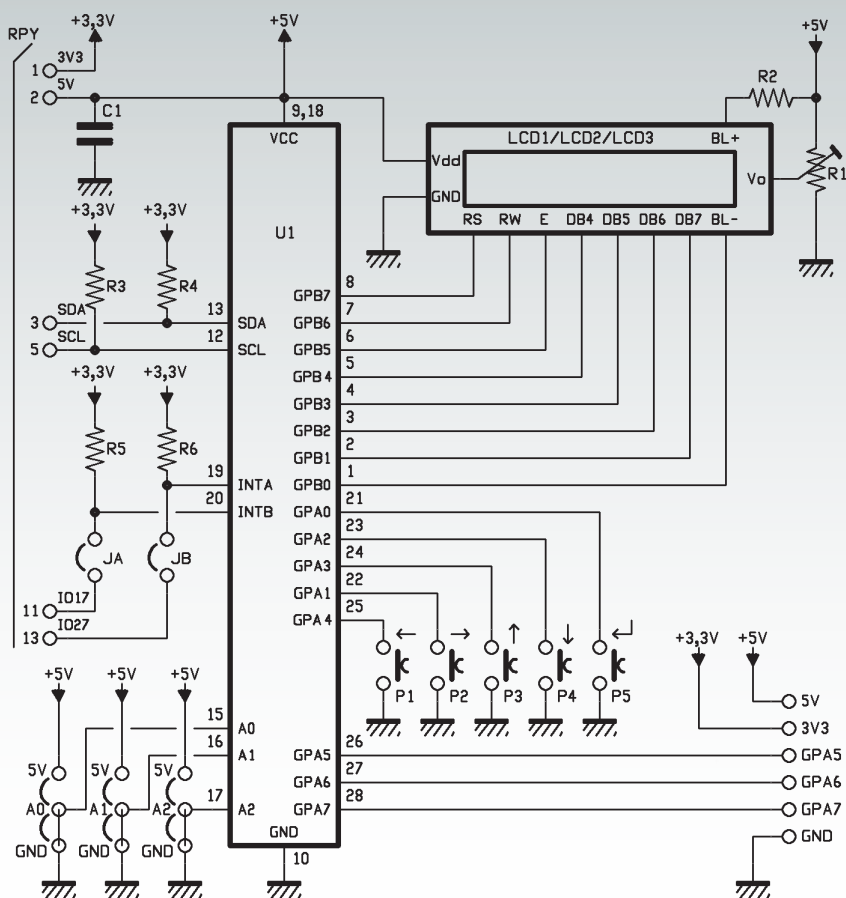
Register Name	Address (hex)	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	POR/RST value
IODIRA	00	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IODIRB	01	IO7	IO6	IO5	IO4	IO3	IO2	IO1	IO0	1111 1111
IPOLA	02	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0	0000 0000
IPOLB	03	IP7	IP6	IP5	IP4	IP3	IP2	IP1	IP0	0000 0000
GPINTENA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

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

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

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

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

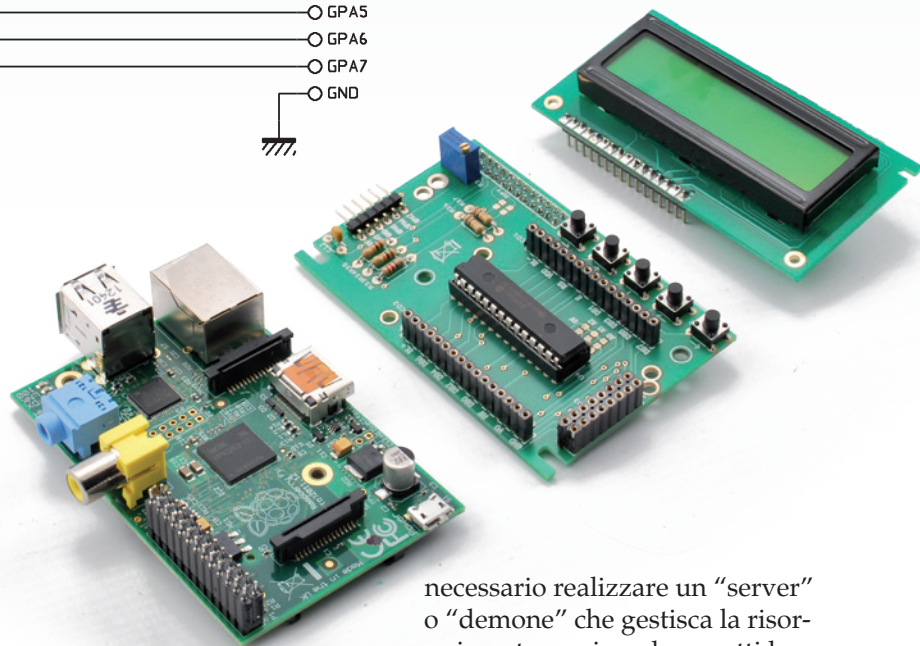


za di corrente o di connettività di rete o qualsiasi altra cosa sia necessaria.

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

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

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



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

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

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

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

SCHEMA ELETTRICO

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

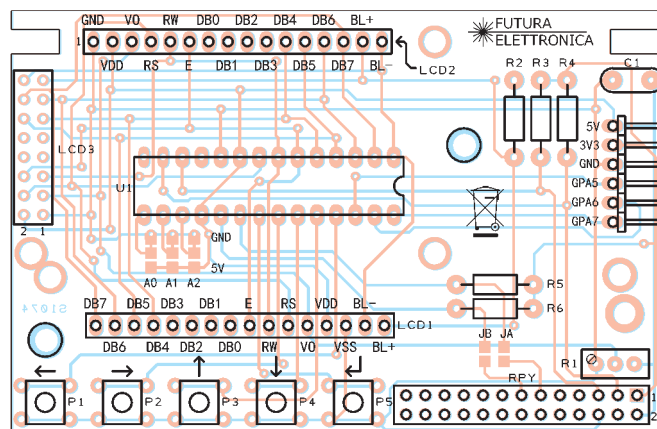
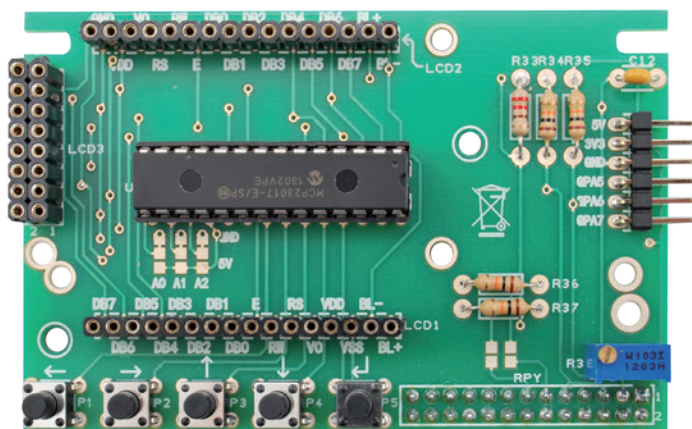
di Raspberry Pi abbiamo utilizzato l'integrato di Microchip MCP23017 che offre 16 ingressi/uscite digitali pilotabili mediante il bus I²C. Così facendo, impegnando in modo non esclusivo due soli pin del connettore di Raspberry Pi, possiamo pilotare uno schermo LCD, interfacciare cinque interruttori a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

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

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



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



Elenco Componenti:

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

U1: MCP23017-E/SP

Varie:

- Display LCD
- Torretta F/F 18 mm

- Vite 10mm 3 MA (2 pz.)

- Zoccolo 14+14

- Strip maschio 6 poli 90°

- Strip femmina 8 poli (2 pz.)

- Strip femmina 16 poli (2 pz.)

- Connettore Femmina 26 poli per RaspberryPi

- Circuito stampato

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

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

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

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

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

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

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

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

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

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

UTILIZZO PRATICO DELLO SHIELD

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

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

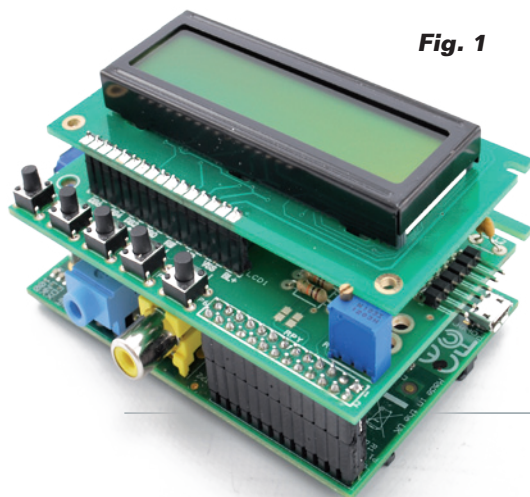


Fig. 1



Fig. 2

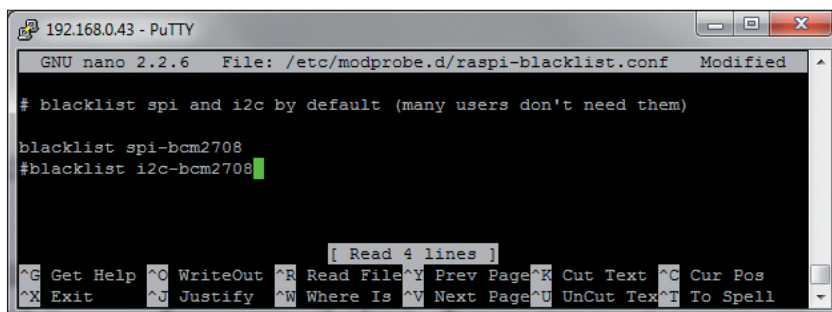


Fig. 3

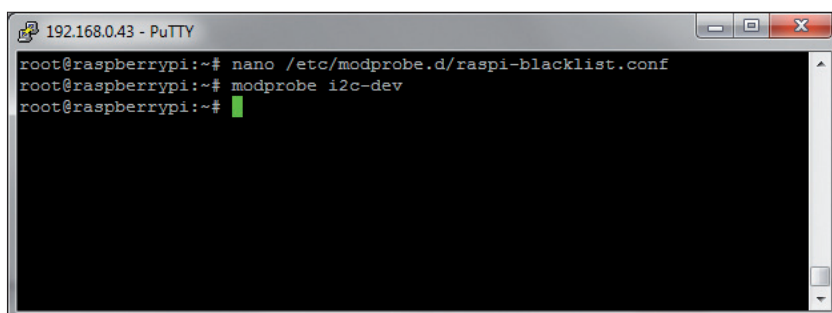


Fig. 4

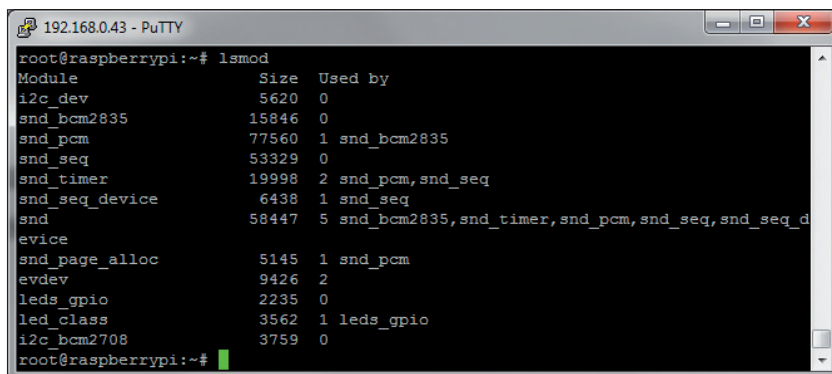


Fig. 5

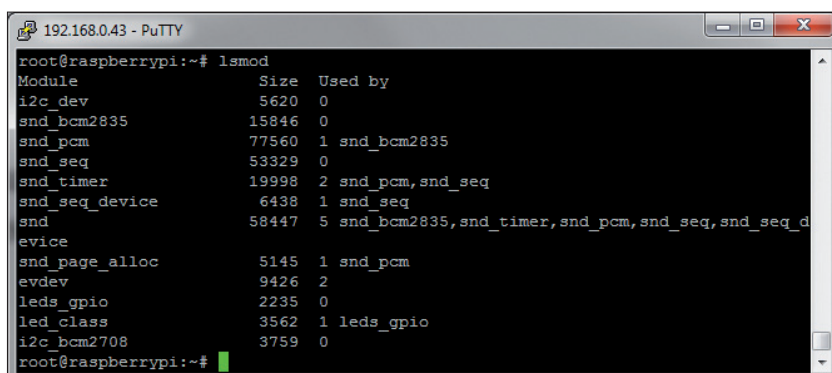


Fig. 6

Fig. 7

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

Fig. 8

```
192.168.0.43 - PuTTY
GNU nano 2.2.6      File: /etc/modules      Modified

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
i2c-dev
```

Fig. 9

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

Fig. 10

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

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

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

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

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

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

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

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

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

Eseguiamo un reboot per rendere effettive le modifiche.

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

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

modprobe i2c-dev

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

lsmod

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

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

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

modprobe -r i2c-dev

Se si desidera che il modulo

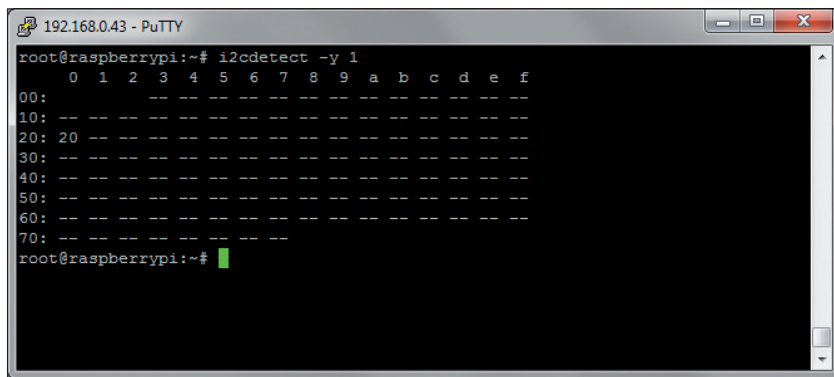


Fig. 11

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

nano /etc/modules

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

i2c-dev

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

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

apt-get install i2c-tools

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

adduser pi i2c

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

reboot

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

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

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

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

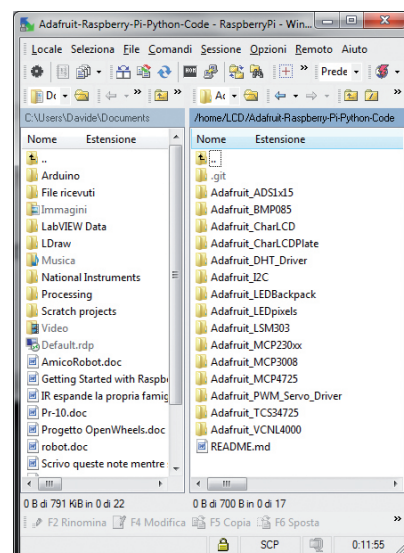
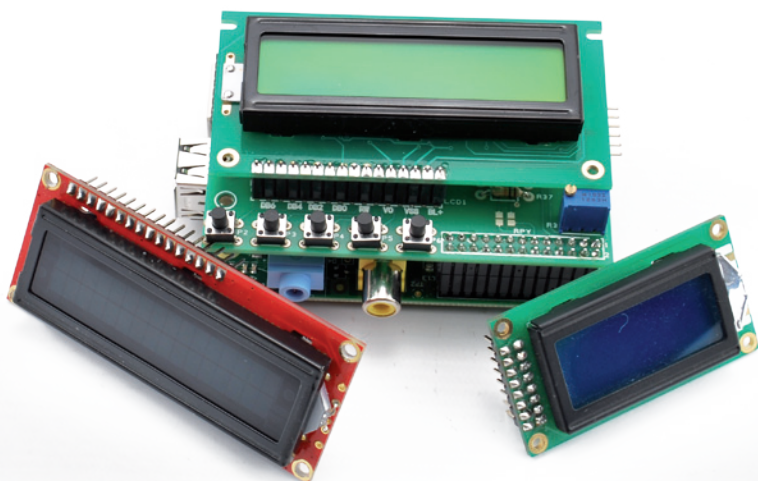


Fig. 12



strumento di gestione delle versioni di software "git".

Per installare "git" usiamo il comando:

```
apt-get install git
```

poi andiamo alla cartella home con il comando:

```
cd /home
```

creiamo una cartella per il nostro progetto, per esempio:

```
mkdir LCD
```

posizioniamoci nella cartella con il comando:

```
CD LCD
```

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

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

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

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

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

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

```
cd Adafruit_CharLCDPlate
```

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

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

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

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

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

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

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

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

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

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

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

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

# Read and return bitmask of pin A7
```

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

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

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

Copiate il programma in un file di nome LCDProva.

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

modprobe i2c-dev

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

python LCDProva.py

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

Listato 1

```
#!/usr/bin/python

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

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

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

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

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

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

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

(Continua)

Listato 1 – SEQUE

```
cpu = CPU()

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

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

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

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

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

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

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



Fig. 13



Fig. 14



Fig. 15



Fig. 16



Fig. 17

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

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

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

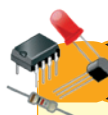
A scopo didattico, in questo programma di esempio, abbiamo voluto presentare il risultato di alcuni comandi che permettono di raccogliere informazioni sullo stato e/o sulle performance del sistema. Inutile ricordare che gli stessi comandi possono essere digitati nella finestra terminale come comandi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

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

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

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

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



per il MATERIALE

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

Il materiale va richiesto a:

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

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

Elettronica In

www.elettronica.in.it

oltre l'elettronica

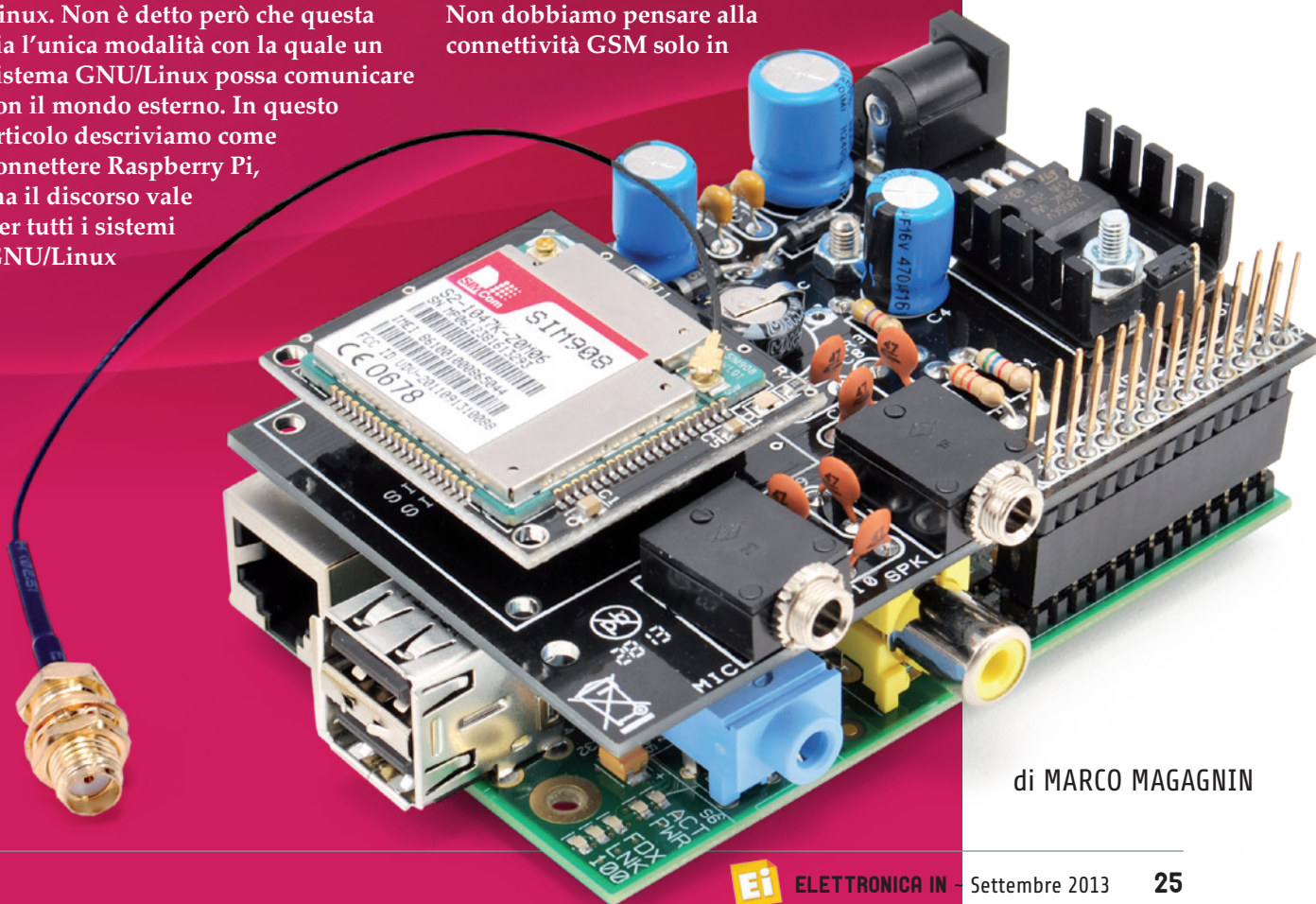
SHIELD DI ESPANSIONE GSM/GPRS/GPS PER RASPBERRY PI

Il sistema operativo GNU/Linux nasce con la "vocazione" della connettività Internet. I protocolli TCP/IP (Transport Control Protocol – Internet Protocol) sono legati a doppio filo alla storia di UNIX prima, e dei suoi successori poi, sino all'attuale GNU/Linux. Non è detto però che questa sia l'unica modalità con la quale un sistema GNU/Linux possa comunicare con il mondo esterno. In questo articolo descriviamo come connettere Raspberry Pi, ma il discorso vale per tutti i sistemi GNU/Linux

dotati di porta seriale, ai moduli della famiglia SIM9XX. In questo modo è possibile estendere le funzionalità di Raspberry Pi per applicazioni che richiedano connettività mobile e, impiegando il modulo SIM908, anche le funzionalità GPS.

Non dobbiamo pensare alla connettività GSM solo in

Shield di espansione per Raspberry Pi col quale controllare i moduli GSM/GPRS SIM900 e SIM908 (con GPS), per realizzare applicazioni che possono essere controllate e che possono comunicare in remoto tramite rete telefonica mobile.



di MARCO MAGAGNIN

applicazioni strettamente “mobile” come quelle su automezzi o in luoghi non collegati alla rete internet. Pensiamo semplicemente ad un sistema di domotica, di antifurto o di controllo di un impianto non presidiato.

Nel caso venga a mancare l'energia elettrica, nella maggioranza dei casi, a queste applicazioni verrebbe a mancare anche la connettività internet. In questo caso, l'unica possibilità di inviare un messaggio di avviso sullo stato delle cose. Ovviamente il sistema di controllo deve essere dotato di un sistema di batteria tampone della massima affidabilità, ed in grado di poter comunicare con il



sistema GNU/Linux. Ci stiamo pensando. Per ora torniamo al nostro shield GSM/GPRS e a come utilizzarlo con Raspber-

ry Pi. Come già anticipato nell'articolo nel numero 178 della rivista, gli shield dedicati a Raspberry Pi sono progettati per poter essere impilati l'uno sull'altro e poter gestire più funzioni contemporaneamente, in modo da poter integrare diversi “servizi” in un'unica soluzione integrata. Potrebbe essere il caso, appunto, di un sistema domotico, dotato di sensori e attuatori, che utilizzi l'LCD per l'attivazione e disattivazione, ma che renda possibile la stessa operazione anche da remoto, sia via rete internet che da GSM. Inoltre a questo possono essere aggiunti servizi di sicurezza per anziani, attivabili sia in locale che da remoto, con funzionalità vocali o di invio SMS a numeri predefiniti. Possiamo aggiungere la presa di immagini con videocamera, associata all'apertura delle porte e con riconoscimento facciale, o di movimento, con l'invio di messaggi e delle stesse immagini in caso di ... sospetto. L'unico limite è la fantasia e ... la capacità di progettare e realizzare le applicazioni. Applicazioni di questa portata superano le possibilità dei microcontrollori in genere, anche se questi sono utilizzabili con profitto nella gestione dei sensori e dei dispositivi periferici. Tornando sul piano concreto, chiaramente i diversi shield non devono utilizzare gli

I moduli GSM/GPRS utilizzati



Lo shield per Raspberry Pi presentato in questo articolo può ospitare sia il modulo SIM900 (GSM/GPRS) che il modulo SIM908 (+GPS). Più precisamente il modulo SIM900 è un Quad-band compatibile GSM/GPRS, operante sulle frequenze di 850/900/1800/1900 MHz, con SMS e accessi a Internet mediante stack TCP/IP integrato, gestito da un processore AMR926EJ-S che si occupa anche della gestione della comunicazione telefonica mediante comandi AT. Il modulo, compatibile GSM fase 2/2+, è di classe 4 (2 W) a 850/ 900 MHz e di classe 1 (1 W) a 1800/1900 MHz; SIM900 integra



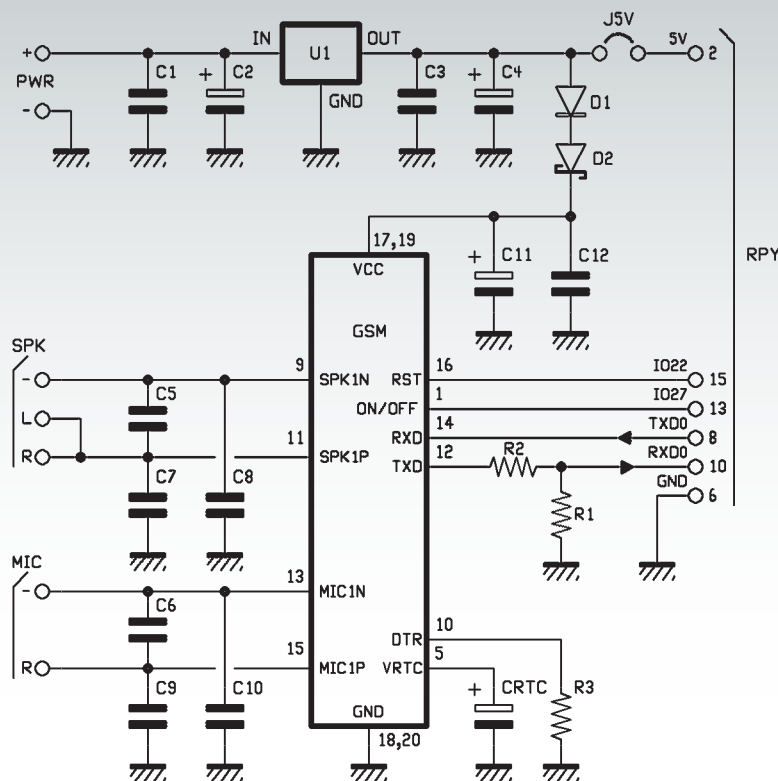
un'interfaccia analogica, un A/D converter, un RTC, un bus SPI, uno I²C, ed un modulo PWM. Il modulo si alimenta a tensione continua di valore compreso fra 3,4 e 4,5 V, assorbe un massimo di 0,3 A (continuativi) in trasmissione. È dotato di connettore per antenna esterna.

Il SIM908 integra un ricevitore GPS (con antenna separata) a 42 canali capace di una sensibilità di -160 dB in tracking e -143 dB in acquisizione, di una precisione nella localizzazione di soli 2,5 m e di tempi di operatività di 1 s per l'Hot Start e 30 secondi per il Cold Start. Il set di comandi AT è esteso per l'accesso al GPS. I manuali sono reperibili in PDF in internet cercando “SIM900 AT command” e “SIM908 AT command”.

stessi pin, alimentazione a parte, in modo da evitare conflitti non gestibili via software. Nello shield GSM/GPRS/GPS vengono utilizzati i pin del connettore GPIO di Raspberry Pi collegati alla porta seriale e due ulteriori pin di I/O per l'accensione/spegnimento ed il reset del modulo SIM9XX.

SCHEMA ELETTRICO

Lo schema elettrico dello shield FT1075 è finalizzato a fornire l'alloggiamento fisico per i moduli SIM900 e SIM908, a fornire un'alimentazione autonoma al modulo GSM, a esportare l'ingresso microfono e l'uscita audio su due jack dedicati ed a connettere correttamente i pin del connettore GPIO di Raspberry Pi con quelli dei moduli GSM. Iniziamo dal circuito di alimentazione composto dal regolatore U1, un integrato 7805 e dalla rete di condensatori di filtraggio e stabilizzazione della tensione (C1, C2, C3 e C4). Attraverso il diodo D1 e lo zener D2 viene portata la tensione positiva ai pin VCC dei moduli SIM9XX, ulteriormente filtrata e stabilizzata dai condensatori C11 e C12. Tramite il ponticello J5V è possibile alimentare anche Raspberry Pi dalla stessa fonte dello shield. Al connettore PWR è possibile collegare un alimen-



[schema ELETTRICO]

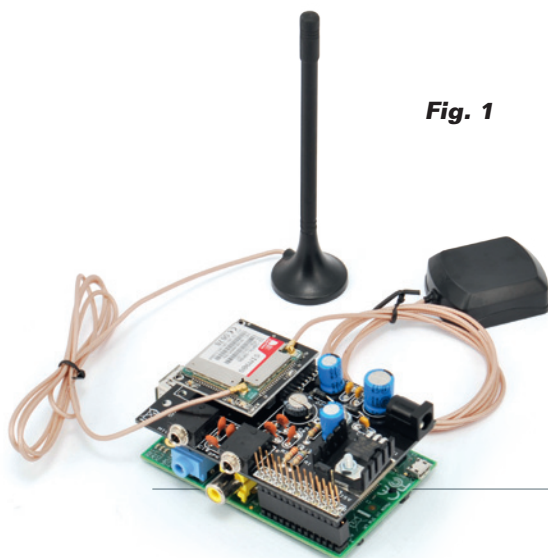
tatore in continua che eroghi una tensione tra i 7 e i 12 Volt. Il pin ON/OFF del SIM9XX è collegato al pin 13 del GPIO di Raspberry Pi (GPIO27) per l'accensione e lo spegnimento del modulo da programma. Il pin di reset RST del modulo SIM9XX è collegato al pin 15 di Raspberry Pi (GPIO22), sempre per permetterne l'azionamento da programma. Il collegamento seriale è attestato sui pin 8 (TXD) e 10 (RXD) di Raspberry Pi. Sulla linea in ingresso RDX l'adattamento al livello di tensione di 3,3 V richiesti da Raspberry Pi è realizzato dal partitore di tensione composto da R1 ed R2. Nessun adattamento di tensione è richiesto nel senso opposto in quanto i 3,3 V in uscita da Raspberry Pi sono compatibili con l'ingresso dei SIM9XX. Il DTR è tenuto costantemente a livello basso mediante la resistenza R3. Il pin per l'alimentazione in tampone del Real-Time Clock (VRTC) è collegato al condensatore CRTC.

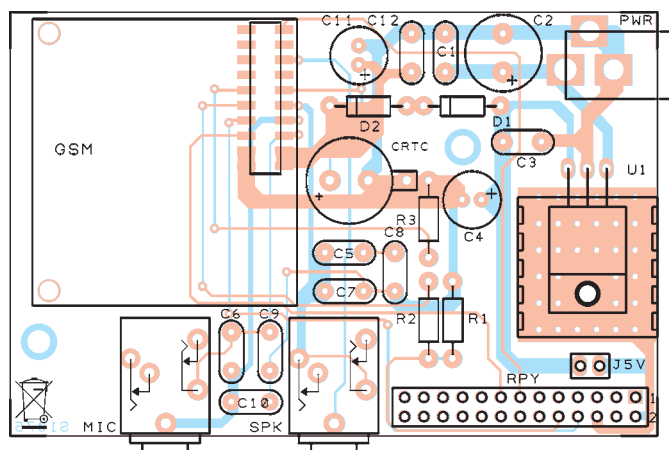
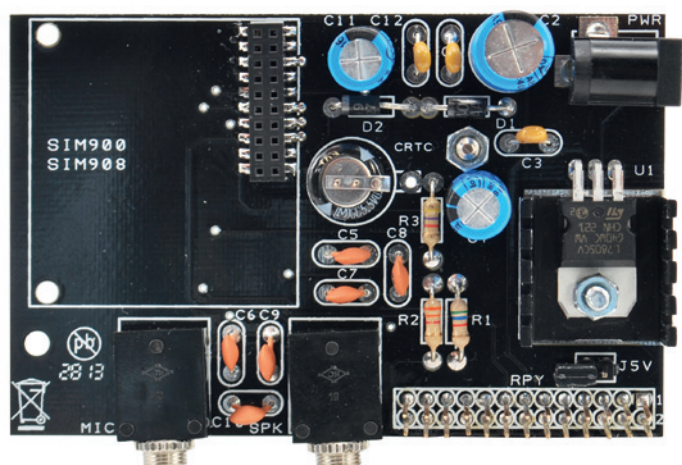
Infine le linee di ingresso microfonico e di uscita audio sono collegate ai rispettivi connettori MIC e SPK ciascuno con la sua rete di condensatori di filtraggio, C6, C9 e C10 per MIC e C5, C7 e C8 per SPK.

Nel piano di montaggio è possibile vedere la disposizione dei componenti. Unico suggerimento, a parte le usuali attenzioni, è di posizionare correttamente il doppio connettore a pettine in modo da distanziare lo stampato dello shield da quello di Raspberry Pi per permettere il montaggio "a pacchetto", nel caso lo si voglia utilizzare congiuntamente ad altri shield, come lo shield LCD che vi abbiamo presentato nel numero 178. Pertanto, prima di saldare i pin del connettore conviene posizionare il connettore a 26 pin procedendo come descritto di seguito:

- Posizionare il distanziale esagonale, inserendo l'estremità filettata dal lato saldature nel

Fig. 1





Elenco Componenti:

R1: 5,6 kohm
R2: 3,3 kohm
R3: 4,7 kohm
C1: 100 nF multistrato
C2: 470 μ F 25 VL elettrolitico
C3: 100 nF multistrato
C4: 470 μ F 16 VL elettrolitico
C5÷C9: 47 pF ceramico
C10: 47 nF ceramico
C11: 470 μ F 16 VL elettrolitico

C12: 100 nF multistrato
U1: 7805
D1: 1N4007
D2: 1N5819
CRTC: Condensatore basso profilo 0,1F

Varie:
- Plug alimentazione
- Presa jack stereo 3,5 mm da CS (2 pz.)
- Dissipatore

- Vite 8 mm 3 MA
- Vite 12 mm 3 MA
- Dado 3 MA (2 pz.)
- Jumper
- Strip maschio 2 poli
- Presa MMS SMT 2x10 vie passo 2 mm
- Connettore 13x2 RaspberryPi
- Torretta M/F 18 mm
- Circuito stampato

foro vicino al condensatore CTRC e fissarlo con il dado M3;

- Posizionare il connettore a 26 pin nelle piazzole identificate con RPY, tenendo il lato con i terminali femmina verso il basso;
- Applicare lo shield su Raspberry Pi facendo appoggiare su quest'ultima il distanziale esagonale. Unire i due connettori da 26 pin, e fissare il distanziale con la vite M3;
- Una volta che le schede sono allineate e distanziate correttamente è possibile saldare sullo shield i terminali maschio del connettore.

Sempre per garantire la possibilità di "impacchettamento" posizionare con attenzione l'aletta di raffreddamento dell'integrato U1, in modo che non vada a toccare l'eventuale shield montato al di sopra.

UTILIZZO PRATICO DELLO SHIELD

Come prima operazione monteremo sullo shield uno dei moduli GSM compatibili e poi, se non ancora fatto, monteremo lo shield sul connettore di Raspberry Pi, facendo attenzione che la parte inferiore dello shield non vada in contatto con i connettori USB o Ethernet. In caso di dubbio proteggeremo i connettori stessi con del nastro isolante. Colleghiamo le periferiche, la rete e l'alimentazione a Raspberry Pi nel solito modo. Colleghiamo l'alimentazione allo shield FT1075 e ... diamo tensione.

In questo articolo vi proponiamo un approccio didattico all'utilizzo dello shield GSM in modo da dare la possibilità di sperimentare il più possibile prima di iniziare a scrivere applicazioni più o meno complesse. Ricordiamo che utiliz-

zando GNU/Linux possiamo realizzare applicazioni che svolgono molte funzioni, condividendo risorse come lo shield GSM. Ad esempio possiamo scrivere un'applicazione che resti in attesa di eventi esterni come l'arrivo di una chiamata voce od un messaggio SMS, ma sia anche in grado di reagire ad eventi interni come l'attivazione di ingressi di I/O, sia digitali che analogici, oppure alla richiesta di esecuzione di chiamate voce, invio di SMS, singoli o di massa, invio di file, messaggi di posta elettronica ecc. Nel frattempo si può desiderare di acquisire dati dalla sezione GPS del modulo. Per fare questo, come già descritto per lo shield LCD, è necessario progettare un'architettura modulare in grado di condividere le risorse senza creare condizioni di conflitto. Ne risulta un'architettura composta da

moduli server che interfacciano e controllano le risorse e i moduli operativi ciascuno specializzato in una specifica funzione che possono essere richiamati ed eseguiti al verificarsi di specifiche condizioni e richiedere servizi di comunicazione ai moduli server, che provvederanno a soddisfarli secondo una logica che garantisca le priorità ed eviti i conflitti. Per fare questo è necessario approfondire fin nei dettagli le esigenze funzionali dell'applicazione ed è il motivo per il quale, prima di addentrarci, nei prossimi numeri, nella progettazione di applicazioni client/server, vi proponiamo un paio di modalità che permettano di sperimentare in modo agile l'utilizzo dello shield.

La prima modalità permette di riutilizzare con poche modifiche i programmi scritti per Arduino, in modo da prendere dimestichezza con lo shield per eredità dal mondo Arduino.

In questo modo si potranno utilizzare gli esempi proposti nei numeri scorsi della rivista nella serie di articoli "Gestire i moduli GSM con Arduino". Per questa modalità utilizzeremo la libreria arduPi realizzata da Libelium Comunicaciones Distribuidas S.L., messa a disposizione in licenza GNU General Public License e reperibile da CookingHacks che vi abbiamo già presentato nell'articolo "Shield di espansione ADC per Raspberry Pi" nel numero 174 della rivista.

La seconda modalità è una modalità diretta. Ci colleghiamo alla porta seriale con un programma TTY e comandiamo il modulo GSM dando e ricevendo direttamente i comandi di gestione. Questo ci permette di provare le sequenze di comandi necessari a realizzare una specifica funzionalità senza la necessità di scrivere e testare ogni volta un programma.

INVIAMO UN SMS IN STILE ARDUINO

Della libreria arduPi abbiamo già parlato nel numero 174 della rivista. Ci limitiamo a ricordare brevemente le modalità di installazione ed utilizzo. Per installare la libreria arduPi è sufficiente scaricarla dall'indirizzo:

http://www.cooking-hacks.com/skin/frontend/default/cooking/images/catalog/documentation/raspberry_arduino_shield/arduPi_1-5.tar.gz

La scompattiamo nella cartella /home e, per comodità, ne modifichiamo il nome in "ArduPi". Sempre per comodità metteremo nella stessa cartella i programmi che andremo a realizzare. Prima di compilare la libreria apriamo il programma arduPi.cpp e, se diversa, sostituiamo alla riga 64 il riferimento alla porta seriale esistente con la riga:

```
serialPort="/dev/ttyAMA0";
```

Ora andiamo nella cartella ArduPi (Fig. 2) con il comando

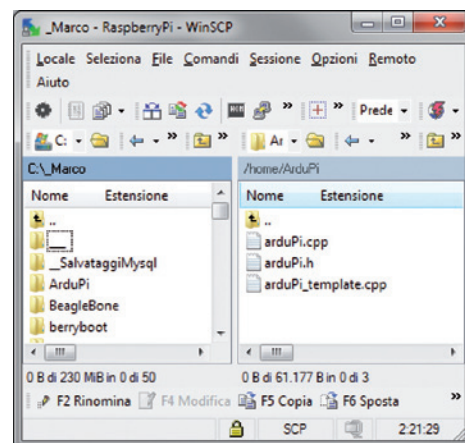
```
cd /home/ArduPi
```

e compiliamo la libreria con il comando (Fig. 3):

```
g++ -c arduPi.cpp -o arduPi.o
```

Per quanto riguarda i programmi vi proponiamo tre moduli. Due permettono di accendere e spegnere il modulo GSM e ci saranno molto utili in futuro.

Il terzo ci permette di inviare un SMS ad un telefono cellulare. Per



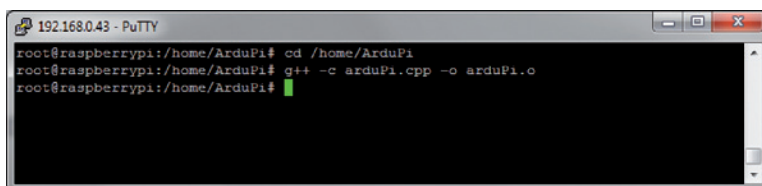
accendere il modulo è necessario alzare il livello del pin GPIO27 per due secondi e poi portarlo di nuovo a livello basso, stessa operazione per spegnerlo, se questo è acceso. Per il reset è necessario alzare e mantenere a livello alto per mezzo secondo il pin GPIO22, per poi riportarlo a livello basso. I programmi sono scaricabili liberamente dal sito www.elettronica.in.

Prima di tutto, per poter utilizzare la porta seriale, dobbiamo eseguire una modifica al processo di boot per disattivare la console seriale, che altrimenti impegnerebbe la porta seriale.

DISABILITIAMO LA CONSOLE SERIALE

Pressoché tutti i sistemi GNU/Linux attivano, alla partenza, la funzionalità console, cioè la possibilità di collegarsi alla linea seriale con un terminale - reale o virtuale - per gestire tutto il sistema tramite questa interfaccia di comunicazione. Un ricordo dei tempi nei quali ci si collegava ai computer via modem utilizzando le normali linee telefoniche. A volte è ancora l'unica possibilità di accesso ai sistemi embedded o server. Per

Fig. 3



Listato 1

```

/*
 * GSMAccendi
 */

//Include Arduino library
#include "arduino.h"

int resetModulePin = 9;
int onModulePin = 8;

void switchModuleOn(){
    digitalWrite(onModulePin,HIGH);
    delay(2000);
    digitalWrite(onModulePin,LOW);
}

void resetModule(){
    digitalWrite(resetModulePin,HIGH);
    delay(500);
    digitalWrite(resetModulePin,LOW);
    delay(100);
}

int main (){
    Serial.begin(115200);
    delay(2000);
    pinMode(resetModulePin, OUTPUT);
    pinMode(onModulePin, OUTPUT);
    Serial.flush();
    printf ("zero\n");
    Serial.print("AT");
    delay(1000);
    if (Serial.available()==0)
    {
        printf ("uno\n");
        resetModule();
        delay(2000);
    }
    Serial.print("AT");
    delay(1000);
    if (Serial.available()==0)
    {
        printf ("due\n");
        switchModuleOn();
    }

    return (0);
}

```

disabilitare la console utilizziamo WINScp, andiamo nella cartella "boot" con il comando:

```
cd /boot
```

e apriamo, con un doppio clic, il file /boot/cmdline.txt eliminiamo, facendo attenzione, i parametri di configurazione (Fig. 4):

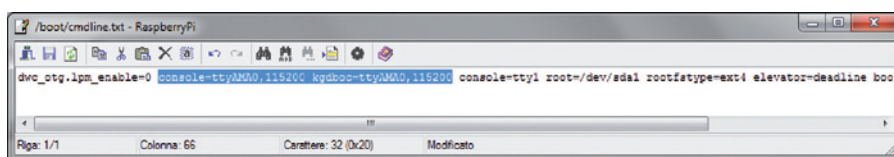


Fig. 4

```
console=ttyAMA0,115200 kgdboc=ttyAMA0,115200
```

Salviamo il file e chiudiamolo.

ESEMPI DI PROGRAMMI (ON/OFF DEL MODULO E INVIO SMS)

Vediamo la logica del programma di accensione del modulo, ispezionabile nel Listato 1. Come prima considerazione potete verificare che i programmi sono codificati per eseguire un'unica sequenza di istruzioni, contrariamente ai programmi per Arduino, che funzionano all'infinito all'interno di un ciclo loop(). Questo permette di utilizzare i singoli programmi come moduli di un'applicazione di livello superiore.

Il processo di accensione del modulo GSM prevede i seguenti passi:

- Controllo, mediante l'invio sulla seriale di un comando "AT" per verificare che il modulo non sia già acceso;
- In caso di risposta negativa (all'invio di un comando AT, in situazioni normali si riceve di ritorno la stringa "OK"), come ulteriore controllo, viene eseguita la procedura di reset con il successivo invio di un secondo comando "AT";
- In caso di ulteriore risposta negativa viene eseguita la procedura di accensione che consiste nel portare il pin GPIO27 a livello alto per due secondi.

Lo scopo dei controlli preliminari è dovuto al fatto che il processo di accensione e spegnimento è lo stesso, quindi, se si esegue il programma di accensione senza

controlli, se il modulo GSM è già acceso, lo si spegne. Per compilare il programma GSMAccendi si utilizza il comando:

```
g++ -lrt -lpthread GSMAccendi.cpp
arduino.o -o GSMAccendi
```

Per eseguire il programma, dopo essersi posizionati nella cartella Arduino con il comando:

```
cd /home/Arduino
```

si digita:

```
./GSMAccendi
```

Possiamo seguire la procedura di accensione del modulo GSM osservando il LED verde, all'accensione rimane acceso per circa due secondi, poi si spegne ed inizia a lampeggiare con intervalli di circa mezzo secondo, segno che il modulo è alla ricerca della rete GSM. Quando la rete è agganciata il LED lampeggia con una frequenza molto più bassa, circa un lampeggio al secondo.

Nel Listato 2 è visibile il programma per lo spegnimento del modulo, che esprime una logica simile al programma precedente:

- Controllo, mediante l'invio sulla seriale di un comando "AT" per verificare che il modulo non sia già spento;
- In caso di risposta negativa, come ulteriore controllo, viene eseguita la procedura di reset con il successivo invio di un secondo comando "AT";
- In caso di risposta positiva viene eseguita la procedura di spegnimento che consiste nel portare il pin GPIO27 a livello alto per due secondi.

Anche in questo caso lo scopo dei controlli preliminari è dovuto al fatto che il processo di accensione e spegnimento è lo stesso, quindi, se si esegue il programma di

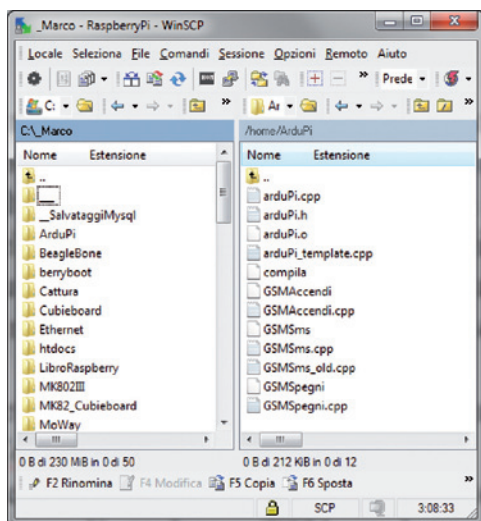


Fig. 5

- Infine vengono inviati i caratteri CTRL-Z per chiudere la chiamata.

Per compilare il programma si digita il comando:

```
g++ -lrt -lpthread GSMsms.cpp
arduPi.o -o GSMsms
```

Per eseguire il programma si esegue il comando:

```
./GSMsms
```

Prima di eseguire il programma è necessario eseguire il programma di accensione del modulo GSM, per sincerarsi che il modulo sia effettivamente acceso e funzionante. Quando si andrà a sviluppare l'applicazione di controllo, sarà compito di quest'ultima gestire la sequenza corretta di chiamata dei moduli, con i relativi controlli di stato e gestione delle condizioni anomale.

Per sperimentare altri programmi potete riferirvi agli esempi presenti nella serie di articoli "Gestire i moduli GSM con Arduino", eventualmente trasformandoli in moduli monofunzionali, come quelli descritti in questo articolo. Nella Fig. 5 è visibile lo stato della cartella dopo la scrittura e compilazione dei programmi. Nella Fig. 6 è visibile la sequenza dei comandi di compilazione e di esecuzione dell'accensione del modulo GSM, invio SMS e spe-

gnimento senza controlli, se il modulo GSM è già spento, lo si accende. Per compilare il programma GSMSpegni si utilizza il comando:

```
g++ -lrt -lpthread GSMSpegni.cpp
arduPi.o -o GSMSpegni
```

Per eseguire il programma, sempre dopo essersi posizionati nella cartella ArduPi, si utilizza il comando:

