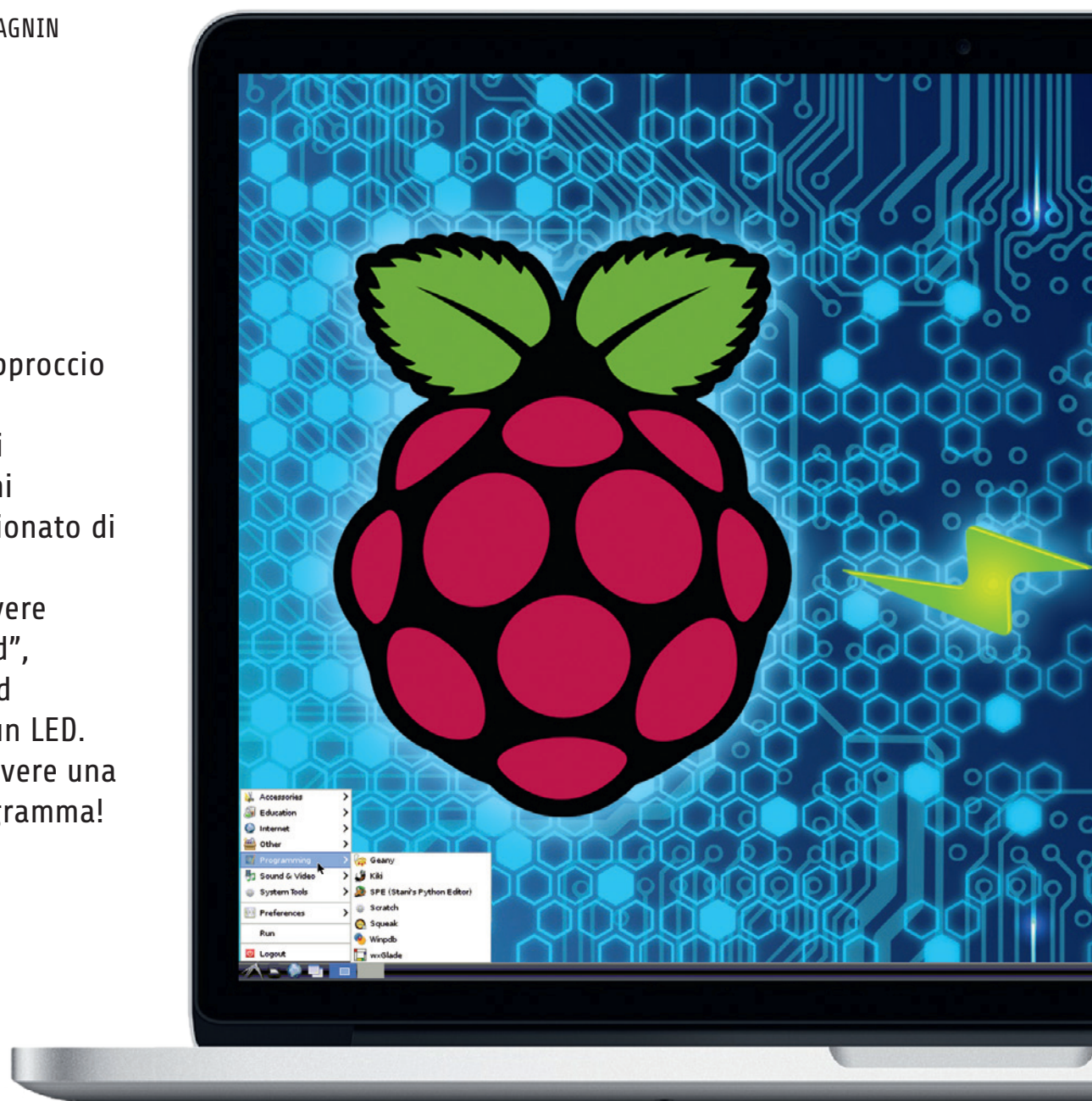
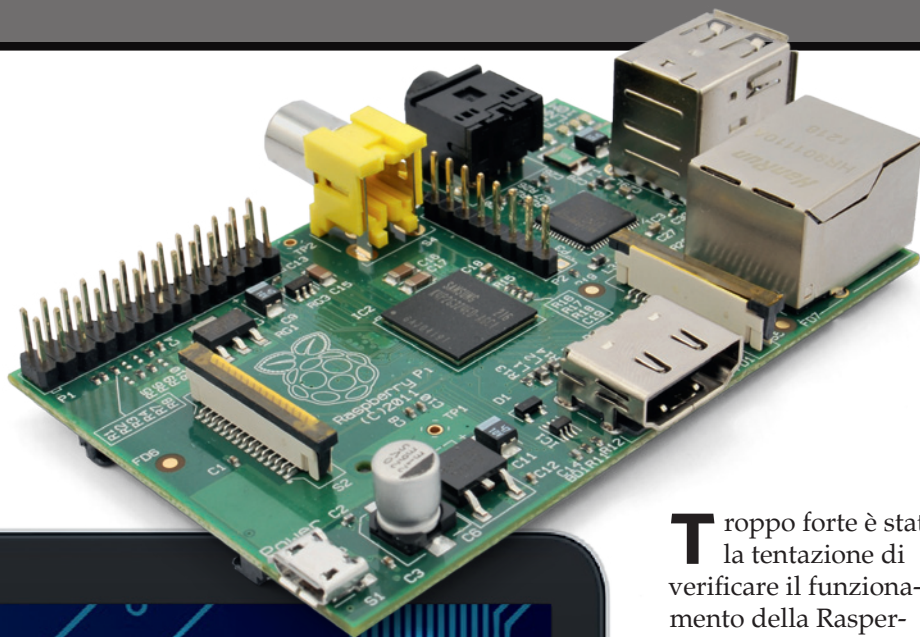


# METTIAMO IN FUNZIONE LA BOARD *RASPBERRYPI*

di MARCO MAGAGNIN

Un primo approccio alla scheda RaspberryPi con gli occhi dell'appassionato di elettronica. Oltre a scrivere "Hello World", proviamo ad accendere un LED. E senza scrivere una riga di programma!



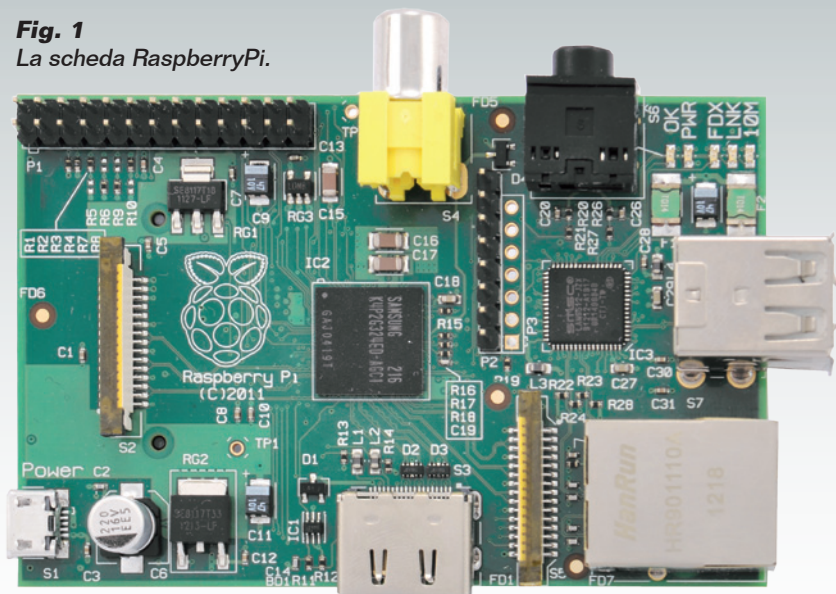


**T**roppo forte è stata la tentazione di verificare il funzionamento della RaspberryPi appena giunta in Redazione e, come avete visto nelle prime news pubblicate on-line, i risultati sono stati decisamente incoraggianti, tanto da indurci a dare il via ad una serie di articoli e progetti che avevamo in mente da tempo ed il cui scopo è quello di avvicinare gli appassionati di elettronica al mondo dei sistemi embedded. Già in passato abbiamo dovuto adeguarci a nuove realtà con l'arrivo dei circuiti integrati prima e dei microcontrollori poi; con questi ultimi siamo stati costretti ad occuparci di software e programmazione anche se, con i micro, le architetture a nostra disposizione supportano un solo programma (firmware) in esecuzione permanente sull'hardware. Dall'altra parte del "mondo" i computer seguivano

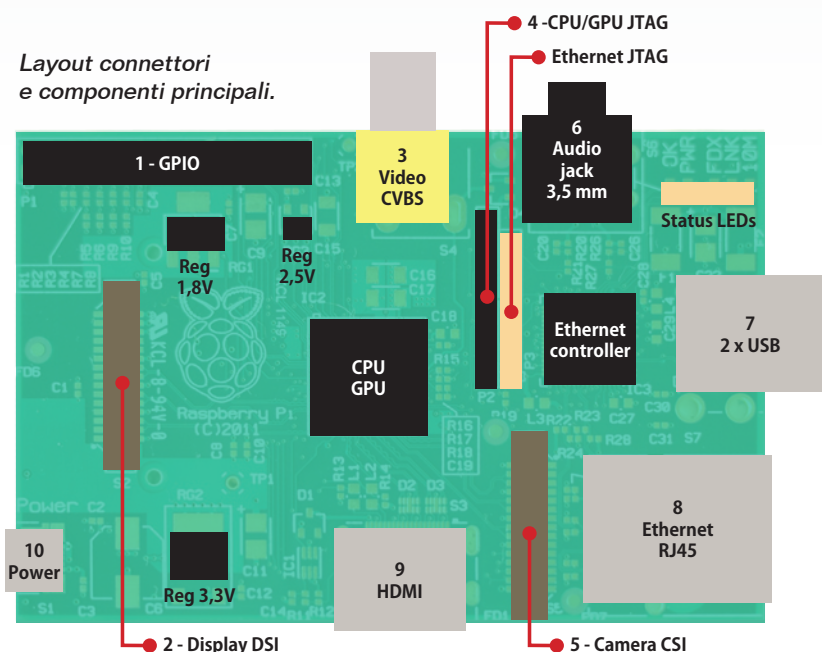
la stessa evoluzione costringendo gli informatici ad una analoga corsa all'aggiornamento. Lo sviluppo dei sistemi embedded, computer dalle dimensioni ormai paragonabili a quelle dei nostri moduli a microcontrollore ma con a bordo un vero e proprio sistema operativo, ha avvicinato questi due mondi tanto che lo scoccare dell'arco ormai è inevitabile. Da una parte i nostri sistemi a microcontrollore si evolvono mettendo a disposizione strumenti che schiudono la porta sul mondo della comunicazione Internet e Web in particolare, dall'altra schede PC-embedded si dotano di pin di I/O. Questo fatto non ci ha sorpresi, se ci riferiamo a schede come la Fox G20 e la sua nuova evoluzione Aria (di cui ci occuperemo prossimamente), ma nel caso della RaspberryPi ci ha incuriositi e fatto pensare. Già, perché la RaspberryPi nasce proprio con un connettore che espone una serie di pin di I/O, una interfaccia seriale, una SPI, una I2C e una alimentazione di 5V e 3,3V. Proprio per questo motivo ci siamo ripromessi di approfondire l'argomento con una serie di articoli destinati agli elettronici puri per capire come integrare e far comunicare tra di loro e con il mondo Internet e Web le schede



**Fig. 1**  
La scheda RaspberryPi.



Layout connettori  
e componenti principali.



più eterogenee come appunto la RaspberryPi, le schede Fox G20 e Aria, le OpenPicus, Arduino, Android e così via, sfruttando le opportunità di ciascuno dei singoli sistemi e le potenzialità dei sistemi embedded che, nella maggior parte dei casi sono basati su Linux.

Già, Linux, universalmente noto come il sistema operativo dalle potenzialità pressoché infinite ma che respinge sistematicamente i tentativi di avvicinamento da parte di chi non è "nato" Linux.

In genere ci si prova, si acquista un portatile apposta per sperimentare con Linux, meglio se già installato, si prova a configurare o personalizzare qualcosa e si entra nel file system spaesati; alla fine, delusi, si riformatta tutto e si torna al più amichevole Windows.

Bene, in questa serie di appuntamenti vorremmo provare a sfatare questa fama di Linux proponendo un approccio pratico che permetta di "fare qualcosa da subito", approfondendo poi

man mano i diversi argomenti ed aspetti per entrare sempre più a fondo nel mondo Linux Embedded. In più, proveremo a fare tutto questo dal punto di vista del cultore dell'elettronica; in altre parole, se l'esercizio base di tutti i tutorial destinati al mondo dell'informatica è ottenere il famoso "Hello World!", il nostro obiettivo sarà quello - tipico dei tutorial delle board hardware - di fare accendere inizialmente un LED. E, per sfatare ancora di più la nozione di Linux, aggiungiamo: senza dovere scrivere alcun programma!

Da questo punto di vista la RaspberryPi ci offre una serie di vantaggi, costa poco (e prima o poi arriverà), ospita la distribuzione di Linux Debian, universalmente ritenuta la più difficile, così non ci facciamo mancare nulla, tutto il software sta su una SD card, sistema operativo compreso, così se per caso combiniamo qualche pasticcio irreversibile, niente paura, rigeneriamo la SD e ripartiamo; infine, possiede quel curioso pettine di pin con gli I/O che ci permetteranno di utilizzarla come una vera board Linux embedded. Come ulteriore grossa opportunità, tutto ciò che apprenderemo con questa scheda (pardon, vero e proprio computer), lo potremo riutilizzare con altre schede Linux embedded, come la Fox G20, la nuova Aria e tante altre ancora come la MK802, un altro vero microcomputer delle dimensioni di una pen drive, nato per Android ma che gradisce anche Linux. E infine, ovviamente, tutto questo ci spianerà la strada verso Linux desktop, che sta prendendo sempre più piede. In un articolo destinato ad appassionati di informatica avremmo iniziato con la storia di Linux, per capire le motivazioni alla base della sua architettura e delle

modalità di configurazione ed utilizzo.

Visto che ci dobbiamo rivolgere ad appassionati di elettronica, iniziamo invece con la descrizione della scheda RaspberryPi rimandando gli altri argomenti a box specifici.

## LA RASPBERRYPI

La scheda RaspberryPi è un vero e proprio computer su scheda singola delle dimensioni di un cracker.

È stata sviluppata nel Regno Unito dalla RaspberryPi Foundation con l'intento di realizzare un computer a basso costo per stimolare l'insegnamento dell'informatica nelle scuole, in modo particolare nei paesi in via di sviluppo.

Il cuore del computer è il SoC (System On a Chip) Broadcom BCM2835 che include un processore ARM1176JZF-S da 700 MHz, un processore grafico (GPU) VideoCore IV, un processore di segnali digitali (DSP) e 256 Mb di RAM.

Non sono necessari dischi fissi o allo stato solido in quanto il sistema operativo e la memoria di massa sono allocati su SD Card che funge anche da unico device di boot.

Nello schema in Fig. 1 sono riportati i connettori ed i componenti principali presenti sulla scheda RaspberryPi:

1. Connettore GPIO
2. Connettore per un Monitor LCD/OLED con interfaccia DSI (Display Serial Interface) del tipo montato sui telefoni cellulari. Verificheremo una lista di modelli compatibili;
3. Connettore per un monitor o televisore con uscita video composito;
4. Connettore JTAG;
5. Connettore CSI (Camera Serial Interface). Tutte le videocamere

- utilizzate negli Smartphone con sensore CMOS dovrebbero essere compatibili con la RaspberryPi come, ad esempio, la economica OV5650 con interfaccia CSI. Anche videocamere con interfaccia I2C come la TCM8240MD dovrebbero poter essere utilizzate con una interfaccia CSI-2 Camera Bridge IC;
6. Jack audio in uscita (in ingresso bisogna utilizzare un microfono USB);
  7. Due connettori USB. Per espandere le porte USB si può usare un HUB esterno, però con alimentazione autonoma; a questo connettore si collegano la tastiera ed il mouse;
  8. Connettore Ethernet RJ45;
  9. Connettore HDMI per collegarsi ad un monitor video ad alta qualità;
  10. Connettore di alimentazione, in formato micro USB, quello dei caricabatteria dei telefonini, per alimentare la RaspberryPi è opportuno utilizzarne uno da almeno un Ampere.

## IL CONNETTORE GPIO

Vediamo subito come è configurato il connettore GPIO che

dovremo usare per accendere il nostro LED e per interfacciare la board con hardware esterno.

Da un punto di vista fisico si presenta come un connettore a passo 2,54 mm su due file di 13 pin ciascuna, per un totale di 26 pin (Fig. 2).

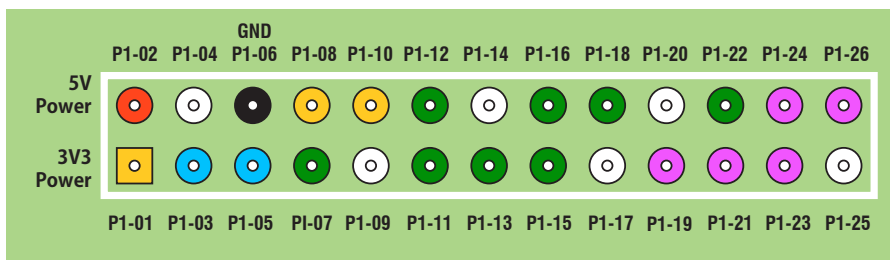
Per l'orientamento bisogna tenere conto che il pin 2 è quello in alto a sinistra tenendo la scheda con il connettore della SD Card posto sulla sinistra.

Su questo connettore sono presenti l'alimentazione a 3,3V sul pin P1-01, 5V sul pin P1-02, la massa sul pin P1-06 e poi altri 17 pin di input/output, che possono essere riconfigurati per ulteriori funzioni come I2C, SPI, UART, Seriale e altre. Ricordiamo da subito, prima di iniziare con esperimenti e applicazioni, che i pin di I/O funzionano con livelli di tensione 0 e 3,3V e NON tollerano il livello a 5V; inoltre non è prevista una protezione dalle sovratensioni. In sostanza, i pin GPIO sono connessi direttamente ai pin del processore. Tutto ciò costringe gli utilizzatori a progettare schede di interfaccia con buffer, conversione di livelli, I/O analogico e

**Tabella 1 - Specifiche tecniche della scheda RaspberryPi.**

| Caratteristiche tecniche    | Descrizione                                                                                                                                    |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| SoC:                        | Broadcom BCM2835 (CPU, GPU, DSP, SDRAM)                                                                                                        |
| CPU:                        | processore ARM1176JZF-S a 700 MHz (famiglia ARM11)                                                                                             |
| GPU:                        | Broadcom VideoCore IV,[55] OpenGL ES 2.0, 1080p30 h.264/MPEG-4 AVC high-profile decoder                                                        |
| Memoria (SDRAM):            | 256 MB (in condivisione con la GPU)                                                                                                            |
| Porte USB 2.0:              | 2 (USB hub integrato)                                                                                                                          |
| Uscite Video:               | Video composito RCA (PAL & NTSC), HDMI (rev 1.3 & 1.4), Pannelli LCD via DSI, 14 risoluzioni HDMI da 640×350 a 1920×1200 (standard PAL e NTSC) |
| Uscita Audio:               | jack da 3.5 mm, HDMI                                                                                                                           |
| Memoria di massa:           | slot per scheda SD / MMC / SDIO                                                                                                                |
| Network:                    | 10/100 Ethernet (RJ45)                                                                                                                         |
| Low-level peripherals:      | 8 × GPIO, UART, bus I <sup>2</sup> C, bus SPI con due chip selects, +3.3 V, +5 V, ground                                                       |
| Alimentazione:              | 700 mA (3.5 W)                                                                                                                                 |
| Ingressi tensione:          | 5 volt via MicroUSB o connettori GPIO                                                                                                          |
| Dimensioni:                 | 85.60 × 53.98 mm                                                                                                                               |
| Peso:                       | 45 g                                                                                                                                           |
| Sistemi operativi previsti: | Debian GNU/Linux, Fedora, Arch Linux ARM, RISC OS                                                                                              |





**Fig. 2** - La pinout del connettore GPIO.

dispositivi di protezione, in modo da evitare il rischio di danneggiare la RaspberryPi.

Vedremo in articolo come accedere ai pin di I/O, come configurarli e cambiarne il livello da alto a basso e viceversa; per il momento teniamo presente che la configurazione predefinita è in direzione OUT.

Nella **Tabella 2** sono descritti i nomi relativi ai vari pin da utilizzare nei programmi e nei comandi. Ricordiamo che ciascun pin è contraddistinto da un numero che ne identifica la posizione sul connettore (da P1-01 a P1-26) e da uno o più nomi che ne specificano la o le funzioni. Ad esempio, il pin P1-07, viene identificato col nome GPIO4. Tuttavia, come abbiamo già accennato, i pin del connettore, possono essere riconfigurati per poter accedere a molti dei segnali del processore. Per fare questo bisogna utilizzare i nomi alternativi di ciascun pin il cui elenco completo è disponibile al link [http://elinux.org/RPi\\_BCM2835\\_GPIOs](http://elinux.org/RPi_BCM2835_GPIOs). In questa

pagina riportiamo solamente alcune delle funzioni più utilizzate che sono elencate nella terza colonna della **Tabella 2** chiamata ALTØ. Ad esempio, il pin P1-05 nel caso normale è denominato GPIO1, con funzione di porta di ingresso e uscita. Lo stesso pin è anche assegnabile al clock del bus I2C; in questo caso, in un ipotetico programma, dovrà essere identificato col nome GPIO1\_ALTØ.

Alcune note aggiuntive sui pin presenti sul connettore:

- I pin 3 e 5, che possono essere utilizzati per il bus I2C sono dotati di resistenze di pull up da 1,8 kohm;
- Il pin 12 supporta il PWM;
- La massima corrente disponibile sul pin 3,3 volt è di 50 mA;
- La massima corrente disponibile sul pin 5 volt è di 300 mA.

Complessivamente i pin di ingresso e uscita presenti sul connettore GPIO sono 17. Infine un'ultima nota sul significato dei cinque LED di stato presenti sulla scheda:

- OK - Significa che è presente una scheda SDCard valida;
- PWR - Si accende in presenza di alimentazione;
- FDX - Indica una connessione LAN Full Duplex;
- LNK - Indica attività sulla rete LAN;
- 10M - Velocità della connessione LAN; quando acceso la connessione è a 100Mb.

## DIAMO VITA ALLA RASPBERRYPI

Per prima cosa colleghiamo alla RaspberryPi le periferiche necessarie al suo funzionamento e cioè:

- Tastiera, una qualsiasi tastiera USB, non c'è preferenza di porta, il sistema operativo riconosce automaticamente le periferiche;
- Mouse, anche qui un qualsiasi mouse USB nella porta rimasta libera;
- Una connessione di rete Ethernet, via cavo. Nel caso di rete cablata non c'è problema, utilizzate una delle terminazioni disponibili; in casa collegatevi ad una porta del router ADSL;
- Un monitor od un televisore, che può accettare in ingresso il segnale Video Composito o HDMI; consigliamo vivamente di optare per una soluzione HDMI, avrete una risoluzione decisamente migliore; in caso contrario non preoccupatevi, soffrirete un poco di più all'inizio, ma abbiamo già sperimentato il monitor in remoto con VNC che vi permetterà di vedere la vostra RaspberryPi sul monitor del vostro PC con la risoluzione che preferite; in seguito vi spiegheremo passo passo come installarlo e utilizzarlo;
- Un alimentatore a 5V (per ora non collegatelo!) con l'uscita micro USB, per intenderci quelli per ricaricare i telefo-

**Tabella 2** - Funzioni assegnate ai pin del connettore GPIO.

| Numero pin | Nome pin | Funzione ALTØ | Numero pin | Nome pin | Funzione ALTØ |
|------------|----------|---------------|------------|----------|---------------|
| P1-02      | 5V0      |               | P1-01      | 3V3      |               |
| P1-04      | DNC      |               | P1-03      | GPIO0    | I2C0_SDA      |
| P1-06      | GND      |               | P1-05      | GPIO1    | I2C0_SCL      |
| P1-08      | GPIO14   | UART0_TXD     | P1-07      | GPIO4    |               |
| P1-10      | GPIO15   | UART0_RXD     | P1-09      | DNC      |               |
| P1-12      | GPIO18   |               | P1-11      | GPIO17   |               |
| P1-14      | DNC      |               | P1-13      | GPIO21   | PCM_DIN       |
| P1-16      | GPIO23   |               | P1-15      | GPIO22   |               |
| P1-18      | GPIO24   |               | P1-17      | DNC      |               |
| P1-20      | DNC      |               | P1-19      | GPIO10   | SPIO_MOSI     |
| P1-22      | GPIO25   |               | P1-21      | GPIO9    | SPIO_MISO     |
| P1-24      | GPIO08   | SPIO_CEO_N    | P1-23      | GPIO11   | SPIO_SCLK     |
| P1-26      | GPIO07   | SPIO_CEO1_N   | P1-25      | DNC      |               |

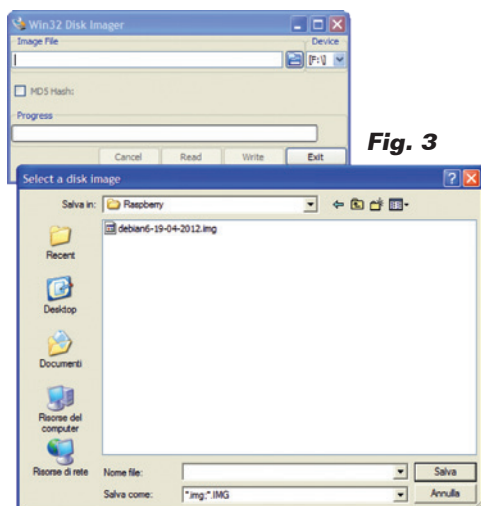


Fig. 3

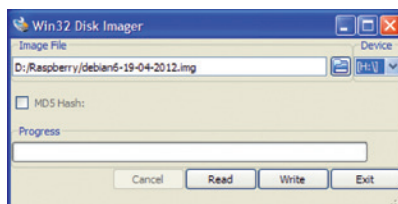


Fig. 4

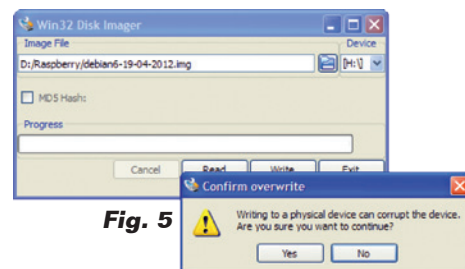


Fig. 5

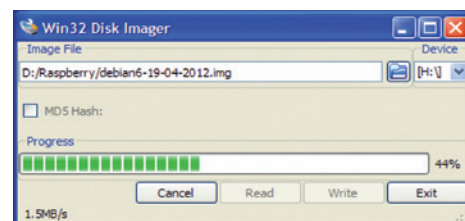


Fig. 6

nini; anche qui in alternativa possiamo collegare un alimentatore stabilizzato a 5V direttamente al positivo e negativo del connettore GPIO, il positivo al piedino 2, il negativo al 6; fate attenzione a non scambiare la polarità.

Passiamo ora al sistema operativo, che è il motore della nostra RaspberryPi. Per sua caratteristica progettuale la scheda accetta il boot, cioè il trasferimento in memoria dei primi programmi necessari al successivo caricamento del sistema operativo, unicamente da SDCard, che rappresenta l'equivalente del disco fisso principale del nostro PC.

In seguito esploreremo come velocizzare le operazioni di trasferimento dei file impiegando altre memorie di archiviazione di massa come chiavette USB o dischi fissi, magari allo stato solido. Come curiosità il termine boot, che è l'abbreviazione di bootstrap, è il laccetto posto dietro a scarponi e stivali che serve per aiutare a calzarli, tirandolo da dietro.