```
./GSMSpegni
```

Nel Listato 3 vediamo il programma che permette di inviare un SMS.

- Nella variabile phone_number[] si inserisce il numero di telefono da chiamare;
- AT+CMGF=1 imposta la modalità SMS in formato testo;
- AT+CMGS= invia il messaggio SMS impostato nell'istruzione successiva;

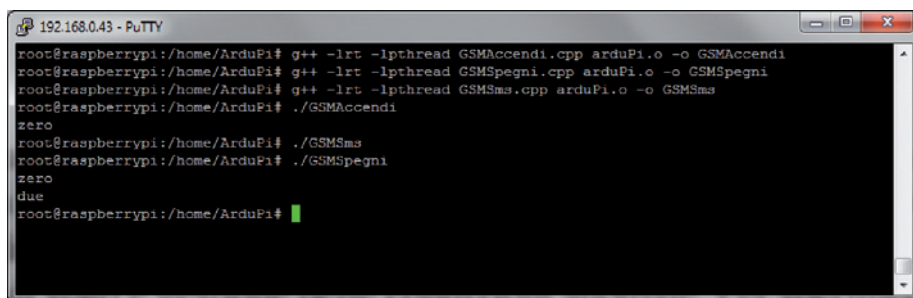


Fig. 6

gnimento del modulo.

Vediamo ora la seconda modalità, quella che permette di interagire direttamente con la porta seriale mediante il programma di emulazione TTY "CuteCom".

E siccome abbiamo impostato Raspberry Pi per essere gestita in remoto tramite protocollo SSH, continuiamo nella stessa direzione in modo da utilizzare l'emulatore TTY con le stesse modalità.

INSTALLIAMO L'EMULATORE TTY SERIALE

Listato 2

```
/*
 * GSMSpegni
 */

//Include ArduPi library
#include "arduPi.h"

int resetModulePin = 9;
int onModulePin = 8;

void switchModuleOff() {
    digitalWrite(onModulePin, HIGH);
    delay(2000);
    digitalWrite(onModulePin, LOW);
}

void resetModule() {
    digitalWrite(resetModulePin, HIGH);
    delay(500);
    digitalWrite(resetModulePin, LOW);
    delay(100);
}

int main () {
    Serial.begin(115200);
    delay(2000);
    pinMode(resetModulePin, OUTPUT);
    pinMode(onModulePin, OUTPUT);
    Serial.flush();
    printf("zero\n");
    Serial.print("AT");
    delay(1000);
    if (Serial.available() == 0)
    {
        printf("uno\n");
        resetModule();
        delay(2000);
    }
    Serial.print("AT");
    delay(1000);
    if (Serial.available() > 0)
    {
        printf("due\n");
        switchModuleOff();
    }

    return (0);
}
```

Listato 3

```

/*
 * GSMSms
 */

//Include ArduPi library
#include "arduPi.h"

int timesToSend = 1;      // Numero di SMS da inviare
int count = 0;
int ok = 0;
int numCar = 0;

char phone_number[]="3.....9"; // * Numero telefono dest. SMS

void setup(){
    Serial.begin(115200);    // UART baud rate
    delay(2000);
    Serial.println("AT+CMGF=1");    // sets the SMS mode to text
    delay(100);
}

void loop(){
    while (count < timesToSend){
        delay(1500);
        Serial.print("AT+CMGS=\"");    // send the SMS number
        Serial.print(phone_number);
        Serial.println("\"");
        while(Serial.read()!='>');
        Serial.print("Se arriva funziona"); // the SMS body
        delay(500);
        Serial.write(0x1A);    //sends ++
        Serial.write(0x0D);
        Serial.write(0x0A);
        delay(5000);
        count++;
    }
}

int main (){
    setup();
    while(1){
        loop();
        if (count == timesToSend)
        {
            break;
        }
    }
    return (0);
}

```

Per la gestione della porta seriale abbiamo scelto di utilizzare il programma "CuteCom". Lo installiamo con i soliti comandi:

```

apt-get update
apt-get upgrade
apt-get install cutecom

```

INSTALLIAMO XMING

Xming è un server X open source per Windows che permette di visualizzare sul desktop del PC Windows le finestre grafiche delle applicazioni che girano sul sistema GNU/Linux in remoto. Non viene trasferito l'intero

desktop ma unicamente la o le finestre grafiche dei singoli applicativi, che possono essere posizionate a piacere sul desktop del PC locale, assieme alle finestre del PC stesso o di altri sistemi remoti. Vi lasciamo immaginare gli aneddoti sorti al riguardo, nel mondo degli amministratori di sistema, sui comandi dati guardando la finestra di amministrazione di un

sistema e digitando in quella di un sistema diverso. Comunque la funzionalità è affascinante.

X è l'applicazione basata sul modello client/server che permette di esportare su una macchina il front-end grafico di un'applicazione in esecuzione in un sistema diverso.

Si appoggia sul protocollo SSH e sull'applicazione Putty (o Kitty) che abbiamo presentato nei numeri precedenti della rivista. La denominazione client/server spesso confonde gli utilizzatori di X perché i termini sembrano invertiti: sul desktop Windows dell'utente è in esecuzione il "server" che viene utilizzato dai programmi applicativi (client). X fornisce un servizio di visualizzazione ai programmi, in quest'ottica agisce da server, mentre il programma applicativo (che può anche essere gestito tramite connessione remota) utilizza i servizi, agendo come un client.

Il protocollo di comunicazione tra server e client opera in modo trasparente rispetto alla rete: entrambi possono risiedere sulla stessa macchina o su macchine diverse, anche con architetture e sistemi operativi differenti; il server e il client possono anche comunicare in modo sicuro attraverso la rete sfruttando un tunnel cifrato (spesso viene utilizzato SSH).

Il server X può essere:

- Un programma di sistema che controlla l'output video di un computer;
- Un componente hardware dedicato (i cosiddetti X terminal: calcolatori dotati solo dell'hardware necessario ad eseguire il server X, pensati per visual-

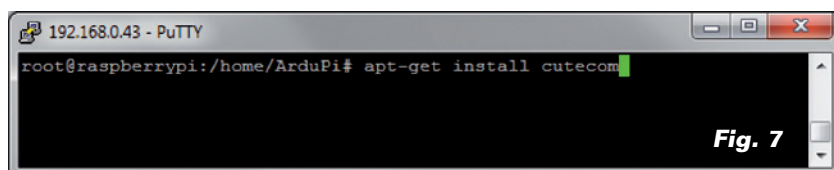


Fig. 7

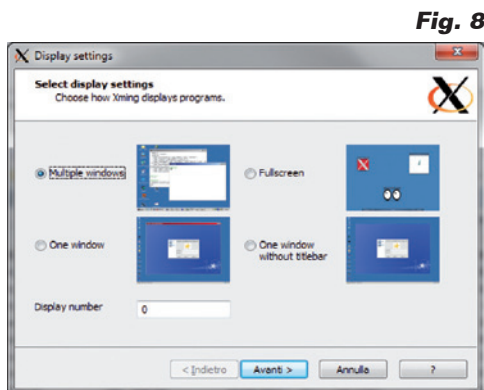


Fig. 8

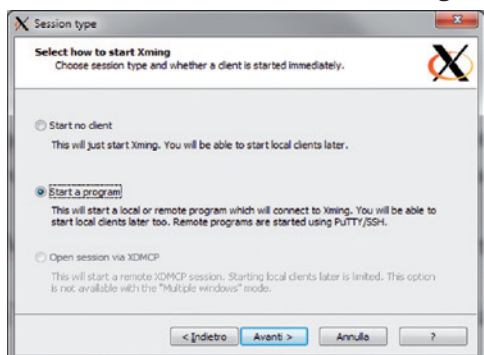


Fig. 9

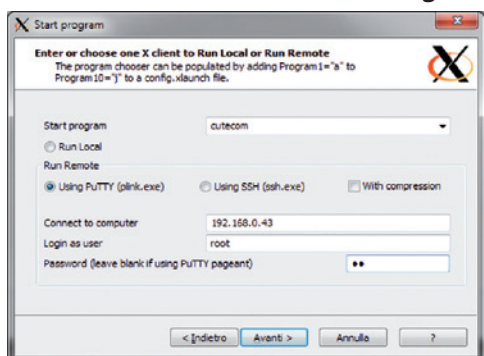


Fig. 10

lizzare applicazioni eseguite su appositi server applicativi);

- Un'applicazione che mostra dati su una finestra di un altro sistema grafico.

Xming è scaricabile da:

<http://sourceforge.net/projects/xming/>

Lo scarichiamo e installiamo come un qualsiasi applicativo Windows.

La nostra attenzione va al programma XLaunch, lo troviamo nel menu di Windows nella scelta Xming, che ci permette di lanciare in remoto quante singole applica-

zioni vogliamo in finestre separate di Windows. Una volta installato Xming lanciamo CuteCom in remoto sul desktop del nostro PC. Apriamo XLaunch (Fig. 8). Selezioniamo "Multiple Windows" e "Avanti". Nella videata successiva selezioniamo "Start a program" (Fig. 9) e ancora avanti. Ora selezioniamo "Using Putty" (Fig. 10). Nel campo Start Program digitiamo "cutecom", in "Connect to computer" inseriamo l'indirizzo IP di Raspberry Pi. In "Login as user" digitiamo "root" e nel campo password la relativa password. Ancora "Avanti" e "Avanti" anche nella pagina successiva (Fig. 11). Arriviamo all'ultima pagina dove confermiamo con "Fine" (Fig. 12). Diamo "OK" e "NO" in successione agli eventuali messaggi e ... aspettiamo. Alla fine otteniamo la finestra di Fig. 13.

GESTIAMO IL MODULO GSM DALL'EMULATORE TTY

La finestra di CuteCom è divisa in "zone" funzionali. In alto sono presenti i campi ed i pulsanti per configurare, attivare e disattivare la comunicazione seriale, in basso il campo di testo per inserire i comandi da inviare sulla porta seriale, un'area con la "history" dei comandi inviati ed alcuni comandi di configurazione che permettono di:

Fig. 11

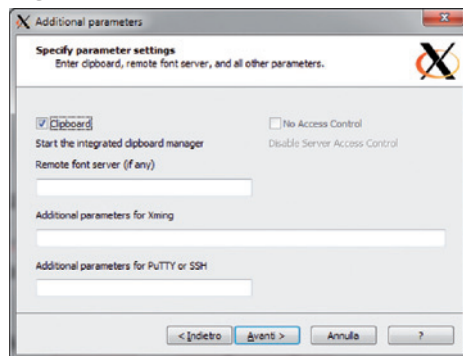
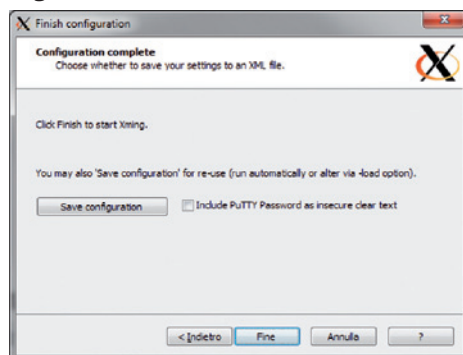


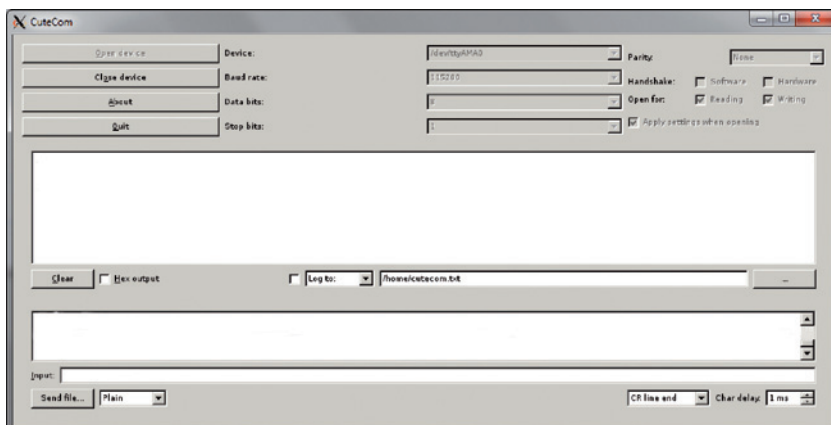
Fig. 12



- Attivare e disattivare un log;
- Aggiungere al comando caratteri di controllo come il fine linea;
- Trasmettere e/o ricevere i dati in formato ascii o esadecimale.

Infine al centro possiamo vedere l'area dove vengono evidenziati i messaggi in arrivo dalla porta seriale. La prima operazione da fare è configurare il nome di device (e relativo percorso) della porta seriale alla quale vogliamo connetterci, nel nostro caso:

Fig. 13



10^a FIERA REGIONALE DELL'ELETTRONICA SCANDIANO (REGGIO EMILIA) 26/27 OTTOBRE CENTRO FIERISTICO 2013



MERCATINO DELLE PULCI RADIOAMATORIALI
TELEFONIA - COMPONENTISTICA
COMPUTER - HI-FI CAR - RADIANTISMO
CB e OM - VIDEOREgistRAZIONE

ORARI

Sabato 26 Ottobre dalle 9 alle 18,30
Domenica 27 Ottobre dalle 9 alle 18



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

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

/dev/tty/AMA0

Se non la troviamo nella tendina dei valori preimpostati, la scriviamo direttamente nel campo di configurazione. Configuriamo la porta impostando 115200 per la velocità, 8 per la lunghezza dei caratteri, 1 per lo "Stop bit" e "none" per la parità. Non selezioniamo alcuna opzione per l'handshake. Spuntiamo invece entrambi i check box per l'apertura in lettura e scrittura. Lasciamo senza spunta i check box "Hex output" e "Log to". Nel campo di testo vicino a "Send file" selezioniamo "Plain", nel penultimo campo in basso a destra selezioniamo "CR line end" e nell'ultimo "1 ms". Bene, possiamo iniziare. Prima però assicuriamoci che il modulo SIM9XX abbia una SIMCard inserita, sia acceso e connesso alla rete mobile verificando che il LED lampeggi lentamente. In caso contrario lo accendiamo eseguendo il programma:

./GSMAccendi

Ora possiamo premere il pulsante "Open Device" in alto a sinistra nella finestra di CuteCom. Se tutto è in ordine vedremo diventare grigia l'area di configurazione in alto e potremo iniziare ad inviare comandi inserendoli nell'apposito campo in basso. Iniziamo con "AT", lo digitiamo nel campo apposito e premiamo il pulsante "Send file", dovremmo ricevere in risposta "OK", in caso contrario potrebbe essere che il modulo non si trovi in modo comandi. Attiviamo il "Command Mode" con il comando "+++".

Facendo riferimento al manuale per il modulo che stiamo utilizzando SIM900 o SIM908, possiamo provare l'effetto di ciascun comando, sia per la configurazione del modulo che per l'esecuzione di sequenze di comandi operativi.

Qualche esempio. Colleghiamo un altoparlante autoalimentato all'uscita audio ed un microfono all'ingresso MIC oppure utilizziamo una cuffia con microfono con gli spinotti separati. Proviamo ad eseguire e a ricevere una chiamata.

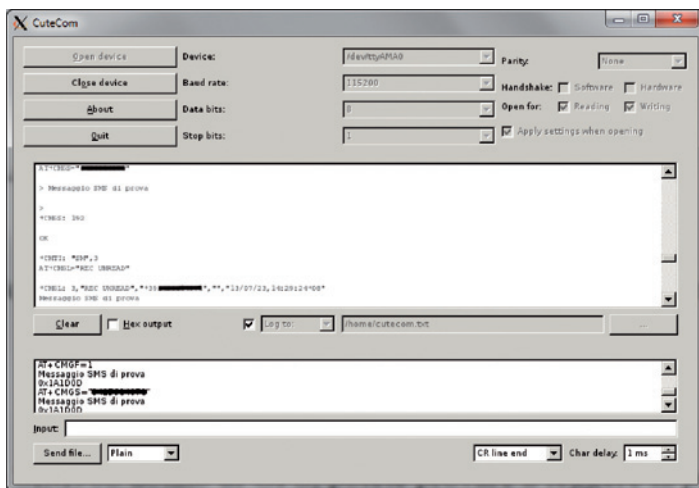
Prima diamo il comando:

AT+CALM=0

che permette di attivare la suoneria (è possibile anche impostare la musicchetta, consultate il manuale). Ora diamo il comando

ATD<numero_di_telefono> seguito da ";", ad esempio
ATD+393.....7;.

Fig. 14



Quando il telefono chiamato accetta la chiamata potremo parlare con il nostro interlocutore, per interrompere la chiamata dal lato nostro diamo il comando:

ATH

Se un telefono esterno ci chiama, sentiremo la suoneria nell'altoparlante e ad ogni squillo vedremo la scritta "RING" nell'area centrale. Per rispondere alla chiamata digitiamo il comando:

ATA

Per inviare un SMS digitiamo la sequenza di comandi:

AT+CMGF=1

AT+CMGS="<numero_di_telefono>"

ricordando di inserire il numero di telefono tra apici. Attendiamo il messaggio di ritorno

">"

Digitiamo il testo del messaggio, per esempio:

Messaggio SMS di prova

questa volta senza apici, infine selezioniamo, nel penultimo campo in basso a destra, "Hex input" e inseriamo:

0x1A1D0D e invio

(equivale a CTRL-Z per chiudere il messaggio SMS).

Per proseguire ricordiamo di riportare il valore del penultimo campo in basso a destra al valore "CR

line end". Per leggere la lista dei messaggi SMS ricevuti e non ancora letti:

AT+CMGL="REC UNREAD"

Otteniamo (Fig. 14):

+CMGL: 3,"REC UNREAD","+393.....1","",13/07/23,14:29:24+08"

Messaggio SMS di prova

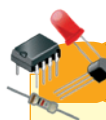
dove "3" è l'indice del messaggio. Possiamo leggere il messaggio singolo, utilizzando l'indice, anche con il comando:

AT+CMGR=3

Infine, se avete installato un modulo SIM908 dotato di GPS, potete provare a leggere la posizione (con l'antenna all'aperto) utilizzando il comando:

AT+CGPSINF=0

Otterrete come risultato una stringa composta dai campi mode, longitudine, latitudine, altezza, data/ora, TTFF (Time To First Fix), numero di satelliti, velocità e direzione di bussola. Potete sbizzarrirvi ad utilizzare i comandi nativi per prototypare le funzionalità che potrete poi codificare in programmi strutturati. Aiutatevi con gli esempi presenti negli articoli della serie "Gestire i moduli GMS con Arduino" nei numeri 175 e seguenti della rivista. Approfondiremo la realizzazione di applicazioni strutturate e modulari nei prossimi articoli. Per ora buon divertimento e ... occhio alla bolletta.



per il MATERIALE

Lo shield di espansione GSM/GPS/GPRS per Raspberry Pi è disponibile in scatola di montaggio al prezzo di Euro 13,90 (cod. FT1075K). Il kit non comprende il modulo GSM che può essere acquistato separatamente (cod. FT900M, Euro 49,00) nè l'antenna (cod. ANTGSMSTL-F01, Euro 8,00). È anche possibile utilizzare lo shield col modulo GSM/GPS Cod. FT971 (Euro 65,00) al quale vanno collegate le due antenne (ANTGPS-SMA, Euro 14,00 e ANTGSMSTL-S01, Euro 8,00) mediante due connettori Cod. CAVOUFLSMA, Euro 6,00 cad. Tutti i prezzi si intendono IVA compresa.

Il materiale va richiesto a:

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

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

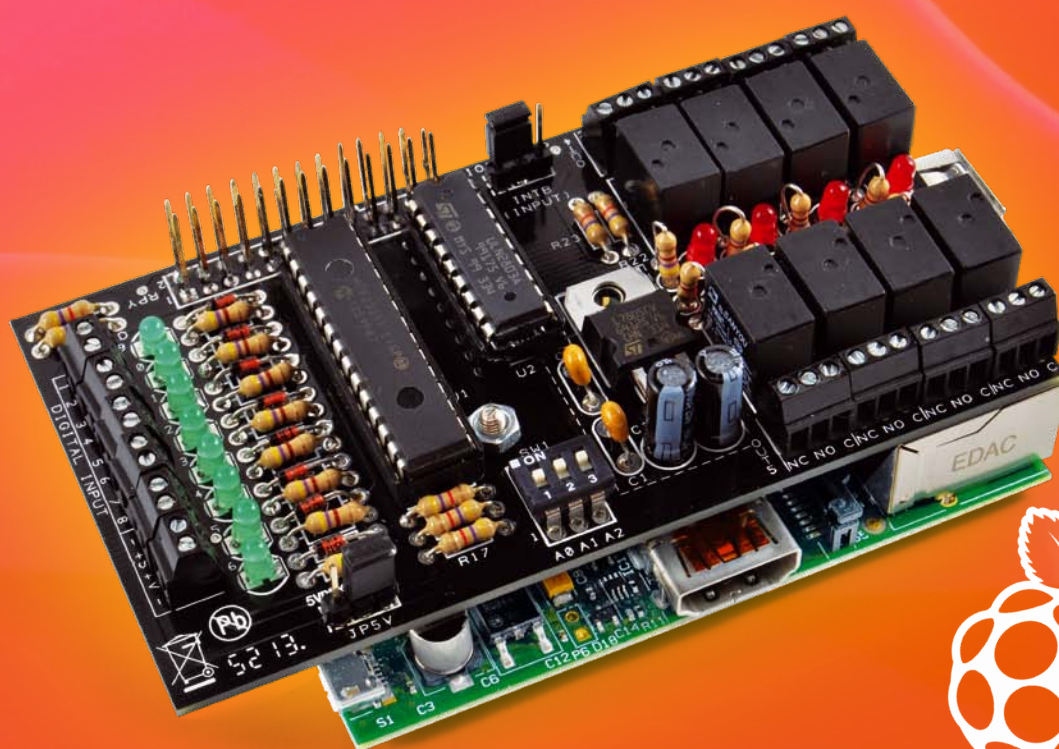
Elettronica In

www.elettronica.in.it

oltre l'elettronica

SHIELD I²C DI ESPANSIONE I/O PER RASPBERRY PI

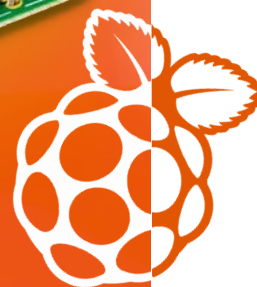
Comandiamo fino a 64 uscite a relé e leggiamo un massimo di 64 ingressi digitali impegnando due sole linee di Raspberry Pi: quelle del bus I²C.



Dopo gli shield di espansione per i display LCD e GSM presentati nei fascicoli, rispettivamente, n° 178 e n° 179, aggiungiamo alla collezione un nuovo shield, basato sull'integrato MCP23017 (lo stesso utilizzato nello shield del display LCD), che permette di aumentare il numero di ingressi/uscite digitali da mettere a disposizione

delle nostre applicazioni. Lo shield di espansione qui descritto rende disponibili otto ingressi e altrettante uscite digitali. Lo stato degli otto ingressi viene evidenziato da un LED per ciascuno. Ognuna delle otto uscite pilota un relé al quale possono essere collegati carichi esterni. Anche su ciascuna uscita è presente un LED, che ne rende visi-

bile lo stato. Per accedere agli ingressi/uscite dello shield si utilizza il bus I²C. L'indirizzo dello shield è configurabile mediante un DIP Switch a tre posizioni, così da permettere il collegamento e la gestione di un massimo di otto shield contemporaneamente, ciascuno con un indirizzo diverso; ciò consente di espandere il sistema fino a un totale di 64 ingressi e



di MARCO MAGAGNIN

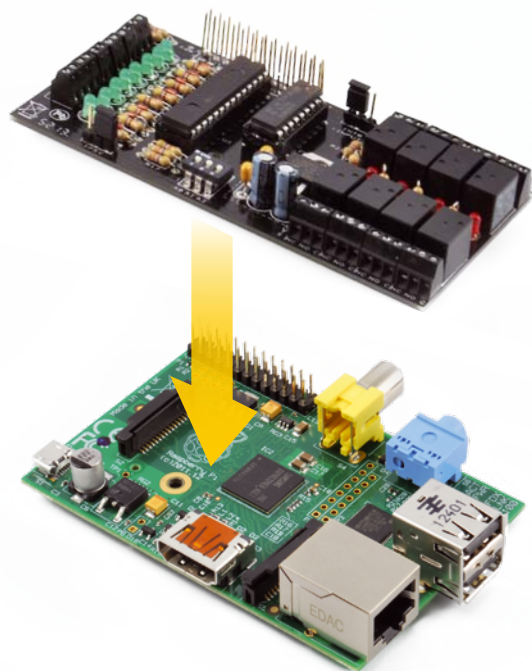
64 uscite pilotabili singolarmente. In questo articolo vi presentiamo anche la libreria Wiring-PI2, una libreria Open Source in licenza GNU LGPLv3 utilizzabile in Python, che permette di accedere ai singoli ingressi e pilotarli in modo decisamente semplice. Ricordiamo che lo shield di espansione presentato in questo articolo convive tranquillamente con gli altri shield che abbiamo presentato ed anche con alcuni di quelli di terze parti. Una ricchezza di possibilità di connessione che lascia spazio alla realizzazione di applicazioni in grado di gestire ambienti anche notevolmente complessi. La realizzazione di applicazioni di questo tipo richiede un salto di livello rispetto ai progetti che abbiamo presentato sino ad ora. Principalmente è necessario progettare e realizzare architetture software in grado di disaccoppiare l'utilizzo fisico delle periferiche esterne rispetto ai moduli di elaborazione veri e propri. In particolare, bisogna realizzare programmi in grado di reagire agli "eventi" esterni gene-

rati dagli ingressi e dai sensori ad essi collegati che, generalmente, si presentano in modo asincrono. È anche necessario progettare correttamente i programmi che gestiscono le periferiche che insistono sulla stessa risorsa di comunicazione. Per esempio, diverse applicazioni possono voler accedere a periferiche differenti, ma che utilizzano lo stesso bus I²C: è proprio il caso dello shield di espansione che presentiamo in questo articolo. Se un'applicazione tiene già impegnato il bus, per leggere o scrivere lo stato di un I/O di sua "competenza", le altre applicazioni che richiedono di impegnare lo stesso bus per gestire gli I/O di loro competenza, trovandolo occupato (busy), non potranno fare altro che andare in errore. La prima soluzione che potrebbe venire in mente è di realizzare un'unica enorme applicazione che gestisca tutte le funzionalità in modo monolitico, un po' come in un enorme microcontrollore. È intuitivo che questo modo di procedere comporta più aspetti negativi che positivi: rigidità dell'applicazione, difficoltà di progettazione, realizzazione, manutenzione e aggiornamento, difficoltà di temporizzazione tra le diverse parti, fragilità (se una parte, anche insignificante, va in errore, cade tutta l'applicazione). Per risolvere un'esigenza applicativa di questo tipo bisogna rivolgersi ad architetture in grado di supportare applicazioni concorrenti e collaboranti e di gestire le comunicazioni tra i diversi attori, basate su server che orchestrano l'utilizzo di risorse da condividere e dove le diverse funzionalità applicative devono essere realizzate come client che richiedono l'utilizzo delle risorse "centralizzate" mediante richieste basate su protocolli condivisi. Vi sono molte alternative possibi-

li: socket TCP/IP, funzionalità database, semplici file e così via. L'importante è che i meccanismi di comunicazione siano in grado di gestire correttamente il fenomeno, per esempio generando code o gestendo "semafori" e "blocchi" per coordinare correttamente le richieste dei client. I client, infine, quando hanno bisogno di accedere ad una delle risorse gestite centralmente, non vi devono accedere direttamente ma devono utilizzare il canale di comunicazione predefinito per quella risorsa, con il protocollo e le modalità previste. In sintesi, nel mondo embedded la progettazione non deve essere rivolta alla singola applicazione ma deve tenere conto dell'architettura che si vuole realizzare, nella sua totalità. Meglio o peggio rispetto ai microcontrollori? Questa domanda non ha una risposta definitiva. Dipende da quello che si vuole realizzare e dai requisiti associati. Sicuramente, in futuro, ci troveremo sempre più nella necessità di integrare e far collaborare questi due ambienti, insieme ad altri come FPGA e PLC. Dai prossimi numeri della rivista inizieremo ad affrontare questo tipo di problematiche, forse più attinenti alla disciplina "informatica" che a quella elettronica, ma ormai, come abbiamo sostenuto molte volte, dobbiamo fare i conti con la necessità, per chi realizza applicazioni basate su dispositivi elettronici destinati al mondo "embedded", di allargare le proprie competenze anche al mondo dell'informatica e della gestione delle reti di comunicazione.

SCHEMA ELETTRICO

Il circuito è progettato attorno all'integrato MCP23017, di Microchip, che offre 16 ingressi/uscite digitali pilotabili mediante il bus I²C. In questo modo, pos-

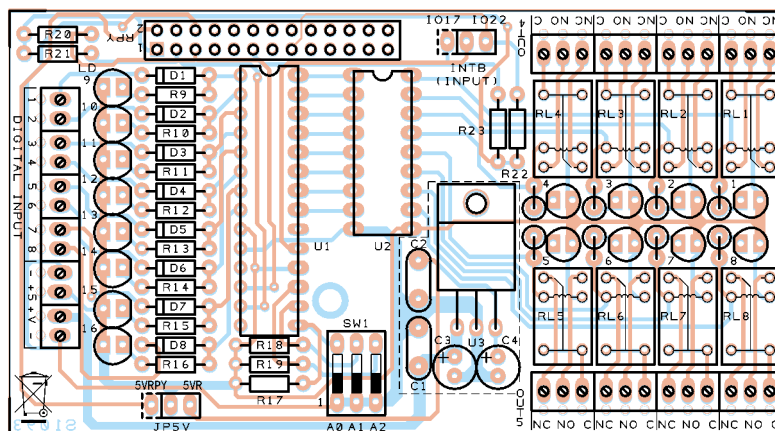
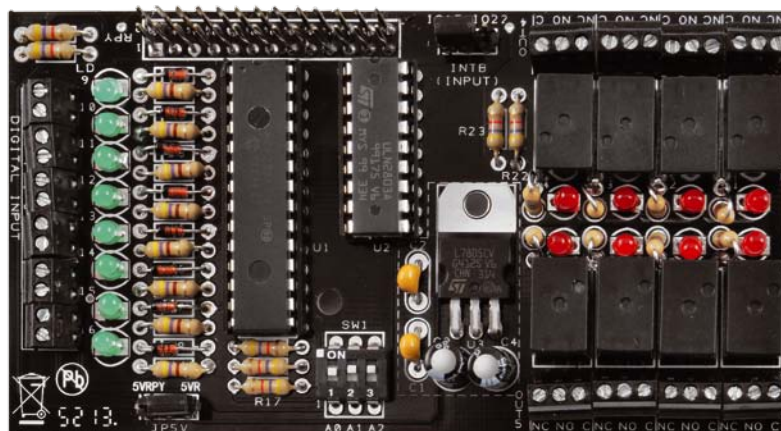


Elenco Componenti:

R1÷R16: 470 ohm
 R17÷R23: 4,7 kohm
 C1, C2: 100 nF multistrato
 C3, C4: 100 µF 16 VL elettrolitico
 D1÷D8: 1N4148
 LD1÷LD8: LED 3 mm rosso
 LD9÷LD16: LED 3 mm verde
 SW1: Dip-Switch 3 vie
 RL1÷RL8: Relé miniatura 5V
 U1: MCP23017-E/SP
 U2: ULN2803
 U3: 7805

Varie:

- Zoccolo 14+14
- Zoccolo 9+9
- Strip maschio 3 poli (2 pz.)
- Jumper (2 pz.)
- Connettore Femmina 2x13 cod. CTF/13+13
- Morsetto 2 poli passo 2,54 mm (6 pz.)
- Morsetto 3 poli passo 2,54 mm (8 pz.)
- Torretta M/F 18 mm
- Dado 3 MA
- Vite 8 mm 3 MA
- Circuito stampato



siamo pilotare gli otto ingressi e le otto uscite digitali del nostro shield impegnando i due soli pin del connettore di Raspberry Pi che fanno capo al bus I²C. Ovviamente il bus non viene utilizzato in modo esclusivo dal nostro shield, ma vi si possono connettere altre periferiche, purché configurate con indirizzi diversi. L'integrato MCP23017 (U1 nello schema elettrico), che abbiamo descritto approfonditamente nel numero 178 della rivista, mette a disposizione 16 linee di I/O suddivise in due banchi da otto, denominati GPA e GPB. Ciascuna linea dell'integrato è configurabile singolarmente come ingresso o come uscita. Nel nostro shield, il banco GPA è tutto

dedicato alla gestione delle uscite con i relé mentre il banco GPB è dedicato alla gestione degli ingressi digitali. Le linee di uscita digitale, provenienti dal banco GPA dell'integrato MCP23017, sono collegate ai pin di ingresso dell'integrato U2, un ULN2803, appositamente progettato per pilotare carichi induttivi, come le bobine dei relé.

All'interno dell'integrato, il cui schema è visibile in Fig. 1, sono presenti otto stadi a Darlington collegati con gli emettitori in comune (terminanti sul pin 9) e i collettori alle uscite, ciascuna delle quali può erogare fino a 500mA. Per inciso, anche se non è il nostro caso, per pilotare carichi che richiedono correnti

superiori ai 500mA, possono essere collegati in parallelo più stadi dell'integrato. All'interno del chip sono presenti anche i diodi di protezione, necessari per pilotare carichi induttivi. Applicazioni tipiche sono il controllo di motori, l'azionamento di relé, solenoidi, lampade, ecc. Gli ingressi 1-8, caratterizzati da un'alta impedenza di ingresso, possono essere pilotati direttamente dalle porte di I/O di un microcontrollore o di un integrato come l'MCP23017. Ma torniamo al nostro schema elettrico. Analizziamo una sola linea di uscita delle otto disponibili in quanto lo schema delle altre sette si ripete in modo identico. Quando, ad esempio, il pin GPA7 si trova

a livello logico alto (oltre i 2,5V fino a 5V), la base del transistor Darlington collegata all'ingresso 1 dell'integrato ULN2803, è polarizzata mediante il partitore resistivo RA, RB ed RC, visibile nello schema di **Fig. 1**. In questa condizione il transistor va in saturazione ed il suo collettore si porta ad un potenziale di poche centinaia di millivolt, lasciando fluire corrente nella bobina del relé. Lo stato di conduzione del transistor pilota anche l'accensione del LED LD1, polarizzato per mezzo della resistenza R1. Quando il pin GPA7 si trova a livello logico basso, il Darlington corrispondente si trova in stato di interdizione, e quindi il suo collettore, come un interruttore aperto, non permette passaggio di corrente nella bobina del relé. Anche il LED, non essendo alimentato, non si accende. Il diodo DA serve a sopprimere le extratensioni inverse generate dalla bobina del relé quando passa dallo stato eccitato allo stato di riposo, tensioni che potrebbero danneggiare il transistor. Sui morsetti a vite della scheda sono disponibili tutti i terminali del relé: il terminale centrale C, il terminale "normalmente chiuso" NC ed il "normalmente aperto" NO. La connessione ai terminali sui morsetti è visibile nel piano di montaggio. Come già anticipato, gli ingressi digitali fanno capo ai pin del banco GPB dell'integrato MCP23017. Anche in questo caso la descrizione di un modulo vale per tutti gli altri. Prendiamo l'ingresso digitale 1, che fa capo al al pin GPB0. La resistenza di pull-up R9, a riposo, mantiene l'ingresso a livello alto. Infatti è collegata al positivo della tensione di alimentazione tramite il LED LD9 che, in questo stato, rimane spento. Il diodo D1 serve a proteggere

l'ingresso dell'integrato, limitando a 5V la tensione massima che può giungere al pin d'ingresso. Infatti, applicando una tensione superiore, D1 non conduce più ed il pin GPB0 "vede" solamente i 5V dell'alimentazione, per effetto della presenza di R9/LD9. Questo accorgimento consente di fare funzionare il circuito con livelli logici, in ingresso, di 12 e più volt, salvaguardando sempre gli ingressi. Tutti gli ingressi digitali hanno la massa in comune, che corrisponde al negativo di alimentazione; negativo e positivo sono presenti anche su due contatti della morsettiera, in modo da avere facilmente a disposizione i 5V con i quali fornire alimentazione ai circuiti esterni. Lo shield richiede un'alimentazione esterna con una tensione compresa tra 9 e 12V, separata rispetto all'alimentazione di Raspberry Pi, in quanto l'uscita a 5V di quest'ultima non è in grado di fornire la corrente necessaria all'alimentazione dello shield. Il circuito integrato U2 (LM7805) provvede ad abbassare e a stabilizzare la tensione d'ingresso, quindi a fornire i 5V necessari alla scheda. Lo shield, invece, è in grado di alimentare Raspberry Pi. Se si desidera mantenere separate le alimentazioni dello shield e della scheda Raspberry Pi, il ponticello JP5V deve essere posizionato tra il pin centrale ed il pin 5VR. Per fare in modo che l'alimentatore dello shield alimenti anche la scheda Raspberry

Pi, il ponticello JP5V deve essere posizionato tra il pin centrale ed il pin 5VRPY. L'uscita INTB, che permette di gestire l'interrupt proveniente dagli ingressi del banco GPB può essere collegata, mediante il ponticello INTR, al GPIO22 o GPIO17 di Raspberry Pi, in modo da renderla disponibile ad eventuali applicazioni. Il Dip Switch collegato ai pin A0, A1 e A2 dell'integrato, permette di impostare l'indirizzo I²C da assegnare all'integrato. Nel piano di montaggio è possibile vedere la disposizione dei componenti. Unico suggerimento, a parte le usuali attenzioni, è di posizionare il doppio connettore a pettine in modo da distanziare lo stampato dello shield da quello di Raspberry Pi in modo da permetterne il montaggio "a

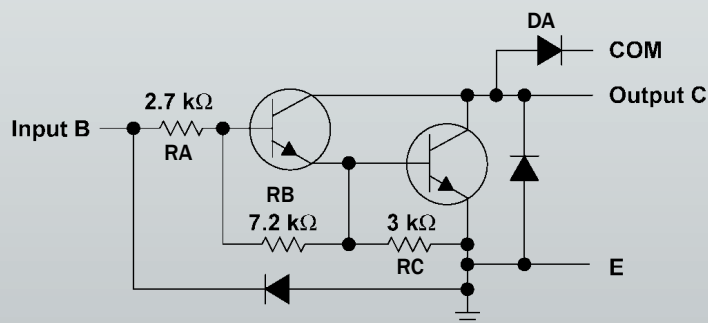
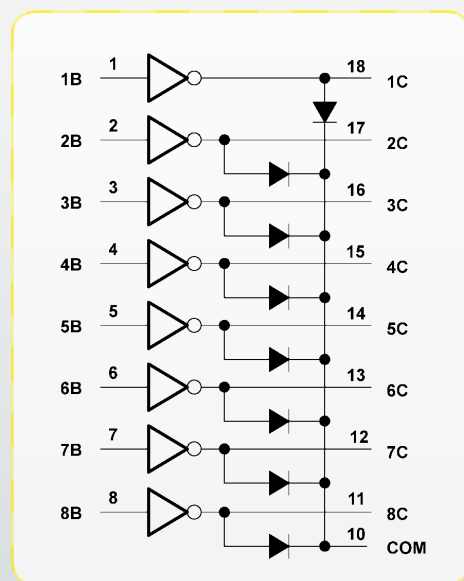
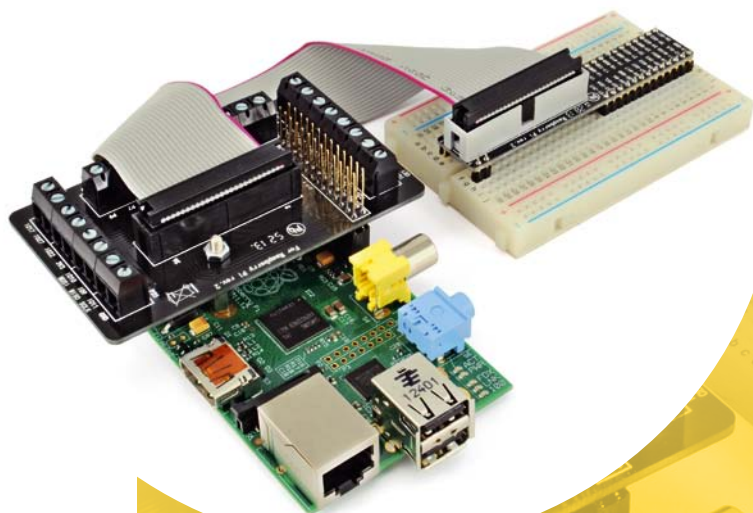


Fig. 1





Adattatore con morsettiera per Raspberry Pi

Lo shield adattatore per Raspberry Pi (FT1104) semplifica la vita a chi desidera sperimentare le possibilità del mondo embedded. È possibile realizzare circuiti o collegare periferiche esterne senza la necessità di realizzare circuiti stampati o di dover utilizzare il saldatore. Lo shield è dotato di morsettiera a 26 poli, pin strip M/F a passo 2,54 mm e connettore maschio a 26 pin, che consentono di portare all'esterno tutte le linee I/O presenti sul connettore a 26 pin della board Raspberry Pi. Il connettore passante a 26 poli dello shield permette di installare altri shield a "catasta" per realizzare complesse applicazioni multifunzione. Utilizzando il cavo piatto incluso nel kit dello shield e l'adattatore FT1072K, (non incluso) è possibile collegare in modo molto semplice Raspberry Pi ad una comune breadboard sulla quale è stata realizzata la propria applicazione. Il vantaggio principale di questa soluzione è la grande flessibilità di utilizzo non disgiunta dalla maggior sicurezza di utilizzo di GPIO, che rimangono protetti da rischi di cortocircuiti o contatti accidentali dei pin tra di loro o con altre parti metalliche di Raspberry Pi. Questi prodotti sono disponibili presso Futura Elettronica (www.futurashop.it).

Tutti gli altri pin di Raspberry Pi sono semplicemente passanti.

UTILIZZO PRATICO DELLO SHIELD

Collegiamo lo shield sulla scheda Raspberry Pi, facendo attenzione alla corrispondenza dei pin dei rispettivi connettori e verificando che la parte inferiore dello shield non vada in contatto con i connettori USB o Ethernet. In caso di dubbio, proteggiamo i connettori stessi con del nastro isolante. Colleghiamo un alimentatore da 9 Vcc ai morsetti di alimentazione dello shield (+V e -) e alimentiamo la scheda Raspberry Pi. Per comunicare con l'integrato MCP23017 è necessario utilizzare il bus I²C e di conseguenza dobbiamo attivare il modulo di gestione del bus stesso che, come ormai saprete, se ci seguite sulla rivista, nell'installazione predefinita di Raspbian, è disabilitato. Per chi ci segue costantemente, questa operazione non costituisce un problema; per chi è nuovo all'ambiente Raspbian, il sistema operativo GNU/Linux di Raspberry Pi, rimandiamo alle indicazioni riportate nel libro "Raspberry Pi il mio primo Linux embedded" (venduto da www.futurashop.it). Nell'articolo descriviamo solo le operazioni specifiche per utilizzare lo shield

pacchetto" nel caso lo si voglia utilizzare congiuntamente ad altri shield, come ulteriori shield di espansione o, per citarne uno, lo shield GSM che abbiamo presentato nel numero 179 della rivista. La possibilità di impilare più schede è data dal fatto che l'integrato MCP23017 è collegato al GPIO di Raspberry Pi unicamente con i pin necessari alla comunicazione I²C ed all'eventuale alimentazione.

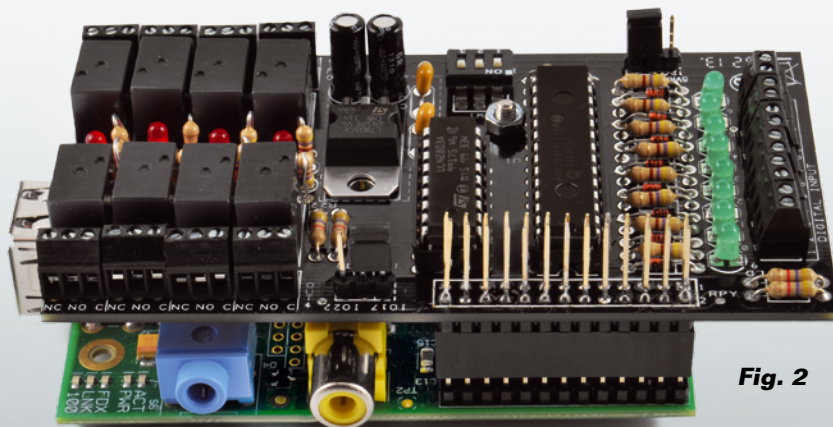


Fig. 2

di I/O. Ricordiamo che per gli articoli sulla rivista dedicati a Raspberry Pi abbiamo adottato il sistema operativo Raspbian (in costante aggiornamento e miglioramento) e gli strumenti di gestione remota, tramite protocollo SSH, utilizzando PuTTY (o KiTTY) e WINScp per la connessione. Dunque, posizioniamo tutti i DIP switch di indirizzamento dello shield su "ON" (GND), il jumper JP5V nella posizione 5VR (alimentazione separata), montiamo lo shield sul connettore di Raspberry Pi, eseguiamo tutti i collegamenti e diamo tensione. Niente fumo? Bene, siamo già a buon punto!

Ora dobbiamo abilitare il driver per la gestione del bus I²C, installare la libreria Python per la gestione dell'integrato MCP23017 e realizzare i primi programmi per accertare che tutto funzioni a dovere. Ricordiamo brevemente il processo per aggiornare il sistema operativo ed abilitare il driver per il bus I²C: innanzitutto utilizziamo PuTTY per aprire una finestra di terminale, ci logghiamo con l'utente "root" e digitiamo i comandi necessari ad aggiornare il database dei pacchetti del sistema operativo:

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

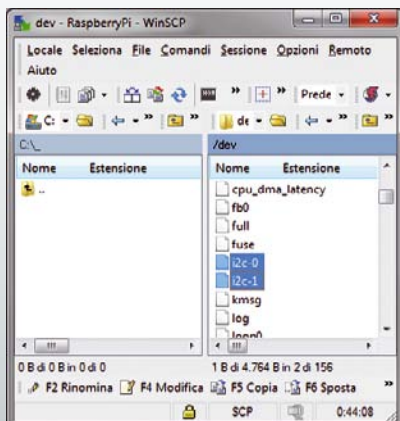


Fig. 7

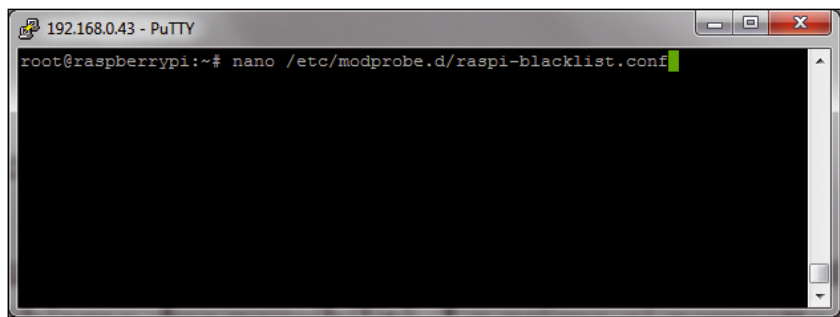


Fig. 3

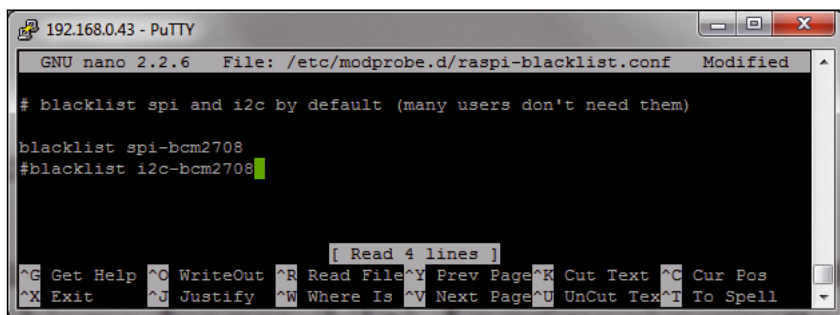


Fig. 4

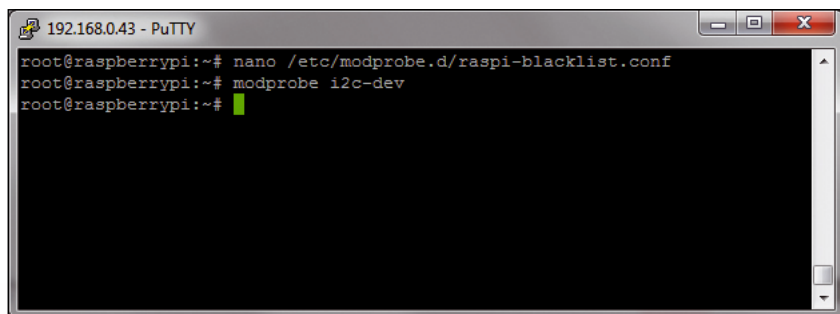


Fig. 5

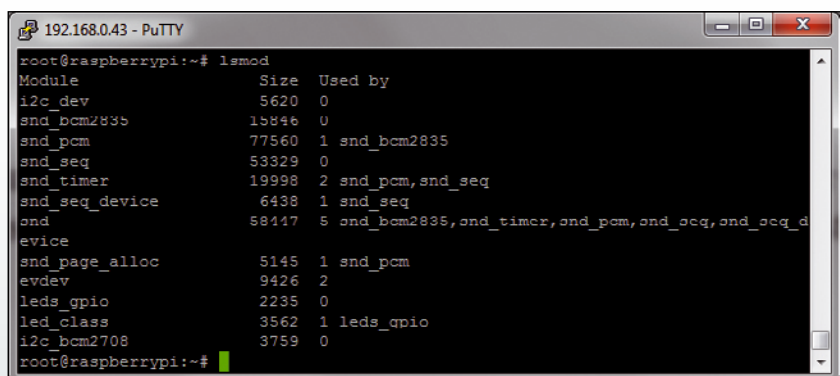


Fig. 6

Per ... resuscitare e rendere utilizzabile il modulo di gestione del bus I²C è necessario toglierlo dalla blacklist che lo tiene "prigioniero" e poi "aggiungerlo" all'insieme di moduli conosciuti dal kernel. Apriamo il file di con-

figurazione che contiene l'elenco dei moduli blacklisted (oscurati), con il comando (Fig. 3):

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

Nano è un editor di testo mini-

Fig. 8

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

Fig. 9

```
192.168.0.43 - PuTTY
GNU nano 2.2.6 File: /etc/modules Modified

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd_bcm2835
i2c-dev
```

Fig. 10

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

Fig. 11

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

Fig. 12

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

male che funziona in ambiente terminale.

Eliminiamo il modulo I²C dalla blacklist cancellando la riga o, come abbiamo preferito noi, commentandola con un “#” (Fig. 4). Premiamo Ctrl-x e poi y alla richiesta di salvare il file, dopo le modifiche.

Eseguiamo un reboot per rendere effettive le modifiche.

Ora dobbiamo fare in modo che il modulo “liberato” venga caricato e diventi parte integrante del kernel. Per questa operazione abbiamo due possibilità: la prima ci permette di caricare il modulo a comando, ed ha validità per tutto il tempo nel quale RaspberryPi rimane acceso. Al boot successivo, il modulo dovrà essere caricato nuovamente da comando.

La seconda possibilità ci permette di caricare il modulo, in automatico, al boot del sistema operativo, e renderlo subito disponibile alle applicazioni, condizione indispensabile in un sistema server unattended.

La prima possibilità richiede l'utilizzo del comando `modprobe`. Digitiamo (Fig. 5):

```
modprobe i2c-dev
```

Possiamo vedere il buon esito dell'attivazione dei driver con il comando che mostra la lista di tutti i moduli installati (Fig. 6):

```
lsmod
```

Dato che in Linux tutto (o quasi) è un file se andiamo nella cartella /dev vediamo apparire i file di collegamento ai device i2c-0 ed i2c-1 (Fig. 7).

Il comando `modprobe` permette di caricare e di scaricare i moduli a run time e mantiene i suoi effetti fintanto che RaspberryPi è acceso. In caso di spegnimento, o anche solo di reboot, i moduli

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

`modprobe -r i2c-dev`

Se si desidera che il modulo `i2c-dev` venga caricato all'accensione di RaspberryPi, è necessario abilitare il caricamento permanente, che si realizza modificando il file di configurazione `/etc/modules`, che contiene la lista dei driver da caricare al momento del boot. In caso contrario dobbiamo ricordarci di caricare il modulo ad ogni accensione con `modprobe`. Per modificare il file possiamo usare il comando (Fig. 8):

`nano /etc/modules`

ed aggiungere una nuova linea, al file di configurazione, che contenga il comando (Fig. 9):

`i2c-dev`

Premete CTRL-x e poi y per salvare le modifiche al file ed uscire. Ora installiamo il pacchetto `i2c-tools`, che ci fornisce una serie di funzioni utilizzabili a linea di comando per verificare il funzionamento del bus I²C, non prima, come sempre, di avere aggiornato la lista dei pacchetti della distribuzione con il comando:

`apt-get update`

Proseguiamo con il comando (Fig. 10):

`apt-get install i2c-tools`
Aggiungiamo il nostro utente pi

al gruppo I²C (Fig. 11), con:

`adduser pi i2c`

Per attivare le nuove configurazioni, eseguiamo il reboot di Raspberry Pi tramite il comando:

`reboot`

Dopo che RaspberryPi si è riavviata, vi siete riconnessi con PuTTY; impartito, se necessario, il comando `modprobe i2c-dev`, verificate se l'integrato è visibile sul bus I²C, tramite il comando:

`i2cdetect -y 0` per RaspberryPi rev. 1

oppure con il comando:

`i2cdetect -y 1` per RaspberryPi rev. 2

Dovrete ottenere un risultato simile a quello visibile in Fig. 12, dove l'indirizzo 0x20 identifica l'integrato MCP23017.

Ora è tempo di installare la libreria per la gestione dell'integrato MCP23017. Abbiamo optato per la libreria `wiringpi2`, un eccellente sistema di gestione del GPIO di Raspberry Pi scritta in C da Gordon "Drogon" Henderson e "adattata" all'utilizzo con il linguaggio Python da Phil "Gadgetoid" Howard. Questa libreria, distribuita con licenza GNU LGPLv3, include diversi moduli, finalizzati all'utilizzo di specifici integrati che utilizzano i bus I²C e SPI: uno di questi è l'MCP23017, utilizzato nel nostro shield.