A questo punto, dunque, dobbiamo preparare una SD Card con il software necessario all'avvio della scheda ed al caricamento del sistema operativo Linux. Nel nostro caso abbiamo utiliz-

zato una SDCard da 8 Gb, ma è possibile impiegare card da 2 a 32 Gb a seconda delle proprie esigenze (la SD verrà utilizzata anche come memoria di massa, ovvero come una sorta di hard-disk). Raccomandiamo di sceglierne una ad accesso veloce in lettura e scrittura (Class 4) perché da questa caratteristica dipendono i tempi di caricamento dei programmi e dei dati.

A questo scopo utilizziamo il nostro PC connesso a Internet, che deve disporre di uno slot o un adattatore per SDCard.

Descriviamo la procedura riferendoci ad un PC con sistema operativo Windows in quanto riteniamo che quanti utilizzano un PC con Linux non abbiano bisogno di queste note.

La procedura consiste nello scaricare dalla sezione Downloads del sito ufficiale di RaspberryPi (<http://www.raspberrypi.org/downloads>) il file "debian6-19-04-2012.zip" o una versione più recente identificata da una data successiva. Il file

scaricato contiene una copia in formato binario (ovvero adatto ad una SDCard) del sistema operativo Linux che vogliamo utilizzare e che comprende anche il boot. Nel nostro caso abbiamo scaricato la distribuzione di Linux denominata Debian; nel mondo Linux con la parola "distribuzione" si intende un insieme di componenti di sistema operativo di base con l'aggiunta di applicazioni utente destinate a soddisfare una specifica categoria di utilizzatori, ad esempio grafici, webmaster, sviluppatori di applicazioni e così via. La distribuzione Debian è la più generalista possibile ed anche la preferita dagli hacker. Scaricato il file in una qualsiasi cartella del PC (nel nostro caso

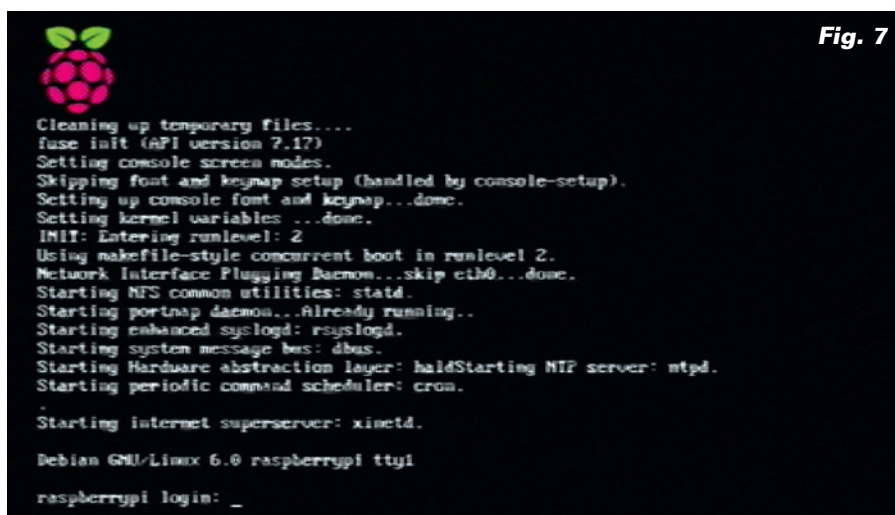


Fig. 7



# La storia di **LINUX**

La storia di Linux parte da lontano, un po' come quella di Internet e del suo protocollo di comunicazione TCP/IP.

Agli inizi dell'era informatica i giganteschi e costosissimi elaboratori disponibili allora si comportavano più o meno come i nostri microcontrollori. Accettavano un unico programma alla volta e di conseguenza un solo utente. Di fatto erano privi di un sistema operativo, ovvero di un "programma" centralizzato sempre in memoria che potesse coordinare il funzionamento di diversi programmi, ripartendo risorse e potenza di calcolo e di conseguenza accettare diversi utenti in contemporanea.

I primi sistemi operativi furono sviluppati negli anni

'50. Tra gli esempi si può citare il GMOS, il sistema operativo della General Motors sviluppato per l'IBM 701, oppure il FORTRAN Monitor System (FMS) sviluppato dall'aviazione americana per l'IBM 709.

Ciascun produttore investiva in quegli anni per sviluppare sistemi operativi adatti alle architetture delle proprie macchine; tra gli altri, negli anni '60, l'MIT, i laboratori della AT&T Bell e la General Electric iniziarono a sviluppare in consorzio il sistema operativo sperimentale Multics per il calcolatore GE-645 (**Fig. A**), ai tempi molto innovativo per quanto riguardava la flessibilità di gestione delle risorse e la scalabilità della configurazione. Purtroppo non ebbe molto succes-

so per via dei numerosi problemi incontrati nella progettazione e realizzazione. Ma l'idea era buona ed i laboratori AT&T Bell, lentamente si allontanarono dal consorzio e iniziarono una riprogettazione del sistema su una scala più ridotta che denominarono Unics. I ricercatori Ken Thompson, Dennis Ritchie, M. D. McIlroy, e J. F. Ossanna legarono il loro nome a questo progetto, sviluppando, insieme ad altri, su un PDP-7 usato, il primo file system organizzato gerarchicamente, il concetto di processo, i file di mappatura dei dispositivi, oggi noti come driver, e il primo interprete a linea di comando.

Il passo successivo fu lo sviluppo e l'integrazione del linguaggio C in Unics che nel

frattempo cambiava nome in UNIX. Space Travel è il video game che segna il raggiungimento degli obiettivi di UNIX come sistema operativo multiutente, multitasking, in grado di gestire la memoria virtuale, portabile, aperto e basato sul linguaggio C. Apertura e portabilità sono le caratteristiche che ne consentiranno il successo; apertura soprattutto verso la comunicazione con gli altri computer in quanto implementava nativamente il TCP/IP, il protocollo di Internet; portabilità perché la sua architettura "a cipolla" prevedeva un kernel centrale che si occupava di tutta la gestione dell'hardware che lo ospitava; per portare UNIX su un altro tipo di hardware ed adattarlo alla nuova architettura bastava

abbiamo chiamato la cartella Raspberry), dobbiamo trasferirlo su SDCard. Per fare ciò dobbiamo utilizzare il programma Win32DiskImager che può essere scaricato gratuitamente dal link <https://launchpad.net/win32-image-writer/+download> e che va installato sul PC.

Scompattiamo nella cartella in cui lo avevamo salvato, lo zip contenente Debian ed otteniamo il file "debian6-19-04-2012.img". Inseriamo la SDCard nell'apposito slot del PC, se disponibile, altrimenti utilizziamo un Multi Media Card Reader.

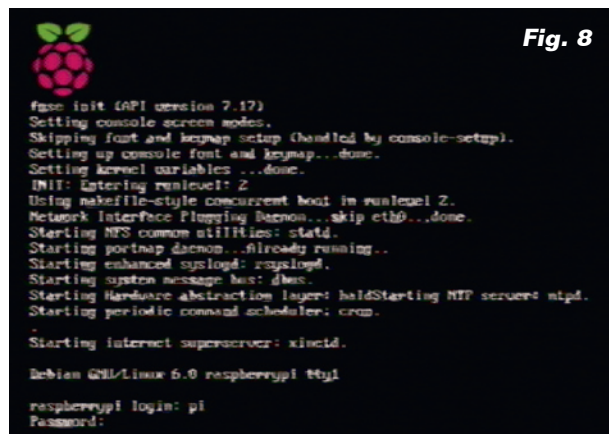
Apriamo Win32DiskImager e carichiamo il file di Debian che avevamo scompattato in precedenza (**Fig. 3**) e scegliamo come destinazione l'unità che identifica la SDCard (**Fig. 4**), premiamo "Write" (**Fig. 5**) e poi diamo l'OK. In **figura 6** si vede l'avanzamento del trasferimento su SDCard. Come notate esiste anche un pulsante "Read" che permette di fare un backup della SDCard ogni volta che vogliamo, ad esempio per creare una SDCard per una seconda scheda con le stesse configurazioni.

Lo spazio occupato dalle infor-

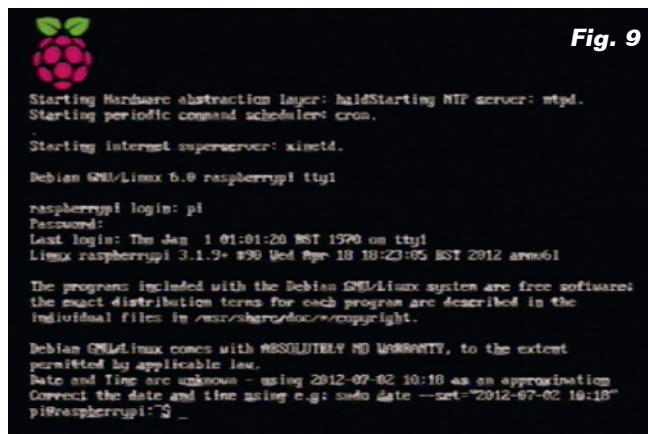
mazioni trasferite occupa circa 2Gb ma l'eventuale spazio eccedente non potrà essere al momento utilizzato; vedremo in seguito come fare in modo da rendere disponibile al sistema tutta la capacità della SDCard.

A questo punto dimentichiamoci del nostro PC, preleviamo la scheda SDCard e inseriamola nello slot della RaspberryPi alla quale, come abbiamo detto in precedenza, avremo già collegato video, tastiera, eccetera.

La scheda non dispone di interruttore di accensione, per accendere si dà alimentazione, per



**Fig. 8**



**Fig. 9**

Vent'anni dopo, negli anni '90, sfruttando proprio le caratteristiche di portabilità, Andrew Tanenbaum scrive una versione di UNIX®, chiamata MINIX (MInimal uNIX) basata su un microkernel, che aveva la capacità di girare sui PC. Ed è proprio a questo progetto che, nei primi anni 90, si ispirò Linus Torvalds, il creatore di Linux.

Linux si è rapidamente diffuso da progetto di una singola persona a progetto di sviluppo mondiale grazie al coinvolgimento di una comunità di migliaia di programmatori e dalla adozione della licenza GPL

(GNU General Public License). Sotto la GPL, il kernel Linux è stato protetto da sfruttamento commerciale ed ha anche tratto benefici dai programmi disponibili in user-space che facevano parte del progetto GNU (di Richard Stallman, che ha scritto tanto di quel codice che in confronto il sorgente del kernel Linux appare minuscolo). Questo ha permesso una crescita esplosiva, la diramazione in diverse “distribuzioni” con l’inglobamento di altri progetti, package e applicativi, shell a linea di comando e ambienti grafici note come Debian, Fedora, Ubuntu, Knoppix e infinite altre. Alcune in grado di partire direttamente da CD, da chiavetta USB o da “SD Card”.



Alcune distribuzioni sono destinate a specifici campi di utilizzo, come la sicurezza informatica, firewall, gateway, grossi provider di servizi o anche gestione di CNC o la gestione video e grafica. Tutto questo ha portato allo sviluppo di nuove professionalità, molto ricercate nel mondo del lavoro

come specialisti di networking, di service providing e di sicurezza (questi ultimi spesso provenienti dalle file degli hacker). Nel frattempo il “piccolo” kernel ha raggiunto la dimensione di circa 6 milioni di righe di codice, sempre portabili però, come dimostra la nostra RaspberryPi.

spegnerla la si toglie. Alcuni LED si accenderanno e dopo qualche attimo vedremo l'immagine di un piccolo lampone sullo schermo, seguito da una lista di comandi che scorreranno velocemente (Fig. 7); aspettate alcune decine di secondi fino al termine ed alla comparsa di una riga contenente:

*raspberrypi login:*

Poche righe prima, probabilmente, troverete l'indirizzo IP della rete interna assegnato al vostro dispositivo dal router DHCP, che sta a significare una

corretta connessione alla rete. Soffermiamoci ora su un concetto importante: Linux permette l'accesso e l'esecuzione di comandi solo a utenti riconosciuti e debitamente autorizzati (userid e password). L'amministratore che ha tutti i diritti di accesso e intervento è l'utente SuperUser al quale competono anche i diritti di "root". Root è la radice dell'albero delle cartelle e dei file che costituiscono la struttura del sistema operativo. Vedremo in seguito come attivare l'utente SuperUser per poter eseguire quei comandi che richiedono

questo livello di autORIZZAZIONE. Ogni distribuzione di Linux ha all'interno un utente generico già attivato, che permette di poter iniziare a lavorare da subito. Nel nostro caso l'utente già attivato è "pi" e la password "raspberrypi" (Fig. 8 e 9). Quindi rispondiamo:

```
raspberrypi login: pi
```

password: raspberry

Bene, siamo operativi, e andiamo a risolvere subito il primo problemino di configurazione ovvero la tastiera che, di default, è quella inglese e quindi i carat-



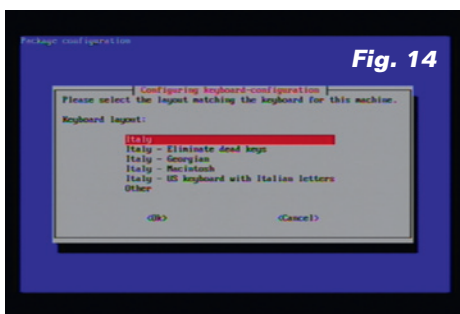


Fig. 14

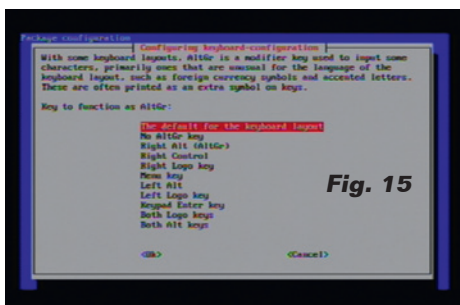


Fig. 15

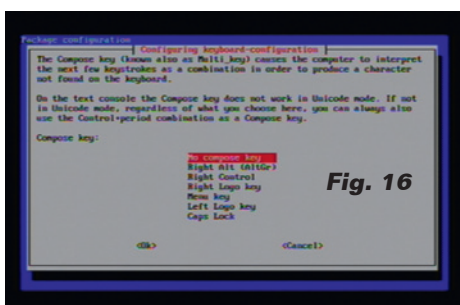


Fig. 16



Fig. 17

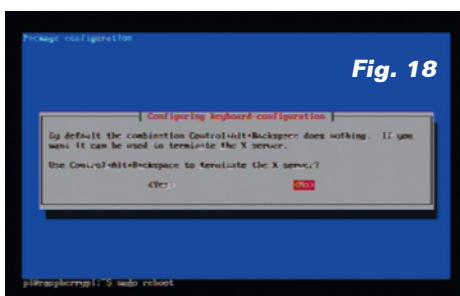


Fig. 18

teri speciali non corrispondono. Come prima operazione configuriamo le impostazioni di tastiera, per adottare quella italiana, con il comando:

*sudo dpkg-reconfigure keyboard-configuration*

Per ottenere, in questo momento, il carattere - (meno), è necessario digitare sulla tastiera (italiana...) il carattere ? (punto di domanda).

Il comando *dpkg-reconfigure* fa parte di quelli che richiedono i privilegi di SuperUser, il comando *sudo* (superuser do) assegna temporaneamente i privilegi di SuperUser all'utente corrente (Fig. 10), in un modo simile a quando si seleziona in Windows l'opzione "Esegui come Amministratore".

Eseguito il comando, appariranno le finestre di configurazione della tastiera da Fig. 11. a Fig. 18, assolutamente intuitive.

Per rendere effettive le configurazioni eseguiamo il reboot (riavvio) della scheda con il comando:

*sudo reboot*

Dopo aver effettuato nuovamente il login, attiviamo ora l'utente SuperUser che ha sempre come nome "root" ed al quale va definita la password per poterlo attivare. Digitiamo il comando (Fig. 19):

*sudo passwd*

ed in seguito inseriamo una password, per due volte, e poi annotiamola accuratamente (Fig. 20 e 21).

Attenzione, se digitate unicamente

*passwd*

cambierete la password all'utente corrente, e non attiverete il SuperUser.

Fate attenzione con le password perché in Linux è difficile recuperare le password e pressoché

Linux è un sistema operativo multitasking e multiutente, ossia può gestire più processi e più utenti contemporaneamente. Un processo è una unità minima di elaborazione, se vediamo la funzione di multitasking di un sistema operativo come un tavolo di lavoro con sopra molte scatole, ciascuna delle quali contiene un compito da eseguire, tipo montare giocattoli di legno, e chi si occupa dei montaggi passa da una scatola all'altra per esempio incollando due parti e poi lasciarle asciugare mentre nel frattempo incolla altre parti in una scatola differente, questo è precisamente il comportamento del sistema, dove in ciascuna scatola sono contenuti uno o un insieme omogeneo di quelli che noi chiamiamo programmi.

Il cuore del sistema operativo Linux è il kernel, che in questo sistema è rigidamente l'unico ad avere accesso a basso livello alle risorse di sistema e a presentarle ai livelli superiori in modo standardizzato e indipendente dal processore hardware. I servizi e le applicazioni a livello superiore non devono curarsi su quale hardware verranno eseguite e quindi non devono essere modificate per adattarsi ad hardware diversi. Questa caratteristica di Linux ha contribuito alla sua fortuna, in quanto lo ha reso estremamente portatile, per "portare" Linux su un nuovo processore è sufficiente adattare il kernel alla nuova architettura, tutto il resto rimane invariato.

Anche le interfacce grafiche desktop come Gnome o KDE, non sono funzionalità offerte dal kernel, bensì normalissimi programmi

impossibile recuperare quella del SuperUser.

Al momento non abbiate timore: fortunatamente se qualcosa va storto ricreate una SD con il file d'immagine originale presente sul vostro PC e ricominciate da capo.

Ora concentriamoci sull'accensione del nostro LED; per l'accesso ai pin del GPIO è necessario utilizzare privilegi di SuperUser, e quindi nel nostro caso ci autenteremo come utente "root" con il comando:

*su*

e con la password relativa. Ora colleghiamo un LED rosso con in serie una resistenza da 120 Ohm tra il pin 7 (GPIO4) del



esterni, al pari di un OpenOffice.org o Firefox. Ecco perché in Linux potete scegliere più interfacce grafiche, è semplicemente un programma come tutti gli altri. In Windows invece l'interfaccia grafica è integrata nel sistema operativo, ed in alcune versioni, anche il browser per la navigazione Internet.

Quella che comunemente viene chiamata distribuzione, Debian, Ubuntu, openSUSE, RedHat solo per fare qualche nome fra le più conosciute, è l'insieme del sistema operativo Linux più una raccolta coerente di programmi usati dagli utenti per le loro normali attività: programmi di grafica, produttività personale, navigazione Internet, player audio/video e così via.

Vediamo ora quali sono le funzioni svolte dal kernel per eseguire i processi, ovvero i contenitori delle applicazioni dei nostri programmi. In figura sono rappresentati schematicamente i tre sottosistemi principali:

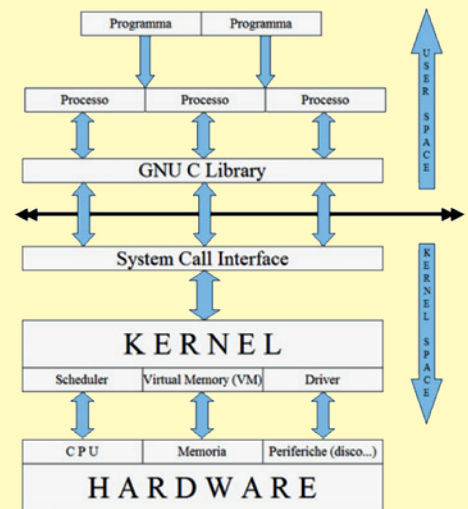
- Gestione dei processi tramite lo Scheduler. Questa parte del kernel si occupa di tutto ciò che riguarda la gestione di un processo, ad esempio di stabilire l'ordine di esecuzione dei processi, il tempo da dedicare al processo, la sospensione dell'elaborazione, la ripresa, lo stop e così via;
- Gestione della memoria tramite la memoria virtuale (VM). La memoria fisica non

è liberamente accessibile ai processi. E' il VM che, anche con l'aiuto del Memory Management Unit (MMU), apposita unità presente direttamente nell'hardware dei microprocessori, rimappa opportunamente quella fisica in uno spazio virtuale che rende disponibile al processo, la scatola dell'esempio iniziale. In questo modo, ogni processo non accede mai alla memoria reale del sistema, ma solo a quella resagli disponibile dal VM. Rinchiuso nel suo spazio di memoria dedicato, il processo non può creare problemi accedendo per sbaglio allo spazio di un altro processo (chi ha usato le prime versioni di Windows, sa quanto fossero frequenti i blocchi dell'intero sistema causati proprio da problemi di questo tipo). Quando la memoria RAM poi per qualche motivo diventa insufficiente, è sempre il VM a gestire lo swap, lo spostamento cioè di parti della memoria sul disco fisso per liberare spazio, nel caso della RaspberryPi nella SD card.

- Gestione delle periferiche tramite i Driver. Questo sottosistema del kernel si occupa di gestire gli accessi alle periferiche rendendole disponibili ai programmi. La particolarità assolutamente vincente di questo sottosistema è che fornisce una modalità comune (un'interfaccia) di accesso a periferiche anche molto diverse fra loro, porte seriali, dischi rigidi, scheda audio tanto per fare degli esempi, trattandole tutte come se fossero dei

file; da qui uno dei motti di Linux: tutto è un file. Ricordiamocene sempre quando cercheremo di installare e utilizzare qualche nuova periferica recalcitrante sul nostro Linux. Ce ne ricorderemo anche quando dovremo accendere il nostro LED.

Ecco, questa è in estrema sintesi l'architettura di un sistema Linux, giusto per iniziare a capire; in realtà le cose sono ben più complesse, come potreste vedere dalle dimensioni di qualunque libro e manuale dedicato al kernel di Linux.



Schema della architettura di Linux.

connettore della Raspberry, e la massa (GND, pin6).  
Digitiamo (**Fig. 22**):

```
echo "4" > /sys/class/gpio/export
```

Questo comando assomiglia molto ai comandi DOS di Microsoft e di fatto significa: esegui il programma export che sta nella cartella `/sys/class/gpio` passando gli il parametro "4", che è il pin che vogliamo usare.

Questo comando crea una struttura di file nella cartella `/sys/class/gpio/gpiochip0/subsystem` mettendo il device GPIO4 in condizione di essere gestito con i comandi successivi.

Prossimamente spiegheremo i metodi per scorrere le cartelle ed i file, sia da linea di coman-

do che utilizzando l'interfaccia grafica. Per ora ci interessa sapere che in questa cartella sono presenti, tra gli altri, due file: `direction` e `value`, entrambi normali file di testo. In `direction` è contenuta la direzione del pin in o out, la direzione predefinita assegnata quando si esporta un device di tipo pin è out. Nel file `value` vi è il livello del pin: "1" per livello alto "0" per livello basso.

A conferma che in LINUX (quasi) tutto è un file, potete verificare il contenuto di questi file con:

```
cat /sys/class/gpio/gpio4/direction
```

otterremo "out" e

```
cat /sys/class/gpio/gpio4/value
```

otterremo "0".

Per inciso, *cat* è il comando che stampa a console il contenuto dell'argomento del comando, nel nostro caso i file di gestione del pin.

Per assegnare una direzione al pin si usa il comando:

```
echo "out" > /sys/class/gpio/gpio4/direction
```

che scrive "out" nel file `direction`. Per trasformarlo in input, nel caso il pin sia usato come ingresso, si usa il comando:

```
echo "in" > /sys/class/gpio/gpio4/direction
```

Ricordiamo di fare attenzione ai livelli, i pin GPIO NON tollerano tensioni superiori ai 3,3 volt. Ora diamo un'occhiata al nostro

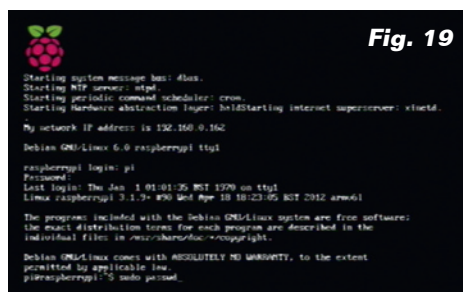


Fig. 19

LED e verifichiamo che sia spento, in quanto il valore del pin è "0" (Fig. 23)

Per accenderlo basta portare a "1" il livello del pin, con il comando:

*echo "1" > /sys/class/gpio/gpio4/value*

Verifichiamo ora che il LED sia acceso. Ce l'abbiamo fatta! (Fig. 24). Ora lo spegniamo col comando:

*echo "0" > /sys/class/gpio/gpio4/value*

Per verificare il valore corrente usiamo sempre il comando:

*cat /sys/class/gpio/gpio4/value*

Bene, alla fine ora "smontiamo" il device GPIO4 con il comando:

*echo "4" > /sys/class/gpio/unexport*

Questo comando corrisponde, in Windows, a "rimozione sicura dell'hardware". Partendo da questo punto si possono scrivere applicazioni in svariati linguaggi con funzionalità locali e ad accesso remoto, come vedremo nei prossimi appuntamenti. Per spegnere la RaspberryPi date il comando:

*halt*

Se invece volete divertirvi con l'interfaccia grafica date il comando:

*startx*

Giocate fin che volete, per uscire, come in Windows scegliete Logout dal menù sulla barra e poi Shutdown.

Nei prossimi mesi approfondiremo una serie di argomenti legati al mondo Linux embedded, per esempio la possibilità di gestire i nostri pin GPIO dall'esterno, via web, memorizzandone gli stati su database, oppure di gestire applicazioni che girano su altri microcontrollori per mezzo di chiamate web service (API), in modo da realizzare un network di controllori eterogenei gestiti centralmente dal sistema Linux embedded, collegare telecamere al web, integrare il text to speech e altro ancora. Difficile? Probabilmente meno di quanto pensiate.