```

192.168.0.43 - PuTTY
root@raspberrypi:~# python
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import wiringpi2
>>> wiringpi2.piBoardRev()
2
>>> exit()
root@raspberrypi:~#

```

Fig. 13

L'approccio adottato nella libreria per l'utilizzo con l'MCP23017 è di assegnare un riferimento numerico a ciascun pin di ingresso/uscita dell'integrato. La numerazione parte da un numero definito a piacere, rendendo possibile utilizzare più di un integrato su una stessa scheda, ciascuno con i pin riconoscibili da un diverso intervallo di numeri. Come vedremo tra poco, noi abbiamo assegnato un intervallo di numeri ai pin dell'integrato che prendono i valori da 65, per il pin GPA0, a 80 per il pin GPB7.

Prima di descrivere qualche esempio di utilizzo della libreria, procediamo ad installarla. La libreria `wiringpi2` è disponibile come "package" aggiuntivo al linguaggio Python. Per gestire i package di Python abbiamo bisogno del gestore di pacchetti Python "pip". Pip è un acronimo ricorsivo che sta per "Pip Installs Python". Quindi in sequenza installiamo il pacchetto "pip" e poi, tramite questo, la libreria "wiringpi2". Aggiorniamo di nuovo la lista dei pacchetti della distribuzione con il comando:

`apt-get update`

Installiamo le dipendenze richieste dal pacchetto pip, con:

`apt-get install python-dev`

ed infine il pacchetto pip stesso:
`apt-get install python-pip`

Listato 1

```
#!/usr/bin/python

# rele_test.py

import wiringpi2 as wiringpi # importa libreria wiringpi2
from time import sleep

pin_base = 65          # imposta la numerazione dei pin a partire da 65
i2c_addr = 0x20        # indirizzo ottenuto con i pin A0, A1, A2 a GND

wiringpi.wiringPiSetup()          # inizializza libreria wiringpi2
wiringpi.mcp23017Setup(pin_base,i2c_addr) # imposta pin e indirizzo i2c
wiringpi.pinMode(65, 1)          # imposta GPA0 come output
wiringpi.digitalWrite(65, 0)     # imposta GPA0 a 0 (0V, off)
wiringpi.pinMode(66, 1)          # imposta GPA1 come output
wiringpi.digitalWrite(66, 0)     # imposta GPA1 a 0 (0V, off)
wiringpi.pinMode(67, 1)          # imposta GPA2 come output
wiringpi.digitalWrite(67, 0)     # imposta GPA2 a 0 (0V, off)
wiringpi.pinMode(68, 1)          # imposta GPA3 come output
wiringpi.digitalWrite(68, 0)     # imposta GPA3 a 0 (0V, off)
wiringpi.pinMode(69, 1)          # imposta GPA4 come output
wiringpi.digitalWrite(69, 0)     # imposta GPA4 a 0 (0V, off)
wiringpi.pinMode(70, 1)          # imposta GPA5 come output
wiringpi.digitalWrite(70, 0)     # imposta GPA5 a 0 (0V, off)
wiringpi.pinMode(71, 1)          # imposta GPA6 come output
wiringpi.digitalWrite(71, 0)     # imposta GPA6 a 0 (0V, off)
wiringpi.pinMode(72, 1)          # imposta GPA7 come output
wiringpi.digitalWrite(72, 0)     # imposta GPA7 a 0 (0V, off)

wiringpi.pinMode(73, 0)          # imposta GPB0 come input
wiringpi.pullUpDnControl(73, 0) # pull-up e pull-down interni disattivati
wiringpi.pinMode(74, 0)          # imposta GPB1 come input
wiringpi.pullUpDnControl(74, 0) # pull-up e pull-down interni disattivati
wiringpi.pinMode(75, 0)          # imposta GPB2 come input
wiringpi.pullUpDnControl(75, 0) # pull-up e pull-down interni disattivati
wiringpi.pinMode(76, 0)          # imposta GPB3 come input
wiringpi.pullUpDnControl(76, 0) # pull-up e pull-down interni disattivati
wiringpi.pinMode(77, 0)          # imposta GPB4 come input
wiringpi.pullUpDnControl(77, 0) # pull-up e pull-down interni disattivati
wiringpi.pinMode(78, 0)          # imposta GPB5 come input
wiringpi.pullUpDnControl(78, 0) # pull-up e pull-down interni disattivati
wiringpi.pinMode(79, 0)          # imposta GPB6 come input
wiringpi.pullUpDnControl(79, 0) # pull-up e pull-down interni disattivati
wiringpi.pinMode(80, 0)          # imposta GPB7 come input
wiringpi.pullUpDnControl(80, 0) # pull-up e pull-down interni disattivati

try:
    while True:
        if not wiringpi.digitalRead(73): # logica invertita (pull-up esterno)
            wiringpi.digitalWrite(65, 1) # imposta GPA1 a 1 (3V3, on)
        else:
            wiringpi.digitalWrite(65, 0) # imposta GPA1 a 0 (0V, off)

        if not wiringpi.digitalRead(74):
            wiringpi.digitalWrite(66, 1)
        else:
            wiringpi.digitalWrite(66, 0)

        if not wiringpi.digitalRead(75):
            wiringpi.digitalWrite(67, 1)
        else:
            wiringpi.digitalWrite(67, 0)

        if not wiringpi.digitalRead(76):
            wiringpi.digitalWrite(68, 1)
        else:
            wiringpi.digitalWrite(68, 0)

        if not wiringpi.digitalRead(77):
            wiringpi.digitalWrite(69, 1)
        else:
            wiringpi.digitalWrite(69, 0)

        if not wiringpi.digitalRead(78):
            wiringpi.digitalWrite(70, 1)
        else:
```

(Continua)

Ora, utilizzando il gestore di pacchetti pip possiamo installare la libreria vera e propria:

pip install wiringpi2

Terminato questo processo, possiamo verificare immediatamente se tutto ha funzionato a dovere. Apriamo l'interprete Python a linea di comando, digitando semplicemente "python" al prompt della finestra di terminale, e verifichiamo la corretta installazione della libreria wiringpi2 digitando le istruzioni (**Fig. 13**):

```
import wiringpi2
wiringpi2.piBoardRev()
```

La risposta, contenente il numero di versione della libreria, ci conferma che tutto funziona correttamente.

Ora utilizziamo la libreria per realizzare qualche programma di esempio. Il primo programma, visibile nel **Listato 1**, permette di agganciare ciascun pin di ingresso del banco GPB dell'integrato, al corrispondente pin di output del banco GPA. I commenti non dovrebbero lasciare dubbi sulla logica del programma, quindi descriviamo unicamente le istruzioni "try: - except:", introdotte per la prima volta nei nostri articoli. Queste istruzioni permettono di "intrappolare" gli errori. Se includiamo un insieme di istruzioni all'interno di un gruppo "try:" (prova), in caso di errore il programma, invece di interrompere l'esecuzione, intercetta l'errore e innesca le istruzioni contenute all'interno del gruppo "except:". In questo gruppo, nel nostro caso, abbiamo inserito le istruzioni necessarie a riportare tutti i pin dell'integrato configurati come input. Questo meccanismo funziona anche quando si interrompe il loop principale del

Listato 1 – segue

```
wiringpi.digitalWrite(70, 0)

if not wiringpi.digitalRead(79):
    wiringpi.digitalWrite(71, 1)
else:
    wiringpi.digitalWrite(71, 0)

if not wiringpi.digitalRead(80):
    wiringpi.digitalWrite(72, 1)
else:
    wiringpi.digitalWrite(72, 0)

sleep(0.05)

except:
    wiringpi.digitalWrite(65, 0) # imposta GPA1 a 0 (0V, off)
    wiringpi.pinMode(65, 0)      # reimposta GPA1 come input
    wiringpi.digitalWrite(66, 0)
    wiringpi.pinMode(66, 0)
    wiringpi.digitalWrite(67, 0)
    wiringpi.pinMode(67, 0)
    wiringpi.digitalWrite(68, 0)
    wiringpi.pinMode(68, 0)
    wiringpi.digitalWrite(69, 0)
    wiringpi.pinMode(69, 0)
    wiringpi.digitalWrite(70, 0)
    wiringpi.pinMode(70, 0)
    wiringpi.digitalWrite(71, 0)
    wiringpi.pinMode(71, 0)
    wiringpi.digitalWrite(72, 0)
    wiringpi.pinMode(72, 0)
    print " "
    print "fine"
```

programma con CTRL-c. Per provare il programma possiamo collegare dei pulsanti tra la massa e ciascun pin di ingresso dello shield; in alternativa possiamo collegare un terminale di un cavetto jumper al morsetto di massa e con l'altro terminale toccare i connettori dei morsetti di ingresso. Con lo strumento WINScp, copiamo il programma del **Listato 1** in una cartella di Raspberry Pi all'interno della cartella `/home`: per esempio, `/home/Rele`. Diamo un nome al programma: per esempio "rele-test.py". Posizioniamoci nella cartella `/home/Rele` e lanciamo il programma con il comando:

```
python rele-test.py
```

Ora, portando a massa (con i pulsanti o con il jumper) gli ingressi dello shield, vedremo accendersi i LED e chiudersi i relé delle corrispondenti uscite, per tutto il tempo durante il quale l'ingresso rimarrà collegato a massa. Per terminare il programma, diamo il comando CTRL-c (**Fig. 14**). Come abbiamo descritto nella parte introduttiva dell'articolo, questo programma, quando funziona, tiene impegnato costantemente il bus I²C, impedendo ad eventuali altri programmi di accedervi. Per superare tale problema è necessario disaccoppiare l'utilizzo del bus rispetto ai programmi.

Un metodo è quello di scrivere programmi che impegnino il bus unicamente per il tempo necessario ad eseguire le azioni richieste, e poi lo liberino per lasciare il campo ad altri richiedenti. Vi forniamo un esempio di questa soluzione nel **Listato 2** e nel **Listato 3**. La soluzione proposta è composta da un programma, da lanciare una sola volta all'accensione di Raspberry Pi, per inizializzare

l'integrato MCP23017 e configurare i pin nel modo richiesto dallo shield. Il secondo programma viene lanciato ogni volta che si desidera modificare lo stato di un pin di uscita. Nel nostro esempio, lo faremo lanciare in momenti temporali definiti utilizzando lo schedulatore cron. In questo modo potremo realizzare un impianto temporizzato come un sistema di irrigazione o un sistema di controllo luci giorno/notte. Il programma del **Listato 2** è derivato dalla parte di configurazione iniziale del programma

mostrato nel **Listato 1**. Anche il programma presentato nel **Listato 3** è derivato da frammenti di codice del **Listato 1** con in più la possibilità di ricevere parametri dall'esterno.

Nel nostro caso ne abbiamo previsti due: il nome del pin del quale vogliamo modificare lo stato, nella forma Ax ed il valore dello stato che desideriamo impostare: "ON" oppure "OFF". Il formato con il quale lanceremo il comando sarà quindi:

```
python rele_pin_set.py A1 ON
```

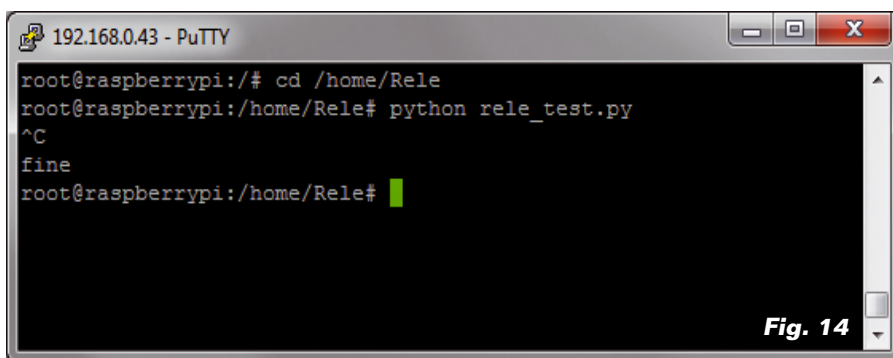


Fig. 14

Listato 2

```
#!/usr/bin/python

# rele_set.py

import wiringpi2 as wiringpi # importa libreria wiringpi2

from time import sleep
import sys

pin_base = 65      # imposta la numerazione dei pin a partire da 65
i2c_addr = 0x20    # indirizzo ottenuto con i pin A0, A1, A2 a GND

try:
    wiringpi.wiringPiSetup()                # inizializza libreria wiringpi2
    wiringpi.mcp23017Setup(pin_base,i2c_addr) # imposta pin e indirizzo i2c
    wiringpi.pinMode(65, 1)                 # imposta GPA0 come output
    wiringpi.digitalWrite(65, 0)            # imposta GPA0 a 0 (0V, off)
    wiringpi.pinMode(66, 1)                 # imposta GPA1 come output
    wiringpi.digitalWrite(66, 0)            # imposta GPA1 a 0 (0V, off)
    wiringpi.pinMode(67, 1)                 # imposta GPA2 come output
    wiringpi.digitalWrite(67, 0)            # imposta GPA2 a 0 (0V, off)
    wiringpi.pinMode(68, 1)                 # imposta GPA3 come output
    wiringpi.digitalWrite(68, 0)            # imposta GPA3 a 0 (0V, off)
    wiringpi.pinMode(69, 1)                 # imposta GPA4 come output
    wiringpi.digitalWrite(69, 0)            # imposta GPA4 a 0 (0V, off)
    wiringpi.pinMode(70, 1)                 # imposta GPA5 come output
    wiringpi.digitalWrite(70, 0)            # imposta GPA5 a 0 (0V, off)
    wiringpi.pinMode(71, 1)                 # imposta GPA6 come output
    wiringpi.digitalWrite(71, 0)            # imposta GPA6 a 0 (0V, off)
    wiringpi.pinMode(72, 1)                 # imposta GPA7 come output
    wiringpi.digitalWrite(72, 0)            # imposta GPA7 a 0 (0V, off)

    wiringpi.pinMode(73, 0)                 # imposta GPB0 come input
    wiringpi.pullUpDnControl(73, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(74, 0)                 # imposta GPB1 come input
    wiringpi.pullUpDnControl(74, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(75, 0)                 # imposta GPB2 come input
    wiringpi.pullUpDnControl(75, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(76, 0)                 # imposta GPB3 come input
    wiringpi.pullUpDnControl(76, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(77, 0)                 # imposta GPB4 come input
    wiringpi.pullUpDnControl(77, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(78, 0)                 # imposta GPB5 come input
    wiringpi.pullUpDnControl(78, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(79, 0)                 # imposta GPB6 come input
    wiringpi.pullUpDnControl(79, 0)          # pull-up e pull-down interni disattivati
    wiringpi.pinMode(80, 0)                 # imposta GPB7 come input
    wiringpi.pullUpDnControl(80, 0)          # pull-up e pull-down interni disattivati

except Exception, e:
    print e
```

Per leggere i parametri all'interno del programma si utilizza il metodo "argv" del modulo "sys". In Python, il modulo "sys" rende disponibili una serie di funzioni e di variabili che permettono di interagire con l'ambiente runtime all'interno del quale viene eseguito il programma Python. In particolare, "argv" contiene la lista degli argomenti passati al programma al momento del lancio. I parametri sono resi disponibili in forma di array dove l'elemento argv[0] contiene il nome del programma e i successivi argv[i]

contengono gli argomenti, nell'ordine nel quale sono stati digitati a linea di comando.

Nel nostro caso avremo:

argv[0] = "rele_pin_set"

argv[1] = "A1"

argv[2] = "ON"

I parametri acquisiti vengono utilizzati per personalizzare le istruzioni di configurazione del pin. Dato che abbiamo assegnato al primo pin il valore 65, per ottenere la corrispondenza tra il parametro passato al programma, ed il corrispondente numero gestito dalla libreria, utilizziamo l'istruzione:

```
Pin_num = int(Pin[1:]) + 65
```

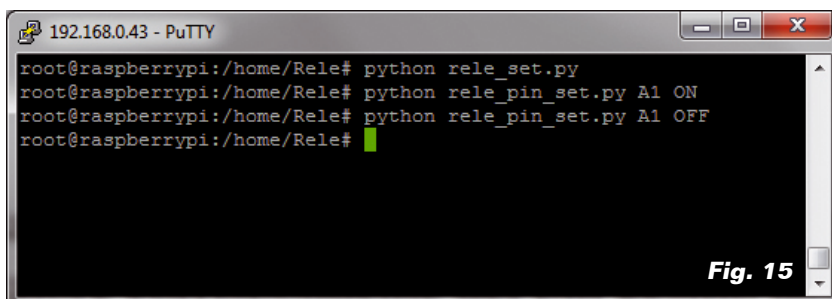
Lo scopo di questa istruzione è di eliminare dal primo parametro la "A" iniziale, e sommare 65 al valore numerico risultante per ottenere il valore corrispondente del pin riconosciuto dalla libreria. Lanciamo manualmente il primo programma per configurare inizialmente l'integrato MCP23017; lanciamo poi il secondo programma, variando i parametri e verifichiamo i risultati che otteniamo (Fig. 15).

Passiamo ora a configurare lo schedatore "cron".

Lo schedatore "cron" è il servizio predefinito, in Linux, per l'esecuzione a tempo di programmi e comandi.

Per funzionare, cron si appoggia a una tabella che contiene le "direttive" sui programmi da eseguire e definisce in quali istanti lanciarli. La tabella utilizzata da cron si gestisce con la funzionalità crontab, la cui sintassi è la seguente:

crontab <opzione>



```
192.168.0.43 - PuTTY
root@raspberrypi:/home/Rele# python rele_set.py
root@raspberrypi:/home/Rele# python rele_pin_set.py A1 ON
root@raspberrypi:/home/Rele# python rele_pin_set.py A1 OFF
root@raspberrypi:/home/Rele#
```

Fig. 15

dove opzione può assumere i valori *-e*, *-l* e *-r*.
Il comando:

crontab -e

ci permette di aprire il file di configurazione predefinito di *crontab*, come si vede in **Fig. 16**. Ogni riga del file di configurazione ha la seguente struttura:

```

* * * * * comando da eseguire
-----
| | | | |
| | | | +---- giorno della settimana (0-6) (0=Domenica)
| | | +----- mese (1-12)
| | +----- - giorno del mese (1-31)
| +----- - ora (0-23)
+----- - min (0-59)

```

Ciascun parametro può essere configurato come serie di valori (ad esempio 0, 3, 7, 9) come un intervallo (ad esempio 3-7) oppure una combinazione dei due (esempio: 0,3,4-6,9) o, ancora, come frequenza **/5* (ogni 5 minuti, ore, ecc.). Nel nostro caso vogliamo che ad ogni minuto pari venga eseguito il programma *rele_pin_set.py*, con i parametri impostati in modo da far chiudere il relé collegato al pin GPBA1 e ad ogni minuto dispari venga rieseguito lo stesso programma con i parametri impostati per far aprire il relé. Nella finestra che ci mostra il contenuto del file, aggiungiamo in fondo le righe di direttiva come in **Fig. 16**:

```
0-58/2 * * * * python /home/Rele/rele_pin_set.py A1 ON > /dev/null
```

che significa far eseguire il programma ad ogni minuto pari di ogni ora, giorno e mese, e:

```
1-59/2 * * * * python /home/Rele/rele_pin_set.py A1 OFF > /dev/null
```

Usciamo dal file con <CTRL>X, rispondiamo "y" alla richiesta di salvataggio e confermiamo il nome del file con "Invio". D'ora in avanti, ogni minuto verrà lanciato il programma che, in

Listato 3

```

#!/usr/bin/python

# rele_pin_set.py

import wiringpi2 as wiringpi
from time import sleep
import sys

Pin = sys.argv[1]
Stato = sys.argv[2]

pin_base = 65      # imposta la numerazione dei pin a partire da 65
i2c_addr = 0x20     # indirizzo ottenuto con i pin A0, A1, A2 a GND

Pin_num = int(Pin[1:]) + 65

try:
    wiringpi.wiringPiSetup()                # inizializza libreria wiringpi
    wiringpi.mcp23017Setup(pin_base,i2c_addr) # imposta pin e indirizzo i2c
    if Stato == "ON":
        wiringpi.digitalWrite(Pin_num, 1) # imposta il pin a 1 (3V3, on)
    else:
        wiringpi.digitalWrite(Pin_num, 0) # simposta il pin a 0 (0V, off)
except Exception, e:
    print e

```

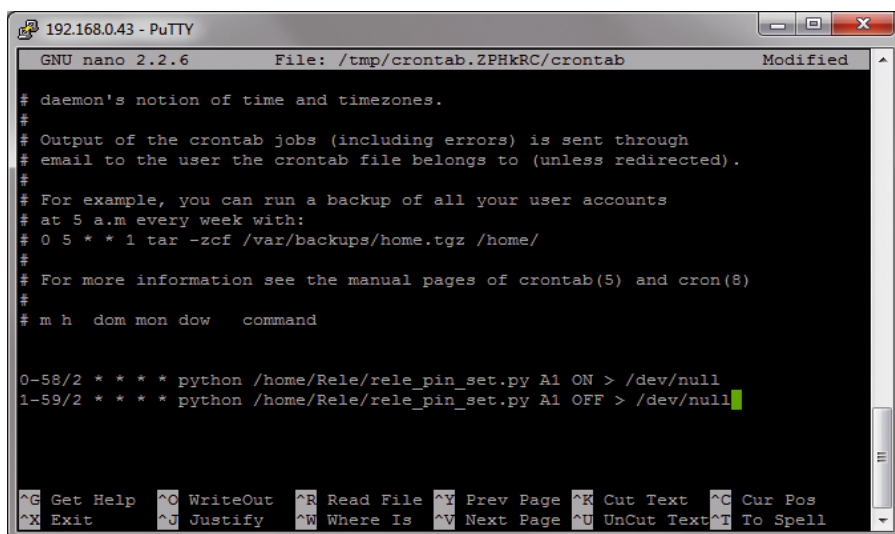


Fig. 16

alternativa, attiverà e disattiverà il relé. Per modificare le impostazioni di cron, richiamiamo la funzionalità:

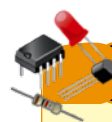
crontab -e

Per vedere le impostazioni utilizziamo l'opzione:

crontab -l

mentre per interrompere l'esecuzione automatica digitiamo

crontab -r



per il MATERIALE

Tutti i componenti utilizzati in questo progetto sono di facile reperibilità e il master del PCB può essere scaricato gratuitamente dal sito della rivista. Il circuito è anche disponibile in kit (cod. FT1093K) al prezzo di 32,00 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>

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

Elettronica In

www.elettronica.in.it

oltre l'elettronica

INTERNET RADIO CON RASPBERRY PI E SHIELD LCD

di MARCO MAGAGNIN

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

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

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



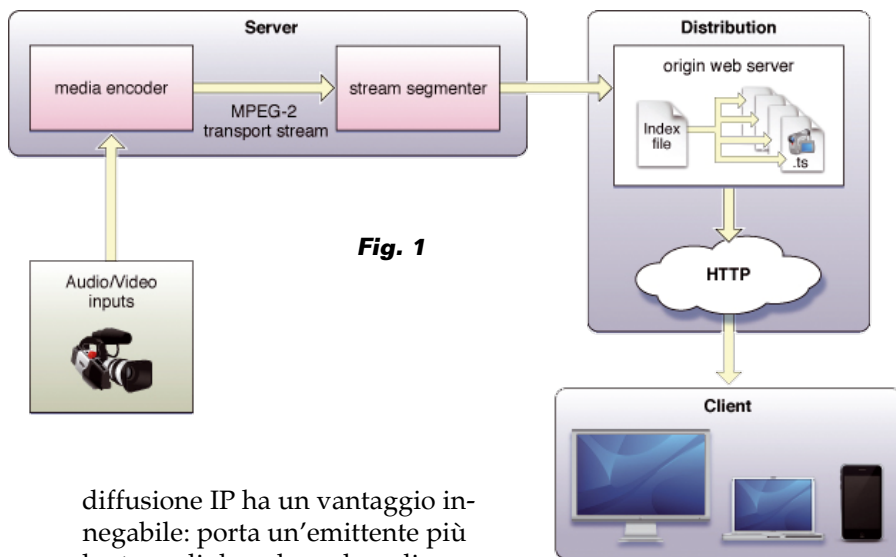


Fig. 1

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

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

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

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

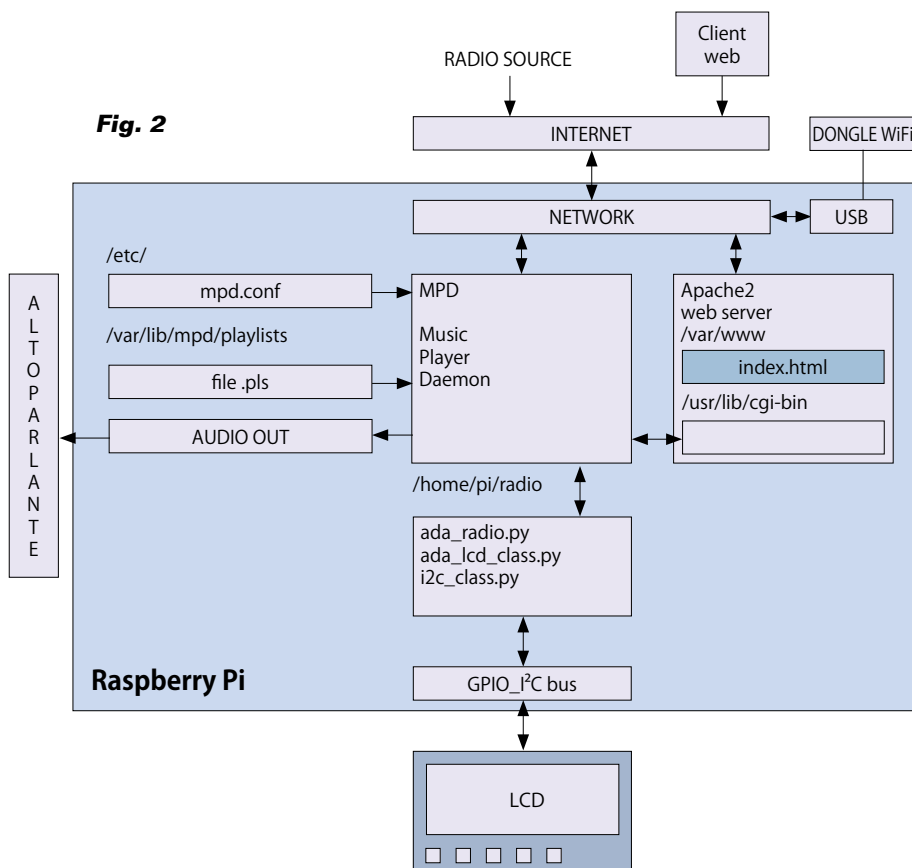


Fig. 2

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

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

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

PREPARAZIONE RASPBERRY PI

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

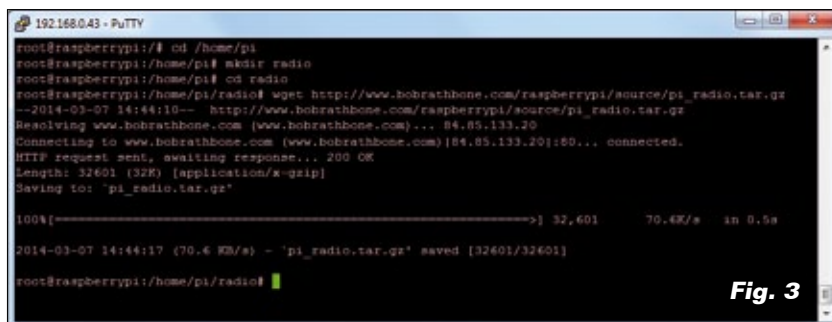


Fig. 3

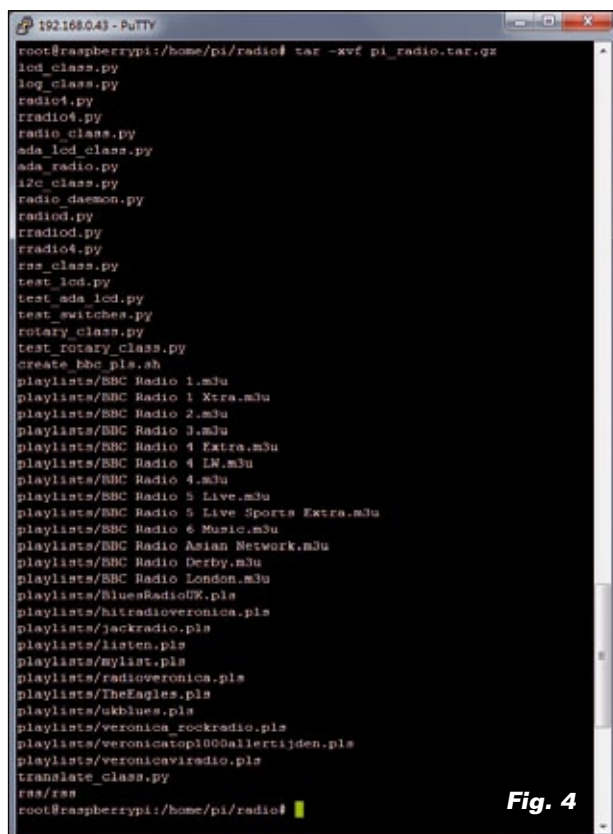


Fig. 4

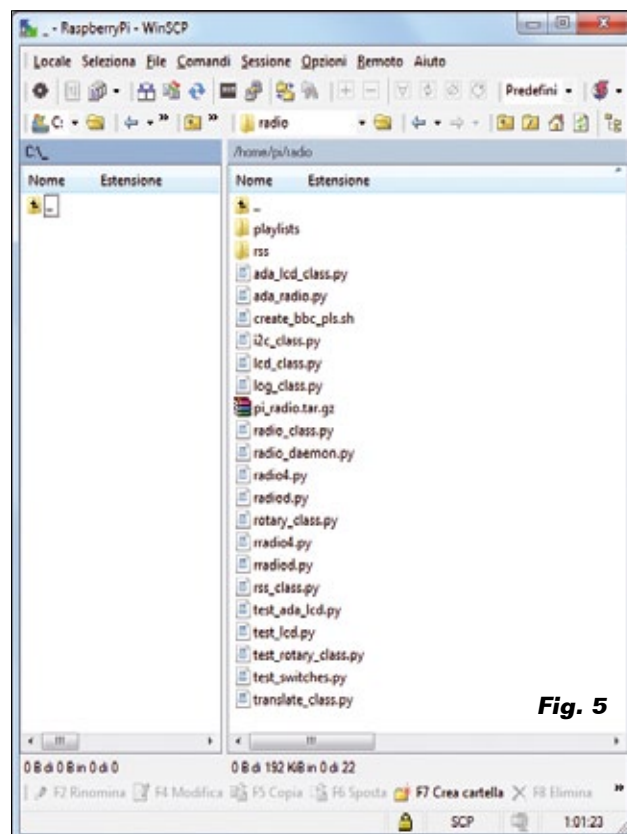


Fig. 5

```

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

```

Fig. 6

```

192.168.0.43 - PuTTY
root@raspberrypi:/home/pi/radio# apt-get install mpd mpc
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libao-common libao4 libaudiofile1 libavcodec53 libavformat53
  libavutil51 libdirac-encoder0 libfaad2 libgsm1 liblame0 libmp3lame0
  libmpeg6 libmpdclient2 libschroedinger-1.0-0 libshout3 libspeex1
  libtheora0 libva1 libvpx1 libwavpack1 libx264-123 libxvidcore4
Suggested packages:
  libeas0 libeas-alsa0 speex avahi-daemon icecast2 mpd-client pulseaudio
The following NEW packages will be installed:
  libao-common libao4 libaudiofile1 libavcodec53 libavformat53
  libavutil51 libdirac-encoder0 libfaad2 libgsm1 liblame0 libmp3lame0
  libmpeg6 libmpdclient2 libschroedinger-1.0-0 libshout3 libspeex1
  libtheora0 libva1 libvpx1 libwavpack1 libx264-123 libxvidcore4 mpc mpd
0 upgraded, 24 newly installed, 0 to remove and 0 not upgraded.
Need to get 6,016 kB of archives.
After this operation, 14,2 MB of additional disk space will be used.
Do you want to continue [Y/n]?

```

Fig. 7

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

INSTALLAZIONE SOFTWARE DI BASE

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

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

```

apt-get update
apt-get upgrade

```

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

`/home/pi/radio`

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

```

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

```

```

tar -xvf pi_radio.tar.gz

```



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

`shutdown -h now`

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



Fig. 8

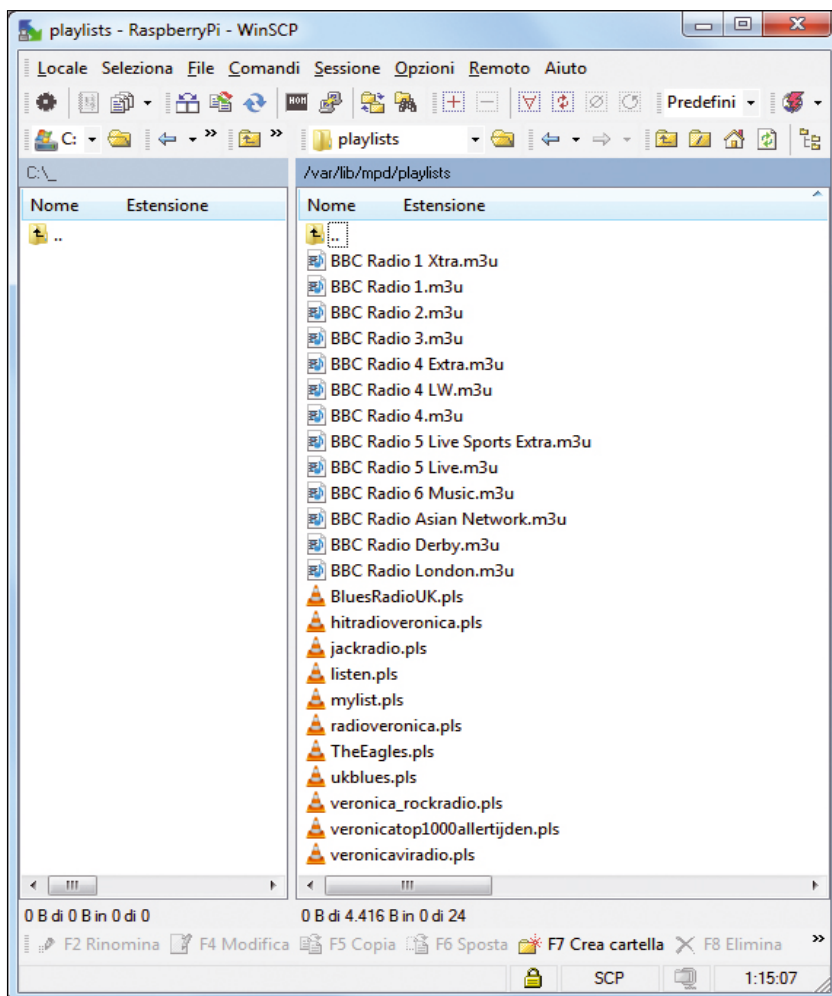


Fig. 9

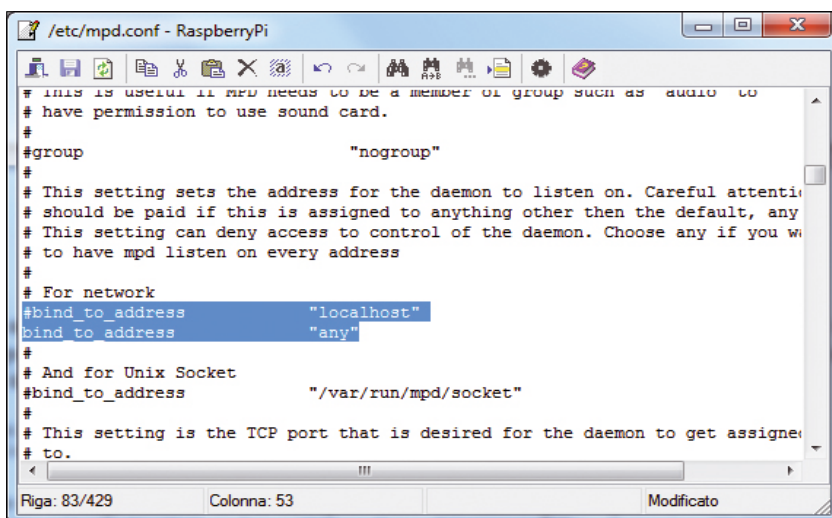


Fig. 10

Il formato dei file PLS (playlist)

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

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

```
[playlist]
```

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

```
NumberOfEntries=2
```

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

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

Nel Listato seguente vediamo un esempio di file playlist.

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

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

```
i2cdetect -y 0 per RaspberryPi rev. 1
```

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

```
i2cdetect -y 1
```

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

INSTALLAZIONE MUSIC PLAYER DAEMON

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

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

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

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

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

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

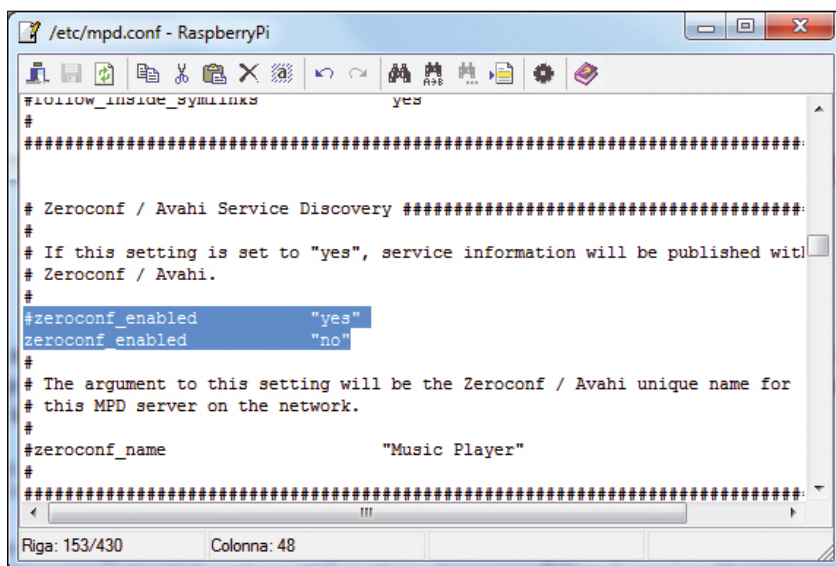


Fig. 11

Fig. B

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

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

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

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

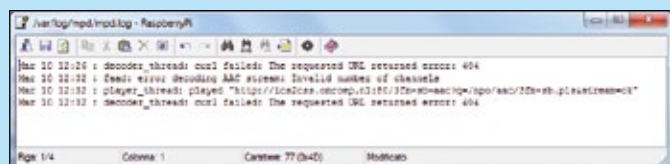
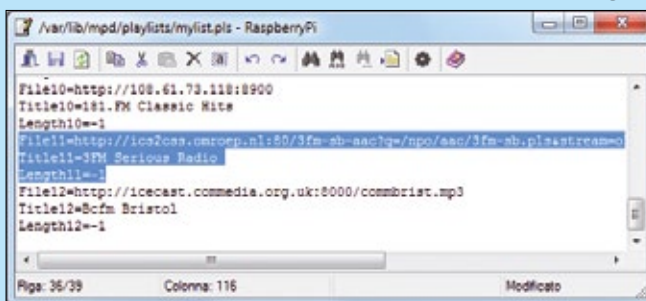


Fig. A



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

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

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

“localhost” ad “any”:

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

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

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

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

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

```
service mpd start
```

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

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

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

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

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

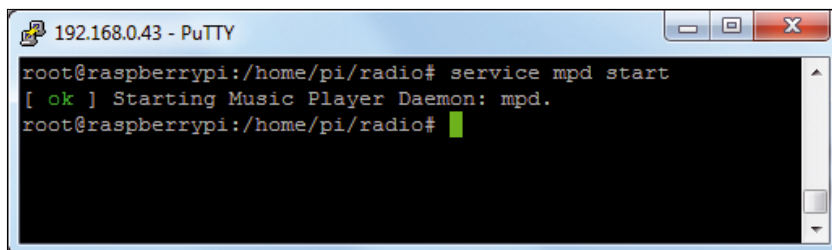


Fig. 12

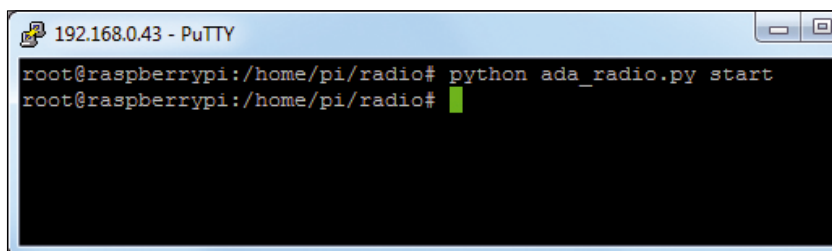


Fig. 13

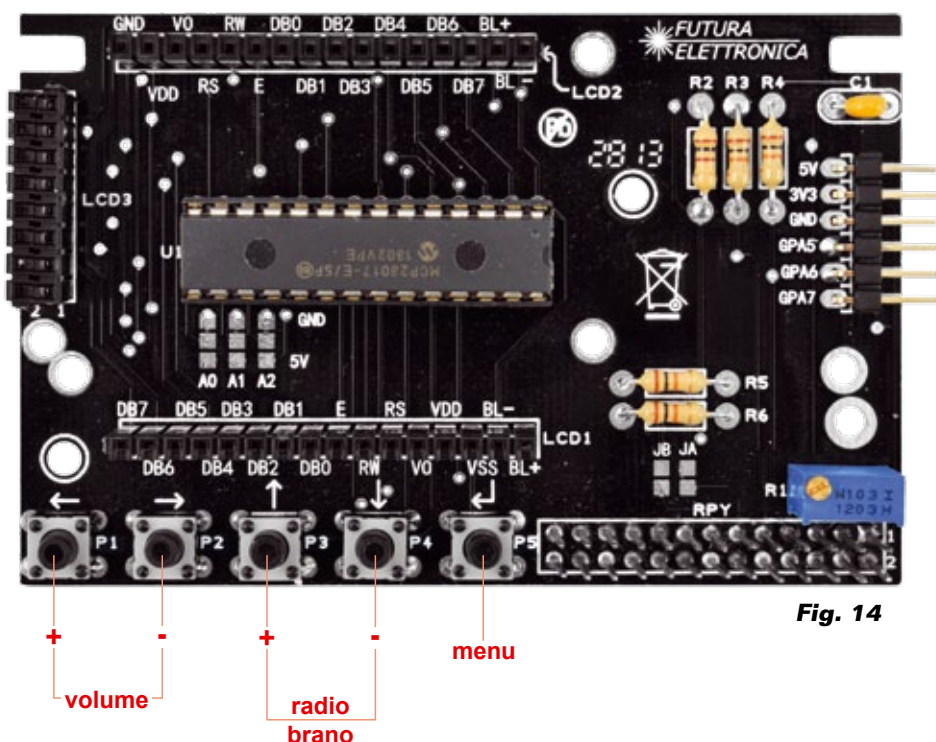


Fig. 14

display LCD;

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

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

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

lancia il server di gestione dell'LCD;

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

interrompe il funzionamento del server di gestione dell'LCD;

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

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

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

visualizza la versione software.

UTILIZZO DELLO SHIELD LCD

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

INSTALLARE L'INTERFACCIA WEB

I componenti principali per

Tabella 1

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

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

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

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

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

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

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

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

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

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

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

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

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

CONFIGURIAMO L'INTERFACCIA WIFI

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

Fig. 15

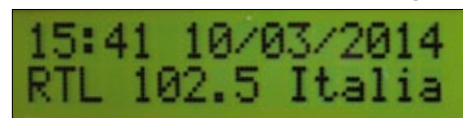


Fig. 16

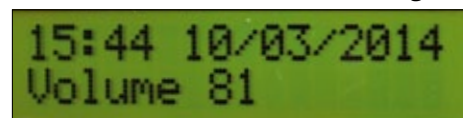


Fig. 17

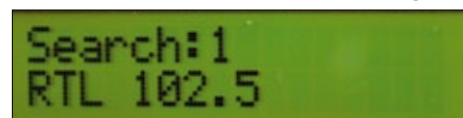


Fig. 18

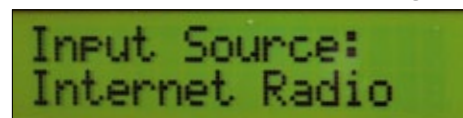


Fig. 19

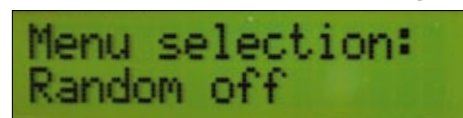


Fig. 20

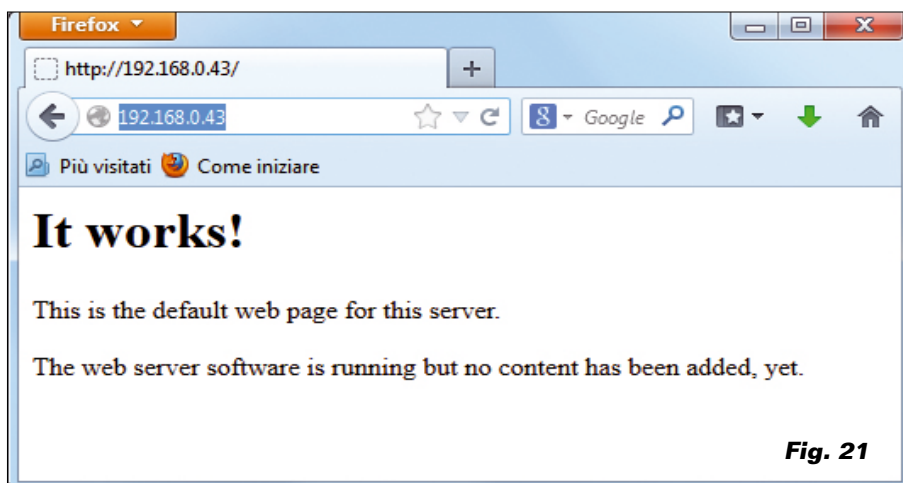
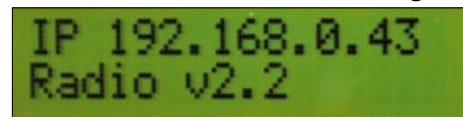
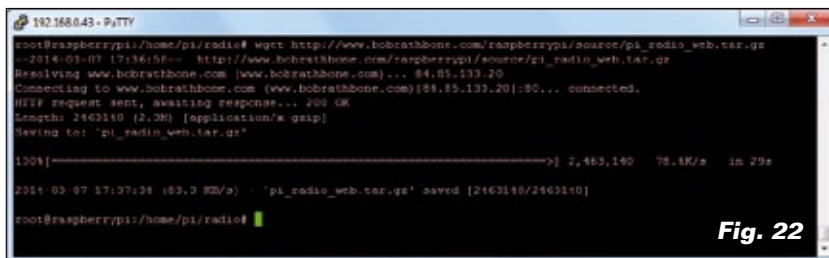


Fig. 21