**La scheda RaspberryPi viene distribuita in esclusiva mondiale da Rs Components e da Farnell.**

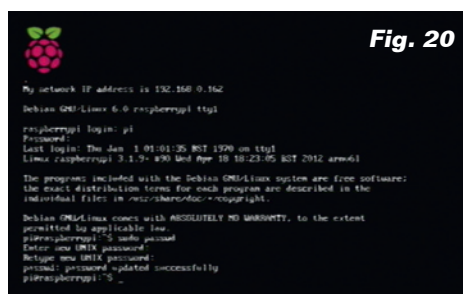


Fig. 20

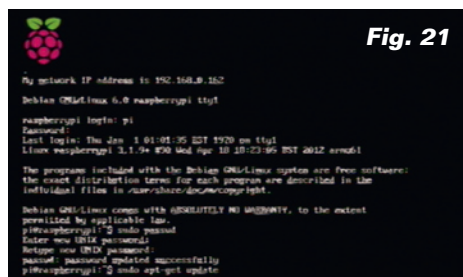


Fig. 21

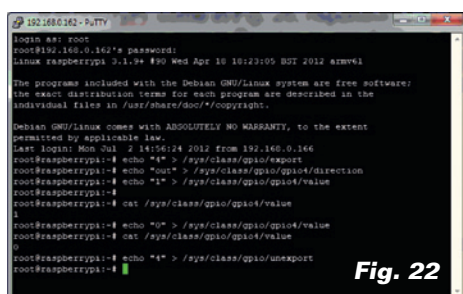


Fig. 22

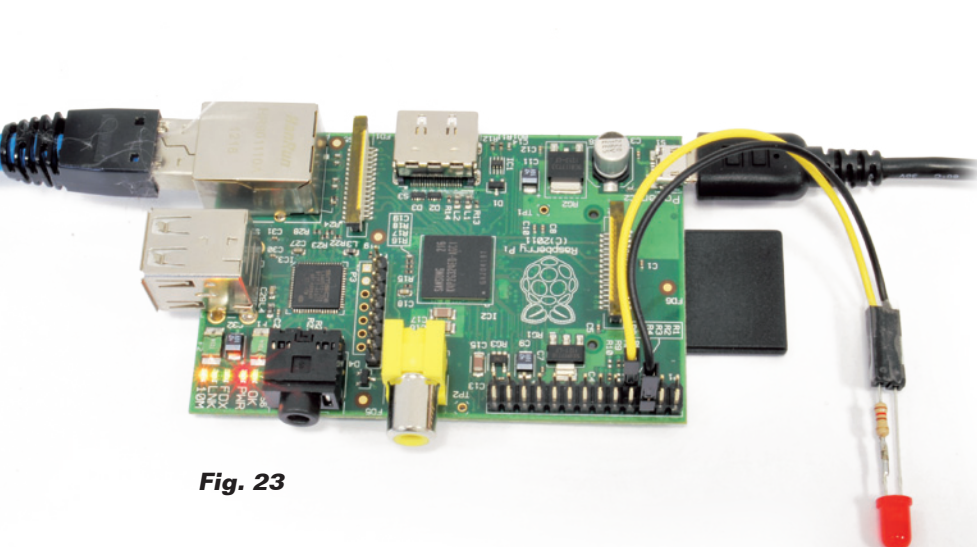


Fig. 23

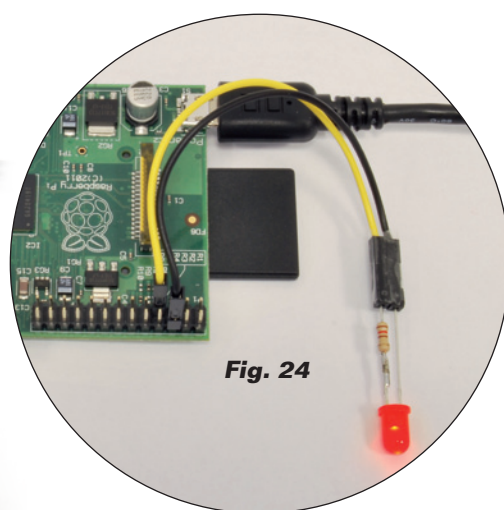
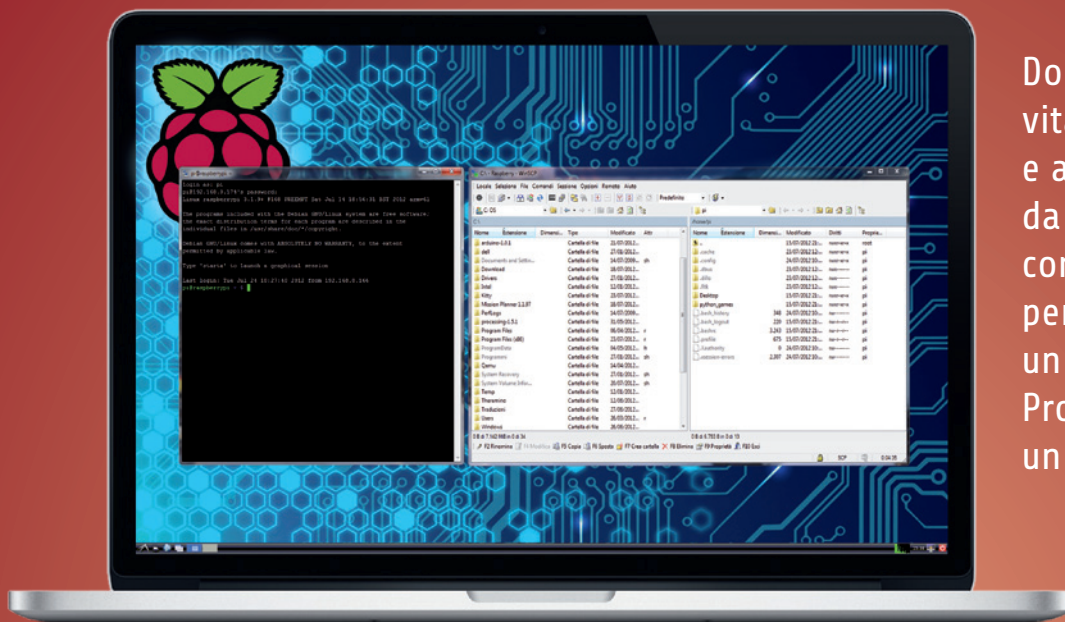


Fig. 24





Dopo aver provato a dare vita alla RaspberryPi e ad accendere un LED da linea di programma, cominciamo a configurarla per farla funzionare come un sistema embedded. Proviamo anche a scrivere un primo programma.

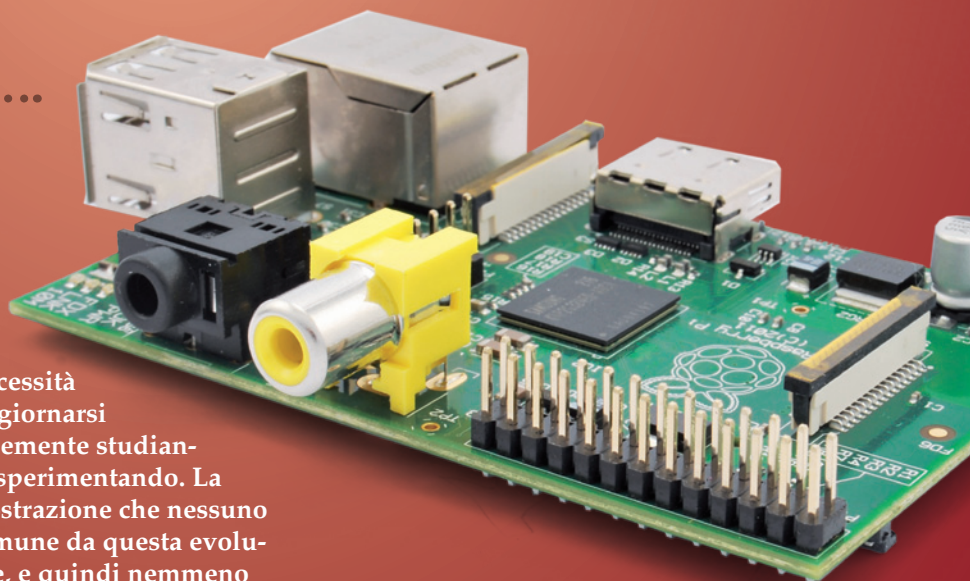


# RASPBERRYPI IL TUO PROSSIMO SISTEMA EMBEDDED

di MARCO MAGAGNIN

**N**el numero di Luglio/Agosto, sia nell'editoriale che in alcune risposte ai lettori nella rubrica Elettronica Insieme, abbiamo sottolineato la sempre più rapida evoluzione degli ambienti tecnologici, la tendenza alla sovrapposizione di discipline e competenze in settori sempre più vasti,

la necessità di aggiornarsi velocemente studiando e sperimentando. La dimostrazione che nessuno è immune da questa evoluzione, e quindi nemmeno





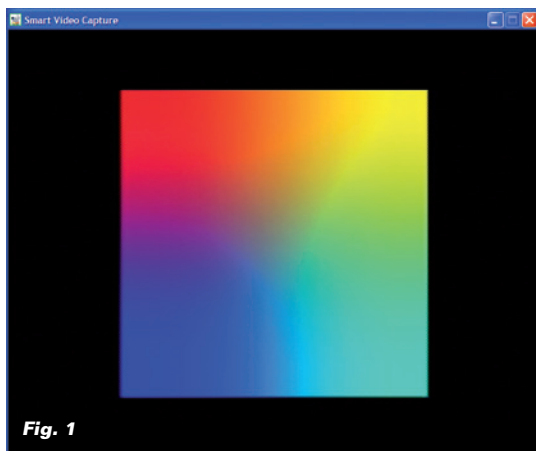


Fig. 1

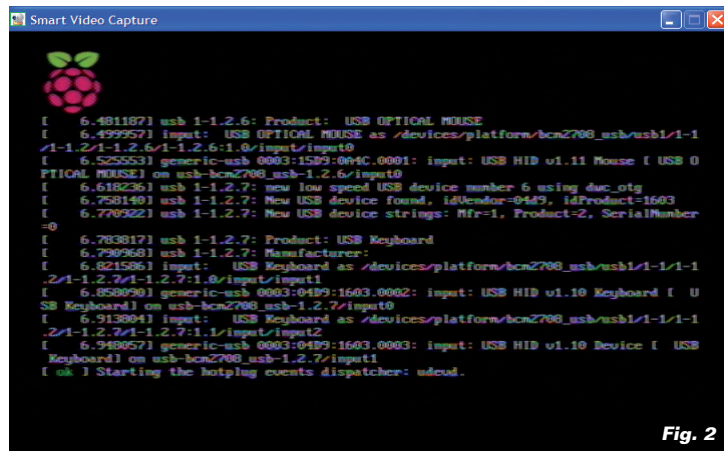


Fig. 2

noi, l'abbiamo avuta non appena terminato il primo articolo di "avvicinamento" a RaspberryPi. Nell'accingerci a preparare questo secondo appuntamento si sono verificati due eventi che ci hanno fatto pensare: il primo è il nuovo sistema operativo - Raspbian - che ha mandato in pensione la versione precedente, addirittura eliminata dai file scaricabili sul sito ufficiale. Il secondo evento è la crescente disponibilità di alternative che, in forme diverse, stanno aparendo sul mercato. Prendiamo come esempio la scheda MK802: nata per Android, ma orientata anche a Linux, e poi Aria, di ACME System, un piccolo gioiello di quattro centimetri quadrati, dal costo ridottissimo e, ovviamente Linux embedded. Per quanto riguarda Raspbian, nessuna preoccupazione: l'abbiamo adottato

immediatamente con molta soddisfazione; ci (e vi) semplificherà la vita nell'avvicinamento al mondo di Linux. Tutto questo ci spinge a continuare sulla strada del mondo embedded con particolare attenzione a Linux, grazie anche alle discussioni ed ai litigi in redazione (pardon, "brainstorming") che ci hanno portato a definire una linea editoriale in cui affiancheremo la visione di Linux come sistema di gestione e coordinamento di servizi, processi e programmi, ad un progetto concreto - da sviluppare passo a passo - proponendo ogni volta un tema che permetta di "alzare l'asticella" della conoscenza e della comprensione. In questa puntata vi proponiamo una breve introduzione al sistema Raspbian, alla sua installazione e alla configurazione, poi vi suggeriamo un'ipotesi di progetto in grande, pensato per coprire molte delle funzionalità e svisce-

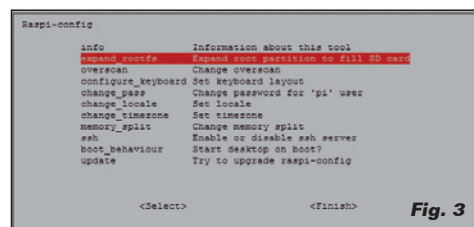


Fig. 3

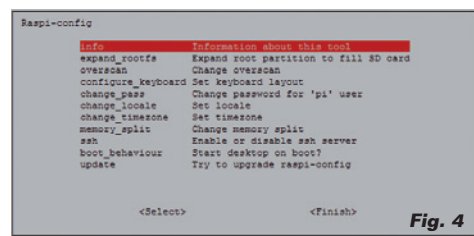


Fig. 4

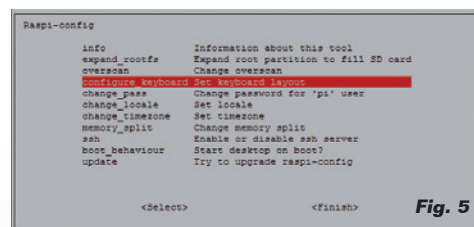
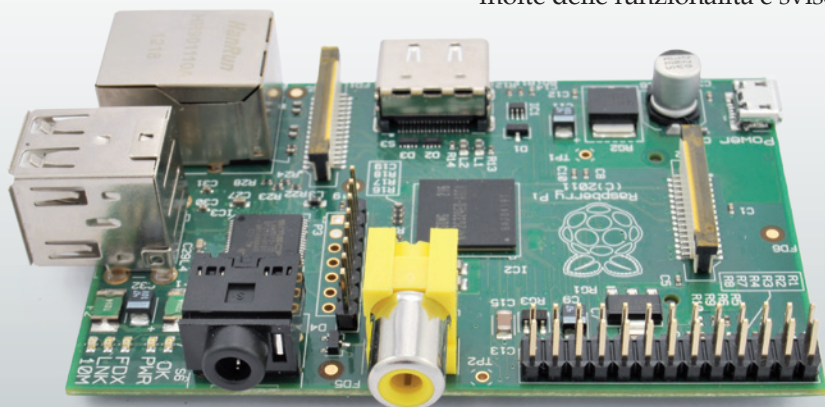


Fig. 5

rare sempre di più funzionalità e opportunità per l'adozione di un sistema embedded, quindi - per rimanere nel concreto - una serie di configurazioni per far apparire RaspberryPi come un sistema embedded e infine un primo piccolo programma - scritto in linguaggio Python - per dimostrare, se ancora ce ne fosse bisogno, che Linux è meno difficile di quel che si dice.

## RASPBIAN

Due parole sulla nuova versione di sistema operativo Raspbian: una distribuzione "figlia" di



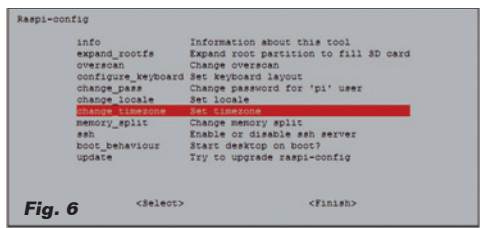


Fig. 6

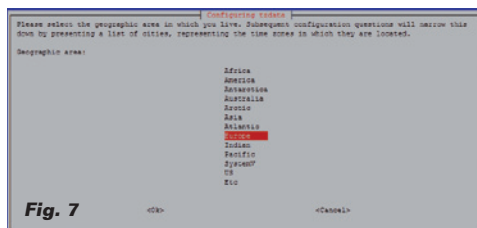


Fig. 7



Fig. 8

Debian (versione “wheezy”), ottimizzata per l’uso su Raspberry. Raspbian sfrutta meglio le risorse a bordo della scheda come la memoria reale e virtuale, ha un pieno utilizzo del processore in virgola mobile ed ha una velocità maggiore di navigazione web. Inoltre - molto apprezzabile dal nostro punto di vista - dispone di un’applicazione di configurazione che parte subito dopo la prima installazione e che può essere richiamata in ogni momento. Questa applicazione semplifica molto una serie di impostazioni iniziali che, fra l’altro, avremmo dovuto per forza spiegarvi in questo articolo, introducendo una quantità di nuovi comandi e relativi concetti.