```

root@raspberrypi:/home/pi/radio# wget http://www.bobrathbone.com/raspberrypi/source/pi_radio_web.tar.gz
--2014-03-07 17:36:58-- http://www.bobrathbone.com/raspberrypi/source/pi_radio_web.tar.gz
Resolving www.bobrathbone.com [www.bobrathbone.com]... 84.85.133.20
Connecting to www.bobrathbone.com (www.bobrathbone.com)[84.85.133.20]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2463140 (2.3M) [application/x-gzip]
Saving to: 'pi_radio_web.tar.gz'

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

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

root@raspberrypi:/home/pi/radio#

```

Fig. 22



```

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

```

Fig. 23

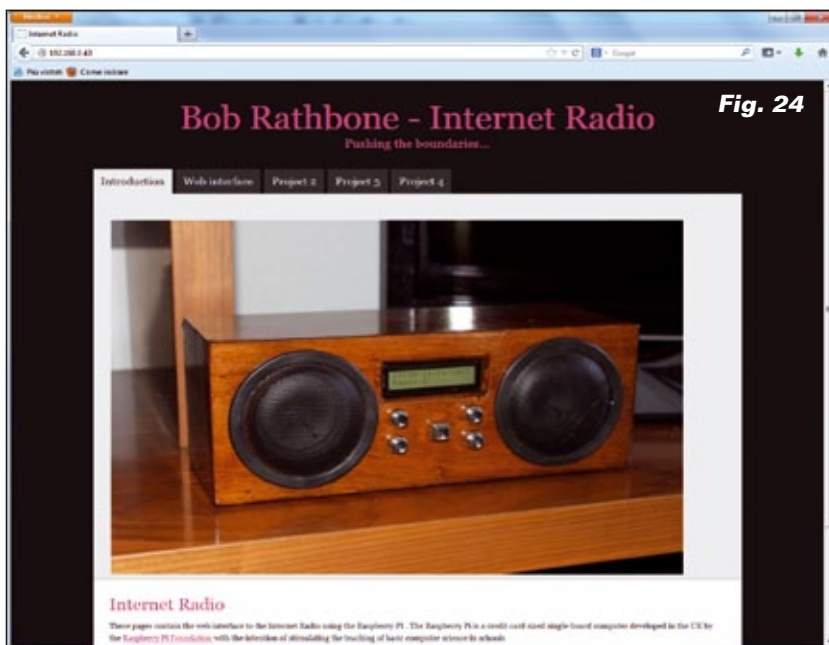


Fig. 24

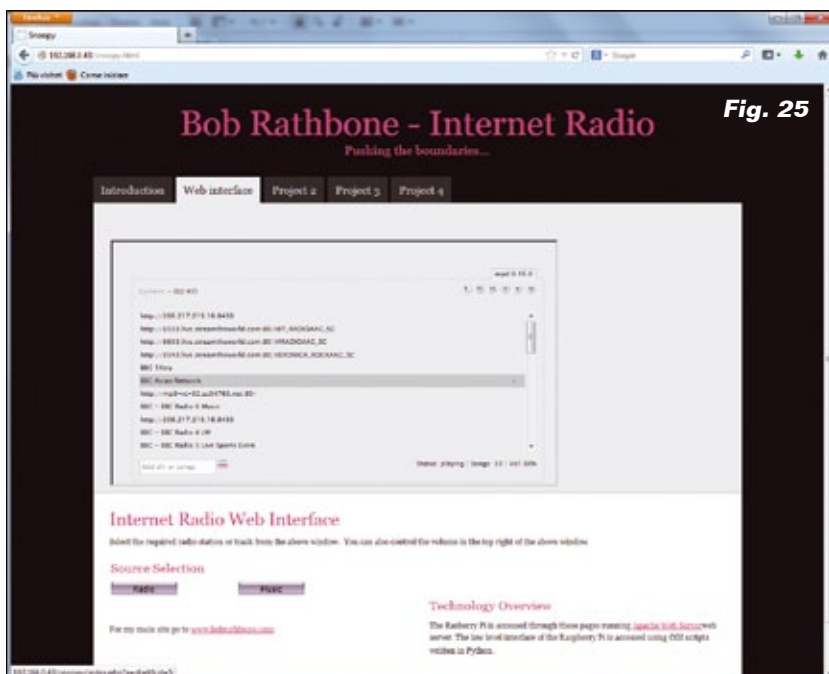


Fig. 25

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

lsusb

Invece con il comando:

ifconfig

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

```

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

```

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


```

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

```

Fig. 26

```

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

root@raspberrypi:~#

```

Fig. 27

Tabella 2

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

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

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

Sotto queste linee aggiungete le seguenti:

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

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

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

```
ifdown wlan0
ifup wlan0
```

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

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

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

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

```
ifconfig
```

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

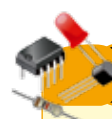
```
#!/bin/bash
/home/pi/radio/ada_radio.py start
```

Rendiamo eseguibile il file con il comando:

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

Tutto il processo è descritto nel "libro".

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



per il MATERIALE

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

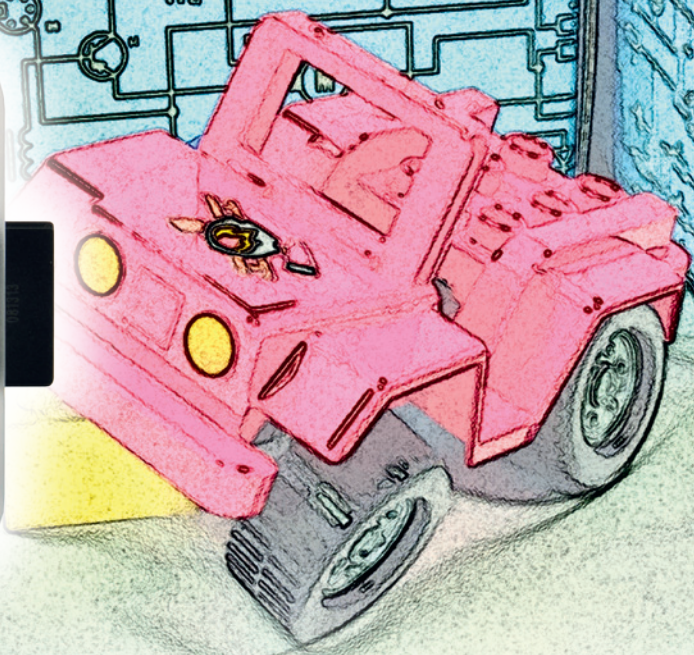
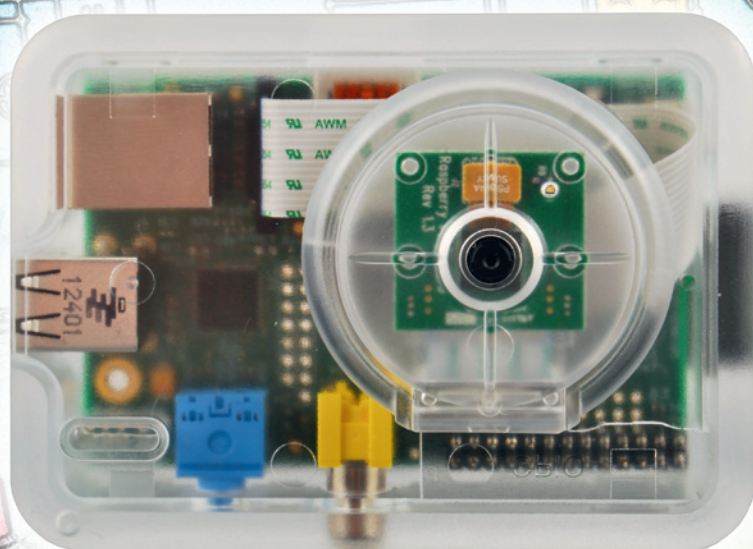
Il materiale va richiesto a:
Futura Elettronica, Via Adige 11,
21013 Gallarate (VA)
Tel: 0331-799775 • Fax: 0331-792287
<http://www.futurashop.it>

Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

Elettronica In

www.elettronica.in.it

oltre l'elettronica



CAMERA PI IL SENSO DELLA VISTA PER RASPBERRY PI

Iniziamo una serie di articoli dedicati all'approfondimento della visione computerizzata utilizzando la nostra scheda Raspberry Pi tutt'altro, e la videocamera dedicata Camera Pi, della quale presentiamo anche le caratteristiche.

di MARCO MAGAGNIN

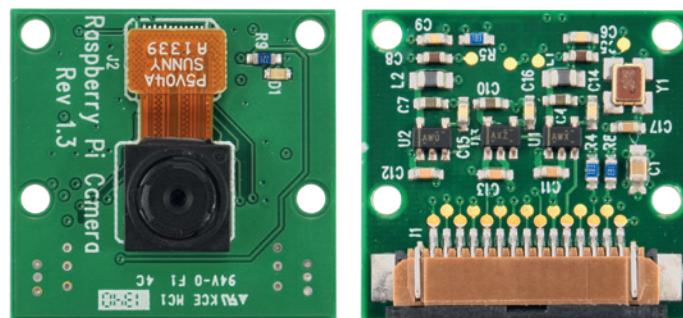
I robot dotati di capacità operative autonome e soprattutto di organi di senso simili a quelli umani, hanno da sempre sollecitato la fantasia di scrittori e sceneggiatori di fantascienza. La ricca produzione di romanzi e film ne è la dimostrazione pratica. Come sempre accade, ad un certo punto della storia anche la scienza ufficiale ha cominciato ad occuparsi dell'argomento, dapprima con le cosiddette "tesi forti", dove l'obiettivo era di riprodurre, per sostituirle, le capacità proprie degli umani. Una volta comprese le dimensioni e le problematiche associate al tema, le ricerche si sono orientate verso obiettivi più concreti e per i quali si poteva pensare di costruire soluzioni tecnologicamente utilizzabili. Allo stato

attuale sono state realizzate una molteplicità di applicazioni che trovano ampio impiego in diversi settori industriali e di servizi: dalla lettura di codici a barre, dove la tecnologia è passata dalla scansione laser di una semplice sequenza di barre verticali bianche e nere, all'interpretazione dei codici noti come QR Code (Quick Response Code) per mezzo di una telecamera. Altre applicazioni molto diffuse si trovano nelle aziende di lavorazione e confezionamento di alimenti, nella raccolta sul campo di prodotti agricoli, nella produzione automobilistica, elettronica e componentistica in genere. In questa serie di articoli, molto più modestamente ed economicamente, vi presenteremo una serie di

esperienze che ci permetteranno di prendere confidenza con questa disciplina. Come molti dei progetti che vi abbiamo presentato nel passato, centrati su Raspberry Pi, anche quelli che vi presenteremo implicano l'integrazione tra diverse discipline e tecnologie. Un sistema di visione, infatti, è costituito da diverse tecnologie integrate tra di loro, come ottica, elettronica, informatica, sistemi di comunicazione e, perché no, meccanica. Per iniziare forniamo una breve panoramica dello scenario associato alla visione computerizzata. Trascuriamo per un attimo la semplice acquisizione di immagini a scopo di trasmissione e/o di registrazione, come nel caso delle webcam, delle videocamere e dei sistemi di videosorveglianza "passivi". In questi casi le immagini acquisite vengono modificate di formato per poter essere trasmesse su canali di trasmissione analogici o digitali ed eventualmente essere archiviate su un supporto elettronico. I temi che noi vogliamo affrontare richiedono invece, in forme diverse, l'elaborazione e l'interpretazione del contenuto delle immagini, per estrarne informazioni utili a pilotare dei processi decisionali automatici o semi automatici. Diamo quindi una prima suddivisione, a grandi linee, delle possibili finalità o problematiche che interessano la visione artificiale, in ordine di complessità:

- **Rilevamento** – Elaborazione dell'immagine per verificare se sono avvenute modifiche tra successivi fotogrammi, nell'area complessivamente inquadrata o in una sua parte. Questa tecnologia è impiegata diffusamente nei DVR integrati in sistemi di videosorveglianza, nei quali una o più telecamere inquadrano aree

Camera Module per Raspberry Pi (vista frontale e posteriore).



sensibili dove si vuole verificare se avvengono delle intrusioni, nel qual caso si richiede di documentare l'intrusione stessa, registrando le immagini per tutta la durata dell'evento. La successiva visione da parte di operatori addetti permetterà di determinare cosa e chi ha determinato l'intrusione e le modalità di svolgimento;

- **Riconoscimento** – L'elaborazione delle immagini ha lo scopo di riconoscere al suo interno delle forme appartenenti a determinati classi. Per esempio oggetti di forma circolare, o rettangolare, o caratterizzati da contorni chiusi, o ancora aree di un determinato colore o che, data la loro forma, possano essere ricondotte ad una determinata classe nota, come ad esempio una faccia o una parte di essa, come gli occhi o il naso.
- **Identificazione** – Questa è un'ulteriore sofisticazione delle elaborazioni precedenti, che permette di "identificare" all'interno delle immagini la presenza di una specifica istanza di una forma appartenente ad una classe nota. Tornando all'esempio delle facce, dopo aver determinato che una certa forma all'interno dell'immagine è una faccia, si tenta di determinare se la faccia è riconducibile ad una faccia nota, la mia, quella di una persona conosciuta oppure non appartenente all'insieme delle facce conosciute;

- **Ricostruzione** – Questo tipo di elaborazione di immagini parte da due o più immagini dello stesso oggetto prese da angolazioni differenti per tentare di generare una superficie 3D completa dell'oggetto. La difficoltà principale di questo tipo di elaborazione è ritrovare e ricongiungere i punti comuni dell'oggetto proveniente dalle differenti immagini.

Nel prosieguo degli articoli vedremo altri aspetti che condizionano la visione computerizzata e come queste influiscono sulle condizioni di acquisizioni e sulle opportunità di sfruttamento dal punto di vista applicativo. Tra le condizioni associate citiamo le fonti d'illuminazione. Normalmente la scena di ripresa è illuminata dalla luce naturale o dalla luce artificiale, simulando le condizioni di visione dell'occhio umano. Le opportunità aggiuntive sono date dalla possibilità di impiegare videocamere sensibili agli infrarossi, che permettono, illuminando la scena con luce infrarossa, di realizzare sistemi di visione notturna, oppure di registrare le radiazioni luminose emesse in base alla temperatura presente nell'area di visione. In questo modo si possono ottenere ed elaborare immagini acquisite anche in campi non esplorabili dall'occhio umano.

Qualunque sia la configurazione del sistema di visione adottato, ci troveremo di fronte un'architettura che integra i seguenti

componenti:

- telecamere e ottiche;
- sistema di illuminazione;
- l'oggetto o la scena da esaminare;
- il sistema di acquisizione e di elaborazione dell'immagine;
- le interfacce uomo macchina;
- le interfacce con l'ambiente esterno.

Il sistema costituito da telecamera e relativa ottica "cattura" l'immagine grazie ad un sensore (CMOS o CCD), composto da una o più matrici di fotorivelatori (camere monocromatiche o a più colori) e la rende disponibile in un formato comprensibile dal sistema di elaborazione. In questo modo le immagini catturate potranno essere elaborate da software opportunamente predisposti, in grado di "leggere" il formato di rappresentazione delle immagini ed "estrarre" le informazioni necessarie ad individuare le caratteristiche e riconoscerne gli aspetti, ad esempio contorni, spigoli, forme, strutture, allo scopo di eseguire i controlli e le verifiche per i quali il sistema è stato concepito.

Sulla base dei risultati dell'elaborazione, il sistema potrà prendere decisioni in merito alle operazioni da eseguire. Per le sperimenta-

zioni e le piccole applicazioni che realizzeremo con Raspberry Pi utilizzeremo la micro videocamera Camera Pi, una videocamera appositamente progettata per essere utilizzata con Raspberry Pi. Non utilizza la connessione USB come le comuni webcam bensì si collega direttamente al connettore CSI presente sulla board. La videocamera è disponibile in due versioni, quella sensibile alla luce diurna e la versione senza filtro IR, utilizzabile per la visione notturna e la cattura di immagini termiche. I motivi che ci hanno fatto scegliere la Camera Pi rispetto ad una comune webcam sono l'alta risoluzione, fino a 2592x1944 pixels per le immagini e 1080p30 per i video, nonché per la disponibilità del software di gestione, appositamente progettato per Camera Pi, in grado di gestire la periferica come una vera e propria foto-video camera, con tanto di regolazioni ed effetti speciali. Le dimensioni della videocamera sono 25mm x 20mm x 9mm. Per standard 1080p30 si intende la capacità di riprendere video con una risoluzione di 1080 righe verticali con refresh di 30 fotogrammi al secondo, la risoluzione orizzontale è di 1920 punti. Sono supportati anche i formati video HD 720p60 con una riso-

luzione orizzontale di 1280 punti per 720 verticali e 60 fotogrammi al secondo ed il formato VGA 640x480p60/90.

INSTALLAZIONE DI CAMERA PI

L'installazione di base della Camera Pi consiste in due fasi, l'installazione hardware e la configurazione del software per il funzionamento standard. Iniziamo dall'installazione hardware. Una raccomandazione, prima di iniziare: la Camera Pi è particolarmente sensibile alle scariche elettrostatiche, quindi prima di estrarre la board dalla sua confezione antistatica, scaricate l'eventuale ecarica elettrostatica toccando un oggetto o apparato collegato a massa. Scintille? Con il vostro sacrificio avete salvato la vostra microcamera.

Ora pensiamo a collegarla a Raspberry Pi. La micro camera dispone di un cavo piatto flessibile a 15 poli, che dovrà essere collegato al connettore CSI (camera Serial Interface). Il connettore CSI è collocato tra la presa LAN e la presa HDMI. Come si vede nello schema di Fig. 1 l'interfaccia CSI prevede due linee dati, una linea clock, un'interfaccia di controllo compatibile con lo standard I2C e le due linee di alimentazione 3,3V e GND.

L'interfaccia CSI di comunicazio-

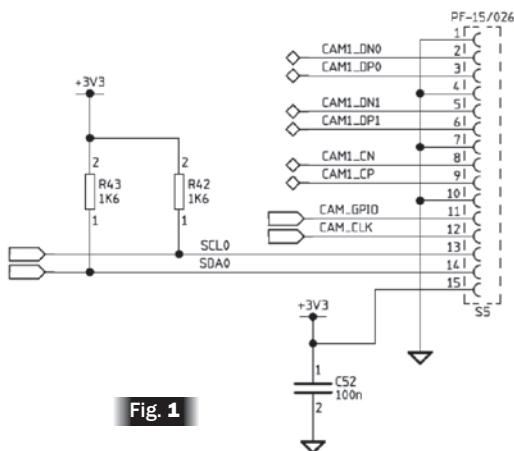


Fig. 1

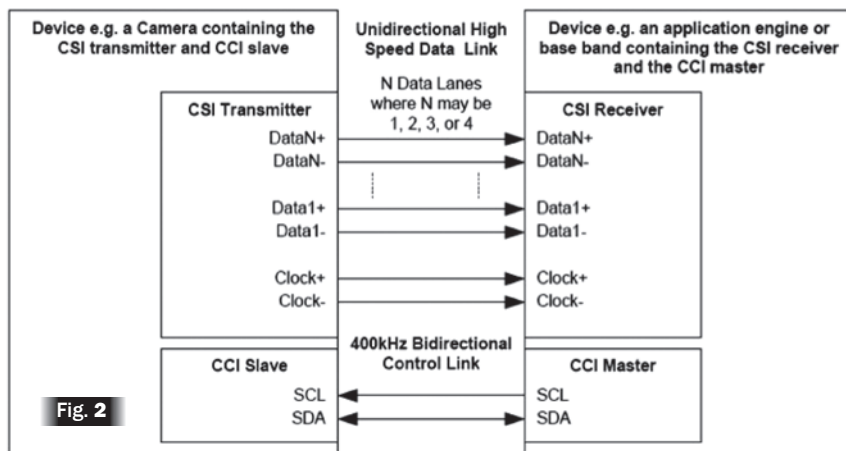


Fig. 2



Fig. 3



Fig. 6



Fig. 7

ne tra la videocamera e Raspberry Pi è un'interfaccia seriale differenziale unidirezionale, con linee di segnale data e clock, conforme agli standard MIPI Alliance Standard per le interfacce D-PHY.

La Fig. 2 illustra il collegamento tra il trasmettitore ed il ricevitore in standard CSI-2. A parte la Camera Pi, l'impiego tipico di questa connessione è il collegamento interno delle videocamere

per gli smartphone ed i dispositivi mobili in genere. L'interfaccia di controllo del sistema è un'interfaccia bidirezionale, basata sullo standard I2C.

Estraiamo la Camera Pi dalla sua confezione, facendo ancora una volta attenzione alle cariche elettrostatiche. Una volta estratta ci troveremo in mano un circuito stampato di circa 5 cm² con montata sopra una telecamera microscopica e con collegato un cavo piatto a 15 conduttori. Sono disponibili diverse combinazioni di contenitori per Raspberry Pi e Camera Pi in modo da poter realizzare uno strumento compatto ed adatto alle diverse esigenze. Quello che abbiamo scelto per le nostre prove è stato realizzato in modo da dare all'insieme Raspberry Pi e Camera Pi l'aspetto di una vera e propria macchina fotografica. Prima di inserire Camera

Pi in un contenitore ricordatevi di togliere la piccola protezione di plastica posta sull'obiettivo, tirandola per l'apposita linguetta.

Se scegliete di utilizzare il contenitore presentato nelle foto, fate passare innanzitutto il cavo piatto in tutte le fessure appositamente predisposte prima di inserirlo nel connettore CSI-2. Per installare fisicamente Camera Pi su Raspberry Pi è sufficiente inserire il cavo piatto nel connettore CSI-2 che si trova tra il connettore Ethernet ed il connettore HDMI. Nell'inserire il connettore, con cautela, verificate che i terminali argentati siano rivolti verso il connettore HDMI ed il dorso blu, di conseguenza,

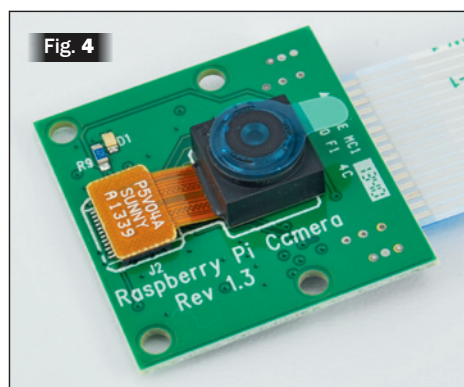


Fig. 4



Fig. 5

Tabella 1

Opzioni di preview	
-preview, -p	Dimensioni della finestra di "Preview" <'x,y,w,h'>. I parametri x e y posizionano la finestra nel video e w, h ne determinano le dimensioni. La finestra di preview rimane sempre sovrapposta alle altre finestre aperte.
-fullscreen, -f	Forza la finestra di "Preview" ad occupare l'intero schermo. In questa configurazione vengono preservate le proporzioni dell'immagine con l'eventuale inserimento di barre di completamento ai lati.
-nopreview, -n,	Non presenta la finestra di "Preview". Le immagini vengono comunque catturate e salvate, se richiesto.
-opacity, -op	Imposta la trasparenza della finestra di "Preview". "0" = completamente trasparente (invisibile), 255 = ricopre completamente la grafica sottostante.

Fig. 8

verso il connettore Ethernet, come visibile in Fig. 6.

Per questo tipo di esperienza è bene utilizzare Raspberry Pi come un PC “classico”, con monitor (va bene un televisore con ingresso HDMI), tastiera e mouse. Non dimentichiamo il collegamento alla rete Internet. Per le prime configurazioni possiamo collegarci in remoto utilizzando il protocollo SSH (Secure SHell). Entrambe le modalità sono descritte nel libro “Raspberry Pi – Il mio primo Linux embedded” reperibile presso Futura Elettronica (www.futurashop.it). Dopo un’ulteriore verifica (avete rimosso la protezione dall’obiettivo?) potete alimentare Raspberry Pi ed attendere il boot. Iniziamo la configurazione. Apriamo una finestra di terminale ed eseguiamo il login utilizzando l’utente “root”. So che è una cattiva abitudine in ambiente GNU/Linux, ma trattandosi di Raspberry Pi più di tanti danni non riusciremo a fare. Al peggio si ricomincia da capo. Al prompt aggiorniamo la configurazione del sistema operativo digitando:

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

Al termine apriamo lo strumento di configurazione di Raspbian digitando:

```
raspi-config
```

Scorrete le voci di menu fino ad evidenziare la voce “Camera” (Fig. 8). Date invio e, nella pagina successiva, selezionate con il tasto “Tab” la voce “Enable” (Fig. 9). Tornati al menu principale selezionate “Finish” (Fig. 10). Verrà chiesto se eseguire il reboot di Raspberry Pi. Scegliete “Yes” (Fig. 11) ed attendete il riavvio della scheda.

A questo punto possiamo fare

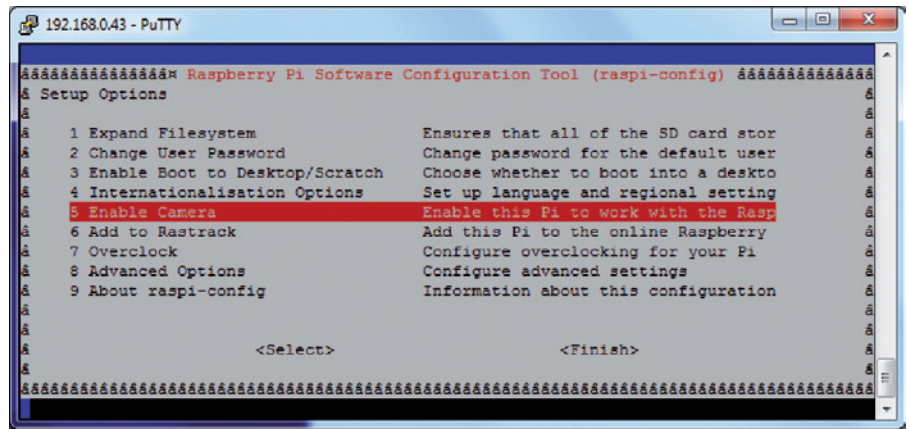


Tabella 2

Controllo generale della videocamera		
--sharpness, -sh	Imposta il livello di definizione (nitidezza) dell'immagine. Valori da -100 a 100. Valore predefinito = 0.	
--contrast, -co	Imposta il livello di contrasto dell'immagine. Valori da -100 a 100. Valore predefinito = 0.	
--brightness, -br	Imposta il livello di luminosità dell'immagine. Valori da 0 (bianco) a 100 (nero). Valore predefinito = 50.	
--saturation, -sa	Imposta il livello di saturazione dell'immagine. Valori da -100 a 100. Valore predefinito = 0.	
--ISO, -ISO	Imposta la sensibilità in scala ISO da utilizzare nelle operazioni di cattura. Valori da 100 a 800.	
--vstab, -vs	Attiva la stabilizzazione video. Funziona solo in modalità video.	
--ev, -ev	Imposta il livello di compensazione dell'esposizione in unità EV (scatti di apertura di diaframma). Valori da -10 a 10. Valore predefinito =	
--exposure, -ex	Imposta il tipo di esposizione. Le possibili opzioni sono:	
	off	
	auto	calcolo automatico dell'esposizione
	night	modalità per riprese notturne
	nightpreview	modalità preview per riprese notturne
	backlight	modalità ottimale per soggetti illuminati posteriormente
	spotlight	modalità per illuminazione mediante luce spot (faretto)
	sports	modalità per riprese sportive e scatti veloci
	snow	modalità ottimale per riprendere paesaggi innevati
	beach	modalità ottimale per riprendere paesaggi che includono spiagge
	verylong	modalità adatta a lunghe esposizioni
	fixedfps	Imposta la frequenza dei fotogrammi fps
	antishake	modalità antitremolio
	fireworks	modalità adatta a riprendere spettacoli di fuochi artificiali
	Alcune delle impostazioni descritte possono essere utilizzate o meno a seconda delle altre impostazioni della videocamera.	
--awb, -awb	Imposta la modalità di bilanciamento automatico del bianco (AWB - automatic white balance):	
	off	disabilita la funzione di bilanciamento
	auto	modalità di bilanciamento automatico (predefinito)
	sun	bilanciamento con pieno sole
	cloud	bilanciamento con cielo nuvoloso
	tungsten	bilanciamento con illuminazione mediante lampade al Tungsteno
	fluorescent	bilanciamento con illuminazione mediante lampade fluorescenti
	incandescent	bilanciamento con illuminazione mediante lampade a incandescenza
	flash	bilanciamento con illuminazione flash
	horizon	modalità Horizon (ad esempio per riprendere tramonti)

(Continua)

Controllo generale della videocamera

-imxfx, -ifx	Imposta gli effetti da applicare alle immagini:	
	none	nessun effetto applicato
	negative	produce un'immagine in negativo
	solarise	polarizza l'immagine
	whiteboard	effetto lavagna bianca
	blackboard	effetto lavagna nera
	sketch	effetto disegno schizzo
	denoise	elimina i disturbi dall'immagine
	emboss	effetto incavato
	oilpaint	effetto dipinto ad olio
	hatch	effetto disegno a trattini
	gpen	effetto disegno a matita
	pastel	effetto pastello
	watercolour	effetto acquerello
	film	effetto film sgranato
	blur	effetto sfuocato
	saturation	immagine a colori saturi
	colourswap	in corso di implementazione
	washedout	in corso di implementazione
	posterise	in corso di implementazione
	colourpoint	in corso di implementazione
	colourbalance	in corso di implementazione
	cartoon	in corso di implementazione
-awb, -awb	Imposta la modalità di bilanciamento automatico del bianco (AWB - automatic white balance):	
	off	disabilita la funzione di bilanciamento
	auto	modalità di bilanciamento automatico (predefinito)
	sun	bilanciamento con pieno sole
	cloud	bilanciamento con cielo nuvoloso
	tungsten	bilanciamento con illuminazione mediante lampade al Tungsteno
	fluorescent	bilanciamento con illuminazione mediante lampade fluorescenti
	incandescent	bilanciamento con illuminazione mediante lampade a incandescenza
	flash	bilanciamento con illuminazione flash
	horizon	modalità Horizon (ad esempio per riprendere tramonti)
-metering, -mm	Impostazione messa a fuoco. Specifica le modalità di calcolo della messa a fuoco per il "Preview" e la cattura dell'immagine:	
	average	media sull'intero campo si ripresa
	spot	messa a fuoco di un soggetto
	backlit	considera l'immagine illuminata da dietro
	matrix	messa a fuoco a matrice (Matrix Metering)
-rotation, -rot	Imposta la rotazione dell'immagine (0-359). Imposta la rotazione, in gradi, da impartire all'immagine, in teoria può essere impostato qualsiasi angolo di rotazione; in pratica le limitazioni hardware di Raspberry Pi permettono di utilizzare unicamente i valori in gradi 0, 90, 180 e 270	
-hflip, -hf	Imposta il ribaltamento orizzontale dell'immagine. L'immagine ripresa viene mostrata in "Preview" e memorizzata su file, ribaltata orizzontalmente	
-vflip, -vf	Imposta il ribaltamento verticale dell'immagine. L'immagine ripresa viene mostrata in "Preview" e memorizzata su file, ribaltata verticalmente	
-roi, -roi	Imposta la zona di interesse per il sensore della videocamera (ROI - Region Of Interest). Permette di specificare un'area rettangolare all'interno del campo di ripresa da utilizzare come sorgente per le operazioni di "Preview", di acquisizione e di memorizzazione. I parametri richiesti sono la posizione x e y dell'angolo in alto a sinistra del riquadro di interesse e la larghezza e l'altezza del riquadro. Tutti i valori sono in coordinate normalizzate, compresi tra 0.0 e 1. Per descrivere una zona che inizia al centro dell'area di ripresa e si sviluppa per un quarto della stessa a destra e in basso, si specificano i valori: -roi 0.5,0.5,0.25,0.25	

Tabella 2 - (segue)

qualche esperimento con la Camera Pi utilizzando i comandi "nativi" per la gestione da linea di comando. I comandi in questione sono "raspistill" e "raspistillyuv" per la cattura e memorizzazione di fotogrammi singoli e "raspivid" per la cattura e memorizzazione di sequenze video. Associati a ciascun comando abbiamo una ricca dotazione di parametri che possiamo dividere in due categorie. La prima categoria, comune a tutti i comandi, permette di configurare la videocamera, impostando per esempio la saturazione, il contrasto ed il tipo di esposizione, mentre la seconda categoria permette di parametrizzare il funzionamento dei singoli comandi come l'intervallo tra gli scatti singoli, la durata di un video e le indicazioni su dove memorizzare il risultato delle operazioni.

Nella **Tabella 1** trovate i parametri di configurazione della videocamera che sono validi per tutti i comandi; nella **Tabella 2** trovate i parametri utilizzabili nella cattura di singoli fotogrammi, quindi con i comandi "raspistill" e "raspistillyun" mentre nella **Tabella 3** abbiamo elencato i parametri utilizzabili con il comando "raspivid" per la cattura di sequenze video. Vediamo qualche esempio pratico dell'utilizzo di questi comandi. Innanzitutto dobbiamo scegliere se utilizzare o meno la funzione "preview", che ci permette di vedere a video l'anteprima delle immagini elaborate dai comandi. In caso affermativo dobbiamo utilizzare Raspberry Pi come un PC, con collegati video, tastiera e mouse, ed operare direttamente dall'interfaccia grafica del sistema operativo (**Fig. 12**).

In caso negativo possiamo anche operare in modalità terminale remoto utilizzando il protocollo di comunicazione SSH con gli strumenti Putty e WINScp. Vediamo

Tabella 3

Opzioni fotocamera – programma raspistill	
<code>--width, -w</code>	imposta la larghezza dell'immagine <size>
<code>--height, -h</code>	imposta l'altezza dell'immagine <size>
<code>--quality, -q</code>	imposta la qualità JPEG <0 to 100> La qualità 100 restituisce l'immagine praticamente non compressa, 75 può essere considerato un buon valore di compromesso tra qualità e compressione.
<code>--raw, -r</code>	aggiunge ai metadati ipeg i dati "raw Bayer"
<code>--output, -o</code>	nome del file dove memorizzare il fotogramma od il video <filename>. Se non viene specificato alcun file il risultato della cattura non viene salvato. Se il nome file corrisponde a "-" tutto il flusso di output viene inviato allo stdout.
<code>--verbose, -v</code>	durante l'esecuzione del comando vengono prodotti messaggi di informazione e debug
<code>--timeout, -t</code>	intervallo prima della acquisizione dell'immagine. Il comando viene eseguito per il tempo specificato prima di eseguire l'operazione di acquisizione, questo solo se è stato specificato un file di output. Se non specificato l'intervallo viene impostato a 5 secondi.
<code>--timelapse, -tl</code>	modalità Timelapse (filmato accelerato). Il valore specifica l'intervallo in millisecondi tra le acquisizioni dei singoli fotogrammi. Per mantenere ordinati i fotogrammi è possibile specificare il qualificatore %04d nel punto del nome del programma dove volete inserire il contatore dei fotogrammi. Per esempio, combinando i due parametri: <code>-t 30000 -tl 2000 -o image%04d.jpg</code> si ottiene una acquisizione di immagine ogni due secondi durante un periodo complessivo di 30 secondi. Le immagini saranno nominate: image1.jpg, image0002.jpg...image0015.jpg. Il qualificatore %04d produce un numero di quattro cifre con gli zeri di riempimento a sinistra, se necessari. La cifra 04 indica il numero di cifre che caratterizzano il contatore, per esempio %08d produce un contatore a otto cifre.
<code>--thumb, -th</code>	imposta le caratteristiche delle immagini di anteprima (thumbnail) inserite nel file JPEG (x:y:quality). Se non specificati i valori predefiniti configurano una dimensione di 64x48 con una qualità JPEG di 35.
<code>--demo, -d</code>	esegue una elaborazione demo per il numero di millisecondi indicati. Questa opzione mostra una serie di funzionalità della videocamera, senza prevedere alcuna acquisizione. La demo termina alla fine del periodo indicato indipendentemente dalla fase in cui si trova.
<code>--encoding, -e</code>	codifica del file di output. I valori gestiti sono jpg, bmp, gif e png. I tipi di immagini non accelerate (gif, png, bmp) richiedono più tempo per essere salvate rispetto ai tipo ipg, che sfrutta l'acceleratore hardware. Il tipo impostato è prioritario rispetto ad un eventuale suffisso specificato per il file di output.
<code>--exif, -x</code>	elenco dei tag EXIF (Exchangeable image file format) da inserire nel file di output JPEG. Il formato di ciascun tag è del tipo <chiave=valore>. Questo parametro permette di inserire sino a 32 tag EXIF ai metadati dell'immagine. Un piccolo sottoinsieme di questi dati è gestito automaticamente dai comandi di Camera Pi. Le scelte operate con questo parametro prevalgono su quelle automatiche.
<code>--fullpreview, -fp</code>	modalità full preview. Questo parametro imposta la finestra di "Preview" alla stessa risoluzione impostata per l'acquisizione. In questa modalità il numero massimo di fotogrammi al secondo è 15 e la finestra di "Preview" mostra lo stesso campo visivo dell'immagine in acquisizione. In questa modalità, ancora in fase di sviluppo, le operazioni di acquisizione avvengono più velocemente.

mo alcuni esempi di comandi per scattare fotogrammi e per registrare filmati.

SCATTARE FOTO CON CAMERA PI

Il comando per scattare fotogrammi è:

```
raspistill
```

seguito dai parametri e dalle configurazioni che trovate elencate nelle Tabelle 1 e 2. La combinazione di questi parametri permette di utilizzare Camera Pi come una vera e propria macchina fotografica di buon livello. Potete impostare i valori di esposizione, messa a fuoco e applicare una notevole quantità di effetti. Potete decidere di visualizzare l'immagine a video e/o registrarla su un file. Il comando più semplice che scatta una foto e la archivia in un file in formato JPEG è:

```
raspistill -o foto001.jpg
```

Per salvare un fotogramma dopo due secondi di "preview" (l'intervallo è specificato in millesecondi):

```
raspistill -t 2000 -o foto001.jpg
```

Per salvare un fotogramma di dimensioni e risoluzione ridotte:

```
raspistill -t 2000 -o image.jpg -w 640 -h 480
```

Per salvare un fotogramma di qualità decisamente ridotta per salvare spazio:

```
raspistill -t 2000 -o image.jpg -q 5
```

Per posizionare la finestra di "preview" sullo schermo alle coordinate 100, 100 con dimensioni 300 per 200 pixel:

```
raspistill -t 2000 -o image.jpg -p
```

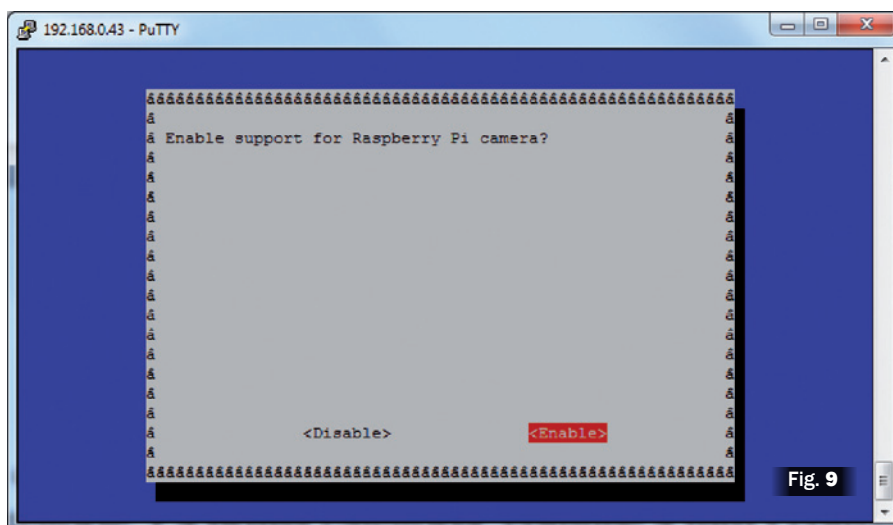


Fig. 9

Opzioni videocamera – programma raspivid	
-width, -w	Imposta la larghezza dell'immagine <size> Valori in pixel tra 64 e 1920.
-height, -h	Imposta l'altezza dell'immagine <size> Valori in pixel tra 64 e 1080.
-bitrate, -b	Imposta il bitrate di trasferimento in bit per secondo. 10MBits/s viene espresso come -b 10000000. Per un flusso di alta qualità H264, 1080p è consigliabile un bitrate di 15Mbits/s o più.
-output, -o	Nome file di output <filename>. Se non viene specificato alcun nome di file il filmato non viene salvato. Se il nome file corrisponde a "-" allora il flusso di output viene inviato a stdout.
-verbose, -v	Vengono prodotti messaggi di informazione e debug durante l'esecuzione del comando
-timeout, -t	Intervallo prima della acquisizione video. Il comando viene eseguito per il tempo specificato prima di eseguire l'operazione di acquisizione, questo solo se è stato specificato un file di output. Se non specificato l'intervallo viene impostato a 5 secondi. Il valore "0" configura il comando per l'esecuzione continua, finché non viene terminata con CTRL-c.
-demo, -d	Esegue una elaborazione demo per il numero di millisecondi indicati. Questa opzione mostra una serie di funzionalità della videocamera, senza prevedere alcuna acquisizione. La demo termina alla fine del periodo indicato indipendentemente dalla fase in cui si trova.
-framerate, -fps	Configura il numero di frame per secondo da memorizzare sul file di output. Nella implementazione attuale il numero di frame per secondo minimo ammesso è 2 mentre il massimo è 30.
-penc, -e	Mostra l'immagine di "Preview" dopo l'operazione di encoding. Visualizzare la finestra di "Preview" dopo la compressione permette di evidenziare eventuali artefatti nella finestra stessa. In condizioni normali la finestra di "Preview" mostra le immagini riprese dalla videocamera prima della compressione. Non è certo che questa funzionalità venga mantenuta nelle release future.

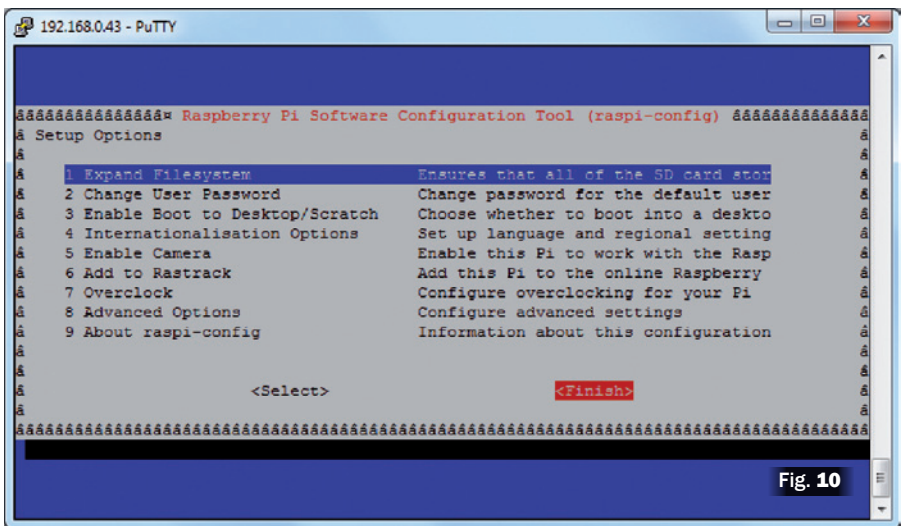


Tabella 4

100,100,300,200

Per disabilitare la finestra di "preview":

```
raspistill -t 2000 -o image.jpg -n
```

Per attivare l'effetto incavato sul fotogramma acquisito:

```
raspistill -t 2000 -o image.jpg -ifx emboss
```

Per acquisire immagini a intervalli regolari, una ogni 10 secondi per 10 minuti (600000 ms) e memorizzarle in file con nomi progressivi foto_numero_1.jpg, foto _ numero _2.jpg, ecc:

```
raspistill -t 600000 -tl 10000 -o foto _ numero _%d.jpg
```

Divertitevi a sperimentare.

REGISTRARE VIDEO CON CAMERA PI

I parametri di impostazione delle immagini e del "preview" sono pressoché tutti uguali, disponibili per l'acquisizione di fotografie più una serie dedicata alla gestione delle riprese video. Le dimensioni predefinite delle riprese video sono 1920x1080 pixel. Per registrare un video di 5 secondi con le dimensioni predefinite (1080p30):

```
raspivid -t 5000 -o video.h264
```

Per registrare un video di 5 secondi con una banda predefinita (3,5Mbits/s):

```
raspivid -t 5000 -o video.h264 -b 3500000
```

Per registrare un video di 5 secondi impostando il numero di fotogrammi per secondo (5fps):

```
raspivid -t 5000 -o video.h264 -f 5
```

Fig. 12

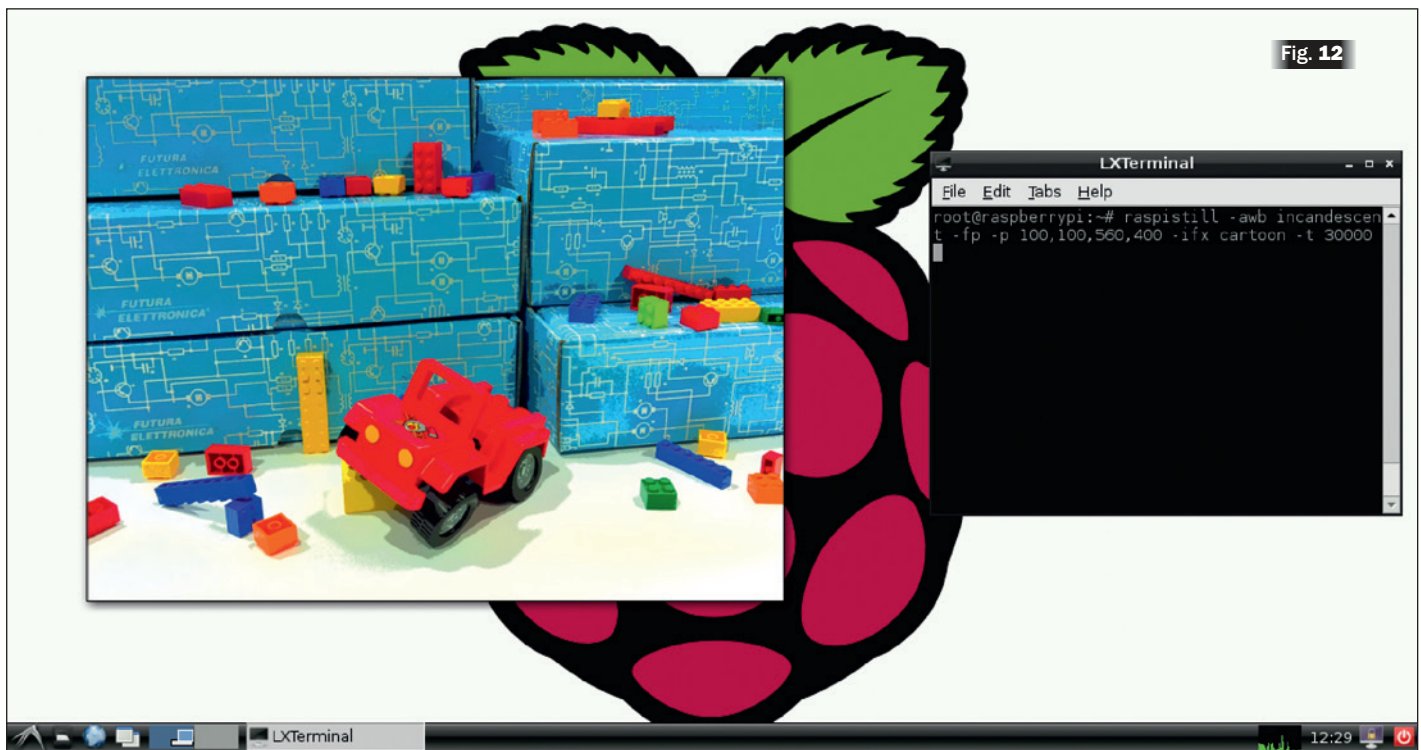


Fig. 13 - norm



Fig. 14 - negative



Fig. 15 - solarise



Per registrare un video di 5s salvandolo su file:

```
raspivid -t 5000 -o - > mio_file.h264
```

UTILIZZARE CAMERA PI DA PROGRAMMI PYTHON

Ovviamente i comandi da finestra terminale non rappresentano l'unica possibilità per utilizzare Camera Pi.

Pressoché tutto ciò che può essere eseguito a linea di comando può essere gestito anche da programma: per gestire la videocamera, ad esempio, via interfaccia web o automatizzare l'acquisizione ed archiviazione delle immagini e dei video. Ma non solo. Il flusso di immagini della videocamera può essere instradato verso altri software di gestione per elaborare le immagini acquisite.

Per gestire la videocamera da programmi Python dobbiamo installare un pacchetto di gestione realizzato da Dave Jones. Siccome è contenuto nella distribuzione Raspbian possiamo installarlo con i comandi ormai familiari:

Fig. 16 - schetch

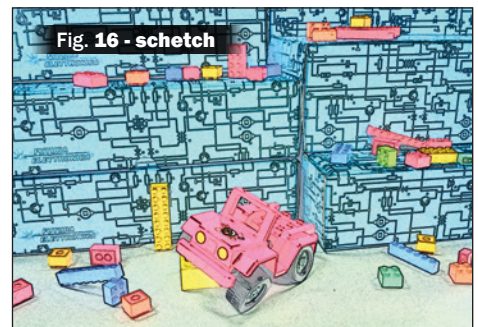


Fig. 17 - emboss

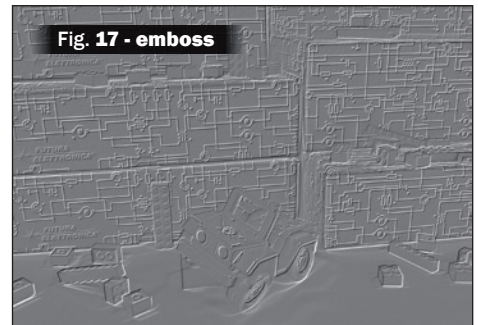
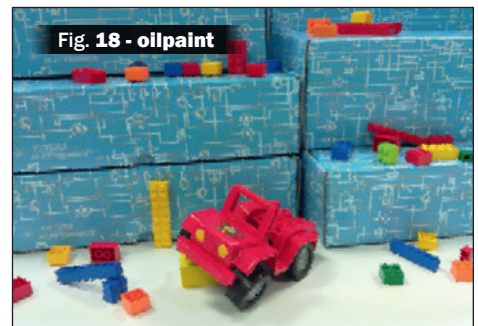


Fig. 18 - ollpaint



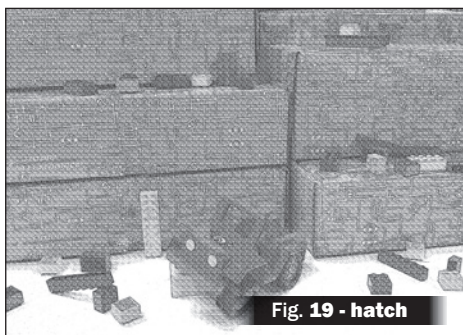


Fig. 19 - hatch

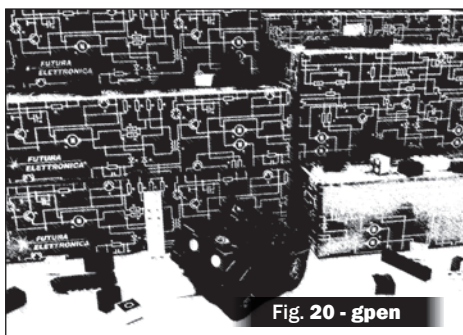


Fig. 20 - gpen

```
apt-get update
apt-get install python-picamera
```

Il pacchetto mette a disposizione moltissime funzionalità che non mancheremo di presentare nei prossimi articoli. Per il momento

Listato 1

```
import time
import picamera

with picamera.PiCamera() as camera:
    camera.resolution = (1024, 768)
    camera.start_preview()
    camera.image_effect = none
    # Ritardo di acquisizione
    time.sleep(10)
    #camera.capture('foto.jpg')
    camera.image_effect = 'sketch'
    time.sleep(10)
    camera.image_effect = 'emboss'
    time.sleep(10)
    camera.image_effect = 'negative'
    time.sleep(10)
```

Listato 2

```
import picamera

with picamera.PiCamera() as camera:
    camera.resolution = (640, 480)
    camera.start_preview()
    camera.start_recording('video.h264')
    camera.wait_recording(60)
    camera.stop_recording()
    camera.stop_preview()
```

```
192.168.0.43 - PuTTY
root@raspberrypi:~# apt-get install python-picamera
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  python-picamera-docs
The following NEW packages will be installed:
  python-picamera
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 53.2 kB of archives.
After this operation, 302 kB of additional disk space will be used.
Get:1 http://archive.raspberrypi.org/debian/ wheezy/main python-picamera armhf 1.2-1 [53.2 kB]
Fetched 53.2 kB in 0s (111 kB/s)
Selecting previously unselected package python-picamera.
(Reading database ... 69317 files and directories currently installed.)
Unpacking python-picamera (from .../python-picamera_1.2-1_armhf.deb) ...
Setting up python-picamera (1.2-1) ...
root@raspberrypi:~#
```

Fig. 21

```
192.168.0.43 - PuTTY
root@raspberrypi:/home# python prova.py
```

Fig. 22

accontentiamoci di ripetere da programma le configurazioni che abbiamo provato da linea di comando.

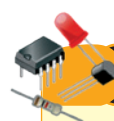
Il programma del **Listato 1** permette di acquisire un fotogramma dopo due secondi di ritardo e salvarlo su file.

Vengono utilizzati i metodi `camera.start_preview()` per iniziare la fase di "preview" e, dopo due secondi, con il metodo `camera.capture` viene eseguita l'acquisizione del fotogramma ed il salvataggio su file.

Il programma del **Listato 2** permette di acquisire un video di 60 secondi in formato 640 x 480 pixel.

Sono possibili molte altre funzionalità, come dividere la registrazione di un video su più file, acquisire singoli fotogrammi durante la registrazione di un video, la registrazione rolling, ecc. Tutte le opzioni presentate in questo articolo sono fondamentali per i successivi, nei quali presenteremo semplici sistemi di elaborazione delle immagini. In questa attività è molto importante poter gestire i parametri di acquisizione delle immagini da

elaborare, come illuminazione, messa a fuoco, formato, risoluzione, bilanciamento, colore o bianco e nero (o entrambe). Tutte caratteristiche ben implementate nel software di gestione di Camera Pi e che vedremo di sfruttare al meglio. Per ora divertitevi a sperimentare l'infinita gamma di opzioni disponibili. ■



per il MATERIALE

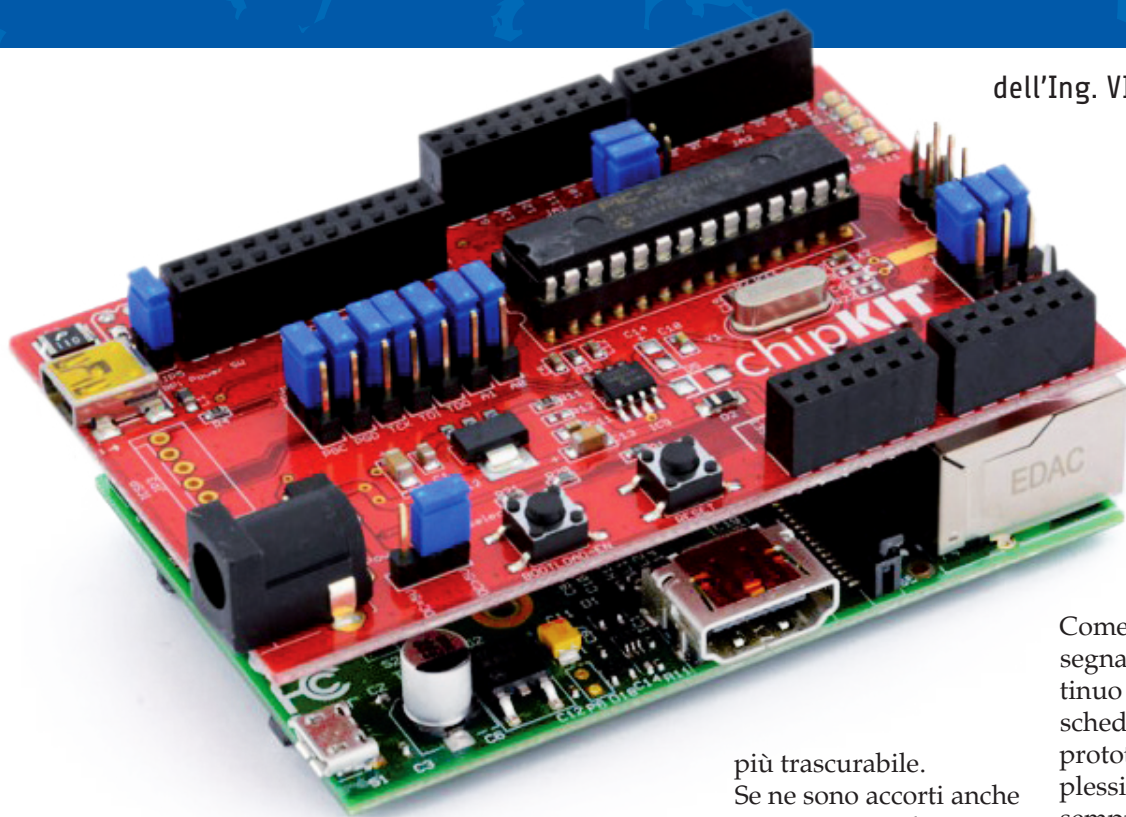
La board Raspberry Pi tipo B (cod. 7310-7568308RS) costa 38,00 Euro mentre la Camera Module per Raspberry Pi (cod. 7310-7757731RS) costa 31,00 Euro. È disponibile anche un contenitore plastico per la sola telecamera (cod. 5100-RPI-CAM.9 Euro 6,80) ed un contenitore trasparente in grado di accogliere sia la telecamera che la board (cod. 5100-TEK-CAM.0, Euro 13,00). I prezzi si intendono IVA compresa.

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

Una nuova scheda di prototipazione irrompe nel panorama dei prodotti embedded per enfatizzare le qualità di Raspberry Pi e Arduino. Scopriamola in anteprima.

chipKIT Pi: COME RENDERE RASPBERRY PI ANCORA PIÙ VERSATILE

dell'Ing. VINCENZO MENDOLA



Il successo riscosso da Raspberry Pi e da Arduino è noto a tutti e questi prodotti hanno conquistato una fetta consistente del mercato

hobbistico che sta diventando di dimensioni considerevoli, tanto da rendere il fenomeno rappresentato da queste schede embedded non

più trascurabile. Se ne sono accorti anche i maggiori produttori di semiconduttori e, più nello specifico, di microcontrollori, i quali nell'era pre-Arduino hanno ignorato o, peggio ancora, snobbato il fenomeno.

Come abbiamo più volte segnalato, c'è un continuo fiorire di nuove schede di sviluppo e prototipazione di complessità e prestazioni sempre più apprezzabili, con costi accessibili a tutte le tasche ma al momento a farla da padroni rimangono Arduino e Raspberry Pi, attorno ai quali si è creata un'enorme comunità di appas-

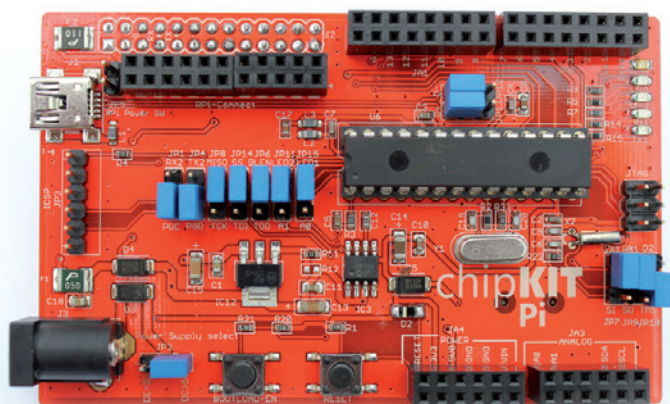


Fig. 1

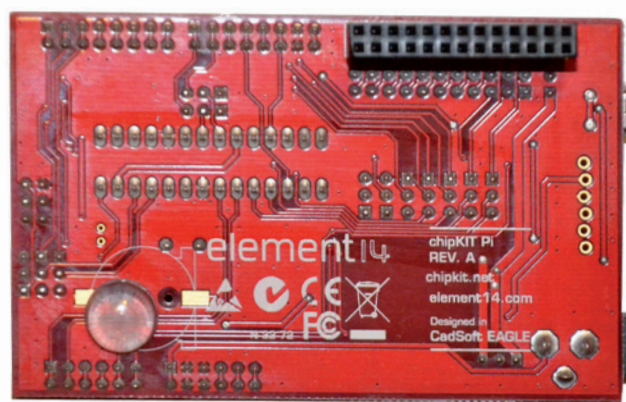


Fig. 2

sionati. Non ci deve pertanto meravigliare che la stessa Microchip sia scesa in campo insieme a Farnell/element14, per realizzare una board "open", nata proprio per fare da ponte tra il mondo di Raspberry Pi e quello di Arduino; la scheda di cui stiamo parlando è chipKIT Pi (Fig. 1).

In virtù dello spirito open source con cui è nato il progetto, gli sviluppatori che hanno lavorato alla sua realizzazione hanno scelto di utilizzare Eaglecad della CadSoft, noto per essere utilizzato da tempo da tutti i progettisti del settore che hanno deciso di rendere disponibili i

sorgenti del loro lavoro. L'utilizzo di questo software, per la realizzazione della scheda, è riportato con orgoglio sia sulla scatola della chipKIT Pi, sia sul retro della board (Fig. 2). Le potenzialità della board Raspberry Pi sono veramente notevoli, come ben sapete anche grazie ai nostri articoli, libri e tutorial sulla materia, quindi per attirare concretamente l'attenzione degli appassionati, i progettisti hanno dovuto scegliere il microcontrollore che gestisce la nuova board "accessoria", in modo che fosse in grado di fornire delle funzionalità avanzate, tra cui applicazioni

touch e audio processing. Per questo motivo il cuore della chipKIT Pi è un potente e flessibile PIC32MX250F128B, un PIC a 32 bit incapsulato in un contenitore plastico dual in line (PDIP) a 28 pin, dal costo veramente interessante, specialmente se rapportato alle caratteristiche offerte.

Vediamo qualche dettaglio saliente inerente a questa MCU (punti 1 e 2 dei **Riferimenti**):

- massima velocità di funzionamento: 50 MHz;
- memoria di programma: 128 kB;
- RAM: 32 kB;
- CODEC Interface (I2S, AC97);
- USB di tipo "host";
- 9 canali ADC a 10 bit, con frequenza massima di campionamento pari a 1.100 ksp/s;
- 5 canali PWM;
- 5 timer a 16 bit;
- 2 oscillatori interni (8 MHz, 32 kHz);
- 3 comparatori;
- 2 SPI;
- 2 I²C;
- tensione di funzionamento compresa tra 2,3 e 3,6V.

Grazie a questa scheda, possiamo aggiungere la potenza di un PIC32 alla flessibilità di Arduino e alle potenzialità di linux embedded espresse attraverso

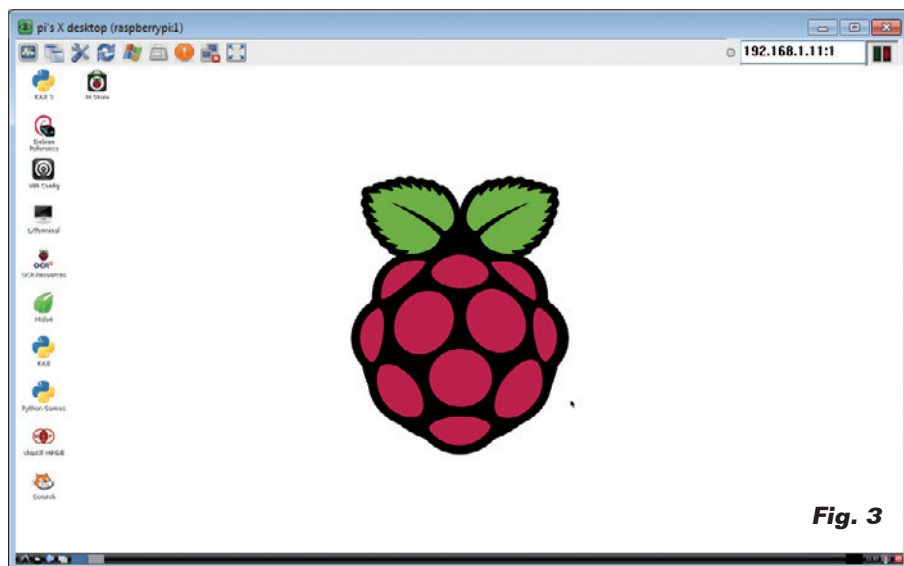


Fig. 3

Raspberry, interfacciandosi direttamente a quest'ultima tramite il connettore di espansione I/O del "lampone".

Il PIC32MX250F128B lavora a 3,3 V come la Raspberry Pi, quindi non sono necessari traslatori di livelli logici; piuttosto, nel caso di interfacciamento con Arduino bisogna fare attenzione agli shield e utilizzare quelli compatibili con la logica a 3,3 V. Qualora si voglia comunque adattare gli shield standard di Arduino a 5V, occorre effettuare le necessarie traslazioni di livello, riferendosi, solo per fare un esempio, a questo documento: http://www.microchip.com/stellent/groups/techpub_sg/documents/devicedoc/en026368.pdf.

Lo schema completo della chipKIT Pi, disponibile alla pagina web dedicata di element14 e raggiungibile all'indirizzo <http://www.element14.com/community/servlet/JiveServlet/download/63873-2-118998/chipKIT%20Pi%20Schematic.zip>, mostra la relativa semplicità del circuito, grazie al gran numero di periferiche integrate nel PIC.

Il manuale d'uso (punto 3 dei **Riferimenti**) fornisce le informazioni essenziali per iniziare subito a lavorare con la scheda, che può essere anche utilizzata in autonomia, senza Raspberry Pi, grazie al connettore di alimentazione di cui dispone e tramite il connettore standard ICSP utilizzato dalla Microchip per connettere i suoi picKIT per programmare e debuggare i propri micro direttamente sui PCB dei circuiti realizzati dai clienti.

La board chipKIT Pi può essere facilmente programmata attraverso l'IDE gratuito ed open source MPIDE, disponibile già integrato nella distribuzione (Raspbian) appositamente adattata <http://downloads.ele>

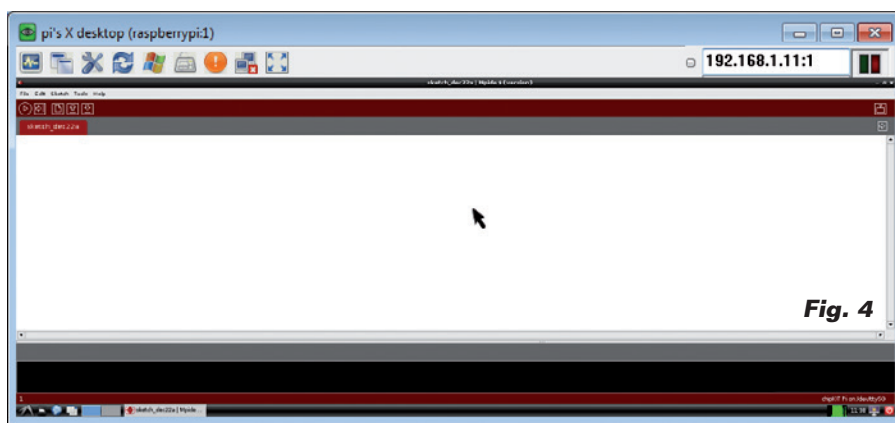


Fig. 4

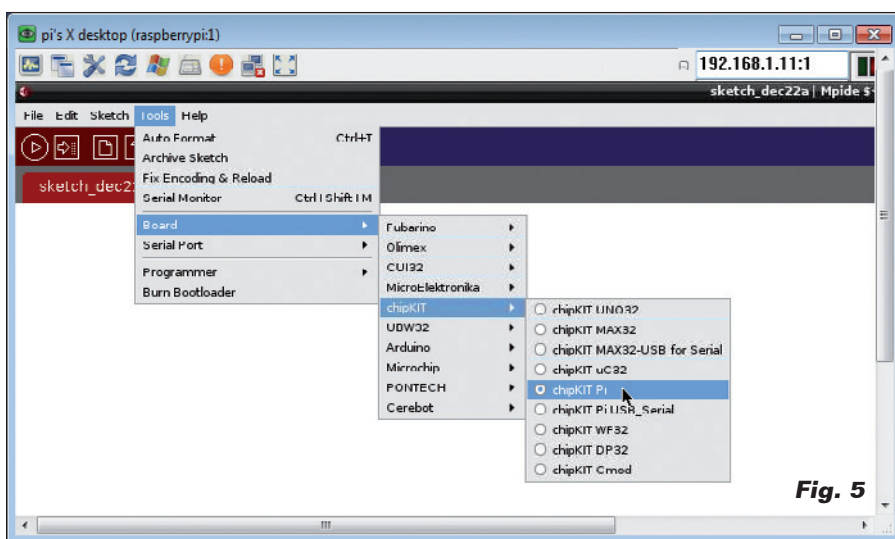


Fig. 5

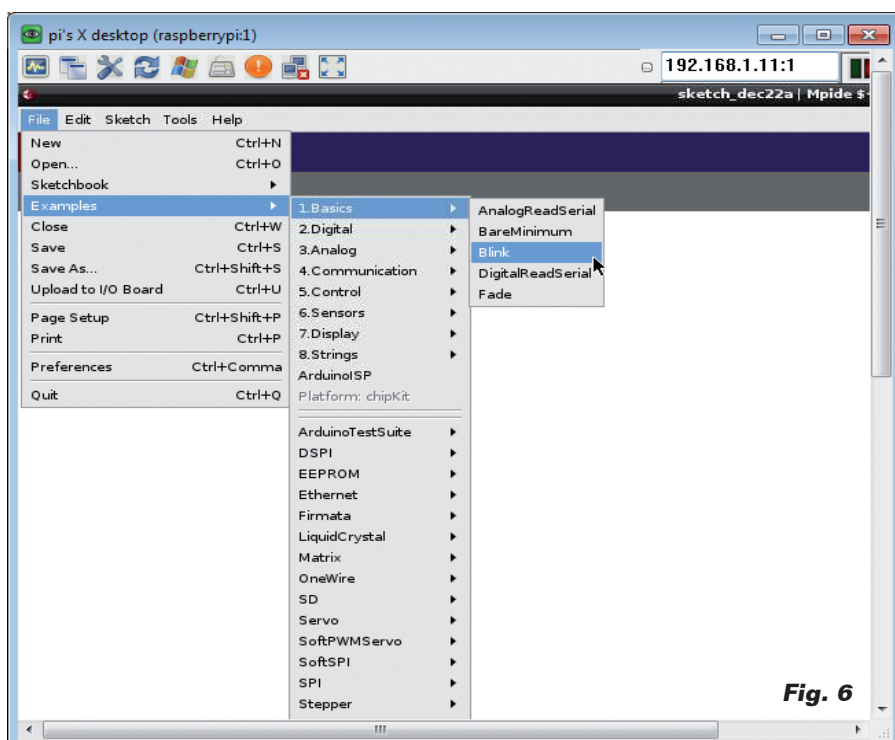


Fig. 6

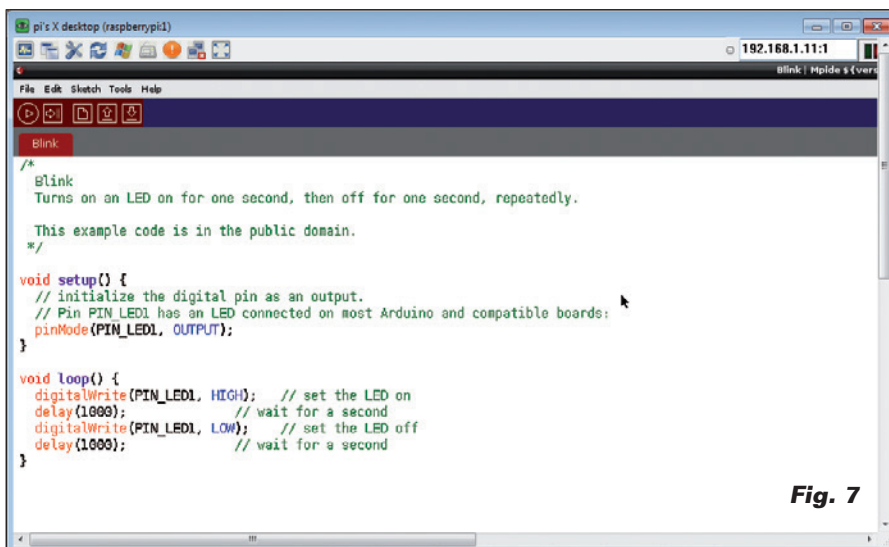


Fig. 7

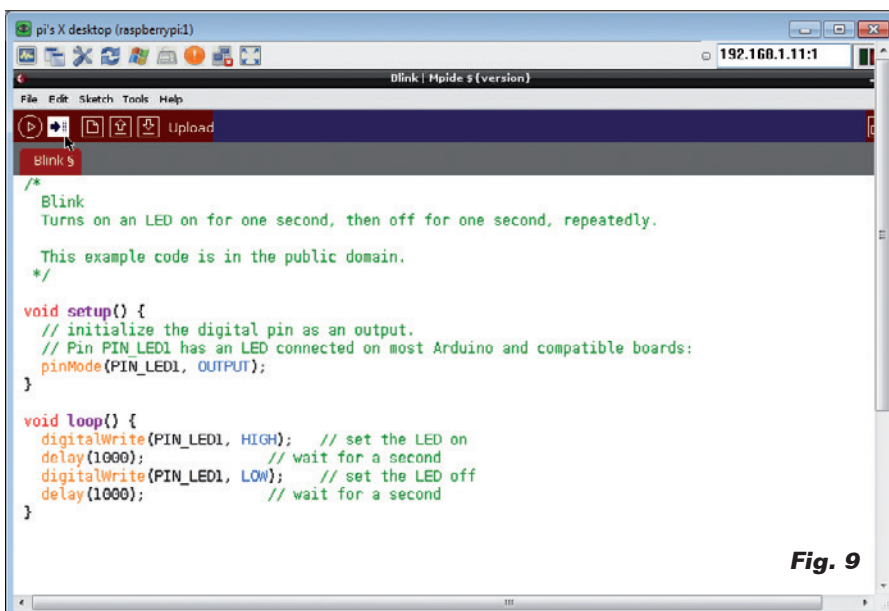


Fig. 9

ment14.com/downloads/chipkit_pi.zip?COM=chipKITPi (Fig. 3 e Fig. 4). Il codice dell'MPIDE è stato scritto per supportare molte schede, tra cui tutte le versioni ed alcuni cloni di Arduino, oltre che le schede di sviluppo della Microchip, quelle della serie chipKIT e molte altre (Fig. 5). Per descrivere l'utilizzo della chipKIT Pi partiamo da un semplicissimo esempio, un "hello world" cui siamo tutti abituati: far lampeggiare il LED1 (Fig. 6

e Fig. 7) della scheda. Prima di poter programmare il chipKIT Pi, dobbiamo configurarlo in modalità "bootloader"; per farlo dobbiamo assicurarci che entrambi i jumper JP1 e JP4 siano posizionati in alto (TX e RX)



Fig. 8

come mostrato in Fig. 8. Una volta fatto ciò, bisogna premere e tenere premuto il pulsante BOOTLOAD -EN, posto in basso a sinistra nella scheda, in prossimità del DC plug J3; mentre questo pulsante è ancora premuto, premere e rilasciare il pulsante di RESET, posizionato alla destra del BOOTLOAD -EN, il quale ora può essere rilasciato. La modalità BOOTLOADER è confermata dall'accensione del LED TX1 e dal lampeggio del LED RX1; tale combinazione di segnalazioni visive e di tasti ci permette di caricare il codice sul PIC32 tramite l'icona "upload" (vedere Fig. 9). Una volta terminato il caricamento, i due diodi luminosi TX1 ed RX1 si spegneranno; nella barra dei comandi visibile in basso nella schermata di lavoro, l'IDE di Microchip ci indicherà che tutto è andato a buon fine ("Done uploading"), allorché si vedrà il LED 1 che inizierà a lampeggiare. Quello descritto è solo un esempio di utilizzo di chipKIT Pi; sul sito ufficiale troverete quanto occorre per realizzare altre applicazioni. ■

Riferimenti

- [1] <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en557425>
- [2] <http://ww1.microchip.com/downloads/en/DeviceDoc/61168E.pdf>
- [3] www.farnell.com/datasheets/1754419.pdf

COMPUTER VISION CON RASPBERRY PI ED IL MODULO CAMERA PI



www.elettronica.in.it



Dopo aver fatto conoscenza con le funzionalità del modulo Camera Pi, video-fotocamera di qualità appositamente progettata per Raspberry Pi, vediamo come impiegarla per acquisire le prime conoscenze di Visione Computerizzata, come riconoscere forme e colori.

di MARCO MAGAGNIN

Camera Pi è un ottimo add-on di Raspberry Pi per scattare fotografie e registrare video di qualità, con la possibilità di applicare una notevole gamma di configurazioni ed effetti. In questo articolo descriviamo come utilizzarla per riconoscere specifici contenuti all'interno delle immagini acquisite. Immediatamente sorge la domanda. Perché mai occuparsi di visione computerizzata (computer

vision)? La risposta è semplice: perché è una disciplina ormai assai diffusa nelle applicazioni del mondo reale ed ormai, entro certi limiti, alla portata anche degli hobbisti. La risposta più articolata parte dalla curiosità e volontà umana di riprodurre nelle macchine capacità sensoriali e manipolative proprie della nostra specie. Ed è anche una delle situazioni nelle quali tra il sogno e la realtà il percorso si è di-

Tabella 1

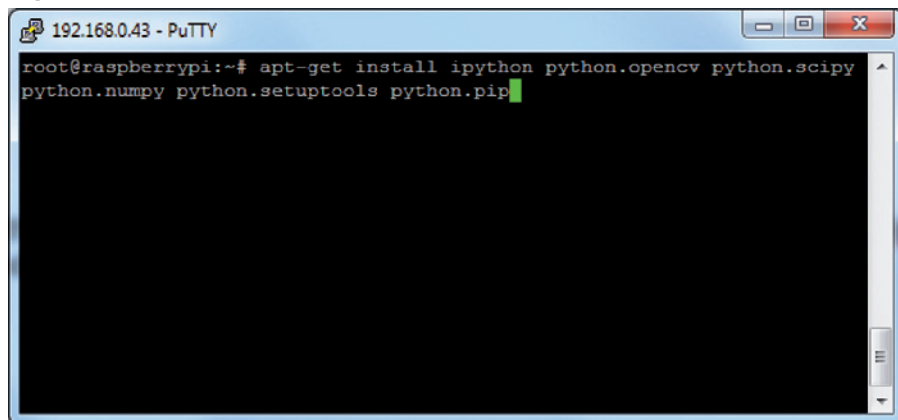
Facile	Difficile
Quanto è grande un piatto? È sporco?	Inquadrare una cucina ed individuare tutti i piatti sporchi.
È cambiato qualcosa in due immagini riprese successivamente?	Individuare e seguire una persona od un oggetto ed inseguirlo in un ambiente affollato.
Misurare il diametro di una ruota. Verificare se non è circolare.	Individuare singole parte di un insieme meccanico.
Di che colore è una foglia?	Di che tipo di foglia si tratta?

mostrato particolarmente lungo e complesso. Infatti la visione è una capacità che la specie umana gestisce straordinariamente bene, praticamente senza sforzo o bisogno di addestramento. Con una semplice occhiata gli esseri viventi acquisiscono una enorme quantità di informazioni che vengono passate al cervello per essere elaborate a seconda delle esigenze del momento. Il tutto in modo pressoché inconscio, mentre il legittimo proprietario di occhi e cervello esegue le sue normali attività, come per esempio scendere in canoa le rapide tumultuose di un fiume in primavera. Facile no? Per le macchine però le cose non vanno allo stesso modo. Strano, basterebbe riprodurre in logica digitale il comportamento del nostro sistema occhio-cervello ed il gioco è fatto. Già. Però forse le cose sono un po' più complesse di ciò che appare a prima vista. Prendiamo, per esempio, l'azione relativa alla lettura di un libro. Gli occhi acquisiscono l'immagine nel campo visivo ed il cervello la elabora per estrarne le informazioni. Per prima cosa il cervello

deve individuare quale parte dell'immagine è occupata dal libro e quale parte è unicamente contorno. La parte "meccanica" dell'operazione richiede la continua messa a fuoco dell'immagine, la visione binoculare permette di valutare la distanza ed aggiustare di conseguenza il cristallino. Siccome l'oggetto è vicino, la messa a fuoco deve essere regolata in continuazione. Una volta che il libro è inquadrato stabilmente, è tempo di cercare di interpretare i segni tracciati nelle pagine. Devono essere individuati i segni che rappresentano delle lettere, separandoli dagli altri contenuti, poi i singoli segni devono essere raggruppati in modo da individuare parole di senso compiuto. Infine, la comprensione dei contenuti, ma questa è tutta un'altra storia. Su queste operazioni insistono ulteriori variabili da considerare ed elaborare correttamente: diversi font grafici, qualche errore nel testo, una diversa angolazione del libro, seguire il testo attorno alle immagini e tra una pagina e la successiva. E se dopo una pausa apriamo il libro ad una pagina sbagliata, oppure

apriamo un libro diverso? Altro problema è il tracciamento di oggetti in movimento, come una palla da tennis durante una partita. Il problema di realizzare algoritmi che riproducano il comportamento di occhi e cervello comincia ad assumere aspetti decisamente complessi. Eppure intere schiere di specialisti di diverse discipline continuano a impegnarsi nello sforzo di riprodurre il senso della vista in macchine e computer in genere. Quali sono i vantaggi portati da queste ricerche? Il vantaggio, immenso, è dato da alcune proprietà dei computer che, in un certo senso, sono migliori rispetto alle corrispettive capacità umane. I computer possono "guardare" la stessa scena per giorni, messi o anni senza mai stancarsi. Oppure, opportunamente programmati, possono eseguire misurazioni precise (lunghezze ed angoli) e ripeterle su tutti i particolari presenti nelle immagini. Possono confrontare immagini successive per verificare se vi sono stati dei cambiamenti, anche impercettibili ed in lunghi intervalli di tempo. Possono riconoscere aree di differenti colori, o sfumature di grigio, stabilendone i contorni e le forme, confrontandole con altre forme memorizzate in precedenza. A questo punto possiamo azzardare una definizione della disciplina Computer Vision come la ricerca e realizzazione di applicazioni dove la capacità visiva dei computer può essere utilizzata in applicazioni difficili o impraticabili per la vista umana, come l'analisi di immagini di diagnostica medica, la sicurezza, la gestione di veicoli autonomi.

Fig. 1



Un settore particolare della Computer Vision è la Machine Vision che ha per obiettivo lo sviluppo di applicazioni per il controllo di qualità, la gestione di processi di produzione (un esempio per tutti è il porzionamento di prodotti alimentari come affettati e formaggi di struttura non omogenea, come il famoso gruviera, e la robotica in generale. A questo punto perché occuparci di Computer Vision, soprattutto a livello amatoriale come in questa rivista. Per diversi motivi. Il primo è, come è già capitato in altri settori, l'assistere al doppio fenomeno di riduzione dei costi di accesso ai dispositivi tecnologici, telecamere e dispositivi di elaborazione, che una volta assolutamente inaccessibili, oggi sono alla portata degli sperimentatori e al diffondersi e, in un certo senso, semplificarsi degli strumenti software di elaborazione delle immagini. In effetti gli strumenti veramente professionali sono ancora inaccessibili ma la tecnologia sottostante è disponibile praticamente a tutti, soprattutto per quanto riguarda gli strumenti software. GNU/Linux e molti strumenti di sviluppo Open Source sono gli stessi (o quasi) utilizzati nelle realizzazioni professionali. Il secondo motivo è che la Computer Vision sta entrando in pressoché tutti i campi tecnologici come arricchimento di funzionalità dei sistemi di supporto a pressoché ogni campo applicativo. Si va dalla sovrapposizione di linee virtuali sui campi sportivi ed al tracciamento di oggetti o veicoli in movimento (pensiamo al canottaggio, al rugby ed alle gare di auto e moto), all'impiego in agricoltura per la gestione di campi coltivati e di frutteti oltre ai già citati settori alimentari, industriali, cinema, arti, medicina, sicurezza, ecc. In questo articolo

descriveremo le basi della Computer Vision con gli strumenti a disposizione di pressoché ogni hobbista: il nostro microcomputer tuttora Raspberry Pi, la Camera Pi per la acquisizione delle immagini (oppure una webcam USB) e gli strumenti di elaborazione di immagini professionali open source come OpenCV, SimpleCV e Python. Di Raspberry Pi ce ne stiamo occupando da molti mesi così

come del linguaggio Python. La Camera Pi e tutte le sue possibilità sono state descritte nel numero 184 di Marzo 2014 della rivista. Per declinare il tutto al servizio della Computer Vision in questo articolo introduciamo due strumenti finalizzati, come riconoscibile anche dai loro nomi, alla elaborazione di immagini. OpenCV (Open Computer Vision) ed il framework SimpleCV (Simple Computer Vision) che ne permet-

Fig. 2

```

192.168.0.43 - PuTTY
root@raspberrypi:~# pip install https://github.com/sightmachine/SimpleCV/zipball/master
Downloading/unpacking https://github.com/sightmachine/SimpleCV/zipball/master
  Downloading master (unknown size): 53.7Mb downloaded
  Running setup.py egg_info for package from https://github.com/sightmachine/SimpleCV/zipball/master

Installing collected packages: SimpleCV
  Running setup.py install for SimpleCV

  Installing simplecv script to /usr/local/bin
Successfully installed SimpleCV
Cleaning up...
root@raspberrypi:~#

```

Fig. 3

```

192.168.0.43 - PuTTY
root@raspberrypi:~# pip install svgwrite
Downloading/unpacking svgwrite
  Downloading svgwrite-1.1.3.tar.gz (174Kb): 174Kb downloaded
  Running setup.py egg_info for package svgwrite

Installing collected packages: svgwrite
  Running setup.py install for svgwrite

Successfully installed svgwrite
Cleaning up...
root@raspberrypi:~#

```

Fig. 4

```

192.168.0.43 - PuTTY
root@raspberrypi:~# simplecv
+-----+
SimpleCV 1.3.0 [interactive shell] - http://simplecv.org
+-----+

Commands:
  "exit()" or press "Ctrl+ D" to exit the shell
  "clear()" to clear the shell screen
  "tutorial()" to begin the SimpleCV interactive tutorial
  "example()" gives a list of examples you can run
  "forums()" will launch a web browser for the help forums
  "walkthrough()" will launch a web browser with a walkthrough

Usage:
  dot complete works to show library
  for example: Image().save("/tmp/test.jpg") will dot complete
  just by touching TAB after typing Image().