Riprendiamo ora il tema del primo avvio del sistema. Se siete partiti con la “vecchia” versione di sistema operativo, passate senza indugi a Raspbian, se partite ora non c’è problema. Preparate una SDCard utilizzando il programma Win32DiskImager, come descritto nel numero 168 di Luglio/Agosto della rivista, solo che questa volta il file di immagine da scaricare dal sito <http://www.raspberrypi.org/> è “2012-07-15-wheezy-raspbian.zip”. Preparata la SDCard, inseritela nella RaspberryPI, collegate tutti i dispositivi esterni come di consueto e data corrente. Questa volta apparirà una veloce videata di test del monitor (Fig. 1), poi una videata con la lista dei comandi di boot in rapido scorrimento (Fig. 2), ed infine la sequenza di boot si fermerà con la schermata di Menu dell’applicazione di configurazione (Fig. 3).

Per prima cosa, usando le frecce, selezioniamo la voce (Fig. 4):

expand\_rootfs

Se vogliamo selezionare le opzioni in basso alla pagina di menu o in quello di livello successivo, useremo il tasto “Tab”.

Expand\_rootfs ci permette di espandere la partizione, ovvero lo spazio occupato da Linux nella SDCard, fino a sfruttare l’intera capacità della SDCard stessa; in altre parole, se abbiamo caricato l’immagine di Raspbian da 1,9 GB su una SDCard da 8 GB, il sistema continuerà a vedere solo 1,9 GB fino a quando non avremo utilizzato questa funzione.

Per eseguire questa operazione con la versione di sistema operativo precedente sarebbe stato necessario eseguire una serie di operazioni, utilizzando il programma Gparted su una seconda versione della SDCard, in modo da espandere lo spazio riconosciuto fino a farlo coincidere con quello disponibile. Se necessario torneremo su questo argomento in futuro. Per ora diamo invio, e poi ancora invio nelle videate successive. Al termine selezioniamo la voce (Fig. 5):

configure\_keyboard

ed eseguiamo le stesse operazioni di configurazione della tastiera che abbiamo visto nel numero scorso. Approfittiamone ora per configurare correttamente le impostazioni di data e ora con la voce (Fig. 6):

change\_timezone

Nelle videate successive scegliamo “Europe” (Fig. 7) e “Rome” (Fig. 8).

Solo due parole sulla voce di menu “memory\_split”: questa funzione ci permette di suddividere la memoria RAM tra il processore ARM ed il processore Video, che ci ricorda una delle più importanti caratteristiche di Linux, e cioè la possibilità di suddividere le risorse disponibili tra i componenti hardware. Nella configurazione predefinita vengono assegnati 224 MB al processore ARM e 32MB al processore video. Le altre configurazioni prevedono una maggiore quantità di memoria assegnata al processore video a scapito di quella assegnata all’ARM. Visto che ci stiamo orientando verso un utilizzo “embedded” di RaspberryPi, con molte applicazioni “interne” e quasi nessun utilizzo del video, lasciamo la configurazione predefinita.

La voce successiva ci permette di anticipare un’ulteriore considerazione; visto che ci stiamo orientando ad utilizzare RaspberryPi come un sistema Linux embedded, dobbiamo tenere presente che questo tipo di sistemi, normalmente, non hanno la possibilità di essere gestiti direttamente collegando video, tastiera e mouse. I sistemi embedded, di solito, nascono per essere gestiti con strumenti come un terminale remoto a linea di comando ed un’applicazione remota di file transfer.

Esattamente le funzionalità messe a disposizione dal ser-



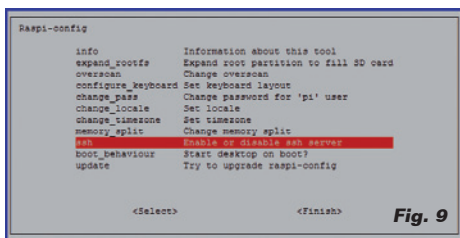


Fig. 9

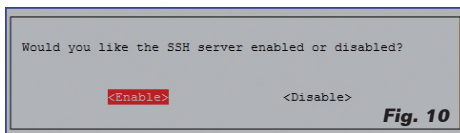


Fig. 10

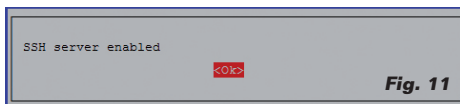


Fig. 11

ver SSH (Secure SHell): la sua attivazione dalla voce di menu "ssh" ci permette di trasformare immediatamente RaspberryPi in un sistema gestibile da remoto. Selezioniamo la voce dal menu (Fig. 9), nella videata successiva selezioniamo "Enable" (Fig. 10) e "OK" in quella dopo ancora (Fig. 11).

Ora una breve spiegazione sulla voce "boot\_behavior": se non modifichiamo nulla, alla partenza RaspberryPi terminerà la procedura di partenza con la richiesta del "login:", se invece scegliamo <YES> nella videata di scelta successiva, alla fine del caricamento verrà avviata l'interfaccia grafica che permetterà di utilizzare Raspberry come un PC qualunque. Lasciate la configurazione su <NO> e se volete provare ugualmente l'interfaccia grafica usate, dopo il login, il comando

```
startx
```

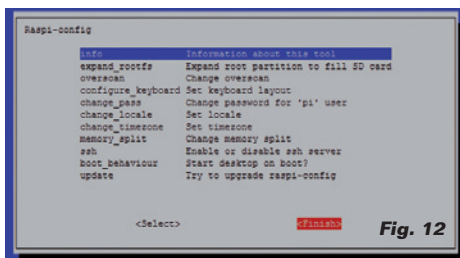


Fig. 12

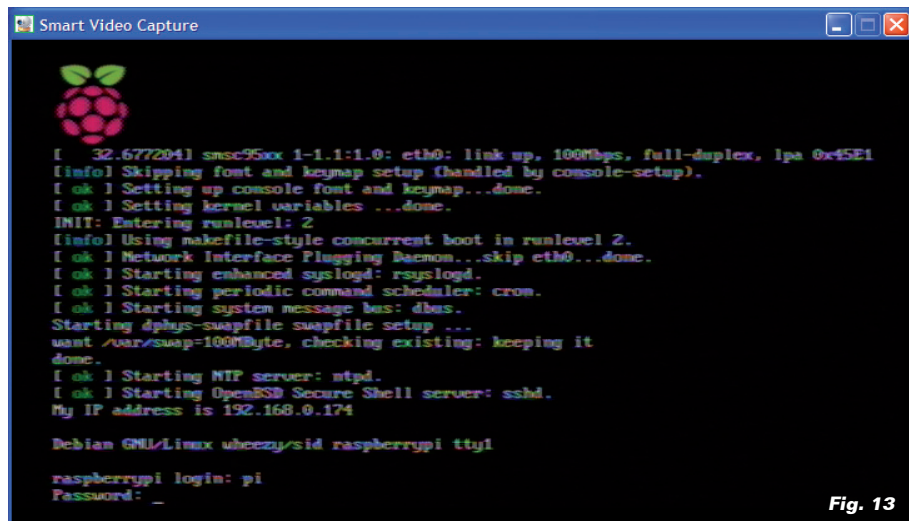


Fig. 13

Finite le configurazioni con la scelta <Finish> (Fig. 12, usate il tasto "Tab") e poi OK nella successiva videata di conferma. RaspberryPi farà il reboot che durerà parecchio tempo, in quanto dovrà riformattare la parte di SDCard prima inutilizzata. Probabilmente tornerà al menu di configurazione, quindi date ancora <Finish> e, finalmente, potrete usare il familiare (Fig. 13):

```
raspberrypi login: pi
password: raspberry
```

In questa videata prendete nota dell'indirizzo IP assegnato a RaspberryPi dal server DHCP della vostra rete; lo trovate in fondo alla riga con il messaggio "My IP address is:", poco sopra la riga di login.

Se volete modificare o consultare qualche voce di configurazione, basta lanciare l'applicazione di configurazione con il comando:

```
sudo raspi-config
```

Di sudo e di come configurare la password di SuperUser (root), abbiamo parlato nel numero 168 di Luglio/Agosto della rivista. Dopo questa semplice procedura di configurazione abbiamo un RaspberryPi ben configurato e adatto a funzionare come sistema

embedded; vi invitiamo a riprovare ad accendere il LED con i comandi che vi abbiamo illustrato nel numero scorso.

Questa attività è molto importante e viene definita *test di non regressione*; il test viene eseguito quando in un progetto si cambia sistema di supporto o vengono eseguite modifiche significative e serve a verificare che la nuova configurazione non influisca negativamente sulle vecchie funzionalità. Noi abbiamo fatto il test e il LED si accende.

## DOVE VOGLIAMO ARRIVARE

Le potenzialità di un sistema embedded sono ben al di sopra di quelle necessarie ad accendere un LED. Nel box di approfondimento dell'articolo scorso, dedicato all'Architettura Linux, abbiamo visto la sua capacità di gestire molti processi e, quindi, molti programmi contemporaneamente, in modo separato o coordinati tra di loro, funzionanti in modo stand-alone od in grado di comunicare con altri sistemi esterni, siano essi altri computer, microcontrollori, sensori o esseri umani. Un vero e proprio, immenso, kit per costruire progetti di ogni dimensione e complessità. Il successo di un progetto di sistema embedded sta in un approccio esaustivo al

problema che si vuole risolvere, ovvero nella sua scomposizione in moduli più piccoli, nella definizione delle caratteristiche di ciascun modulo e nella scelta degli strumenti migliori per realizzarli. Questi strumenti possono essere pacchetti già disponibili nel mondo Linux - da installare e integrare tra di loro - oppure, se proprio non se ne può fare a meno, di programmi scritti da noi per integrare, perfezionare, personalizzare, o realizzare ex novo specifiche funzionalità. In questo caso, utilizzando uno o più strumenti di integrazione, scripting o linguaggi di programmazione.

In **Fig. 14** abbiamo voluto raffigurare questa realtà come un mucchietto di mattoncini dai quali pescheremo di volta in volta quelli necessari alle nostre realizzazioni.

Proviamo ad immaginare un progetto abbastanza complesso che evolva dal nostro LED ad un sistema che possa controllare, oltre al suo GPIO, diversi altri sistemi esterni, archiviandone le informazioni inviate e ricevute in una base di dati e permettendone interrogazione e gestione remota dal mondo Internet con varie modalità.

Partiamo dal nostro LED: vogliamo che non sia più gestito solamente a linea di comando, ma da un processo (programma che gira in continuazione sulla macchina Linux) che lo accenda e lo spenga in conseguenza a eventi esterni (per esempio un pulsante, anch'esso collegato ad un pin GPIO); per questo utilizzeremo un mattoncino "linguaggio di programmazione", per esempio Python.

Ora vogliamo che i dati di stato, acceso/spento e di eventi, siano memorizzati in modo da storicizzare lo stato di tutti gli ingressi e

Nel mondo Linux non esiste una configurazione predefinita di quello che si potrebbe definire come PC standard. In realtà esistono diverse famiglie di distribuzioni basate su dei kernel di riferimento, come Debian, per la nostra RasperryPi, e su una collezione più o meno ricca di applicazioni destinate ad utilizzatori generici o finalizzate a supportare qualche ambiente professionale in particolare, come grafici, amministratori di sistema, progettisti di network, creatori di animazioni, e una infinità di altri.

Ciò non toglie che non si possa arricchire una qualunque distribuzione con quante e quali applicazioni il nostro PC sia in grado di supportare.

Le applicazioni sono disponibili, per la stragrande maggioranza gratuitamente, sotto forma di pacchetti direttamente scaricabili ed installabili.

Quando si parla di **pacchetti** si intende un archivio di file contenente del software (compilato o in forma di sorgenti), pronto per essere installato sulla macchina in modo, solitamente, quasi del tutto automatico.

Se esistono dipendenze tra pacchetti, normalmente vengono risolte in modo automatico, per esempio se una applicazione di gestione audio richiede dei driver suono particolari, la lista di installazione del pacchetto li include e li fa installare prima della applicazione che li richiede. Altre volte, alcune dipendenze devono essere risolte a mano caso per caso installando i diversi pacchetti nella sequenza corretta. La utility solitamente usata per l'installazione, rimozione ed aggiornamento delle applicazioni distribuite in formato pacchetto (in Debian l'estensione è .deb) è l'APT (Advanced Packaging Tool). L'utility mette a disposizione il programma apt-get con diverse opzioni di funzionamento.

apt-get update - aggiorna il "catalogo" di tutti i pacchetti

disponibili per la distribuzione apt-get install *pacchetto* o *lista pacchetti* - scarica, scompatta e installa il o i pacchetti indicati ed in genere tutte le loro dipendenze. Per esempio il comando "sudo apt-get install apache2" installa il server web apache e tutte le sue dipendenze

apt-get remove *pacchetto* - disinstalla il pacchetto dal sistema

apt-get autoremove *pacchetto* - disinstalla le dipendenze non più usate di un pacchetto

apt-get clean - ripulisce i file utilizzati per le installazioni, molto utili con la RasperryPi per liberare spazio. Infine

apt-get upgrade - aggiorna tutta la nostra configurazione aggiornando tutti i pacchetti obsoleti, che non hanno bisogno di nuove dipendenze, mentre se si vuole un aggiornamento più invasivo si può utilizzare

apt-get dist-upgrade

L'utility APT offre anche la funzionalità apt-cache per la ricerca e la verifica dello stato dei pacchetti installati. Gli utilizzi principali sono:

Per trovare pacchetti la cui descrizione contiene *parola*:

apt-cache search *parola*

Per elencare le informazioni dettagliate su un pacchetto:

apt-cache show *pacchetto*

Per elencare i pacchetti da cui dipende un dato pacchetto:

apt-cache depends *pacchetto*

Per presentare informazioni dettagliate sulle versioni disponibili di un pacchetto e le sue dipendenze inverse:

apt-cache showpkg *pacchetto*

le uscite, per esempio per poter eseguire un riavvio della macchina e ripristinare subito gli I/O allo stato che avevano prima dello spegnimento. In un microcontrollore si usa la EEPROM, in un sistema embedded ci si appoggia a un database, solitamente di tipo relazionale come il "mattoncino MySQL".