Documentation:
  help(Image), ?Image, Image?, or Image()? all do the same
  "docs()" will launch webbrowser showing documentation

SimpleCV:1>

```

Tabella 2

Colore	Valori
Red	(255, 0, 0)
Green	(0, 255, 0)
Blue	(0, 0, 255)
Yellow	(255, 255, 0)
Brown	(165, 42, 42)
Orange	(255, 165, 0)
Black	(0, 0, 0)
White	(255, 255, 255)

te l'utilizzo semplificato con il linguaggio Python. OpenCV è una libreria specifica per la Computer Vision sviluppata originariamente da Intel e rilasciata sotto licenza Open Source BSD. La libreria è multiplatforma e può essere utilizzata su sistemi GNU/Linux, Mac OS X e Windows. La libreria è stata progettata principalmente per l'elaborazione di immagini in tempo reale e supporta le seguenti principali funzionalità:

- cattura di immagini in tempo reale;
- import di file video;
- elaborazione di immagini (luminosità, contrasto, soglie, ecc);
- riconoscimento di oggetti (facce, corpi, ecc);
- riconoscimento di forme con caratteristiche omogenee (blob).

La vastità e versatilità delle funzionalità di OpenCV ne rendono abbastanza difficile l'utilizzo da parte degli utenti "normali" che trovano difficoltà nello scegliere e nell'organizzare in applicazioni compiute la miriade di "mattoncini" disponibili.

Per fortuna ci viene in aiuto una seconda libreria che, pur limitando in parte le funzionalità di OpenCV, ne facilita enormemente l'utilizzo. Stiamo parlando di SimpleCV, un "framework" scritto e utilizzabile in Python, "semplice" da utilizzare, che incatola funzionalità della libreria OpenCV e algoritmi di trattamento immagini in "mattoni" di più alto livello che semplificano la vita

degli sviluppatori che desiderano realizzare applicazioni di visione artificiale, senza necessariamente essere in possesso di conoscenze approfondite di computer vision acquisite nei più prestigiosi istituti di ricerca mondiali. Prima di iniziare a "lavorare" diamo un'ultima indicazione sulla complessità dei problemi di visione computerizzata con alcuni esempi (**Tabella 1**) di cosa è facile e cosa non lo è per niente nelle applicazioni di computer vision.

PREPARIAMO L'ATTREZZATURA

Il nostro sistema di visione è composto dalla nostra inseparabile scheda Raspberry Pi, dal modulo Camera Pi che abbiamo presentato nel numero 185 della rivista e dalle librerie che abbiamo descritto in precedenza con le loro dipendenze. In realtà le librerie di visione computerizzata sono state progettate per individuare e gestire le periferiche di visione sulle porte USB mentre il modulo Camera Pi utilizza un driver dedicato che la gestisce interfacciando direttamente il connettore CSI-2. Per sopperire a

questo problema, nei programmi Python di esempio che vi presentiamo, useremo l'accortezza di acquisire un fotogramma mediante i comandi di gestione di Camera Pi (li abbiamo descritti nel numero 185) per poi caricare in memoria la relativa immagine ed elaborarla utilizzando le funzioni della libreria SimpleCV. Se per seguire questi esempi non volete utilizzare il modulo Camera Pi, nel prosieguo di questi articoli descriveremo anche le modalità di utilizzo di una webcam collegata ad una porta USB. Per ora potete consultare la lista delle webcam compatibili all'indirizzo http://elinux.org/RPi_USB_Webcams. Possiamo cominciare. Quello che dobbiamo realizzare è una architettura applicativa con tutti i moduli necessari al corretto funzionamento della libreria OpenCV e SimpleCV all'interno del sistema operativo di Raspberry Pi.

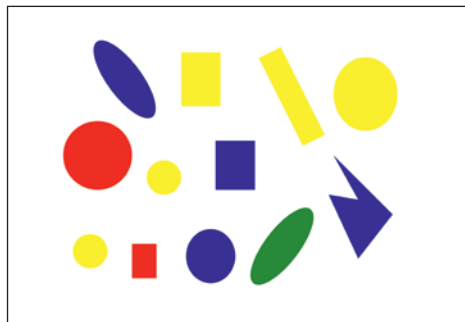
Partiamo dalla configurazione come l'abbiamo lasciata nell'articolo dedicato alla Camera Pi sul numero 185 della rivista, con la Camera Pi installata, Raspberry

Fig. 5

```

-----
Data and other attributes defined here:
AQUAMARINE = (127, 255, 212)
AZURE = (0, 127, 255)
BACKGROUND = (0, 0, 0)
BEIGE = (245, 245, 220)
BLACK = (0, 0, 0)
BLUE = (0, 0, 255)
CHARCOAL = (70, 70, 70)
CRIMSON = (220, 20, 60)
CYAN = (0, 255, 255)
DEFAULT = (0, 0, 0)
FOREGROUND = (255, 255, 255)
FORESTGREEN = (34, 139, 34)

```

Fig. 6

Pi collegata alla rete e con il monitor collegato all'uscita HDMI. Tastiera e mouse sono opzionali, se ci colleghiamo a Raspberry Pi in modalità remota utilizzando PuTTY e WINScp. Come sempre, diamo per acquisite le conoscenze di base per la gestione del sistema operativo, Raspbian, di Raspberry Pi, descritti negli articoli precedenti dedicati alla scheda e nel libro "Raspberry PI – Il mio primo Linux embedded". Prima di iniziare rinfreschiamo il sistema operativo con i comandi digitati dalla finestra di terminale alla quale ci siamo loggati con l'utente "root":

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

Proviamo il funzionamento della Camera Pi con il comando:

```
raspistill -t 5000
```

Dopo esserci accertati che tutto funzioni a dovere possiamo iniziare ad installare i componenti

software del nostro sistema di visione. Iniziamo ad installare la libreria OpenCv, i suoi pacchetti prerequisiti e le sue dipendenze. In particolare IPython è un interprete Python "potenziato" che offre una shell interattiva particolarmente ricca di funzionalità, in grado di integrarsi con altri linguaggi e ambienti di sviluppo, come, appunto, SimpleCV. Installiamo tutto con il comando (Fig. 1):

```
apt-get install ipython
python-opencv python-
scipy python-numpy
python-setuptools
python-pip
```

Poi passiamo ad installare SimpleCV come pacchetto di Python, ricordiamo che pip (Python Package Index) che abbiamo appena installato, è il pacchetto che ci permette di gestire i moduli aggiuntivi del linguaggio Python (Fig. 2):

```
pip install https://
github.com/sightmachine/
SimpleCV/zipball/master
```

Installiamo l'ulteriore dipendenza, necessaria per il funzionamento di SimpleCV, "svgwrite". Svgwrite si occupa della gestione del formato vettoriale delle immagini (SVG sta per Scalable

Vector Graphics). Essendo una estensione al linguaggio Python la installiamo con il comando pip (Fig. 3):

```
pip install svgwrite
```

Una volta terminate le installazioni possiamo eseguire le prime verifiche sfruttando la shell interattiva della quale SimpleCV è provvista (una funzione equivalente a quella disponibile per il linguaggio Python). Utilizziamo la shell per verificare il corretto funzionamento della libreria OpenCV. Digitiamo il comando (Fig. 4):

```
simplecv
```

La shell si apre informandoci sulla versione e presentando un riassunto di comandi. Per uscire digitiamo:

```
exit()
```

Prima di cominciare a sperimentare ci soffermiamo per introdurre, sia pure molto sommariamente, le modalità di digitalizzazione e memorizzazione delle immagini. Al di là dei numerosi formati, che si diversificano per la modalità bianco e nero e o a colori e per la qualità di restituzione (numero di pixel e numero di colori), le immagini sono fonda-

Listato 1

```
import subprocess
from SimpleCV import Image
import time
subprocess.call("raspistill -n -w %s -h %s -o Listato1_1.bmp" % (640, 480), shell=True)
img = Image("Listato1_1.bmp")
img.show()
pixel = img[120, 200]
print pixel
img[120, 200] = (0,0,0)
pixel = img[120, 200]
img.save("Listato1_2.bmp")
print pixel
```


Fig. 7

```

root@raspberrypi:/home# python Listato1.py
error in original mmap
error in original mmap
(127.0, 16.0, 32.0)
root@raspberrypi:/home#

```

mentalmente rappresentate con uno o più array di pixel, piccoli e numerosi puntini che variano al variare della dimensione e qualità dell'immagine. Una immagine in bianco e nero da 640 x 480 pixel (appunto) sarà composta da 307.200 pixel (circa 0,3 megapixel) ciascuno con dimensione di uno o più bit, che contengono la gradazione di colore presente nel punto dell'immagine rappresentata dal pixel stesso. Una immagine rappresentata da pixel di un solo bit sarà un'immagine in bianco (0) e nero (1) senza sfumature di grigio, la cosiddetta immagine "binarizzata". Le immagini a colori richiedono una matrice di pixel per ciascun colore, in genere tre nella rappresentazione RGB (rosso verde blu). Anche qui la dimensione in bit di ciascun pixel caratterizza il numero di colori che possono essere gestiti. Nella **Tabella 2** sono visibili le combinazioni di valori dei pixel che rappresentano i colori principali nel formato a otto bit. In questa rappresentazione le terne di valori identici rappresentano tutte le gradazioni di grigio. Fortunatamente, lavorando con SimpleCV non è necessario conoscere tutte le rappresentazioni numeriche dei colori, in quanto i principali sono richiamabili per nome, come GREEN, RED, YELLOW, ecc. Digitando (**Fig. 5**):

`help(Color)`

nella shell di SimpleCV si ottiene la lista dei nomi di colore riconosciuti.

A proposito, il comando `help(qualcosa)` è il modo per

ottenere aiuto da SimpleCV; il menu dell'help si ottiene con `help()`. Una volta compresa la modalità di rappresentazione di un'immagine, diventa più semplice comprenderne le modalità di elaborazione. Per esempio, il flip orizzontale dell'immagine si risolve nell'invertire orizzontalmente ciascuna riga della matrice di pixel. Il primo pixel di ciascuna riga diventa l'ultimo, l'ultimo diventa il primo e così tutti quelli intermedi. La trasformazione di un'immagine a colori in un'immagine in bianco e nero consiste nel creare una unica matrice a partire dalle tre originarie dove ciascun pixel contiene la media dei tre valori dei corrispondenti pixel delle matrici originarie. Sostituendo i pixel dell'immagine con altri di colori contrastanti è possibile elaborare l'immagine aggiungendo testi e figure geometriche. Mediante operazioni algebriche sulle matrici di pixel è possibile sommare immagini o applicare effetti di vario genere. In un modo più complesso avviene la ricerca di forme all'interno di singole immagini e la ricerca di similitudini o differenze tra immagini. Ora vediamo alcuni semplici esempi per toccare con mano, anche se molto superficialmente, le problematiche legate all'elaborazione di immagini. Come base di riferimento utilizziamo un foglio sul quale abbiamo stampato alcune forme geometriche di formato, dimensione e colori diversi (**Fig. 6**). Sia il documento .doc che i listati di esempio sono disponibili sul sito www.elettronica.in.it.

Utilizzateli per sperimentare

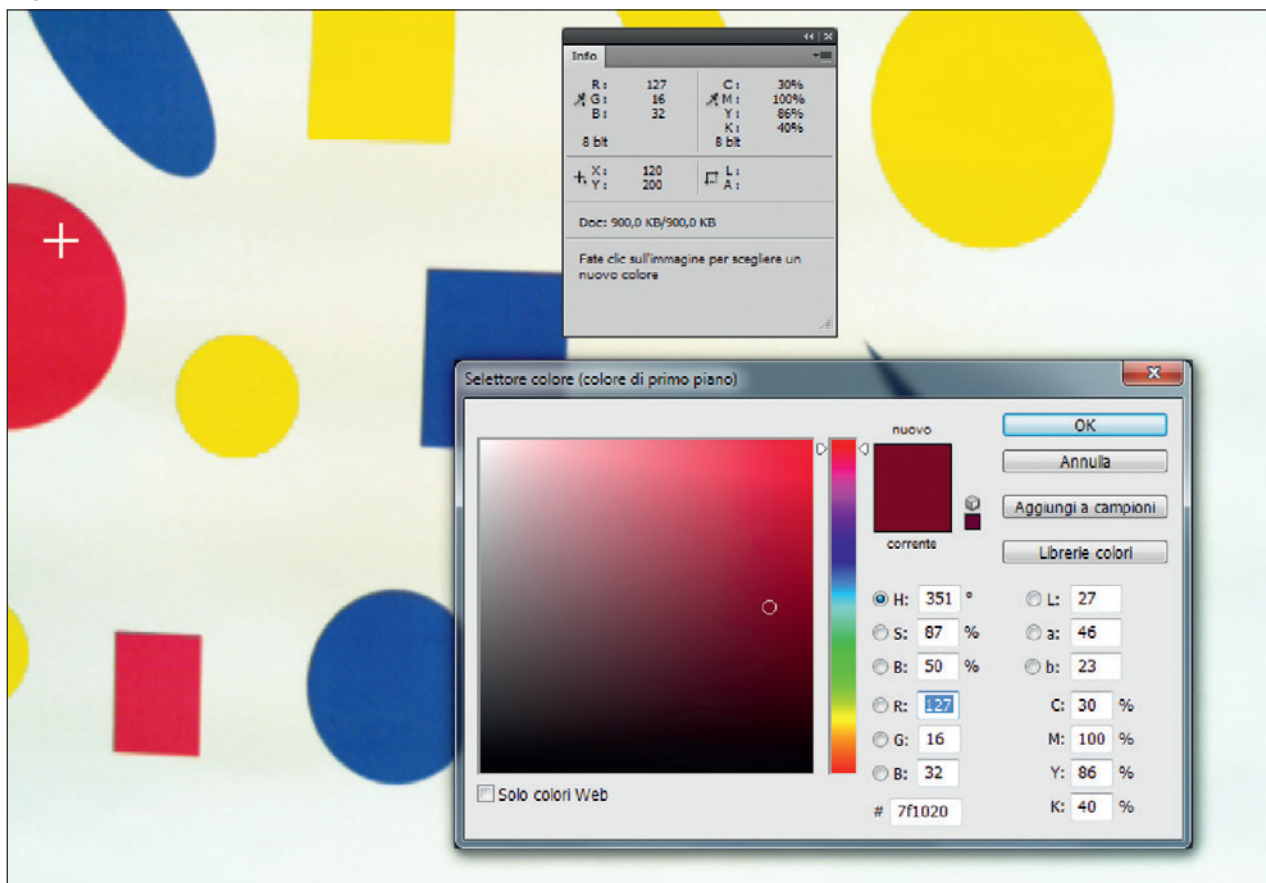
modificando il foglio di test ed i parametri di personalizzazione delle funzioni per comprendere le problematiche associate all'acquisizione ed elaborazione di immagini. L'illuminazione gioca un ruolo primario, così come le distanze, l'angolo di ripresa, le dimensioni, il contrasto e l'uniformità dei colori. Sperimentate. Per iniziare, nel **Listato 1** vediamo il riconoscimento del colore di uno specifico pixel all'interno dell'immagine. Per fare questo teniamo conto che gli elementi sono indicizzati a partire dall'angolo in alto a sinistra dell'immagine. L'indice del primo pixel in alto a sinistra dell'immagine ha valore (0,0). In particolare nel listato estraiamo il valore del pixel (120, 200). Si ottiene contando 120 pixel verso il basso e 200 pixel verso destra, più o meno al centro del primo cerchio rosso (circa) in alto a destra.

Nel **Listato 1**, la funzione "subprocess.call()" ci consente di lanciare un processo all'esterno del programma in esecuzione e di attendere il suo termine prima di proseguire l'elaborazione. Il processo esterno che viene lanciato è il programma "raspistill" con i suoi parametri, che ci permette di acquisire un fotogramma e di salvarlo su memoria di massa con il nome desiderato. La funzione "Image()" carica in una variabile il file immagine specificato. La funzione "show()" mostra l'immagine a video. Infine la funzione "img()" ci permette di accedere in lettura/scrittura ai contenuti dell'immagine stessa. Copiamo il programma del **Listato1** in un file di nome Listato1.py nella cartella /home e lo eseguiamo con il comando:

`python Listato1.py`

In questo ed in ciascuno dei lista-

Fig. 8



ti che seguono abbiamo inserito il salvataggio delle immagini intermedie che si ottengono durante il processo, in modo da poterle analizzare in seguito. Le immagini vengono salvate nella cartella / home, dalla quale possono essere prelevate (per esempio utilizzando WINScp) per aprirle con un programma di visualizzazione e/o editing grafico. Otterremo il risultato visibile in **Fig. 7** che ci presenta il valore del pixel come (127, 16, 32). Possiamo verificare la correttezza del risultato aprendo l'immagine con Photoshop e visualizzando il valore del pixel come visibile in **Fig. 8**. Corrispondono perfettamente. Per inciso chi non ha Photoshop può installare su Raspberry Pi il pacchetto di grafica open source "Gimp". Questo semplicissimo esempio ci permette di verificare concretamente l'aiuto derivante dall'impiego di SimpleCV nel trattamen-

to delle immagini. La funzione `img(120,200)` fa tutto il lavoro per noi. Per impostare un determinato pixel con un determinato colore usiamo l'istruzione:

```
img[120, 200] = (0,0,0)
```

ovviamente sostituiamo i parametri (0, 0, 0) con i valori R, G, B del colore che desideriamo. Già questo ci permette di operare sulle immagini praticamente come vogliamo. Nella libreria SimpleCV sono disponibili una quantità di funzioni che permettono di generare linee, figure geometriche, riempire aree, impostare trasparenze, sovrascrivere testo, ecc. Facciamo un grosso salto ed iniziamo a descrivere come riconoscere una o più forme all'interno di un'immagine. Anche qui partiamo da un esempio "semplice", come il riconoscere delle aree colorate all'interno di

un'immagine con sfondo bianco. Le aree di colore uniforme, per le quali è possibile riconoscere un contorno, nell'elaborazione di immagini, sono definite blob (macchie). Naturale, quindi, che la funzione che riconosce le aree di colore contrastante con il fondo si chiami `findBlobs()`. Nel **Listato 2** vediamo l'applicazione di questa funzione. La funzione "binarize()" trasforma l'immagine originale in un'immagine in bianco e nero senza sfumature di grigio, in modo da "enfaticizzare" le "macchie". Non quelle gialle, però, troppo simili allo sfondo bianco (**Fig. 9**). Potete provare a "fissare" il valore di soglia che discrimina le "macchie" dallo sfondo dando un valore (0-255) al parametro della funzione `binarize()`, per esempio `binarize(130)`. La funzione `findBlobs`, oltre a riconoscere le "macchie" restituisce

Fig. 9**Fig. 10**

```

192.168.0.43 - PuTTY
root@raspberrypi:~/home# python Listato3.py
Areas: [ 2374.5  5592.5  6098.  6745.5  7339.5  7491.  10466. 1
Angles: [ 90.  -88.95837402  90.  54.46232224  45.6902771
Centers: [[171 419]
[330 256]
[206 413]
[146 109]
[550 333]
[410 398]
[ 96 236]]
root@raspberrypi:~/home#

```

anche una serie di informazioni per ciascuna area individuata, come la dimensione, l'angolo (direzione) rispetto all'asse x e le coordinate del centro dell'area (Fig. 10). Queste informazioni sono molto utili nelle applicazioni di robotica.

Nel **Listato 3** troviamo un esempio di visualizzazione delle

informazioni suddette. Vengono evidenziati tre array nei quali sono contenute le informazioni appena citate. L'ordine degli array è per dimensione decrescente dell'area (blob) riconosciuta. Viene indicata per prima la "macchia di dimensioni maggiori e poi, via via, tutte le altre. Possiamo anche introdurre dei filtri di se-

lezione per restringere il campo di ricerca di eventuali aree di colore contrastante. Un criterio è, appunto, il colore. Possiamo decidere di far riconoscere solo le aree di colore blu, ed evidenziarne i contorni con linee di colore contrastante. Lo facciamo con il programma del **Listato 4**. Il "colore" di riferimento (22, 36, 75) lo

Listato 2

```

#!/usr/bin/python

import subprocess
from SimpleCV import Image
import time
subprocess.call("raspistill -n -w %s -h %s -o Listato2_1.png" % (640, 480), shell=True)
img = Image("Listato2_1.png")
img.show()
time.sleep(5)
img = img.binarize()
img.show()
time.sleep(5)
macchie = img.findBlobs()
img.save("Listato2_2.png")
img.show()
time.sleep(20)

```

Listato 3

```

#!/usr/bin/python

import subprocess
from SimpleCV import Image
import time
subprocess.call("raspistill -n -w %s -h %s -o Listato3_1.png" % (640, 480), shell=True)
img = Image("Listato3_1.png")
img.show()
time.sleep(5)
img = img.binarize()
img.show()
time.sleep(5)
macchie = img.findBlobs()
img.show()
time.sleep(5)
print "Areas: ", macchie.area()
print "Angles: ", macchie.angle()
print "Centers: ", macchie.coordinates()

```


Listato 4

```
#!/usr/bin/python

import subprocess
from SimpleCV import Color, Image
import time
subprocess.call("raspistill -n -w %s -h %s -o Listato4_1.png" % (640, 480), shell=True)
img = Image("Listato4_1.png")
img.show()
time.sleep(5)
colore = (22,36,75)
blue_distance = img.colorDistance(colore).invert()
blobs = blue_distance.findBlobs()
blobs.draw(color=Color.BLACK, width=4)
blue_distance.save("Listato4_2.png")
blue_distance.show()
time.sleep(5)
img.addDrawingLayer(blue_distance.dl())
img.save("Listato4_3.png")
img.show()
time.sleep(5)
```

abbiamo individuato sempre con il metodo "Photoshop" o "Gimp". La funzione "colorDistance()" sottrae dall'immagine i pixel che rappresentano colori "abbastanza" diversi dal colore di riferimento, mentre conserva i pixel che rappresentano colori "abbastanza" simili. Una volta riconosciute le "macchie" desiderate, le evidenziamo con un contorno nero di tre pixel di spessore con la funzione "draw()".

Dopo aver lanciato il programma molto probabilmente troverete che viene riconosciuto anche l'ellisse verde scuro, in effetti un colore "abbastanza" simile al blu (Fig. 11 e Fig. 12). Le condizioni

di luce possono influire molto sui risultati del riconoscimento. Ancora una volta ... sperimentate. Il passo successivo è quello di riconoscere contorni di forma particolare, come linee, cerchi e spigoli. Consideriamo una linea come una sequenza di punti che giacciono su un segmento di retta, che generalmente costituiscono parte di un contorno di una forma od una figura. Il metodo utilizzato dalla funzione di SimpleCV "findLine()" si basa sulla trasformata di Hough. In parole povere con questa tecnica si analizzano tutte le possibili linee che passano per i punti del piano bidimensionale (la nostra imma-

gine). A tutti i punti dell'immagine vengono assegnati punteggi crescenti man mano che questi appaiono giacere su un segmento che congiunge due punti, come ad esempio i vertici di una forma geometrica.

Lanciamo il programma e probabilmente troveremo che qualche linea retta non viene riconosciuta (magari il foglio non è perfettamente piano) mentre potrebbero essere riconosciute linee "abbastanza" rettilinee come parti dei "lati" lunghi degli ellissi (Fig. 13). Utilizziamo la funzione find.Circle() per riconoscere figure circolari all'interno dell'immagine. La funzione dispone di tre

Fig. 11

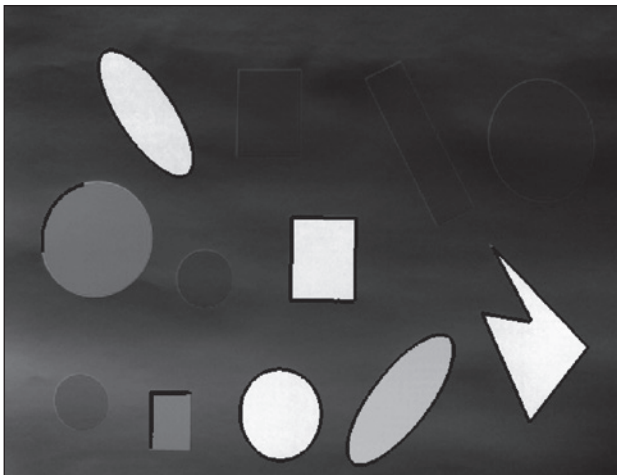
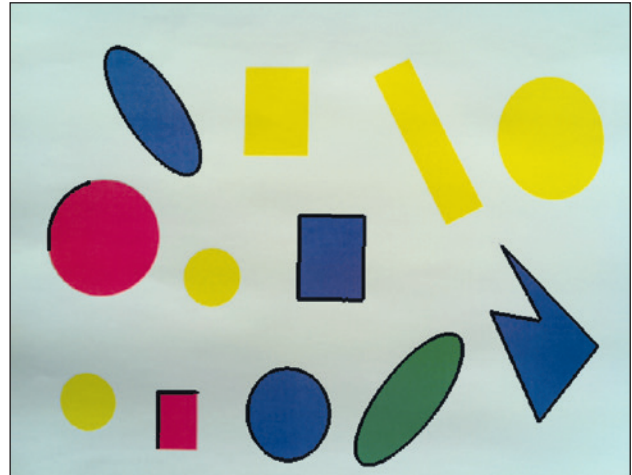


Fig. 12



Listato 5

```
#!/usr/bin/python

import subprocess
from SimpleCV import Color, Image
import time
subprocess.call("raspistill -n -w %s -h %s -o Listato5_1.png" % (640, 480), shell=True)
img = Image("Listato5_1.png")
img.show()
time.sleep(5)
# Imposta il valore di soglia
lines = img.findLines(threshold=30)
lines.draw(color=Color.BLACK, width=4)
img.save("Listato5_2.png")
img.show()
time.sleep(15)
```

Listato 6

```
#!/usr/bin/python

import subprocess
from SimpleCV import Color, Image
import time
subprocess.call("raspistill -n -w %s -h %s -o Listato6_1.png" % (640, 480), shell=True)
img = Image("Listato6_1.png")
img.show()
time.sleep(5)
cerchi = img.findCircle(canny=250, thresh=200, distance=11)
cerchi.draw(color=Color.BLACK, width=4)
cerchi.show()
time.sleep(5)
cerchi = cerchi.sortArea()
cerchi[0].draw(color=Color.RED, width=4)
img_with_circles = img.applyLayers()
img_with_circles.save("Listato6_2.png")
img_with_circles.show()
time.sleep(15)
```

parametri. Canny rappresenta il livello di precisione (astuzia) adottata dall'algoritmo di identificazione delle forme circolari. Il valore intermedio è 100, con un valore più alto, l'algoritmo diventa più schizzinoso e ricono-

sce un numero minore di cerchi. Con un valore più basso l'algoritmo diventa più approssimativo riconoscendo forme anche non perfettamente circolari. Thresh è il livello di soglia, come nel caso del riconoscimento di linee

rettilinee. Distance individua la distanza minima in base alla quale determinare se la forma riconosciuta consiste di un solo cerchio o di più cerchi concentrici. Il resto delle istruzioni ordina i cerchi riconosciuti in ordine cre-

Fig. 13

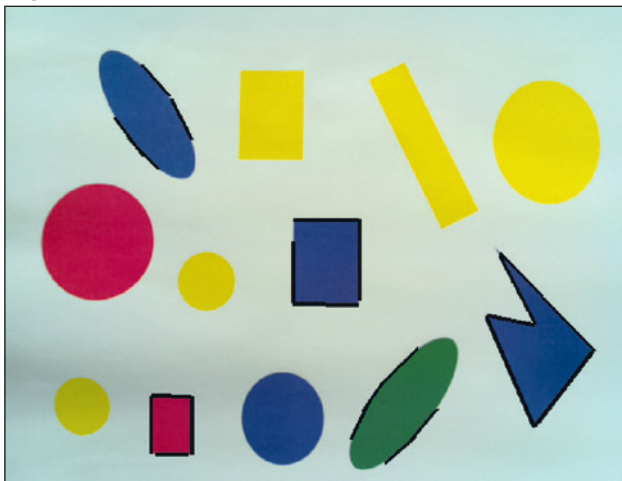


Fig. 14

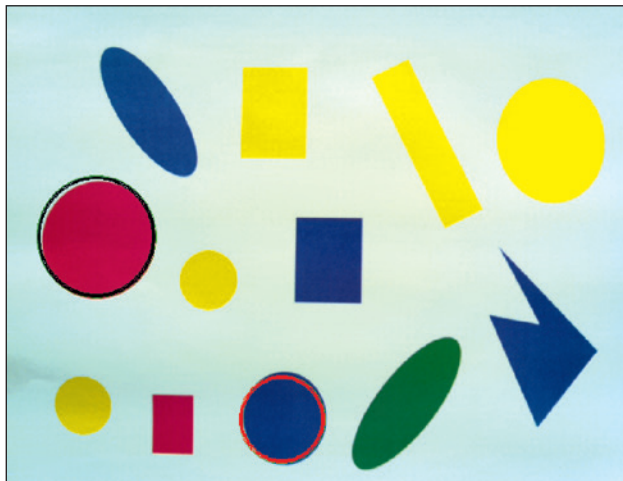
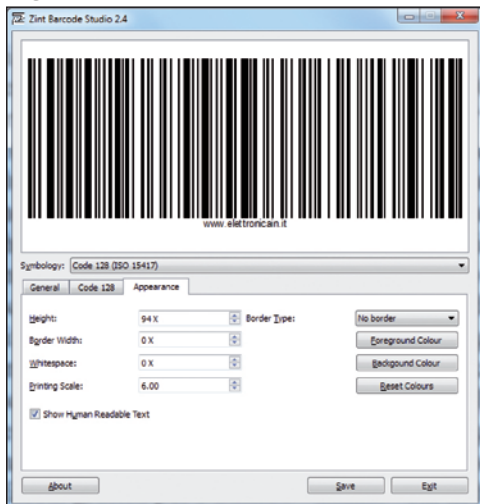
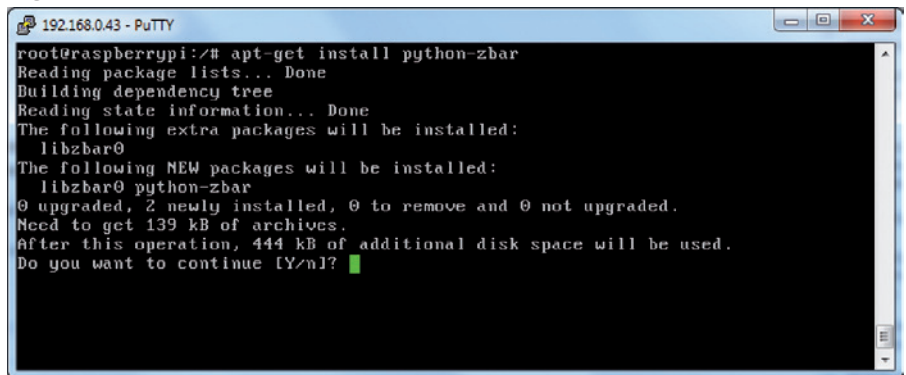


Fig. 15

scenze in base all'area di ciascuno. Il più piccolo, circles[0], viene evidenziato con un contorno rosso mentre tutti gli altri sono evidenziati da un contorno nero (Fig. 14).

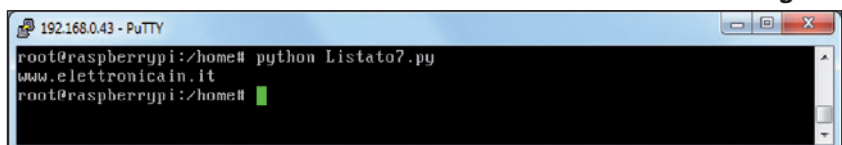
Lanciate il programma e ... divertitevi. Provate a modificare i parametri. Provate con fogli sui quali avete stampato cerchi senza riempimento, con la circonferenza di colori diversi. Con illuminazioni di intensità diverse, ecc. Spero di avervi convinto che la visione computerizzata "non" è una scienza esatta ma richiede di volta in volta, a seconda dei problemi, analisi approfondite e ... molta sperimentazione prima di adottare soluzioni definitive. In compenso stimola la fantasia ed in un certo senso diverte e stupisce. Per concludere questa prima carrellata sull'elaborazione di immagini presentiamo la funzione SimpleCV che permette

Fig. 16

di leggere i codici a barre. Ovviamente si chiama "findBarcode()" ed è in grado di riconoscere in automatico il tipo ed il contenuto dei codici a barre presenti nell'immagine. Sono riconosciuti anche i QR code (Quick Response code), particolari codici grafici bidimensionali ormai diffusissimi per veicolare informazioni eterogenee in particolare verso dispositivi mobili.

Un crittogramma QR Code può contenere sino a 7.089 caratteri numerici o 4.296 alfanumerici.

Per la generazione dei codici utilizziamo il programma Zint Barcode Studio, un prodotto Open Source che ci permette di generare codici a barre e QR Code di ogni tipo e salvarli come immagini sia in formato bitmap che vettoriale (Fig. 15). Zint Barcode Studio è disponibile per diversi sistemi operativi sia con interfaccia grafica per l'utilizzo "umano" che in forma di libreria per l'utilizzo a programma (Zint Barcode Generator). Per poter interpretare le immagini che con-

Fig. 17**Fig. 19****Fig. 18**

Listato 7

```
#!/usr/bin/python

import subprocess
from SimpleCV import Image, Barcode
import time
subprocess.call("raspistill -n -w %s -h %s -o Listato7_3.png" % (640, 480), shell=True)
img = Image("Listato7_3.png")
img.show()
time.sleep(5)
# Load a one-dimensional barcode
barcode = img.findBarcode()
print barcode[0].data
```

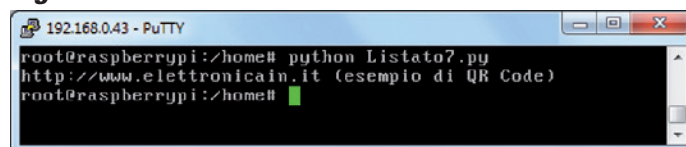

CIRCUITI STAMPATI DI *qualità* AD UN PREZZO ECCEZIONALE

NUOVA **S.C.E.** S.N.C.
dei F.lli D'ASTOLI

PRODUZIONE E CAMPIONATURA
CIRCUITI STAMPATI
ANCHE SU BASE DI ALLUMINIO
DOPPIA FACCIA - MULTISTRATO - CIVILI



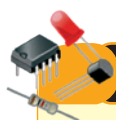
Fig. 20



tengono un codice a barre dobbiamo aggiungere al "corredo" di SimpleCV la libreria "zbar". La installiamo con il comando (Fig. 16):

apt-get install python-zbar

Abbiamo preparato un paio di barcode stampati su fogli di carta e li abbiamo inquadrati con la telecamera. Il programma ve lo presentiamo nel **Listato 7**. Nelle **Fig. 17** e **Fig. 18** sono visibili un codice a barre tradizionale con il risultato dell'elaborazione mentre le **Fig. 19** e **Fig. 20** si riferiscono all'esempio con il QR Code. Più semplice di così. Ancora una nota prima di darci appuntamento ad un prossimo articolo. Tutti gli esempi che abbiamo descritto funzionano anche senza la presenza di una telecamera. Possono tranquillamente elaborare anche serie di immagini acquisite in precedenza o provenienti da fonti diverse. Basta eliminare le istruzioni di acquisizione dell'immagine e "puntare" la funzione "image()" verso il percorso/file che contiene l'immagine che vogliamo elaborare. Nel prosieguo faremo un ulteriore salto e approfondiremo le funzionalità disponibili per il riconoscimento e la elaborazione di contenuti "biologici" nelle immagini, come le facce e parti di esse. Per ora potete allenarvi a sperimentare i diversi modi di "vedere" del nostro cervello e di un sistema computerizzato. ■



per il MATERIALE

La board Raspberry Pi tipo B (cod. 7310-7568308RS) costa 38,00 Euro mentre la Camera Module per Raspberry Pi (cod. 7310-7757731RS) costa 31,00 Euro. È disponibile anche un contenitore plastico per la sola telecamera (cod. 5100-RPI-CAM.9 Euro 6,80) ed un contenitore trasparente in grado di accogliere sia la telecamera che la board (cod. 5100-TEK-CAM.0, Euro 13,00). I prezzi si intendono IVA compresa.

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

INTERFACCIA WEB PER RADIO INTERNET CON RASPBERRY PI

di MARCO MAGAGNIN



Arricchiamo il progetto Internet radio con un'interfaccia utente web, open source, professionale e fruibile da browser su PC e dai dispositivi mobili.

In tutti gli articoli e nei libri dedicati all'argomento embedded abbiamo sempre sottolineato l'importanza, quando si affronta un progetto embedded basato su GNU/Linux, di una buona progettazione e della conoscenza dei "mattoni" già messi a disposizione dalla comunità degli sviluppatori. Parte peculiare della progettazione è la corretta scelta, tra le alternative, dei "mattoni" che ci garantiscano sia la possibilità di portare a termine il nostro progetto, che di espanderlo in futuro ed adattarlo a nuove esigenze senza dover rimettere in

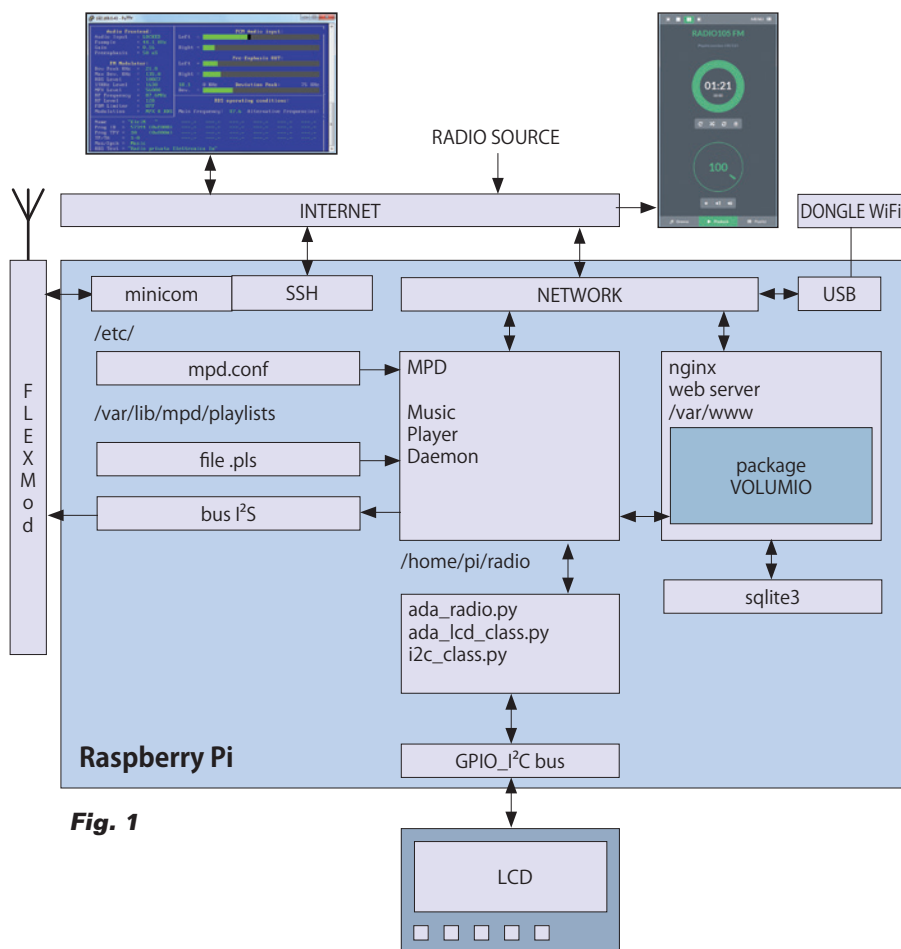


Fig. 1

discussione l'intera architettura. Negli articoli precedenti, dedicati al mondo della radio Internet, sia in ricezione che in trasmissione, tra gli altri componenti, abbiamo privilegiato il pacchetto MPD (Music Player Demo). I motivi per i quali lo abbiamo scelto si possono riassumere nei seguenti, che possono essere utilizzati anche come filo conduttore per adottare componenti in genere, da integrare nei nostri progetti embedded:

- Funziona bene con l'hardware di Raspberry Pi, e con le sue specifiche periferiche audio, oltre ad essere utilizzabile su molte altre piattaforme e con molteplici dispositivi audio. Questo permette di capitalizzare e riutilizzare esperienze e applicazioni già realizzate;
- L'architettura è separata in un modulo server e un'interfaccia

client open source che permette di adottare o realizzare i client adatti alle proprie esigenze;

- È stata adottata da diverse comunità di sviluppo come base di riferimento per le proprie realizzazioni e, in particolare, sono disponibili molte tipologie di client che si integrano con il modulo server.

Avendo scelto questo pacchetto come cuore della nostra applicazione di radio Internet, descritta nel numero 185 della rivista, che permette di gestire una lista di stazioni radio Internet istruendone l'output verso l'uscita audio. Con poche modifiche alle configurazioni abbiamo integrato il progetto iniziale con il modulo FLEXMod della Tecnoroll, che permette di ritrasmettere via etere il segnale audio delle radio Internet. Questa applicazione

sfrutta l'uscita audio I²S, peculiare di Raspberry Pi, in questa gamma di dispositivi. In questo articolo, oltre ad approfondire lo strumento di amministrazione del modulo FLEXMod, descriviamo un'ulteriore espansione della nostra applicazione, che permette di dotarla di un'interfaccia grafica web funzionalmente molto completa e curata dal punto di vista "grafico", in grado di adattarsi a diversi dispositivi di connettività, come browser web su PC, tablet e smartphone. Stiamo parlando del pacchetto Volumio, realizzato, guarda caso, come un client molto elaborato per MPD. Potremo impiegare Volumio sia per gestire la versione "ricevitore" che "trasmettitore" del nostro progetto, funzionante in modo integrato sia con lo shield LCD, che senza quest'ultimo. La configurazione completa è visibile nello schema di **Fig. 1**. Questo, ancora una volta, per dimostrare che la scelta di utilizzare componenti ben progettati e correttamente integrati permette molta flessibilità e, in un certo senso, semplicità, nella realizzazione ed evoluzione delle applicazioni. Volumio è anche l'occasione per presentare un'applicazione di livello professionale realizzata con tecnologie che ne permettono la fruibilità da diverse tipologie di dispositivi adattandosi automaticamente. Stiamo parlando del framework jQuery Mobile. Per descrivere, anche sommariamente, questa tecnologia, dobbiamo introdurre un elemento che non abbiamo mai toccato finora: il concetto di "foglio di stile" o CSS (Cascading Style Sheets). Il CSS è un linguaggio di descrizione di formati grafici, che permette di gestire tutto il layout di un sito Web, in modo da sollevare la componente HTML da questa

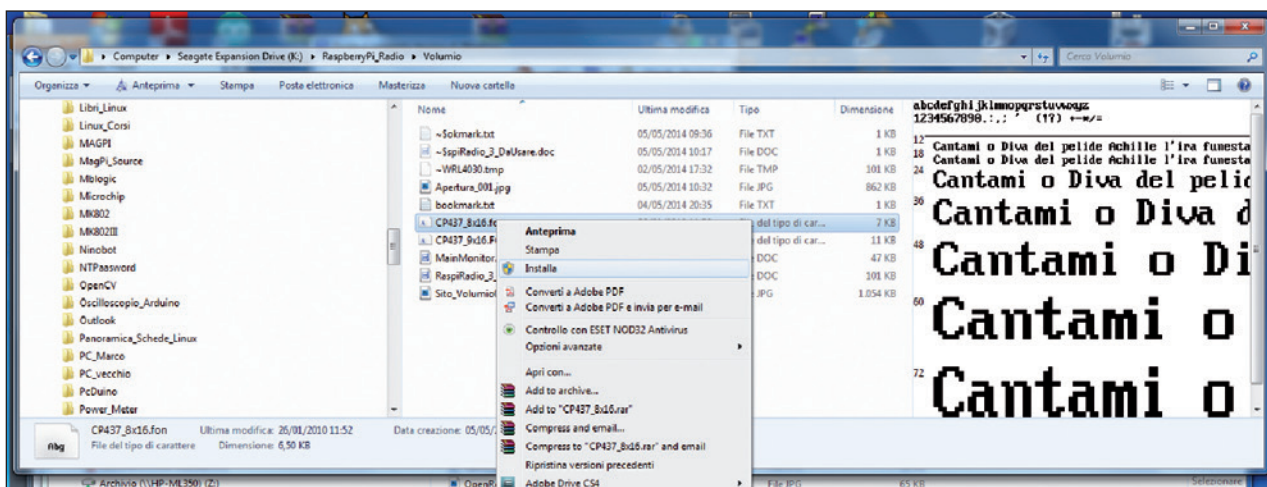


Fig. 2

incombenza. Più semplicemente, permette di tenere separate le descrizioni dei contenuti (HTML) dall'aspetto grafico con il quale le si vuole rappresentare (CSS). In questo modo diventa possibile confezionare gli stessi contenuti in modi diversi ed anche adattarli a diversi dispositivi di visualizzazione. Il framework jQuery, sul quale si appoggia jQuery Mobile utilizza Javascript per combinare codice HTML e CSS ed offrire una serie di strutture "preconfezionate" per la realizzazione di siti web. L'offerta è completata da librerie di oggetti funzionali come pulsanti, campi di testo, liste di selezione a tendina, check box, combo, slider, ecc. che possono essere configurati in diverse forme grafiche. Grazie ad essi è possibile intervenire sulla formattazione del testo, sul posizionamento degli elementi grafici e sulla disposizione che questi elementi avranno rispetto a diversi media e device. jQuery Mobile è uno strato ancora superiore a jQuery (devono essere utilizzati assieme) che permette di adattare "automaticamente" le pagine del sito web in costruzione, ai diversi dispositivi mobili previsti. Approfondire la conoscenza ed imparare a utilizzare questi strumenti richiede uno sforzo di formazione ed un impegno di tempo notevole. Se mai dovete

intraprendere questo cammino, sappiate comunque che porta con sé notevoli vantaggi. jQuery e jQuery Mobile sono sempre più diffusi e utilizzati nello sviluppo di applicazioni web ed in particolare quelle destinate ai dispositivi mobili. Una volta realizzata un'applicazione con questi strumenti, rispettando alcuni vincoli, è possibile trasformarla in un'App per dispositivi specifici ed in particolare per l'ormai sempre più diffuso sistema operativo Firefox OS. In ogni caso le applicazioni realizzate funzionano lato server su pressoché tutte le piattaforme. Non ultimo il fatto che tutti gli strumenti per lo sviluppo, il debug, la simulazione e la gestione di queste applicazioni sono disponibili con licenze Open e quindi una volta tanto studiare costa solo fatica e non impegno economico. Infine, la gestione di queste applicazioni richiede meno risorse lato server e meno carico sulle linee di comunicazione, in quanto molti dei componenti delle pagine vengono memorizzati nella memoria cache del browser.

La sola controindicazione, se si può considerare tale, è che le applicazioni per essere fruite in modo grafico, richiedono l'utilizzo di browser compatibili con le specifiche WebKit. In realtà sono previsti diversi livelli

di compatibilità: A, B e C. Per approfondimenti consultate l'indirizzo <http://stackoverflow.com/questions/14524217/minimum-requirements-for-jquery-mobile>. Tutti si stanno adeguando, ma in particolare i browser Open Source come Firefox e Safari lo sono già da tempo e si mantengono compatibili con le nuove versioni dei framework. Dopo questa descrizione di massima vediamo alcune funzionalità aggiuntive del modulo FLEX-Mod, rispetto a quelle presentate nel numero 185. Abbiamo già visto nel numero precedente che il modulo FLEXMod mette a disposizione le proprie funzionalità di configurazione per mezzo di un'interfaccia a "caratteri" raggiungibile mediante collegamento seriale. In realtà ci siamo collegati dall'esterno utilizzando lo strumento PuTTY ed il pacchetto di emulazione terminale "minicom". Una delle funzionalità più interessanti, soprattutto per i cultori dell'audio di qualità è il monitor principale del modulo, raggiungibile da PuTTY più minicom con il comando:

MainMonitor

Prima però un attimo di pazienza, l'interfaccia MainMonitor è un'interfaccia grafica a "caratteri" gestita dal processore FPGA

Fig. 3

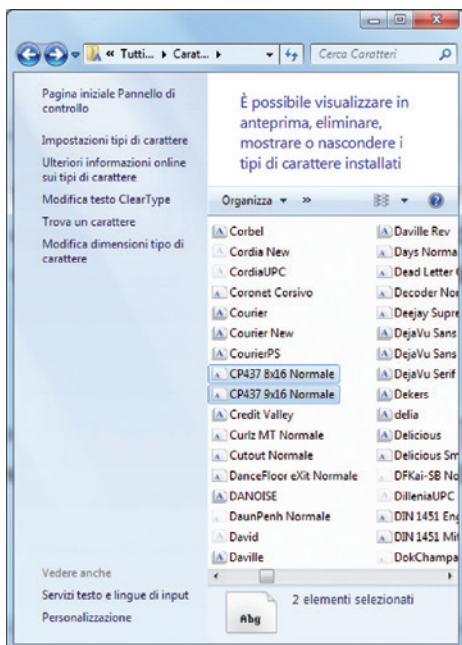


Fig. 4

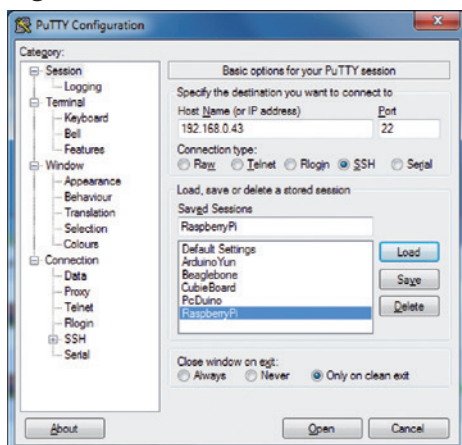
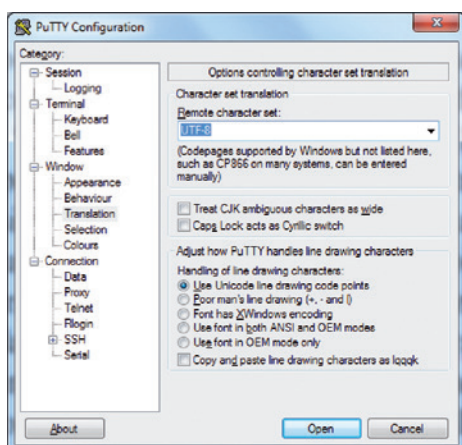


Fig. 5



del modulo stesso, e richiede di impostare gli strumenti di visualizzazione in base alla mappa caratteri CP437, per intenderci, quella dei vecchi terminali IBM. Per mappa caratteri si intende la collezione di "segni" grafici, ciascuno dei quali corrisponde ad uno specifico carattere ASCII o di altra codifica. In questo modo l'applicazione originale compone stringhe di caratteri, nel nostro caso UTF-8, e le invia sulla porta seriale.

L'emulatore di terminale legge la stringa e presenta sul video il "segno" grafico corrispondente utilizzando un font adatto. Per ottenere un buon risultato grafico, quindi, dobbiamo, se già non è presente, aggiungere alla nostra collezione di font Windows, il font CP437 e poi configurare PuTTY in modo da interpretare le stringhe seriali in modo corretto. Chiaramente tutto questo vale unicamente nel mondo Microsoft, chi utilizza PC con GNU/Linux non ha sicuramente bisogno di queste note, basta impostare l'emulazione della finestra terminale come "VT100" (vecchio terminale video di Digital). Cominciamo. Verificate in "Pannello di controllo>>Caratteri" l'esistenza di un font CP437. Se così non fosse, scaricate i font dal sito www.elettronica.in oppure all'indirizzo:

<http://neveradudelikethisone.com/2010/01/cp437-fonts-for-windows/>

Le istruzioni che seguono sono valide per Windows XP. Scompattate i file di caratteri in una cartella e installateli da "Pannello di controllo>>Caratteri" utilizzando la voce di menu "File>>Installa nuovo tipo di carattere...". Selezioniamo i file dove abbiamo scompattato i font

Fig. 6

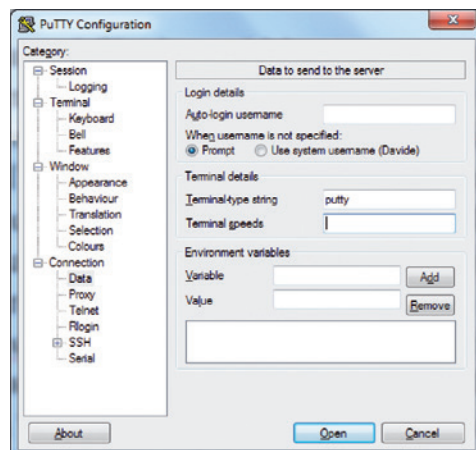
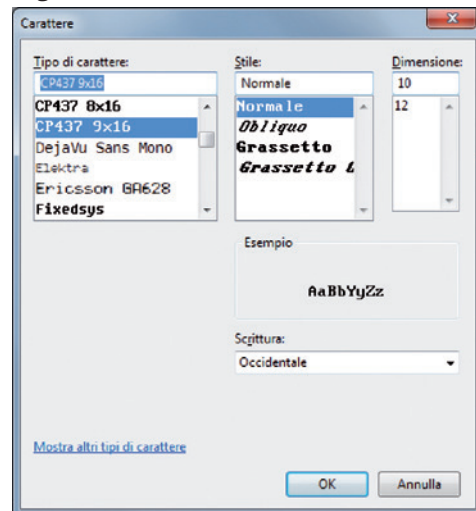
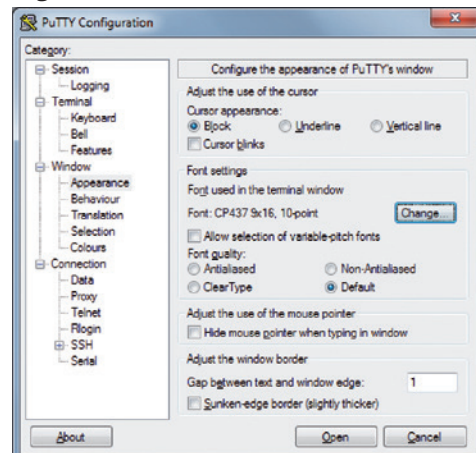


Fig. 7



e premiamo il pulsante "OK". In Windows 7, dopo aver scompattato i file di caratteri è sufficiente cliccarvi sopra con il tasto destro

Fig. 8



e scegliere la voce "Installa" come visibile in Fig. 2. In entrambi i casi possiamo controllare il buon esito dell'operazione visualizzando la finestra "Caratteri" del "Pannello di controllo" (Fig. 3).

Ora passiamo a configurare PuTTY. Lo apriamo e probabilmente avremo già configurato una sessione salvata col nome di "RaspberryPi" o simile. Possiamo caricare questa (pulsante "Load") oppure crearne una nuova (Fig. 4).

Nella sezione "Windows>>Traslation" impostiamo il box "Received data assumed to be in which character set" al valore "UTF-8". Nella sezione "Handling of line drawing character" selezioniamo il check box "Use Unicode line drawing code points" (Fig. 5). In "Connection>>Data" nel campo "Terminal type string" inseriamo il valore "putty". Eliminiamo il contenuto del campo "Terminal speeds" (Fig. 6).

Nella sezione "Window>>Appearance" selezioniamo il font "CP437 9x16" (Fig. 7 e Fig. 8).

Torniamo alla categoria "Session" e premiamo il pulsante "Save" (Fig. 9) per memorizzare le impostazioni in modo permanente.

Fig. 9

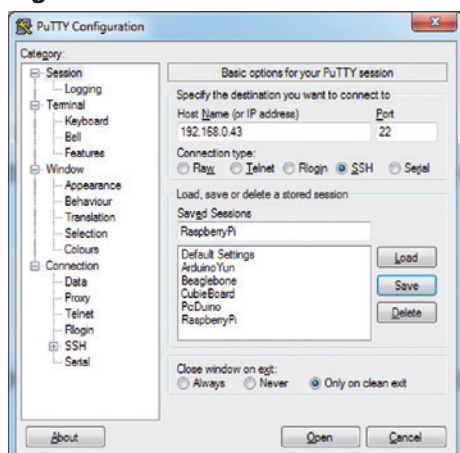


Fig. 10

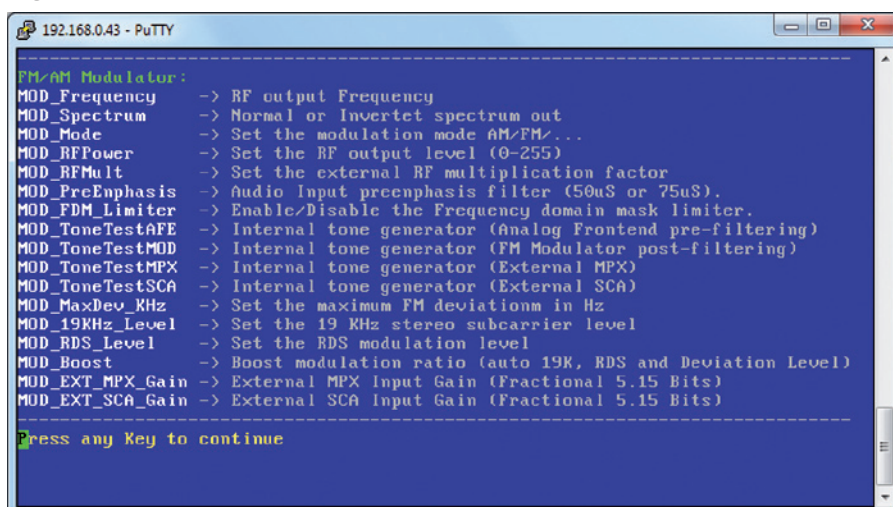
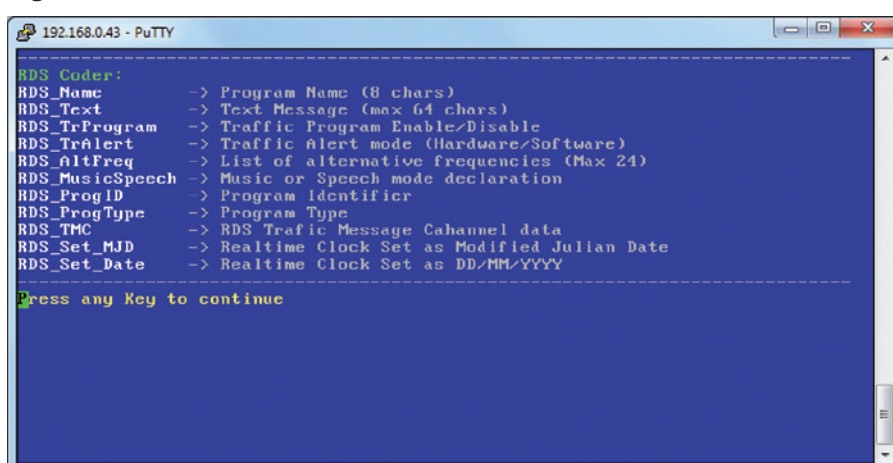


Fig. 11



Verifichiamo che tutto sia collegato correttamente a Raspberry Pi come descritto nel numero 185 e apriamo la finestra di terminale con il pulsante "Open". Entriamo con login (root) e password, ed al prompt del terminale digitiamo:

`minicom -c on`

Ci ritroviamo nello strumento di configurazione del modulo FLEXMod, se vogliamo approfondire i comandi digitiamo "help" e poi scorriamo le pagine con il tasto "Invio" (Fig. 10, Fig. 11, Fig. 12 e Fig. 13) oppure digitiamo direttamente:

"MainMonitor"

Il MainMonitor, visibile nella Fig. 14, è il "cruscotto" che riassume lo stato di funzionamento del modulo FLEXMod. Il pannello è diviso in diverse sezioni, che raggruppano le informazioni relative al funzionamento dei singoli processi del modulo. Tra i principali, vediamo lo stato dell'ingresso audio, la frequenza e la potenza di trasmissione FM, il nome e la descrizione RDS della radio. Il "Prog ID" è un campo particolare, che identifica univocamente l'emittente radio, assegnato dalle autorità di controllo. La finestra con le barre a scorrimento permette di visualizzare le caratteristiche del segnale stereo

Fig. 12

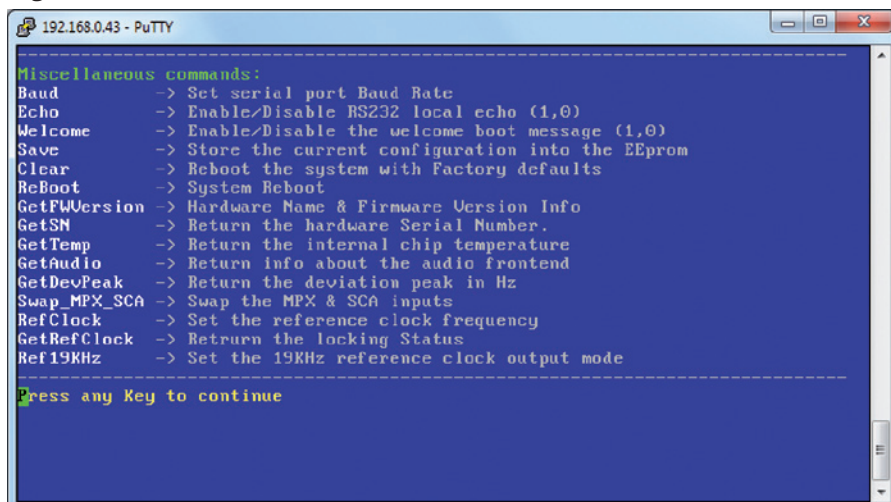


Fig. 13

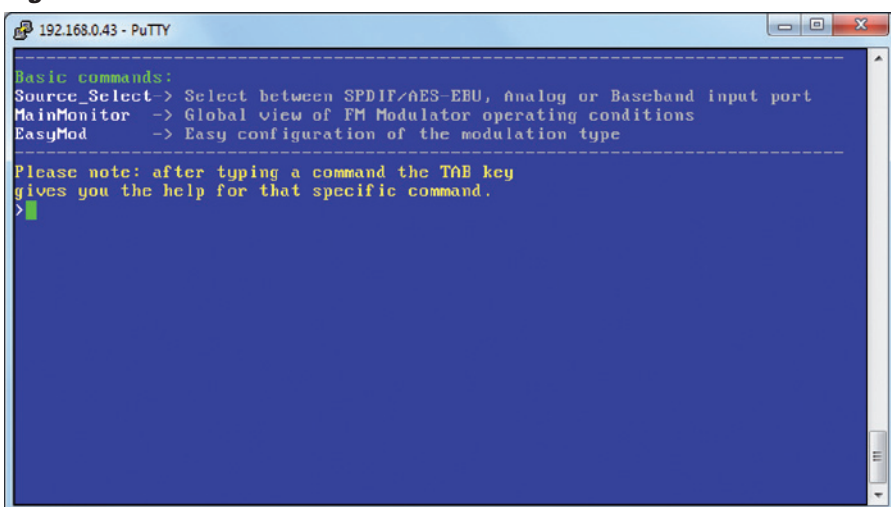
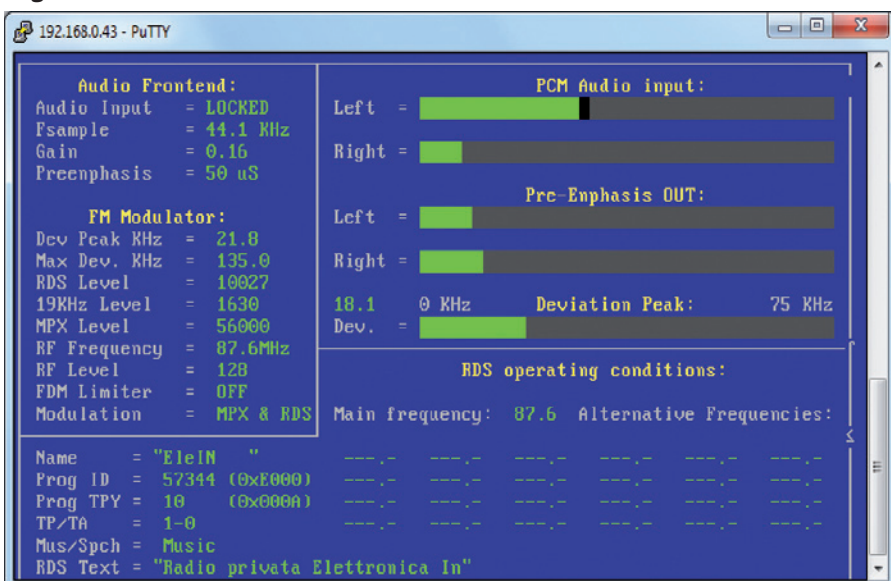


Fig. 14



in ingresso e dopo la fase di preenfasi. L'ultima finestra "RDS operating conditions" ci permette di introdurre una funzionalità interessante, che abbiamo sicuramente sperimentato utilizzando l'autoradio, probabilmente senza averci mai fatto troppo caso. La possibilità di mantenere l'ascolto sulla stessa emittente utilizzando frequenze alternative. Pensiamo di percorrere in auto una valle tortuosa dove la copertura FM non può essere garantita da una sola emittente. Più ripetitori sulla stessa frequenza generano sicuramente delle interferenze nelle aree dove i segnali si sovrappongono. Una possibilità è diversificare le frequenze, ma richiede di ricercare la nostra stazione in continuazione, durante il viaggio, per mantenere la continuità d'ascolto. La soluzione è di fare in modo che questo "inseguimento" avvenga in modo automatico. Ed è proprio ciò che avviene con la funzione RDS. Nella configurazione della stazione trasmittente si impostano tutte le frequenze alternative sulle quali è possibile sintonizzarsi per ascoltare l'emittente desiderata. A questo punto, il ricevitore, se dotato di questa opzione, quando il segnale dell'emittente in ascolto si affievolisce, ricerca automaticamente l'emittente ricevibile con il maggiore livello di segnale, tra quelle la cui frequenza è nell'elenco. Nel numero 185 abbiamo citato alcuni impieghi per la nostra emittente radio, per esempio quello che permette di realizzare una audio guida all'interno delle sale di un museo. Con questa funzionalità possiamo fare in modo che in ciascuna stanza sia udibile una descrizione personalizzata dei contenuti esposti nella stanza stessa, con il passaggio automatico da una descrizione all'altra semplicemente cambian-

Fig. 15

do stanza. La differenza rispetto alle trasmissioni broadcast sta nel fatto che nella valle tortuosa tutti i ripetitori trasmettono lo stesso segnale, mentre nel museo, ciascuno "ripetitore" viene pilotato con il segnale audio personalizzato per la stanza. Nel caso del museo, i muri divisori ci aiutano nel forzare la sintonizzazione su una emittente o l'altra in base al livello di segnale. A seconda della dimensione e delle condizioni di propagazione possiamo perfezionare la configurazione agendo anche sulla potenza di emissione per ciascuna delle emittenti. Il comando che ci permette di impostare le frequenze alternative è RDS_AltFreq seguito dalla lista delle frequenze, come per esempio:

```
RDS_AltFreq 99.4, 101.3, 102.7
```

Il comando per impostare la potenza di uscita FM dell'emittente è:

```
FM_RFPower <valore>
```

Ed accetta valori da 0 a 255, che rappresentano i passi di regolazione della potenza in uscita e non l'effettiva potenza del segnale, 128 è un buon valore di inizio.

Per chi volesse approfondire maggiormente il significato dei numerosi parametri di configurazione rimandiamo al manuale ufficiale del modulo FLEXMod reperibile all'indirizzo http://www.tecnoroll.it/prodotti/modulators/flexmod_fm/Modulatore_FM_V1.0.pdf.

INSTALLAZIONE VOLUMIO

Passiamo ora ad installare il client Volumio, che ci permette di gestire in remoto la nostra stazione di distribuzione audio, con un'interfaccia grafica

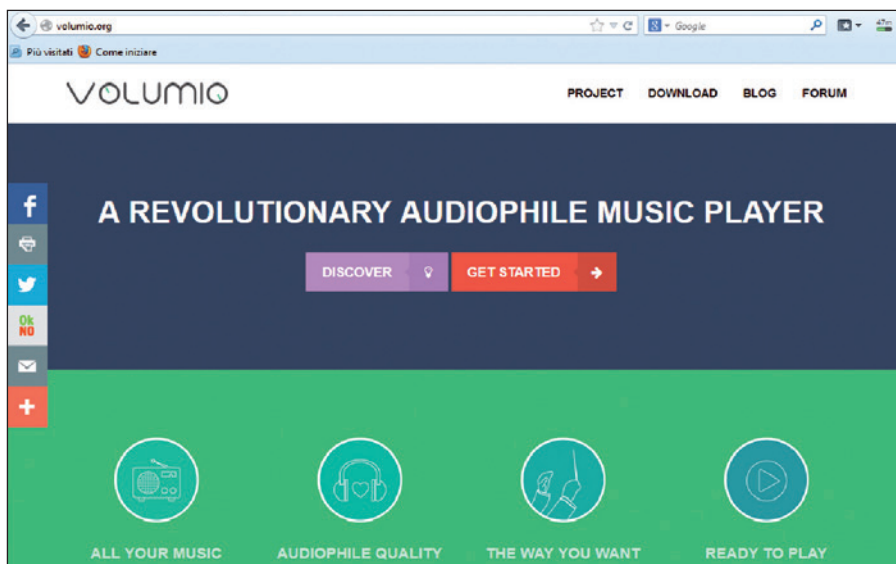


Fig. 16

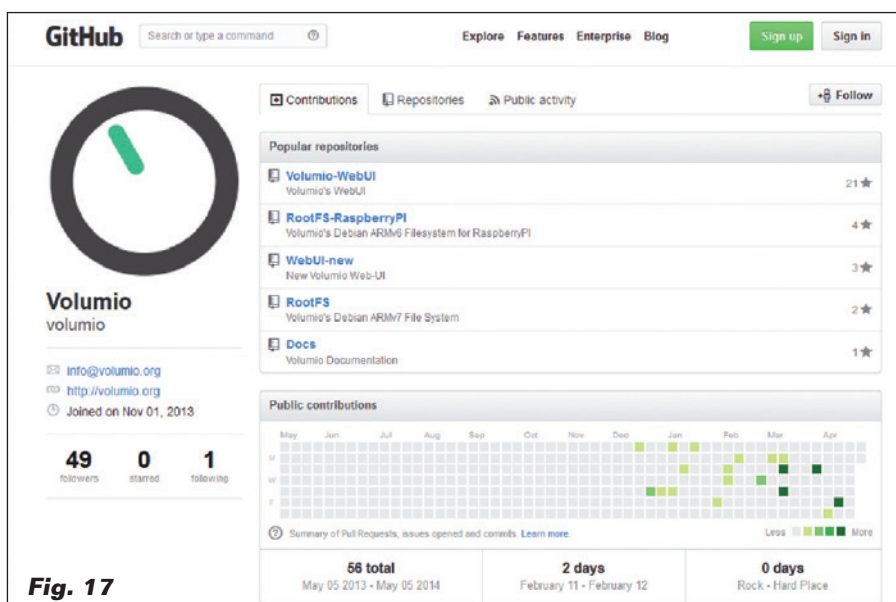


Fig. 17

Fig. 19

```

192.168.0.43 - PuTTY
root@raspberrypi:~# apt-get -y install nginx sqlite3 php5-fpm php5-curl php5-sqlite php5-cli

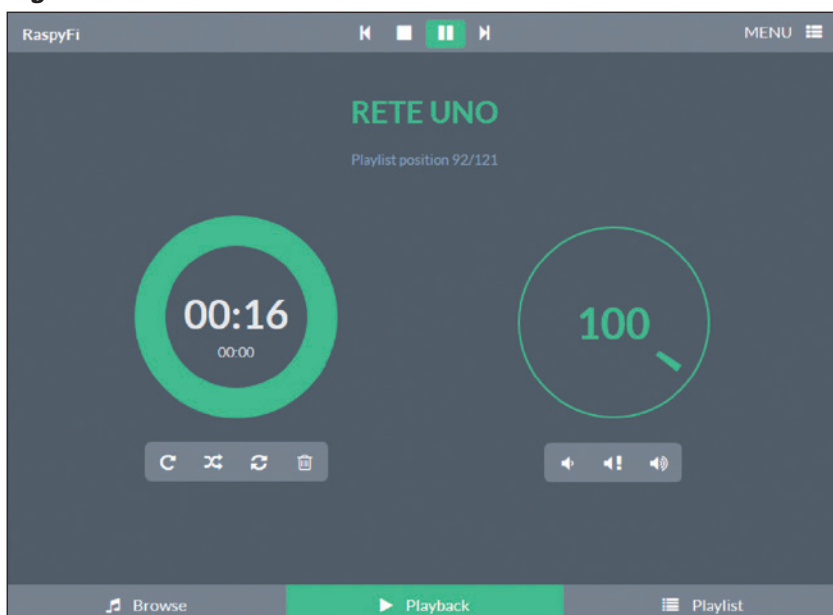
```

Fig. 20

```

192.168.0.43 - PuTTY
root@raspberrypi:~# cd /home
root@raspberrypi:/home# git clone https://github.com/volumio/Volumio-WebUI.git
Cloning into 'Volumio-WebUI'...
remote: Reusing existing pack: 762, done.
remote: Total 762 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (762/762), 1.64 MiB | 17 KiB/s, done.
Resolving deltas: 100% (355/355), done.
Checking out files: 100% (490/490), done.
root@raspberrypi:/home#

```

Fig. 21

decisamente professionale. Il sito di riferimento di Volumio è raggiungibile all'indirizzo <http://volumio.org> (Fig. 15 e Fig. 16) dove potremo trovare supporto e documentazione al prodotto oltre, ovviamente, alla sezione downloads (Fig. 17).

Se andate a curiosare troverete che Volumio è disponibile per diverse piattaforme, tra le quali Raspberry Pi. Per Raspberry Pi la comunità di Volumio ha deciso di confezionare una immagine di SD Card che contiene il sistema operativo Raspbian ed il pacchetto Volumio con tutte le

sue dipendenze già installate e configurate. Oppure è disponibile in formato sia sorgente che pacchettizzato nel repository GitHub, il repository "social" di custodia e controllo versioni di software Open Source. Noi abbiamo preferito installare Volumio pezzo per pezzo sulla configurazione che abbiamo ottenuto sino a questo momento, per diversi motivi. Il primo è che in questo modo salviamo tutto il lavoro fatto fino ad oggi, senza doverlo ripetere su una nuova distribuzione, questo nel rispetto dello spirito

GNU/Linux, piuttosto pigro nel rifare cose già fatte. Secondo, l'immagine scaricabile dal sito di Volumio è stata realizzata a partire da NOOBS (New Out Of Box Software), la distribuzione che offre un sistema di installazione multi sistema operativo, che abbiamo già descritto nel numero 177. Con questo sistema però, si perdono delle funzionalità, come l'impossibilità, utilizzando lo strumento raspi-config, di espandere la partizione fino ad utilizzare tutto lo spazio sulla SDCard. Terzo, operare in questo modo ci permette di addentrarci maggiormente nel mondo GNU/Linux, e capire le problematiche associate alla integrazione di diversi pacchetti in una unica applicazione. Abbiamo già descritto Volumio nella presentazione dell'articolo, approfondiamo qui solo alcuni aspetti tecnici, indispensabili per poterlo installare ed utilizzare. Volumio è un'applicazione web e quindi necessita di un server web per poter essere gestita. Il server web prescelto da Volumio è nginx, un'alternativa ad "apache2", con un funzionamento molto simile. Inoltre Volumio utilizza, come appoggio per i suoi dati di configurazione e gestione, il server database "sqlite". Se ci avete seguito sin qui, nella nostra configurazione avremo attivi sia i server apache2 che mysql. Possiamo semplicemente spegnerli o addirittura disinstallarli se stiamo dedicando il progetto unicamente alla gestione audio. Utilizziamo "rcconf" per disattivare i server.

Digitiamo il comando (Fig. 18):

```
rcconf
```

e nella pagina risultante, con il tasto barra cambiamo la configurazione di apache2 e mysql da "" a ". Scegliamo "OK" con il tasto

“Tab” e diamo “invio”. Possiamo verificare di avere effettivamente spento i server con i comandi:

```
service apache2 status
service mysql status
```

se risultassero ancora attivi li spegniamo con i comandi:

```
service apache2 stop
service mysql stop
```

Ora installiamo i pacchetti richiesti da Volumio per poter funzionare, con il comando (Fig. 19):

```
apt-get -y install nginx
sqlite3 php5-fpm php5-curl
php5-sqlite php5-cli
```

Pazientiamo un po’ e poi installiamo il pacchetto Volumio, andiamo alla cartella home con il comando:

```
cd /home
```

e cloniamo il pacchetto Volumio nella stessa cartella (Fig. 20):

```
git clone https://github.com/volumio/Volumio-WebUI.git
```

Ora dobbiamo copiare il contenuto della cartella /home/Volumio-WebUI nella cartella /var/www in modo che sia visibile al server web:

```
cp /home/Volumio-WebUI /
var/www
```

Modifichiamo i permessi di esecuzione di alcuni file, che abbiamo appena scaricato, con la serie di comandi:

```
chmod 775 /var/www/_OS_
SETTINGS/etc/rc.local
chmod 755 /var/www/_OS_
SETTINGS/etc/php5/mods-
available/apc.ini
```

Fig. 22

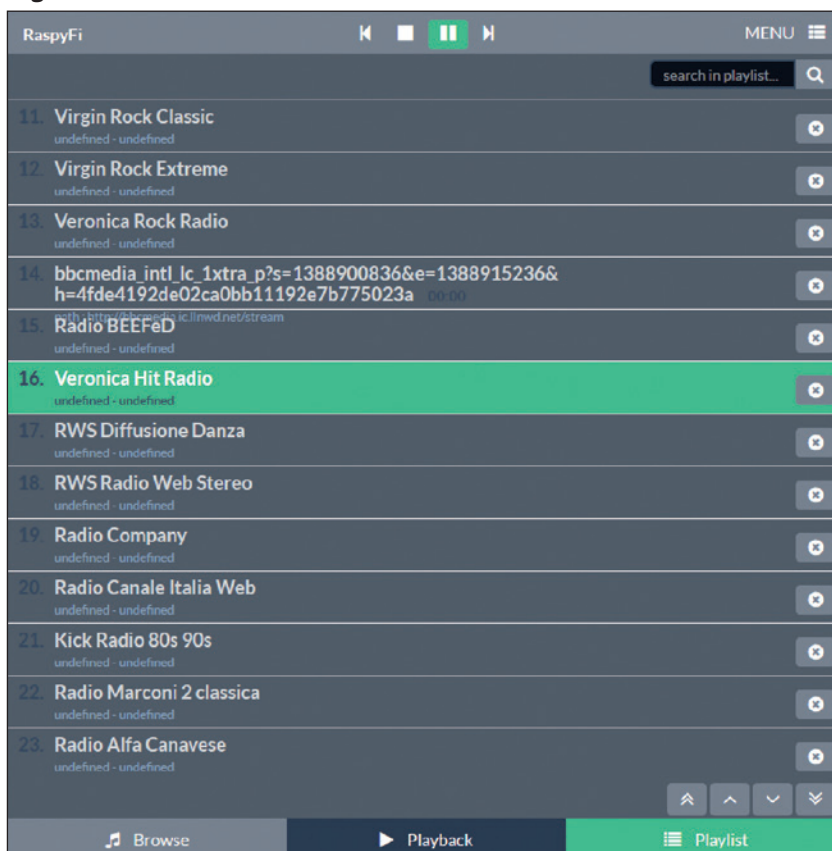
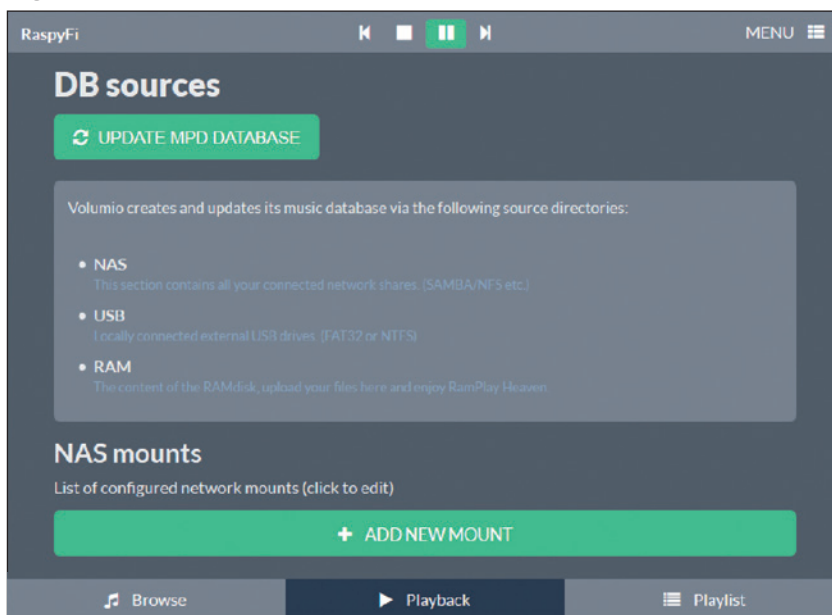


Fig. 23



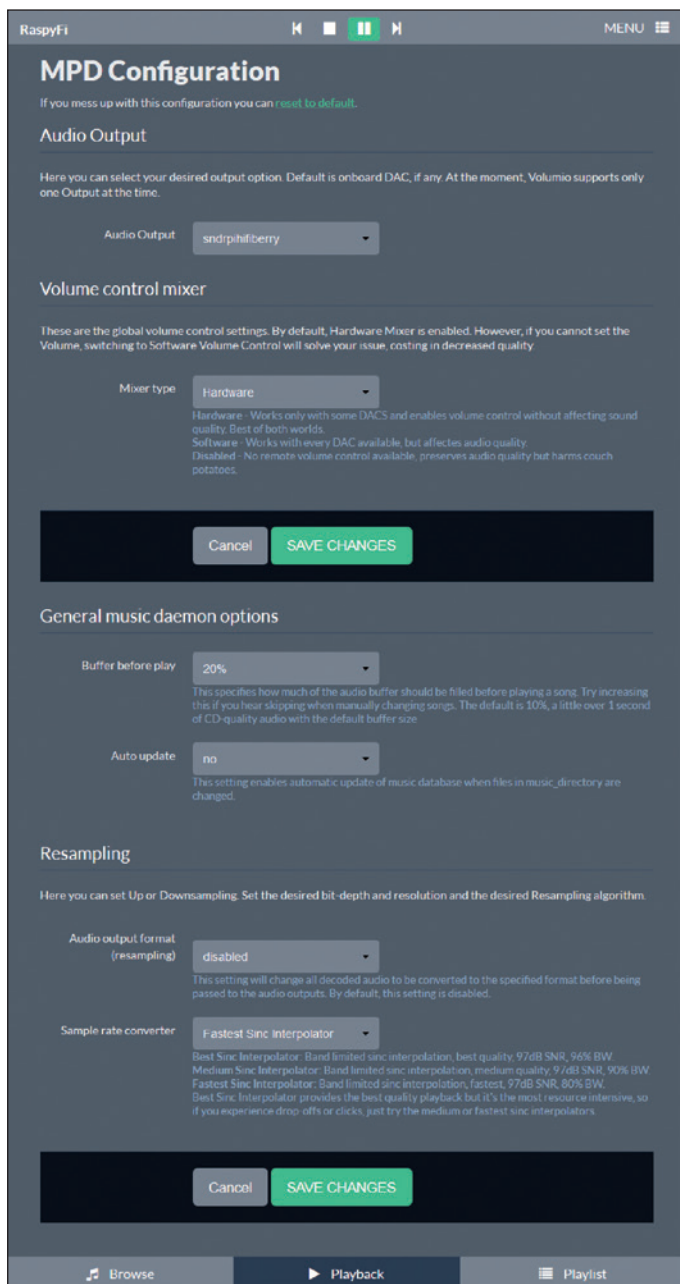
```
chmod -R 777 /var/www/
command/
chmod -R 777 /var/www/db/
chmod -R 777 /var/www/inc/
```

Copiamo in un sol colpo tutte

le configurazioni dei vari moduli nelle cartelle corrette con il comando:

```
cp -var /var/www/_OS_
SETTINGS/etc /
```

Fig. 24



Verifichiamo il contenuto del file di configurazione `/etc/network/interfaces` per verificare che si sia mantenuta l'eventuale configurazione con l'IP statico, se non è così la ripristiniamo come richiesto dalla nostra configurazione.

Bene, complimenti. Ora riavviamo Raspberry Pi, in modo da rendere effettive tutte le installazioni e configurazioni eseguite sino a questo momento con il comando:

`reboot`

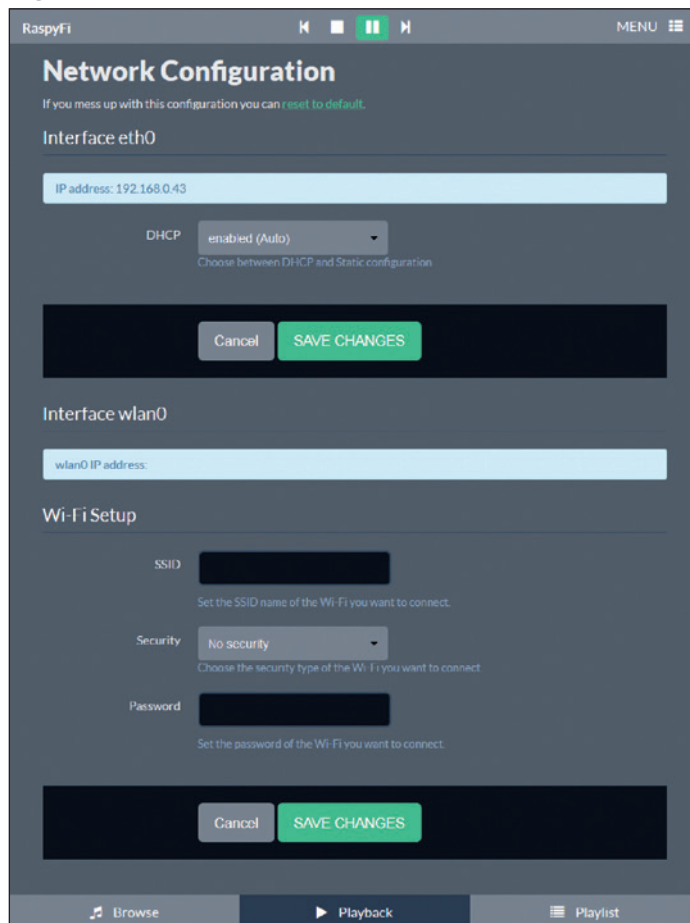
Una volta che tutto sarà ripartito, possiamo verifica-

re il corretto funzionamento. Apriamo un browser e digitiamo nel campo URL l'indirizzo IP di Raspberry Pi. Vedremo apparire la pagina iniziale di Volumio (**Fig. 21**). Lo stesso se apriamo la pagina dal browser del nostro smartphone o del tablet. Lanciamo da finestra da terminale anche il client per l'interfaccia LCD, se questo non parte in automatico al boot di raspberry, digitando il comando:

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

Torniamo alla pagina web di Volumio, visibile in **Fig. 21**. In alto abbiamo la riproduzione dell'RDS con la descrizione della radio attiva al momento. Sotto, a sinistra, un widget indica il corretto funzionamento di tutto il progetto (indicatore in movimento circolare), con al centro il tempo di riproduzione della radio selezionata. A destra il widget circolare per il controllo del volume. Nel caso di utilizzo del modulo FLEXMod deve essere impostato a 100, il massimo valore, nel caso di utilizzo con uscita altoparlante, viene utilizzato effettivamente

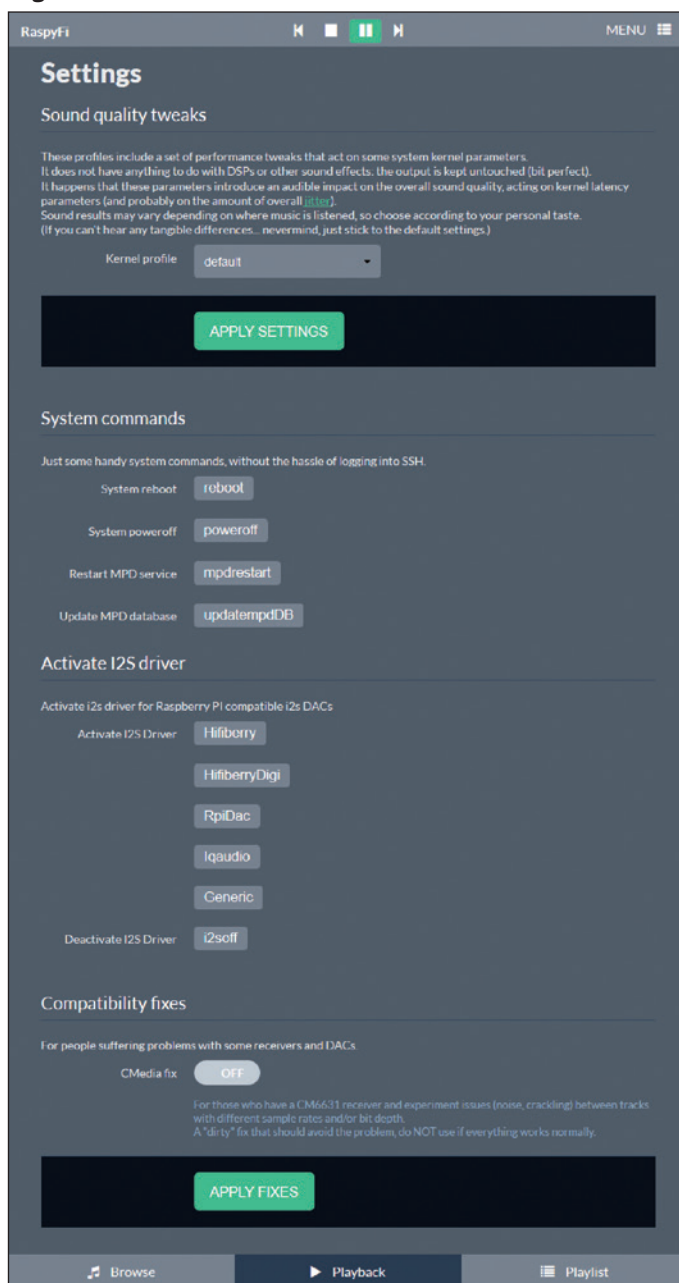
Fig. 25



come controllo di volume. Nella barra di menu in basso clicchiamo su "Playlist". Ci ritroveremo l'elenco di tutte le radio che abbiamo inserito nella nostra playlist. Clicchiamo su un' emittente per selezionarla (Fig. 22). Vedremo cambiare anche la radio impostata sul display LCD. I due client funzionano in modo indipendente ma influenzandosi l'un l'altro. La pagina "Playlist" contiene l'elenco delle radio prese dal file di configurazione `/var/lib/mpd/playlists`, che abbiamo configurato nel numero 185. In alto a destra troviamo l'icona del menu. Clicchiamo sull'icona a griglia e analizziamo le funzioni disponibili. "Main", ovviamente, ci riporta alla pagina principale del programma. "Library" ci permette di aggiungere alle playlist di MPD, quelle contenute su diverse sorgenti in input, come una memoria di massa su chiavetta USB od un disco di rete condiviso (Fig. 23). Approfondiremo l'argomento della condivisione di dischi in rete in un prossimo articolo. La voce "Playback", a dispetto del nome, apre la pagina di configurazione del server MPD (Fig. 24). La configurazione principale consiste nell'impostazione del driver corretto per la gestione dell'uscita audio, nel nostro caso è quello che abbiamo impostato manualmente nel file `/etc/mpd.conf` (vedi numero 185). Gli altri parametri servono ad ottimizzare la riproduzione a seconda delle condizioni di funzionamento di tutto il sistema, sia in termini di buffer assegnati alla riproduzione, per renderla il più fluida possibile, che di normalizzazione dell'audio. Possiamo lasciare invariati questi parametri. L'unico che può interessarci è il parametro "Gapless mp3 playback", nel caso durante la riproduzione di file mp3, questi si interrompessero bruscamente, per migliorare la situazione possiamo impostare il parametro a "no".

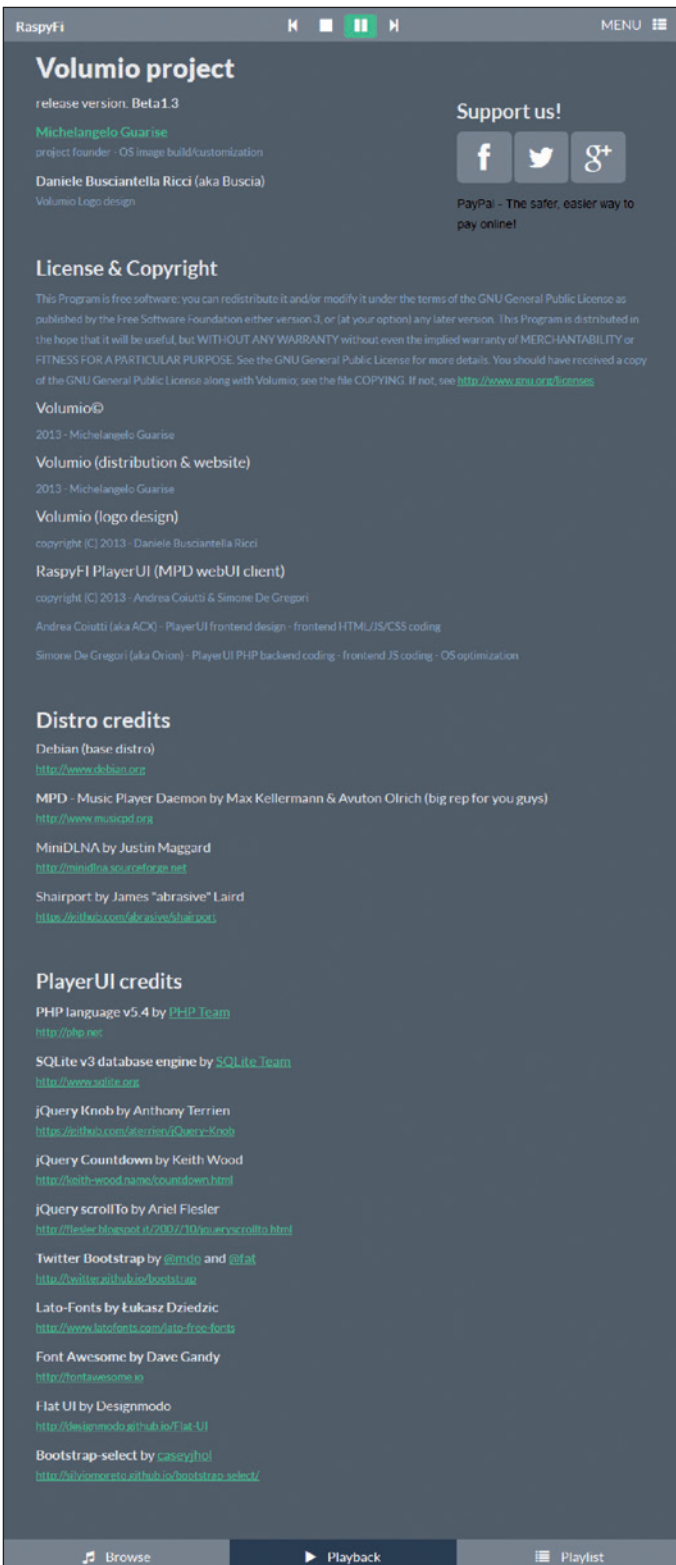
In ogni sezione della pagina, per rendere effettive le modifiche, dobbiamo salvare le impostazioni, cliccando sul pulsante "SAVE CHANGES". La voce "Network" del menu (Fig. 25) ci permette di impostare la configurazione di rete, sia per l'interfaccia Ethernet che per l'interfaccia WiFi, nel caso utilizziamo un dongle WiFi compatibile. La voce "System" ci permette di configurare e/o eseguire alcune funzionalità di sistema direttamente dall'interfaccia grafica di Volumio (Fig. 26), la prima sezione "Sound quality tweaks" ci permette di impostare la latenza del kernel per migliorare la qualità del suono. Il nostro suggerimento è di lasciare le cose come stanno. Se volete fare delle prove ricordate di salvare le modifiche cliccando sul pulsante "APPLY SETTINGS". La sezione "System commands"

Fig. 26



ci permette di eseguire alcuni comandi di sistema dall'interfaccia grafica, tutti comandi che già conosciamo. La sezione "Activate I2S driver" ci permette di attivare e disattivare i driver I²S. Il driver compatibile con il modulo FLEXMod è "HifiBerry". L'ultima sezione, "Compatibility fixes" offre una via per risolvere eventuali problemi legati ad un modulo specifico, il CM6331. La raccomandazione è di non attivarla se tutto funziona correttamente. Se del caso, salvare l'impostazione cliccando sul pulsante "APPLY FIXES".

Apriamo la voce "Credits" potremo farci un'idea di quanto lavoro è richiesto e quante persone sono im-

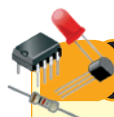
Fig. 27

peginate per realizzare applicazioni Open Source di qualità (Fig. 27). È l'occasione per ringraziarli tutti pubblicamente. L'ultima voce "Turn off", permette di chiudere l'applicazione e spegnere Raspberry Pi da remoto.

Proviamo ora a selezionare una stazione radio dal display LCD e vedremo modificarsi l'impostazione anche nell'interfaccia web. La stessa cosa avviene se utilizziamo l'interfaccia web per cambiare stazione. Il display LCD sarà aggiornato conseguentemente. Vi lasciamo sperimentare con la nuova configurazione del nostro progetto di radio Internet, se volete non utilizzare il modulo FLEXMod e preferite l'audio in uscita sull'altoparlante basta modificare il file di configurazione del server /etc/mpd.conf a riga 204-205 in modo da disattivare l'uscita I2S e ripristinare l'uscita verso il connettore di output audio come segue:

```
## name "sndrpihifiberry [snd_rpi_
hifiberry_dac], device 0: HifiBerry DAC HiFi
pcm5102a-hifi-0 []"
name "My ALSA Device"
```

Nel prosieguo di questi articoli descriveremo come aggiungere al nostro progetto la possibilità di caricare brani da sorgenti esterne come chiavette USB e dischi condivisi in rete, installando e configurando correttamente i pacchetti di condivisione dischi e cartelle. Inoltre descriveremo come rendere Raspberry Pi un access point WiFi, in modo da poterci collegare direttamente con il nostro PC o dispositivo mobile, senza la necessità di una rete domestica. Due funzionalità che sono utili di per sé, anche non associate al progetto della radio Internet. Per ora buona sperimentazione. ■



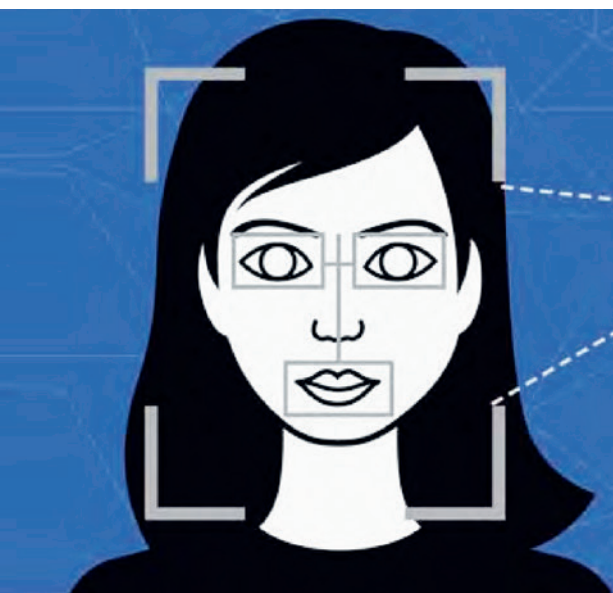
per il MATERIALE

Il progetto pratico a cui si riferisce questo articolo è stato presentato sul fascicolo del mese scorso (n.185). Questo progetto è facilmente realizzabile da chiunque mentre il software di tutte le applicazioni può essere scaricato gratuitamente dal sito della rivista (www.elettronicaonline.it). Il modulo radio FLEXMOD-DAB2 utilizzato costa 390,00 Euro mentre la scheda di alimentazione completa di antenna sulla quale montare il modulo costa 98,00 Euro (cod. FLEXMODPOWER). La board Raspberry Pi mod. B (cod. 7310-7568308RS) costa 38,00 Euro mentre la scatola di montaggio dello Shield LCD (cod. FT1074K) costa 17,00 Euro. Tutti i prezzi si intendono IVA compresa.

Il materiale va richiesto a:

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

RICONOSCIMENTO VOLTI CON RASPBERRY PI



Utilizziamo le funzionalità della libreria SimpleCV per individuare volti e altri particolari del corpo umano all'interno di immagini. Concludiamo con un programma in grado di riconoscere la nostra faccia e attivare un relè, oppure di dare l'allarme in presenza di sconosciuti.

di MARCO MAGAGNIN

Un campo di applicazione della visione computerizzata, che ha sempre affascinato la fantasia popolare, è quello relativo alla capacità di robot e computer di individuare, riconoscere ed interagire con le controparti umane. Oggi queste funzionalità stanno entrando nella nostra esperienza comune, anche se con modalità che spesso non rispecchiano propriamente le anticipa-

zioni della fantascienza. In questo articolo approfittiamo della disponibilità di strumenti economici di calcolo e di acquisizione immagini, come Raspberry Pi e la sua videocamera dedicata Camera Pi e di prodotti software open source di acquisizione ed elaborazione delle immagini come OpenCv e SimpleCV che permettono un approccio di alto livello a questa disciplina e quindi "abbastanza"

semplificato. Tutti gli strumenti ed i prodotti sono stati presentati nei numeri 185 e 186 della rivista. Nel numero 185

abbiamo presentato la Camera Pi e le sue funzionalità, in particolare la possibilità di applicare "effetti" alle



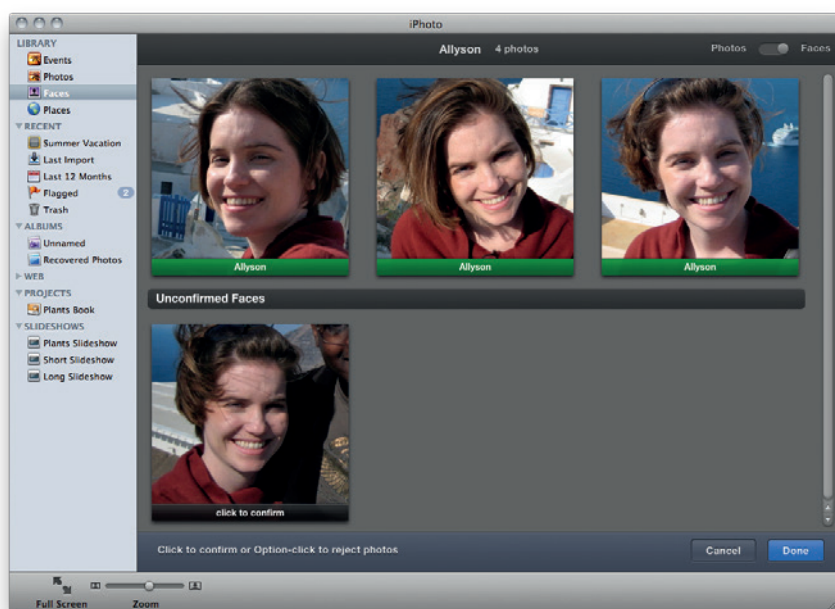


Fig. 1

immagini o ai video acquisiti. Nel numero 186 abbiamo introdotto i diversi possibili obiettivi della visione computerizzata: acquisizione, localizzazione (o rilevamento) e identificazione (o riconoscimento). Nello stesso numero abbiamo descritto e presentato degli esempi su come localizzare semplici oggetti ed estrapolarne forme, colori e contorni. In questo articolo, con lo stesso principio, anche se con un salto "tecnologico" notevole, presentiamo la possibilità di localizzare, all'interno delle immagini, esseri umani o loro particolari come facce, occhi, naso, ecc. Questa funzionalità è presente nelle applicazioni più evolute di raccolte di fotografie e di social network. Una volta caricate le foto, il sistema le scandisce alla ricerca di volti di persone, le individua e dà la possibilità di associarvi un nome (Fig. 1). Se per caso la stessa persona è presente in più immagini viene riconosciuta e "anagrafata" automaticamente, con buona pace della privacy. Quest'ultima

funzionalità è quella che abbiamo citato come identificazione o riconoscimento.

IL METODO DI RICONOSCIMENTO

Per riconoscere oggetti reali, come nel nostro caso le persone e i loro particolari in SimpleCV, ma anche in OpenCV, viene fornito un metodo noto come "Haar features cascade" o anche come metodo Viola-Jones. Il metodo, infatti, è stato proposto nel 2001 da Paul Viola e Michael Jones nel loro articolo "Rapid Object Detection using a Boosted Cascade of Simple Feature" che, tradotto, suona più o meno

come "Individuazione rapida di oggetti mediante una cascata di confronti successivi di caratteristiche semplici". Il metodo è la combinazione di quattro componenti chiave:

- Elementi di confronto, materializzati da matrici rettangolari di pixel, conosciute come caratteristiche (features) di Haar;
- Il calcolo dell'immagine integrale, a partire dall'immagine da elaborare, per velocizzare l'individuazione delle caratteristiche;
- L'applicazione del metodo di apprendimento per sistemi di visione computerizzati AdaBoost;
- Un classificatore a cascata per velocizzare il processo di individuazione.

L'algoritmo di estrazione delle caratteristiche è derivato dalla wavelet di Haar (http://it.wikipedia.org/wiki/Wavelet_Haar) dove, semplificando ai limiti del ridicolo, la wavelet è rappresentata da due matrici quadrate, una che rappresenta la parte superiore della wavelet ed una che rappresenta la parte inferiore. Nel modello utilizzato per l'estrazione delle caratteristiche dalle immagini, le matrici di riferimento hanno forme diverse come quelle visibili in Fig. 2, più adatte a individuare particolari appartenenti al corpo

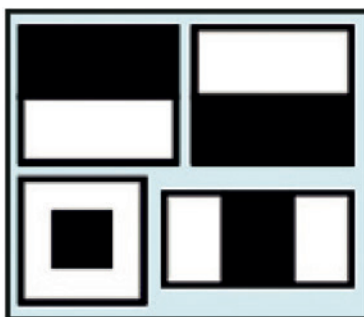


Fig. 2

umano come gli occhi o il naso. Da ciò la loro denominazione di Haar Features, per distinguerle dalla accezione originale (wavelet). La stessa figura mostra la forma delle features utilizzate da OpenCv e SimpleCV. La verifica della presenza o meno di una "feature" Haar in una porzione di immagine avviene sottraendo il valore medio dei pixel presenti nella porzione nera della "maschera" dal valore medio dei pixel presenti nella parte chiara della "maschera". Se la differenza è superiore ad un certo valore di soglia la caratteristica è considerata presente. Il valore di soglia viene determinato, per ciascuna caratteristica,

analizzate di volta in volta, sono costituite in genere da matrici di 24x24 pixel, che devono essere "confrontate" con tutte le features previste. Qui sorge un primo problema. Elaborando una matrice di 24x24 pixel si ottiene qualcosa come più di 160.000 caratteristiche, che richiederebbero per le operazioni di confronto, tempi di elaborazione inaccettabili.

I signori citati in precedenza, hanno introdotto una semplificazione di calcolo che si basa sull'immagine integrale, che permette di eseguire i calcoli per l'intera matrice con soli quattro pixel alla volta (Fig. 3). Per inciso il metodo è stato sviluppato

di integrazione, il calcolo del valore medio dei pixel contenuti in una data porzione dell'immagine, ad esempio l'area "D" in Fig. 3, avviene applicando la formula di seguito ai soli pixel in basso a destra di ciascuna area:

$$D = A+B+C+D - (A+B) - (A+C) + A.$$

Nonostante la semplificazione, il volume di elaborazione è ancora eccessivo per essere efficiente. Gli autori del metodo hanno notato che alcune caratteristiche erano più rilevanti di altre al fine del riconoscimento di alcune parti, per esempio gli occhi, ma non lo sono affatto per rico-

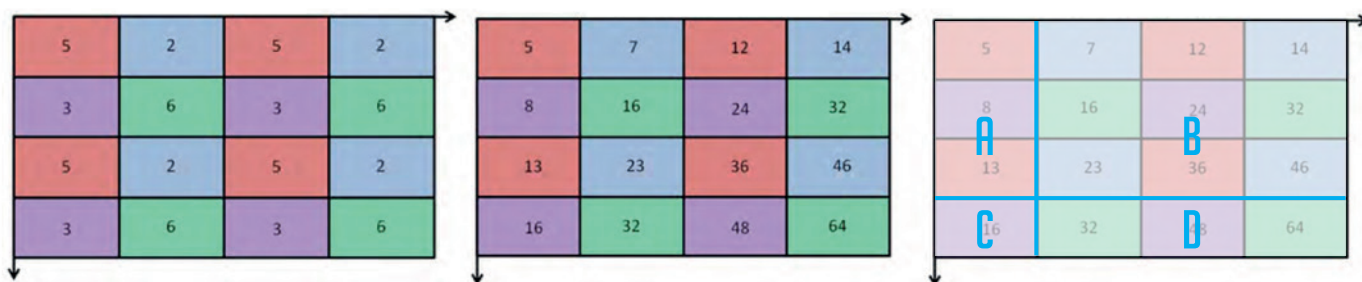


Fig. 3

durante il processo di addestramento della funzione a individuare specifici oggetti come particolari del corpo umano. Il processo di apprendimento si concretizza nel "presentare" al sistema di visione il maggior numero possibile di immagini della famiglia di "oggetti" che si vogliono identificare ed il maggior numero possibile di immagini che non c'entrano nulla con l'oggetto stesso. Dalla mole di dati "studiati" vengono calcolati i valori di soglia per ciascuna delle caratteristiche che, nel caso di OpenCV e SimpleCV, vengono memorizzati in file in formato .xml. Le porzioni dell'immagine,

dal punto di vista matematico nel 1984. Potete approfondire il metodo a questo indirizzo:

<http://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/>
Il processo consiste nell'assegnare ad un dato pixel in una data posizione nella matrice (immagine) la somma di tutti i pixel presenti nell'area al di sopra e a sinistra della sua posizione, come visibile in Fig. 3. Partendo dal pixel in alto a sinistra e proseguendo verso destra e verso il basso, il processo di calcolo incrementale del valore di ciascun pixel è decisamente efficiente. Dopo il processo

noscere, ad esempio le guance. Alcune non sono rilevanti per nulla. Come fare a selezionare le caratteristiche più rilevanti a riconoscere un dato particolare e scartare le altre? Mediante l'algoritmo matematico conosciuto come Adaboost (Adaptive boosting) un algoritmo di "machine learning" realizzato per ottimizzare le prestazioni di altri algoritmi di apprendimento. In parole ultra semplici, da far inorridire i puristi, il suo scopo è di ricercare l'insieme più piccolo possibile che garantisca una percentuale data (per esempio 75%) di accuratezza nel risultato di riconoscere o scartare l'oggetto richiesto. Il lavoro

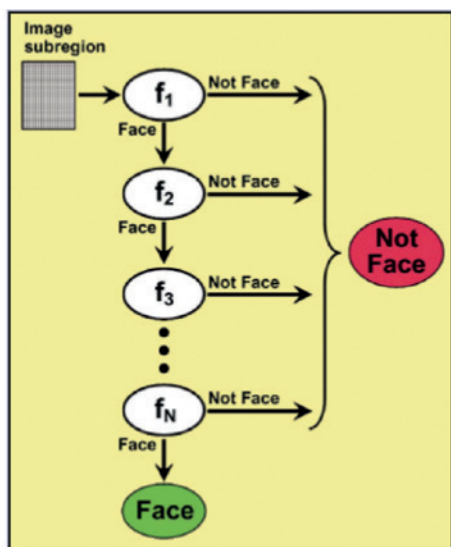


Fig. 4

citato all'inizio dimostra che con 200 caratteristiche è possibile ottenere un grado di localizzazione con una percentuale di accuratezza del 95%. Per completezza e maggiore accuratezza le implementazioni del metodo utilizzano circa 6.000 caratteristiche selezionate dalle 160.000 iniziali. Se ci fossimo fermati qui, per ogni porzione della matrice di 24x24 pixel avremmo dovuto confrontare 6.000 caratteristiche. Saremmo sicuramente... invecchiati prima di essere individuati. Per fortuna un'altra intuizione dei nostri autori ha fornito una soluzione anche a questo problema. Nella maggioranza dei casi, in un'immagine, la maggior parte dell'area non contiene gli oggetti ricercati. Per questo motivo è opportuno definire un metodo semplice in grado di capire se una porzione dell'immagine appartiene all'oggetto ricercato oppure non ne fa parte sicuramente. In questo secondo caso la porzione viene scartata al primo colpo e non viene più processata. Il metodo si deve concentrare su quelle parti che in qualche modo sembrano far parte degli oggetti

ricercati e deve analizzarle in profondità. Questa necessità ha portato ad introdurre il concetto di "cascade" ovvero di cascata di classificatori. Invece di applicare tutte le 6.000 caratteristiche ad ogni porzione dell'immagine, le caratteristiche stesse vengono raggruppate in diversi livelli di classificazione ed applicate all'immagine un livello alla volta. Se una porzione di immagine viene scartata dai classificatori del primo livello non verrà più elaborata. Se invece passa il primo livello, verrà confrontata con le caratteristiche che appartengono al secondo livello, e così via. Se una porzione di immagine supera tutti i livelli allora il particolare appartiene all'oggetto ricercato, come visibile nello schema di **Fig. 4**.

Il metodo di classificazione proposto suddivide le circa 6.000 caratteristiche in 38 livelli di classificazione. I primi cinque livelli sono composti rispettivamente da 1, 10, 25, 25 e 50 caratteristiche. In questo modo, mediamente, è necessario confrontare ciascuna porzione dell'immagine con circa 10 caratteristiche (mediamente!). Prima di passare a qualche esempio pratico ricordiamo che per funzionare, il metodo deve essere addestrato con una grande quantità (600 e più) di immagini che contengono diverse versioni degli oggetti da ricercare e altrettante immagini che non li contengono. Per fortuna, almeno per le principali forme associate al corpo umano

ci troviamo il lavoro già fatto e codificato all'interno sia dei pacchetti OpenCV che SimpleCV. Questo ci permette di fare subito qualche esperimento.

METTIAMOCI LA FACCIA

Dopo la lunga chiacchierata di introduzione al metodo di riconoscimento di oggetti basato sull'algoritmo "HaarFeature-Cascade" sperimentiamo praticamente con qualche esempio. Approfittiamo dell'occasione anche per aggiornare il sistema operativo di Raspberry Pi e per installare una nuova libreria che ci facilita la gestione della Camera Pi. Per i nostri esperimenti serve Raspberry Pi configurata come l'abbiamo lasciata nell'articolo sul numero 186 della rivista, ovvero collegata alla rete locale con un cavo Ethernet e collegata ad un monitor HDMI con l'apposito cavo. Noi preferiamo collegarci a Raspberry Pi da remoto via SSH con PuTTY e WinSCP, ma nulla vieta di collegare una tastiera ed un mouse alle uscite USB (meglio attraverso un hub USB dotato di alimentazione propria) ed eseguire gli esempi dal desktop. Abbiamo descritto nel numero 185 come installare e configurare la Camera Pi e come installare e configurare i pacchetti OpenCV e SimpleCV nel numero 186, ai quali rimandiamo per chi partisse da zero. Le informazioni sulla gestione di base di Raspberry Pi e del sistema operativo Raspbian le trovate nella serie di articoli dedicati alla scheda

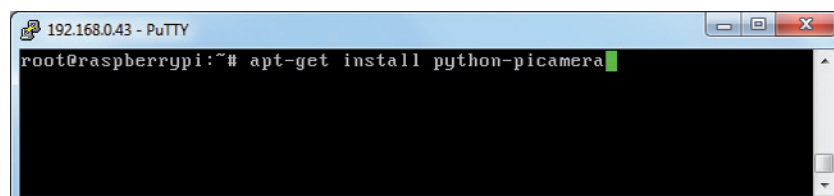


Fig. 5

oppure nel libro “Raspberry Pi – Il mio primo Linux embedded”. Diamo alimentazione a Raspberry Pi, verifichiamo che il monitor funzioni e ci colleghiamo con PuTTY (o come volete voi) con l’utente “root”. Aggiorniamo il sistema operativo, in quanto al momento di scrittura di questo articolo sono apparsi molti miglioramenti, con i comandi:

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

Se la vostra distribuzione non è proprio recentissima, ci vorrà un po’ di tempo per portare a termine gli aggiornamenti. In casi come questi è sempre opportuno un buon:

```
reboot
```

prima di continuare. Ora installiamo la libreria “picamera”, una libreria appositamente sviluppata per l’ambiente Python per interfacciare la Camera Pi. Ricordiamo che SimpleCV ed OpenCV si aspettano di “trovare” le videocamere sulle porte USB, mentre Camera Pi si collega all’opposito connettore CSI

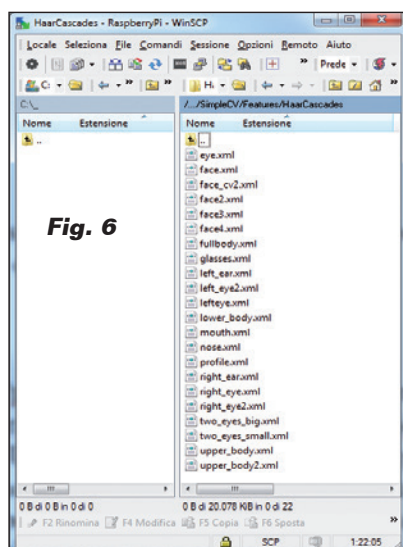


Fig. 6

presente su Raspberry Pi. Per i nostri articoli abbiamo scelto di utilizzare Camera Pi per la qualità delle immagini e dei video che si possono ottenere e per la gamma di funzioni disponibili per l’elaborazione delle immagini. Nell’articolo sul numero 186 abbiamo usato l’artificio di richiamare da sistema le funzionalità di interfaccia di Camera Pi e salvare il fotogramma che poi andavamo ad elaborare con l’istruzione:

```
subprocess.call("raspistill -n -w %s -h %s -o Listato3_1.png" % (640, 480), shell=True)
```

Con la nuova libreria tutto ciò non è più necessario ed andremo ad interfacciare la Camera Pi utilizzando direttamente le funzioni Python messe a disposizione. La documentazione della libreria “picamera”, assieme ad altre utili informazioni sulle funzionalità disponibili e sul funzionamento di Camera Pi sono reperibili, in inglese, all’indirizzo:

<http://picamera.readthedocs.org/en/release-1.5/>

Installiamo la libreria con il comando (Fig. 5):

Intel Pocket Avatar

Questa volta non è un processore bensì una particolare App che decreta l’ingresso di Intel in questo particolare mondo. Pocket Avatar è infatti una nuova applicazione di messaggistica istantanea, che permette agli utenti di chattare utilizzando degli avatar in 3D creati attraverso le espressioni facciali catturate dalla fotocamera dello smartphone. Grazie alla tecnologia di face tracking, i contatti assumeranno le sembianze più diverse, come ad esempio quelle di un cartone animato, ma le espressioni e i movimenti della bocca riprodurranno quelle del “padrone” dell’avatar. Oltre alle più tradizionali funzionalità presenti nelle applicazioni di messaggistica, come sincronizzazione della rubrica, inviti, blocco dei contatti, condivisione sui social network, ecc, con Pocket Avatar è possibile creare brevi video con messaggi personalizzati (fino

a 15 secondi) e inviarli tramite l’applicazione, una mail o un social network. L’app è gratuita per gli smartphone Android e per gli iPhone e dispone di 40 diversi avatar: alcuni di essi saranno a disposizione degli utenti gratuitamente mentre per accedere alle immagini esclusive si dovrà sborsare 0,99 dollari.

L’applicazione costituisce un test per la tecnologia di face tracking: oggi sugli smartphone e domani sulle piattaforme Intel come laptop o tablet, a conferma dell’impegno pluriennale dei Laboratori Intel nella ricerca sul 3D e il videorending.

Listato 1

```
#!/usr/bin/python2.7

# Programma test Haar Features

import picamera
from SimpleCV import Image
import time

with picamera.PiCamera() as camera:
    camera.resolution = (640, 480)
    camera.start_preview()
    time.sleep(10)
    camera.capture('foto.jpg')
    foto=Image("foto.jpg")

    print(foto.listHaarFeatures())
    trovati=foto.findHaarFeatures('face.xml')
    if trovati:
        for trovato in trovati:
            print "Trovato alle coordinate : " + str(trovato.coordinates())
            trovato.draw()
    else:
        print "Non trovato"
    camera.stop_preview()
    foto.save('fotol.jpg')
    foto.show()
    time.sleep(10)
```

apt-get install python-picamera

Negli esempi che seguono ci concentriamo sul riconoscimento di facce e di particolari del corpo umano per i quali abbiamo disponibili le features "HaarCascade". Quali sono? Quelle per le quali gli sviluppatori di OpenCV e SimpleCV hanno provveduto ad eseguire il processo di addestramento e messe a disposizione in entrambe le librerie. Le troviamo nella cartella (**Fig. 6**):

/usr/local/lib/python2.7/dist-packages/
SimpleCV/Features/HaarCascades/

D'accordo, il percorso è un po' lungo ma quello è il loro posto. Anticipiamo un paio di avvertimenti nel caso insorgessero dei problemi. Osservate intanto come la libreria SimpleCV sia installata sotto la cartella di Python2.7: è necessario quindi utilizzare questa versione di Python per eseguire gli esempi. Se lanciate gli esempi sotto Python3 probabilmente otterrete

degli errori. In secondo luogo, per raggiungere il file .xml all'interno della cartella citata, normalmente basta il nome, per esempio "face", come indicato nella maggior parte degli esempi scaricabili da Internet. Se però questa impostazione desse degli errori dovreste utilizzare il percorso completo.

Nel nostro caso abbiamo trovato che funziona chiamare le "Feature" con l'estensione, per esempio "face.xml", ma non sempre è così.

Nella cartella citata troviamo i seguenti file, che possono essere diversi nel caso venga installata una versione di SimpleCV diversa dalla 1.3 che abbiamo installato noi:

eye.xml
fullbody.xml
mouth.xml
two_eyes_big.xml
face2.xml
glasses.xml
nose.xml
two_eyes_small.xml
face3.xml
left_ear.xml
profile.xml
upper_body2.xml
face4.xml
left_eye2.xml
right_ear.xml
upper_body.xml
face_cv2.xml
lefteye.xml
right_eye2.xml
face.xml
lower_body.xml
right_eye.xml

Ciascun file .xml contiene le caratteristiche (Features), risultato del processo di apprendimento finalizzato a individuare un dato particolare, che possiamo utilizzare direttamente per elaborare le nostre immagini. I nomi richiamano il particolare che sono in grado di riconoscere. Per alcune parti del corpo è presente più di una versione di "Features". Alcune "Features" sono più selettive di altre, alcune individuano meglio forme molto evidenti nell'immagine, altre sono più adatte a riconoscere porzioni di immagini più piccole. Nelle applicazioni reali si applicano più Features alla stessa immagine in modo da "provarle tutte". Nei casi più

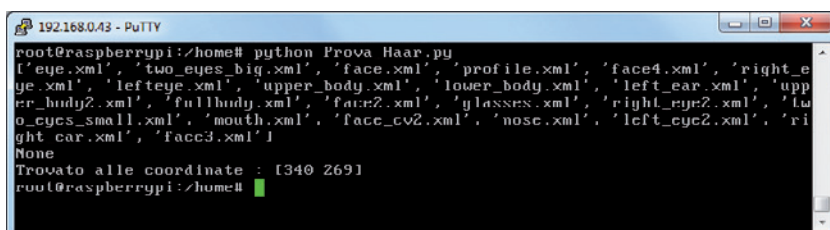


Fig. 7

particolari, o per riconoscere oggetti diversi da facce e particolari del corpo, è necessario procedere a realizzare i propri file di Features con gli opportuni processi di apprendimento. Ma questa è un'altra storia. Ancora una piccola curiosità, nella versione di SimpleCV che abbiamo installato noi, il nome del file "lefteye.xml" è stato scritto senza il trattino basso. Gli errori, in informatica, sono sempre in agguato. Ricordiamoci di questa anomalia se dobbiamo richiamare nei programmi questa "Feature". Per cominciare copiate il programma del **Listato 1** in un file nella cartella /home o dove preferite con il nome Prova_Haar.py. Questo programma esegue i seguenti compiti:

- Importa la libreria "picamera" per gestire la Camera Pi direttamente dal linguaggio Python;
- Importa la funzione Image dalla libreria SimpleCV e la funzione "time" per gestire i delay e lasciarci "vedere" le immagini;
- L'istruzione with permette una gestione "pulita" delle risorse alle quali è associata, in genere risorse esterne come la nostra Camera Pi. La risorsa viene impegnata al momento dell'esecuzione dell'istruzione e viene rilasciata correttamente, sia al termine del programma che in caso di termine del programma in errore. La seconda parte dell'istruzione crea l'oggetto "camera" come istanza della classe Picamera, fornita dalla libreria che abbiamo installato in precedenza;
- Imposta la risoluzione della Camera Pi;
- Inizia il "preview" dell'immagine ripresa dalla videocamera (ricordate "raspistill?");

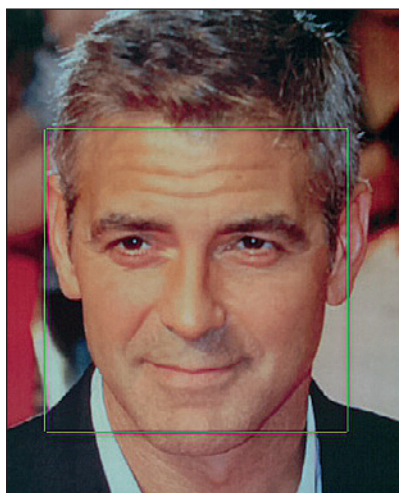


Fig. 8

- Time sleep tiene il preview attivo per 10 secondi, in questo modo abbiamo tempo di posizionarci correttamente davanti alla telecamera. Possiamo anche utilizzare delle foto o delle immagini prese da riviste. Provate diverse alternative. Vi convincerete che forse questo sistema non è il metodo migliore per farsi aprire la porta di casa;
- Catturiamo un fotogramma e lo salviamo in un file;
- Richiamiamo il file come oggetto Image per renderlo disponibile a SimpleCV;

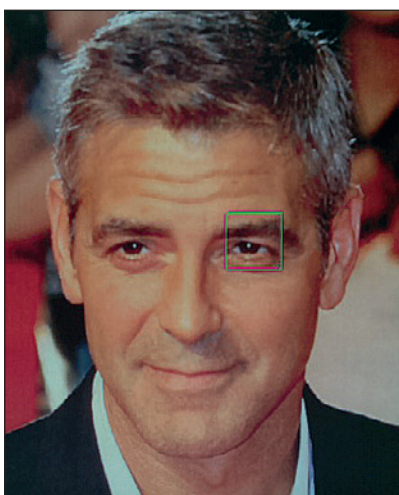


Fig. 9

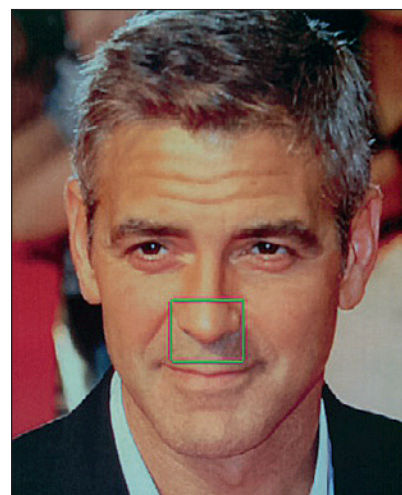


Fig. 10

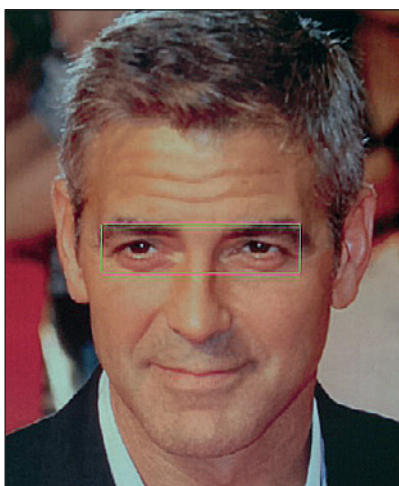


Fig. 11

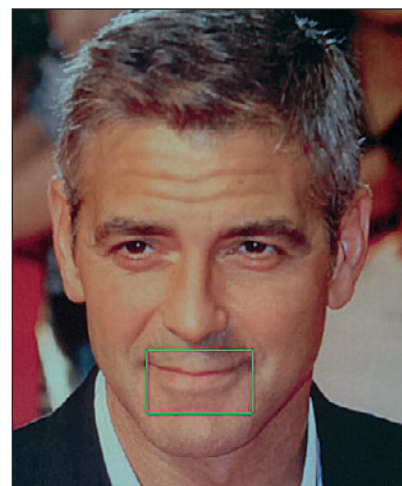


Fig. 12

- Con la funzione `listHaarFeatures()` stampiamo a console la lista delle "Features" disponibili, le stesse che abbiamo visto in precedenza (**Fig. 6**);
- Finalmente eseguiamo la ricerca del particolare "anatomico" indicato dalla corrispondente "Features" richiamata nella funzione;
- Per ciascun "particolare" individuato stampiamo a console le coordinate del "centro" del particolare ed evidenziamo con un rettangolo l'area dell'immagine dove è stato individuato;
- Alla fine del processo interrompiamo il preview, salviamo l'immagine con evidenziati i particolari individuati e

la presentiamo a video per 10 secondi.

Modificando le istruzioni di questo semplice programma possiamo sperimentare tutta la gamma dei particolari riconoscibili e farci un'idea pratica di quanto incidano le dimensioni dell'immagine inquadrata, le condizioni di luce e l'orientamento, per esempio, del viso nell'immagine. Un conto è riconoscere un naso con il viso ripreso in posizione frontale mentre è ben diverso riconoscere un naso di profilo. Mentre vi fate riconoscere il vostro viso inquadrato dalla telecamera, potrete notare che viene riconosciuto anche il viso del nonno nella foto sul mobile del tinello. Per tutti questi test potete utilizzare sempre il programma del **Listato 1** modificando semplice-

mente i parametri della funzione `"findHaarFeatures()"`. Di seguito forniamo alcuni esempi. Nella **Fig. 7** vediamo la console del processo di individuazione del viso nell'immagine. In **Fig. 8** il viso individuato.

Riconosciuto?

Nella **Fig. 9** utilizziamo il programma del **Listato 1** con le `Feature lefteye.xml` per riconoscere ed evidenziare con un riquadro l'occhio sinistro.

```
trovati=foto.  
findHaarFeatures('lefteye.xml')
```

Nella **Fig. 10** il risultato del programma con le `Features nose.xml`.

```
trovati=foto.findHaarFeatures('nose.  
xml')
```

Nella **Fig. 11** le `Features two_eyes_big.xml` ci permettono di individuare le zone dell'immagine dove sono presenti entrambi gli occhi.

```
trovati=foto.  
findHaarFeatures('two_eyes_big.  
xml')
```

Nella **Fig. 12** il risultato del programma con le `Features mouth.xml`.

```
trovati=foto.  
findHaarFeatures('mouth.xml')
```

Nel caso il programma non individui il particolare desiderato provate con le `Features` alternative, ad esempio, nel caso degli occhi `two_eyes_small.xml`. Ora tocca a voi, sperimentate più che potete. Modificate il programma per acquisire un'immagine, anche da file, e ricercare al suo interno i particolari che desiderate: figure intere, mezzi busti, volti, occhi, orecchie, chi ha gli occhiali e chi no. Per concludere vi proponiamo

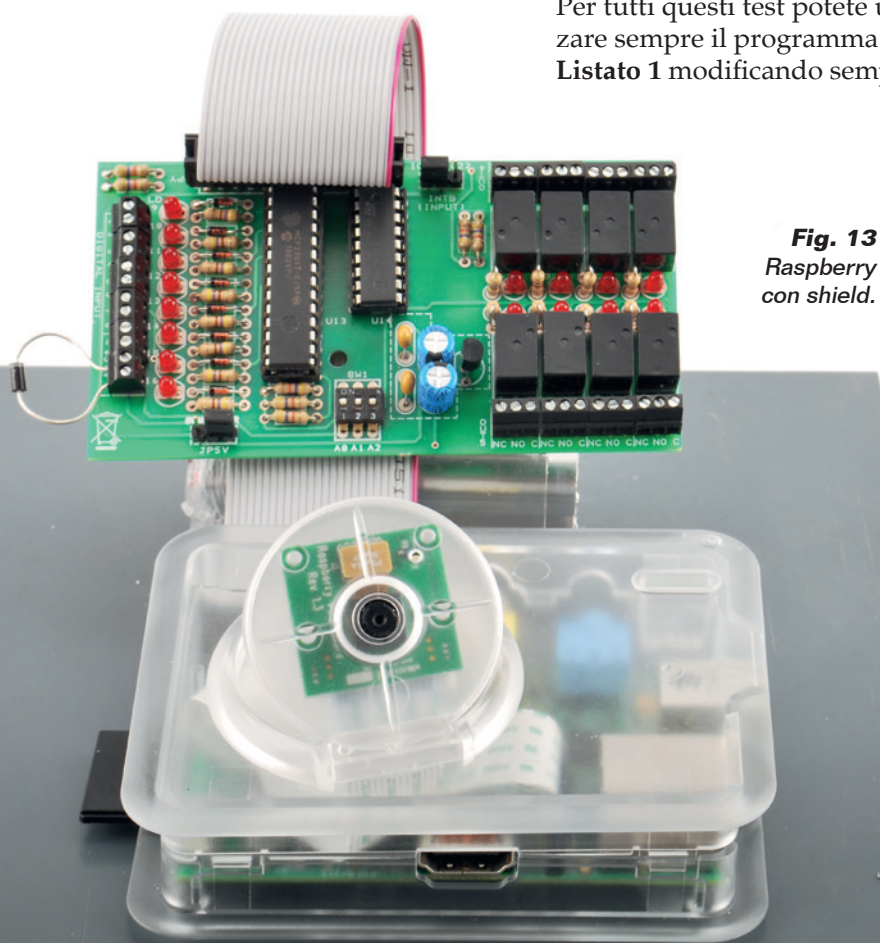


Fig. 13
Raspberry
con shield.

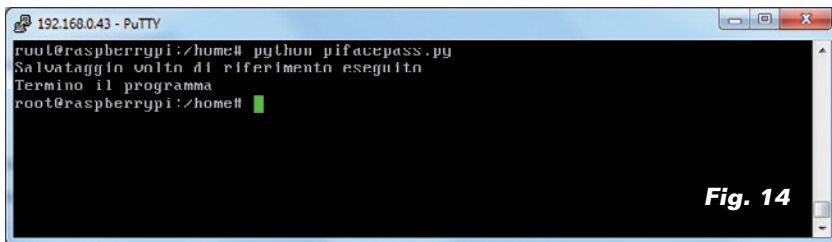


Fig. 14

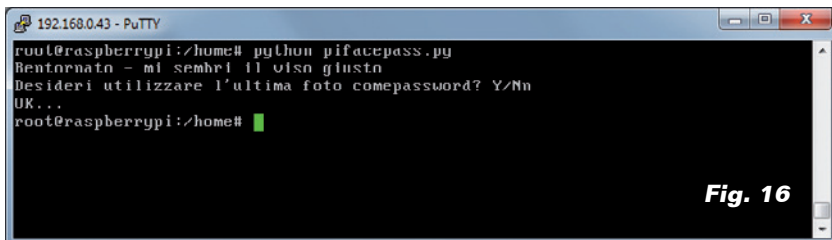


Fig. 16

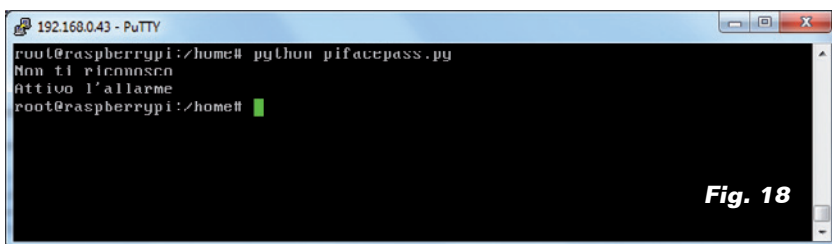


Fig. 18

un altro programma, un po' più complesso, ma che permette di memorizzare un volto e di verificare se quello che si analizza nelle immagini successive è lo stesso oppure no. Una specie di "password umana" che potrebbe essere utilizzata, ad esempio, per permettere o meno l'accesso ad un locale o ad un computer. Vi raccomandiamo di non farlo, se non per gioco o per dimostrazione in quanto il programma è molto semplice e a puro scopo dimostrativo. Viene "addestrato" con una sola immagine, che rimane in formato jpeg. Di conseguenza, per farsi riconoscere bisogna farsi riprendere con un'angolazione ed una posizione molto simile a quella utilizzata per salvare l'immagine di riferimento. Dopo che avrete addestrato il programma provate a far "riconoscere" una vostra fotografia, oppure addestratelo direttamente con una vostra foto. Resterete sorpresi dal com-

portamento di un sistema del genere. Un conto sono i sistemi didattici come quelli che presentiamo, altra cosa è la gestione di sicurezza di accessi a luoghi ed a sistemi. Quest'ultima è una faccenda molto seria da gestire con sistemi professionali progettati e realizzati da specialisti del settore. Vediamo dunque il programma "pyfacepass.py" del **Listato 2**. La versione originale del programma è disponibile all'indirizzo:

<https://gist.github.com/rishimukherjee/2220293>

Noi lo abbiamo adattato per funzionare con Raspberry Pi e Camera Pi e per interfacciare lo shield di espansione I2C FT1039K, dotato di otto uscite relè, in modo da poter attivare un relè in caso di rilevamento positivo o un'uscita di allarme in caso di mancato riconoscimento. La descrizione e tutte

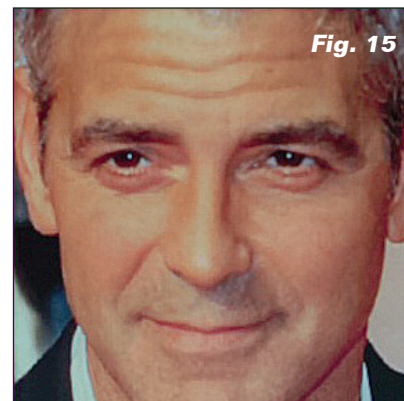


Fig. 15

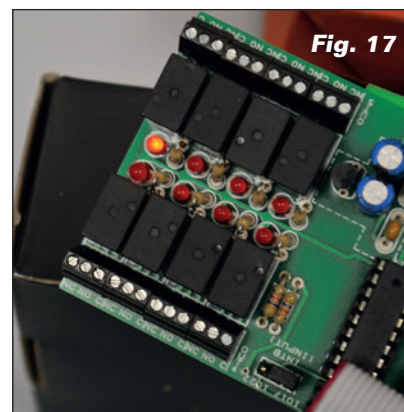


Fig. 17

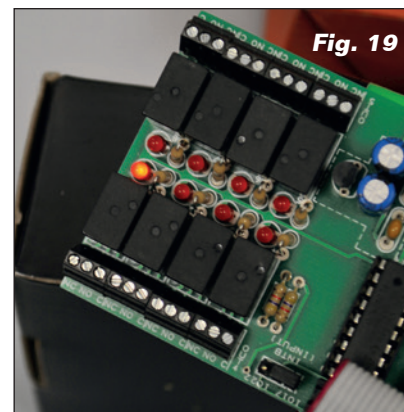


Fig. 19

le istruzioni necessarie per far funzionare lo shield di espansione sono descritte nell'articolo dedicato sul numero 183 della rivista ed è in vendita presso Futura Elettronica. Per non rovinare l'estetica del contenitore per Raspberry Pi con il supporto per la Camera Pi abbiamo utilizzato un cavo piatto per collegare il connettore GPIO di Raspberry Pi con quello dello shield,

```
#!/usr/bin/python

import picamera
import time
from SimpleCV import Color, Image, np
import wiringpi2 as wiringpi

pin_base = 65
i2c_addr = 0x20

quality = 400      # Parametro "quality" per la funzione findKeypointMatch
minMatch = 0.3     # Parametro "minDist" per la funzione findKeypointMatch

try:
    password = Image("password.jpg")
except:
    password = None

mode = "unsaved"
saved = False
minDist = 0.25

wiringpi.wiringPiSetup()
wiringpi.mcp23017Setup(pin_base,i2c_addr)
wiringpi.pinMode(65, 1)      # imposta GPA0 come output
wiringpi.digitalWrite(65, 0) # imposta GPA0 a 0 (0V, off)
wiringpi.pinMode(72, 1)     # imposta GPA7 come output
wiringpi.digitalWrite(72, 0) # imposta GPA7 a 0 (0V, off)

with picamera.PiCamera() as camera:
    while True:
        camera.start_preview()
        time.sleep(10)
        camera.capture('pifacepw.jpg')
        image=Image("pifacepw.jpg")
        camera.stop_preview()
        faces = image.findHaarFeatures("face.xml") # Riconosce il viso mediante il file HaarCFeatures "face"
        if faces:
            if not password:
                faces.draw()
                face = faces[-1]
                password = face.crop().save("password.jpg")
                print "Salvataggio volto di riferimento eseguito"
                print "Termino il programma"
                break
            else:
                faces.draw()
                face = faces[-1]
                template = face.crop()
                template.save("passwordmatch.jpg")
                keypoints = password.findKeypointMatch(template,quality,minDist,minMatch)
                if keypoints:
                    print "Bentornato - mi sembri il viso giusto"
                    wiringpi.digitalWrite(65, 1)
                    wiringpi.digitalWrite(72, 0)
                    domanda = raw_input("Desideri utilizzare l'ultima foto come password? Y/N").strip()
                    if domanda == "Y":
                        image = cam.getImage().scale(320, 240)
                        faces = image.findHaarFeatures("face.xml")
                        tryit = 1
                        while not tryit == 10 or not faces:
                            image = cam.getImage().scale(320, 240)
                            faces = image.findHaarFeatures("face.xml")
                            tryit += 1
                        if not faces:
                            print "Non trovo nessuna faccia"
                            break
                        else:
                            faces.draw()
                            face = faces[-1]
                            password = face.crop().save("password.jpg")
                            face.crop().show()
                            print "Salvataggio eseguito"
                            print "Termino il programma"
                            time.sleep(1)
                            break
                    else:
                        print "OK..."
                        break
                else:
                    print "Non ti riconosco"
                    print "Attivo l'allarme"
                    wiringpi.digitalWrite(65, 0)
                    wiringpi.digitalWrite(72, 1)
                    break
            else:
                break
```

Listato 3

```
#!/usr/bin/python

import time
from SimpleCV import Image, Color

immagine = Image("password.jpg")
keys = immagine.findKeypoints()
keys.draw(color=Color.BLUE)
immagine.show()
immagine.save('foto_2.jpg')
time.sleep(10)
```

facendo particolare attenzione al rispetto della corrispondenza tra i pin. Lo shield richiede un'alimentazione separata con una tensione di 9V. Il jumper di selezione dell'alimentazione deve essere inserito nella posizione 5VR (alimentazione separata). Riassumiamo brevemente le operazioni da eseguire per installare il software di gestione dello shield. "Liberare" il driver di gestione I2C commentandolo nel file `raspi-blacklist.conf` nella cartella `/etc/modprobe.d`. Eseguire un "reboot" per rendere effettiva la nuova configurazione. Dare il comando per caricare il driver di gestione del bus I2C:

```
modprobe i2c-dev
```

Installare la libreria "wiringpi2" con il comando:

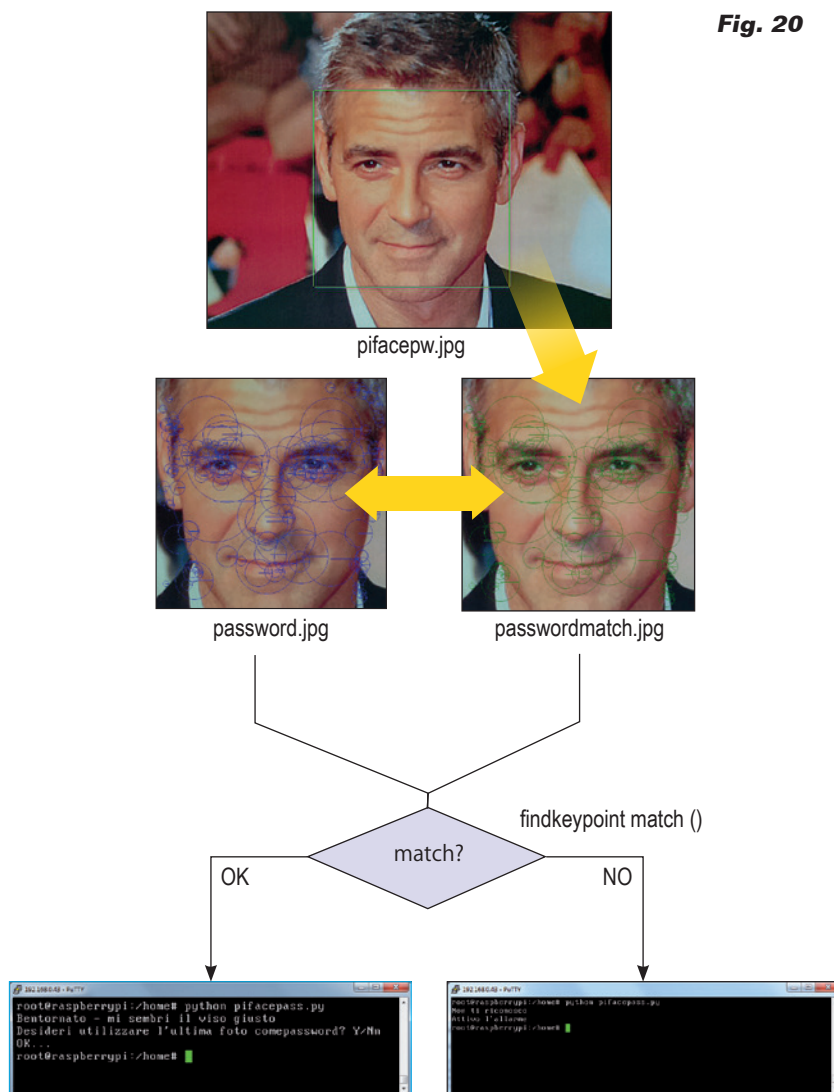
```
pip install wiringpi2
```

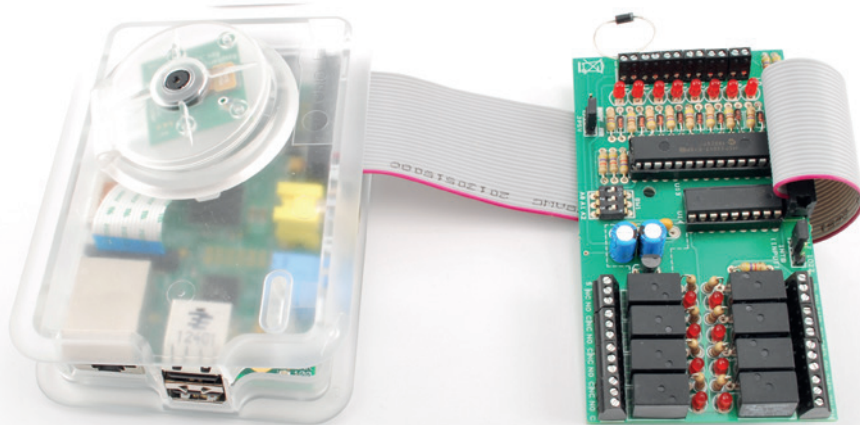
Se qualcuna di queste operazioni vi apparisse oscura o per approfondire la gestione via software degli ingressi/uscite dello shield vi rimandiamo all'articolo citato. Per approfondire in generale le modalità di gestione di Raspberry Pi consultate il libro *"Raspberry Pi – Il mio primo Linux embedded"*. Torniamo al listato del programma. Molte istruzioni le conoscete già. Focalizziamoci un attimo sulla logica del programma. All'inizio del programma vengono importate le librerie necessarie alla gestione delle immagini ed alla gestione dello

shield di espansione. Vengono impostati i parametri per configurare la funzione che individua i punti significativi per confrontare l'immagine di riferimento con quella inquadrata dalla telecamera e vengono inizializzati i pin di uscita dello shield di espansione che utilizzeremo come uscite OK o di allarme, in base al risultato del riconoscimento; rispettivamente l'uscita "0" e l'uscita "7". Dopo le dichiarazioni iniziali il programma cerca l'esistenza di un file che contenga l'immagine di riferimento da utilizzare per il ricono-

scimento del volto inquadrato; se non esiste l'oggetto "password" viene impostato a "None". In questa condizione il volto che si presenta alla videocamera per la prima volta viene memorizzato come volto di riferimento ed il programma termina l'esecuzione (Fig. 14). Come immagine di riferimento viene memorizzato il ritaglio che inquadra il solo volto come visibile nella Fig. 15. Se invece viene trovata un'immagine di riferimento, memorizzata in precedenza, vengono eseguite le seguenti elaborazioni:

- Viene verificato con la fun-





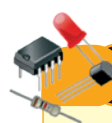
zione findHaarFeatures, che abbiamo descritto in precedenza, se nell'immagine risulta inquadrato un volto. In caso non venga individuato alcun volto viene emesso un messaggio di errore ed il programma viene terminato;

- Viene selezionato il volto in primo piano nell'immagine (quello che nell'inquadratura appare di dimensioni maggiori, in pratica quello in primo piano) e viene creato l'oggetto "template";
- Viene ritagliato il volto individuato ed il "ritaglio" viene salvato con il nome passwordmatch.jpg;
- Gli oggetti "password" (che contiene il volto di riferimento) e "template" (che contiene il volto appena individuato) vengono confrontati con la funzione "findKeypoint-Match()", che confronta i punti significativi estratti dalle due immagini e ne verifica il livello di sovrapposibilità;
- In caso di esito positivo viene stampato il messaggio di riconoscimento e viene attivato il relè corrispondente all'uscita "0" (Fig. 16 e Fig. 17). Questo è il punto dove inserire eventuali ulteriori istruzioni legate al riconoscimento, ad esempio l'invio di un messaggio mail o un tweet. Viene chiesto se si vuole utilizzare l'ultimo volto come nuovo volto di riferimento;

- In caso di risposta affermativa il nuovo volto viene salvato nel file "password.jpg". In ogni caso viene poi terminato il programma;
- In caso l'esito del confronto tra il volto di riferimento ed il volto appena acquisito dia esito negativo viene emesso un messaggio di errore e viene attivato il relè corrispondente all'uscita "7" (Fig. 18 e Fig. 19). Questo è il punto dove inserire eventuali istruzioni da eseguire a fronte del mancato riconoscimento, come inviare un messaggio mail o un tweet. Il programma viene terminato.

Più o meno, relè a parte, questo è il metodo utilizzato dai programmi di archiviazione foto, che abbiamo citato nell'articolo, per individuare, dare un nome e poi riconoscere successivamente i personaggi inquadrati nelle nostre immagini. Prima di concludere, un piccolo approfondimento sui keypoints citati nel metodo di match utilizzato nel programma del Listato 2. I keypoints sono i punti significativi che permettono di "anagrafare" un'immagine e che vengono calcolati ed estratti dalla funzione findKeypoints(). Nella Fig. 20 potete vedere quali aree vengono individuate dalla funzione findKeypoints() e confrontate con il ritaglio dell'immagine che contiene il volto. Potete divertirvi a

sperimentare i risultati di questa funzione lanciando il programma del Listato 3 cambiando il nome del file. Utilizzatelo con le immagini "password.jpg" e passwordmatch.jpg. Corrispondono? E anche per questa volta è tutto. Speriamo di essere riusciti a trasmettere la complessità dell'argomento legato all'individuazione di oggetti e di particolari del corpo umano e di quanto sia stata semplificata la loro applicazione in programmi di individuazione e riconoscimento, grazie alla ricerca di algoritmi e metodi di trattamento di immagini e alla loro trasposizione in funzionalità software. In particolare abbiamo visto come la libreria SimpleCV permetta di condensare in una o comunque pochissime istruzioni una quantità di concetti ed elaborazioni fino a poco tempo fa alla portata di pochi specialisti. Per approfondire l'argomento, che abbiamo affrontato solo superficialmente potete riferirvi ai tanti articoli e libri disponibili sull'argomento. Per la maggior parte, purtroppo, in lingua inglese. ■



per il MATERIALE

Per effettuare gli esperimenti descritti in questo articolo sono stati utilizzati una board Raspberry Pi tipo B (cod. 7310-7568308RS Euro 38,00), il relativo modulo telecamera (cod. 7310-7757731RS Euro 31,00) ed il kit dello shield I²C di espansione IN/OUT per Raspberry Pi (cod. 7100-FT1093K Euro 32,00) presentato sul fascicolo n. 183. Tutti i prezzi si intendono IVA compresa.

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