La presenza di un database ci offre l'opportunità di centralizzare tutte le informazioni gestite dal sistema in modo strutturato e, quindi, ci fornisce il punto di partenza per poterle rendere accessibili e gestirle, ad esempio, on-line via Web. Per realizzare tutto ciò abbiamo bisogno di un server Web, come il "mattoncino Apache".

Con le opportune configurazioni e creando pagine personalizzate ("mattoncino PHP") potremo interrogare lo stato del nostro LED, accenderlo e spegnerlo da una pagina Web. Ad una struttura di questo tipo possiamo ancora aggiungere altre funzionalità: ad esempio la possibilità di gestire con altri processi (programmi) dispositivi esterni come Arduino o FlyPort OpenPICUS, gestendoli con messaggi di tipo WebService, o API a seconda delle specifiche esigenze. Il nostro database potrebbe quindi arricchirsi con

questi dati e visualizzare anch'essi via Web.

Proseguendo nel ragionamento, la nostra pagina Web ci potrebbe permettere di gestire in remoto ed in modo centralizzato tutta una costellazione di dispositivi sparsi sulla rete. Ad esempio, dopo aver imparato a gestire dei WebService (API), potremmo cominciare ad utilizzare servizi disponibili sul Web come i servizi meteo ed integrarli nel nostro sistema che, blocco dopo blocco e funzione dopo funzione, continuerà a crescere. Potremmo anche aggiungere processi di gestione della voce, in modo che alcuni eventi vengano annunciati tramite voce sintetizzata, come utilizzando il "mattoncino espeak".

E, ancora, integrare nel tutto dei servizi via FTP, mentre dal lato Web produciamo statistiche, o forniamo istruzioni sull'utilizzo di un tale sistema con i mattoncini gnuplot, wiki, e molti altri. E poi ... per ora ci sembra che

basti, ma non pensate che sia un'impresa impossibile: un passo alla volta, scomponendo tutto in moduli più piccoli e gestibili, scegliendo, assemblando, personalizzando ed integrando i componenti più opportuni, resterete sorpresi da cosa si può riuscire ad ottenere da un computer grande come un biglietto da visita.

Ovviamente, tutto quello che imparerete con RaspberryPi sarà utilizzabile con pochi adattamenti su altri sistemi Linux embedded. Bando alle ciance e vediamo subito come passare dalla teoria alla pratica: è giunto il momento di utilizzare il nostro RaspberryPi in remoto, come un vero sistema embedded, per poi scrivere il primo programma.

#### **L'UTILIZZO COME SISTEMA EMBEDDED**

In genere i sistemi embedded non possiedono monitor, tastiera e mouse in quanto sono destinati a funzionare dentro una scatola, chiusi in un armadio per apparati. La loro gestione avviene

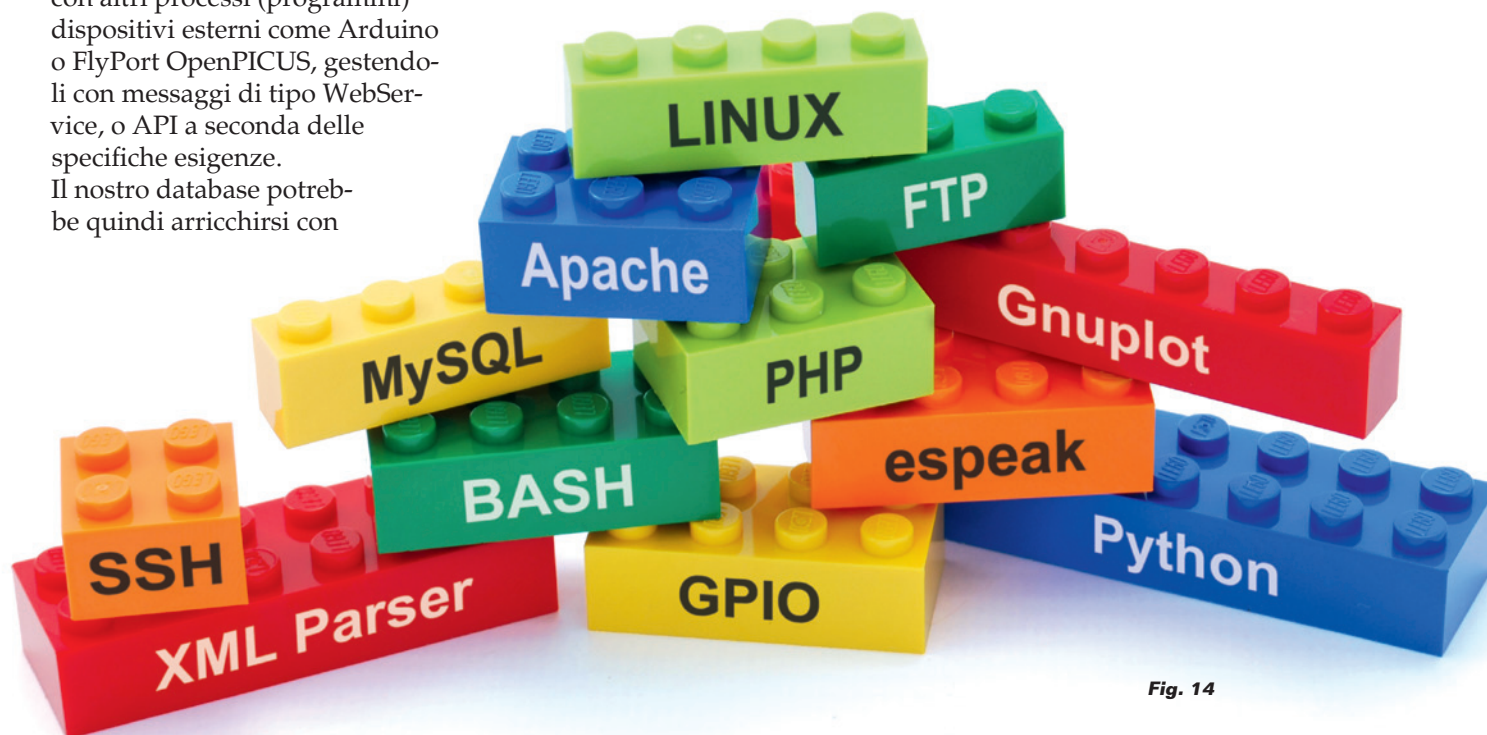


Fig. 14



in remoto, portando all'esterno il terminale a linea di comando (console) ed il gestore dei file (file manager). È anche possibile gestire l'interfaccia grafica in modalità "terminale remoto", giusto per non farsi mancare nulla. Vedremo in seguito quest'ultima modalità. Per ora soffermiamoci sulla console ed il file manager. Metà del cammino è già stato compiuto quando abbiamo attivato il server SSH (Secure SHell) nella configurazione iniziale. Ora dobbiamo predisporre il collegamento ai servizi di tale server sul PC che utilizzeremo per la gestione remota.

Le prossime note riguardano la gestione remota sotto Windows su un normale PC, mentre gli appassionati di Linux non hanno certo bisogno di queste informazioni.

Il PC e la scheda RaspberryPi devono essere collegati alla stessa rete LAN interna.

Ci servono due programmi di interfaccia per PC, uno per la console a linea di comando e l'altro per il file manager.

Per il primo vi raccomandiamo Kitty, una versione migliorata per Windows di PuTTY, il miglior programma di terminale remoto scritto per il mondo Linux. Lo si scarica gratuitamente da <http://kitty.9bis.com/> e non ha bisogno di installazione: è un semplice eseguibile pronto per essere avviato con un doppio clic.

Configuriamolo per la prima volta. All'apertura ci appare la videata riportata in Fig. 15. Iniziamo a configurare una sessione dedicata a RaspberryPi ed a salvarla, in modo da poterla riutilizzare. Per prima cosa digitiamo RaspberryPi (o un altro nome a vostra scelta) nella casella "Saved Session" (Fig. 16), poi premiamo "Save".

Ora inseriamo l'indirizzo IP di

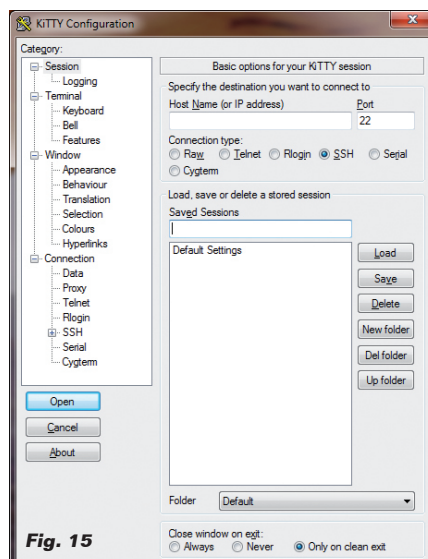


Fig. 15

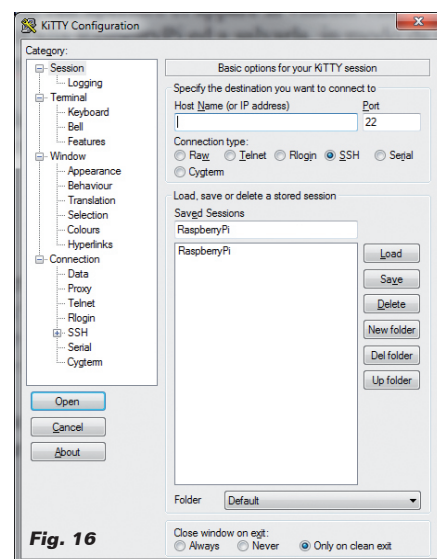


Fig. 16

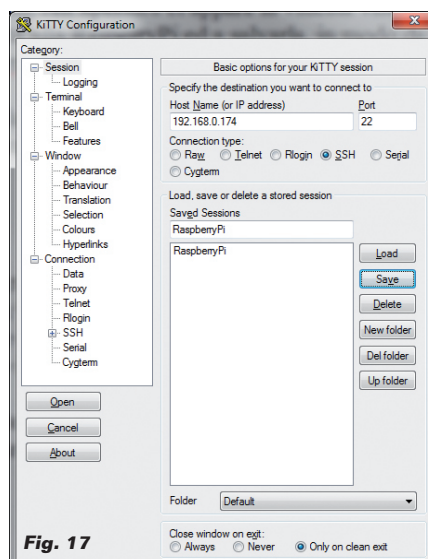


Fig. 17

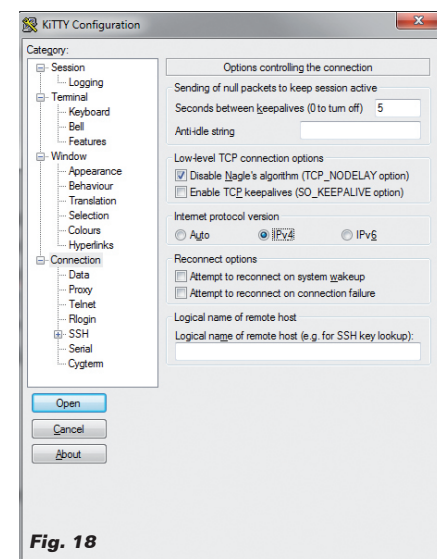


Fig. 18

RaspberryPi nella casella "Host Name (or IP address)" (Fig. 17), l'indirizzo IP dovreste averlo annotato a suo tempo, altrimenti lo potete recuperare con il comando

*ifconfig*

Premete ancora "Save". Ora andata alla sezione "Connection" (Fig. 18) impostate "5" nella casella "Seconds between keep alive" e selezionate il check box "IPv4".

La prima impostazione permette di tenere il collegamento attivo anche se non digitate comandi per diverso tempo, in caso contrario il server SSH, per motivi

di sicurezza, termina le sessioni inattive dopo poco tempo.

La seconda seleziona il protocollo IPv4 come protocollo di comunicazione (IPv4 è il protocollo tradizionale di Internet, quello che usa gli indirizzi ai quali siamo abituati come 74.125.232.103). Premete "Save" e poi "Open". Apparirà una finestra con la richiesta di "login" e a questo punto è come se stessimo usando

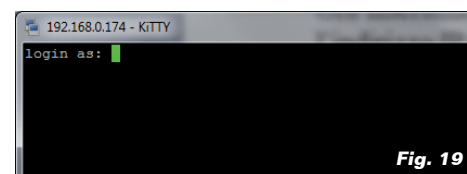


Fig. 19

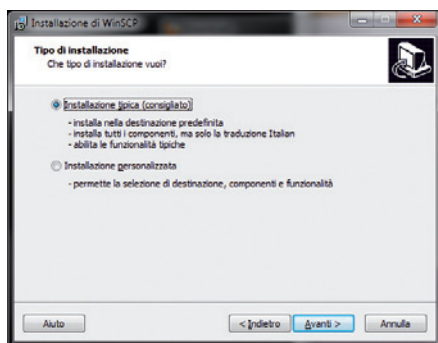


Fig. 20



Fig. 21

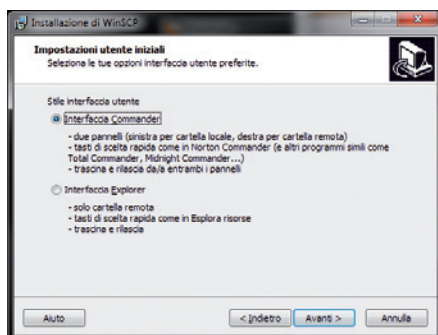


Fig. 22

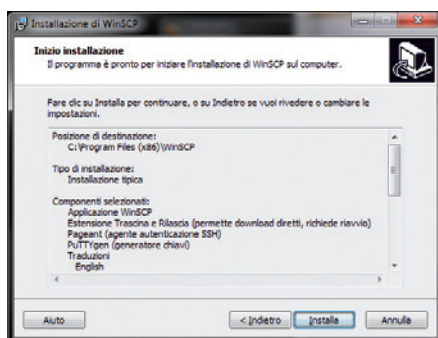


Fig. 23

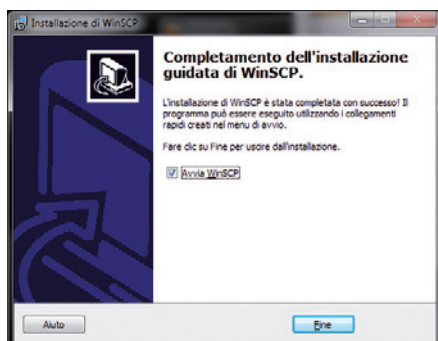


Fig. 24

la RaspberryPi dalla sua tastiera (Fig. 19). Autenticatevi e provate a dare qualche comando o, ancora, ad accendere e spegnere il LED; ricordatevi che l'uso di GPIO richiede i diritti di "root". Ora passiamo al file manager: il programma da scaricare e da installare è "WinSCP". Lo trovate gratuitamente all'indirizzo: <http://winscp.net/eng/download.php>; al momento in cui scriviamo la versione disponibile è WinSCP 4.3.8; voi scaricate l'ultima stabile, evitando le versioni beta. WinSCP può fare tutte le operazioni fondamentali sui file, come il download e l'upload. Consente anche di rinominare file e cartelle, creare nuove cartelle, modificare le proprietà di file e cartelle e creare collegamenti.

Lanciate il setup; dopo le prime videate con la conferma dell'installazione e l'accettazione dei termini della licenza, seguite il processo di installazione con le scelte indicate dalla Fig. 20 alla Fig. 24.

Premendo "Fine" si lancia il programma. Per il futuro vi consigliamo di creare due collegamenti con Kitty e WinSCP sul desktop per facilitarvi il compito. Vi apparirà la videata di Fig. 25 che vi avvisa che non è stata impostata una connessione sicura e crittografata, tra il vostro PC e RaspberryPi; Linux è molto rigido per la sicurezza e, d'altra parte, quelli che stiamo usando sono strumenti di amministrazione. Per ora accontentiamoci e non mettiamo troppa carne al fuoco, dando OK.

Il risultato è la videata di apertura di WinSCP visibile in Fig. 26. Configuriamola come in Fig. 27, con indirizzo IP, utente, password e scegliendo il protocollo "SCP". Infine salviamo le impostazioni con "Salva". Premiamo "Accedi" (Fig. 28) e finalmente il

nostro ambiente di gestione remota è pronto, ed apparirà come in Fig. 29, con le due sessioni aperte e connesse.

In WinSCP a sinistra abbiamo il file system del PC ed a destra quello di RaspberryPi, posizionato sulla cartella personale corrispondente al nostro utente "PI". L'interfaccia familiare dovrebbe togliervi ogni dubbio sul suo utilizzo.

Bene, è ora di pensare a programmare.

## IL PRIMO PROGRAMMA PYTHON

La prima domanda: perché Python? Risponderemo a varie riprese, man mano che procederemo nello sviluppo del nostro progetto.

In seguito, infatti, utilizzeremo diversi linguaggi di programmazione, per darvi la possibilità di conoscerli e confrontarli, mantenendo comunque un occhio di riguardo alla semplicità. Per questo motivo ci siamo orientati ai linguaggi cosiddetti interpretati, cioè che non richiedono una fase di compilazione, ma si scrivono le istruzioni in un file di testo e le si fa eseguire con un opportuno comando.

Tra questi linguaggi troviamo quelli più diffusi ed i più maneggevoli da utilizzare. Tutti, comunque, in grado di essere utilizzati in una gamma infinita di applicazioni. Vi anticipiamo che per la parte Web utilizzeremo il linguaggio PHP, che è anche quello utilizzato per sviluppare Facebook e la maggior parte dei siti web; un ottimo investimento in termini di conoscenza e arricchimento del curriculum. Affronteremo anche la trasformazione in programmi di liste di comandi di console - un po' quello che avviene con i file .bat in Windows - ma molto più potente e di utilizzo diffusissimo.

Infine useremo Python: potentissimo anche se semplice e, soprattutto, già disponibile nel nostro RaspberryPi; tutto questo ci permetterà di avere le prime soddisfazioni nelle vesti di programmatori Linux. Iniziamo con una breve introduzione al linguaggio Python. Per verificare la sua presenza in un computer basta digitare il comando

```
python -V (maiuscolo)
```

Se è presente, l'interprete Python ci risponde con la versione installata (Fig. 30).

Se si digita unicamente `python` si apre l'ambiente di programmazione. Per lanciare un programma si deve usare la forma

```
python nome-file
```

Proviamo, digitiamo `python` al prompt `>>>` digitiamo `print` "Ciao a tutti" e invio ("Hello" è troppo banale).

L'istruzione viene eseguita immediatamente (Fig. 31).

Per chiudere digitate `exit()` con le parentesi.

Passiamo alla modalità con il programma in un file.

Nella sezione di RaspberryPi in WinSCP puntiamo il cursore del

mouse in un'area vuota e col tasto destro selezioniamo "Nuovo" e poi "File" (Fig. 32).

Diamo un nome, ad esempio "Prova", quindi invio; al suo interno scriviamo

```
#!/usr/bin/python
print "Ciao a Tutti"
```

e chiudiamo salvando (Fig. 33 e 34). La prima riga indica all'interfaccia a linea di comando dove trovare il programma al quale consegnare le righe successive come linee di codice da eseguire. Torniamo alla console e digitiamo "python prova" e voilà: benvenuti nella programmazione Python e Linux in generale (Fig. 35).

Ora torniamo ai nostri LED, scrivendo un piccolo programma Python in grado di gestire il GPIO; un primo piccolo ostacolo da superare è la necessità di reperire la libreria per gestire il GPIO, che non è inclusa nell'installazione base.

Non sappiamo il perchè di questa mancanza, ma la trasformiamo nell'occasione per

Fig. 25

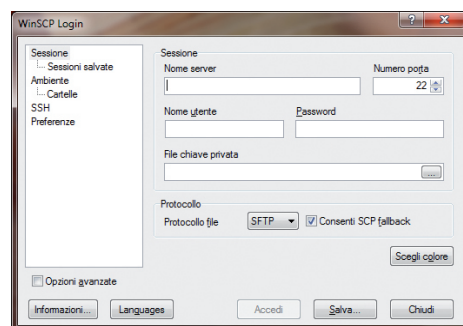
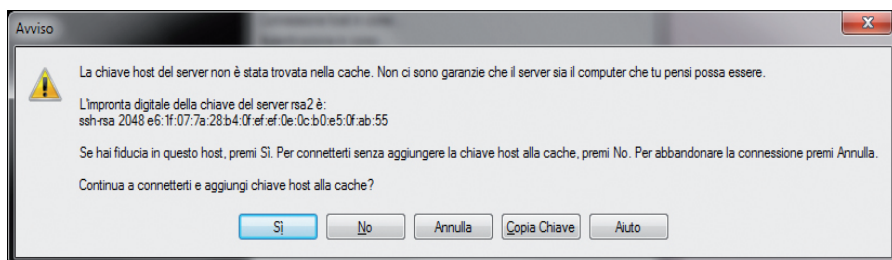


Fig. 26

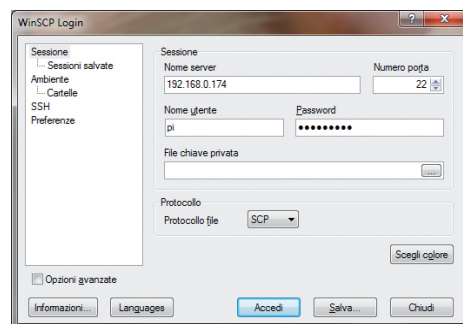


Fig. 27

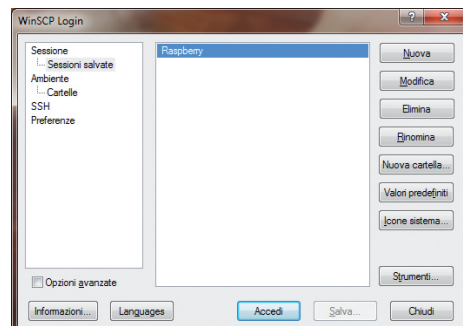


Fig. 28

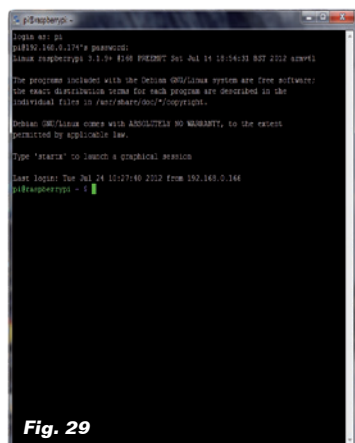


Fig. 29

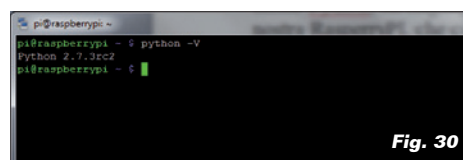
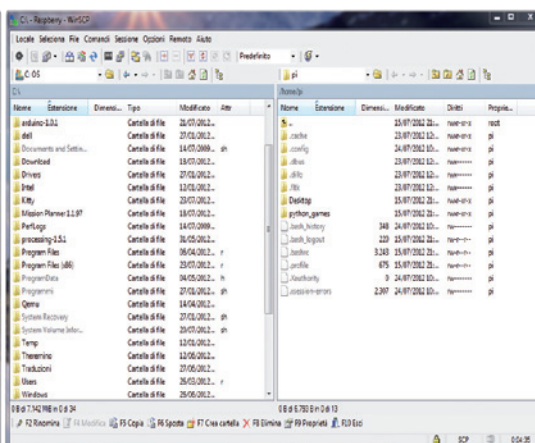


Fig. 30

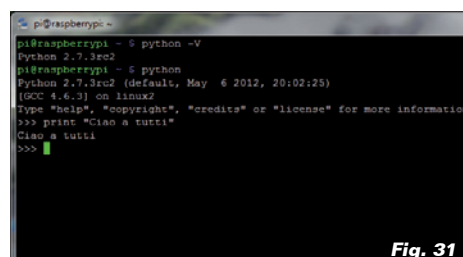


Fig. 31



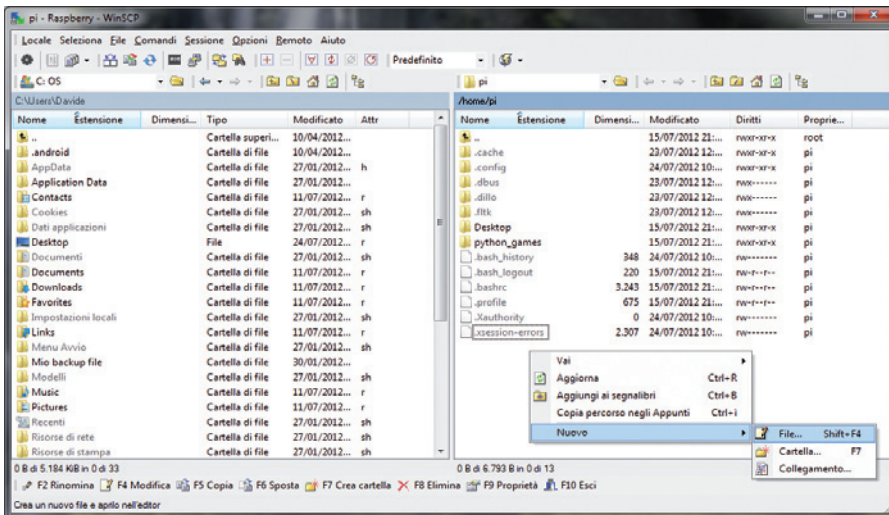


Fig. 32

suo interno. Questo si ottiene con il comando (Fig. 37):  
tar -xvf RPi.GPIO-0.2.0.tar

Che significa Estrai tutti i file (-x) mostrando a console tutti i messaggi di log (v) il contenuto del file (f) in una cartella con lo stesso nome del file (senza il suffisso .tar).

La cartella che otteniamo RPi.GPIO-0.2.0 contiene la nostra libreria in formato sorgente, che dovrà ora essere installata per farla riconoscere a Python. Di fatto l'installazione consiste nella ricompilazione di vari moduli di Linux e Python e nel loro riposizionamento nelle cartelle opportune. Con il comando cd (Change Directory, come in Windows) ci si posiziona nella directory specificata.

```
pi@raspberrypi ~ $ cd
RPi.GPIO-0.2.0
```

all'interno della quale si può digitare il comando di installazione vero e proprio (Fig. 38).

```
pi@raspberrypi ~/RPi.GPIO-0.2.0 $ sudo python
setup.py install
```

Se tutto è andato bene, ovvero se non riceviamo segnalazioni

imparare a installare una libreria in un linguaggio. La possiamo scaricare, sempre gratuitamente, all'indirizzo: <http://pypi.python.org/pypi/RPi.GPIO> dove ne esistono diverse versioni in continuo aggiornamento.

Dovete scaricare la versione RPi.GPIO-0.2.0.tar.gz in quanto abbiamo provato la versione RPi.GPIO-0.3.1a.tar.gz, ma al momento non funziona. Selezionate il link prescelto e, nella pagina seguente, selezionate "Downloads" e infine il file "RPi.GPIO-0.2.0.tar.gz".

Salvate il file dove volete sul PC, poi andate in WinSCP, posizionatevi nella cartella dove avete salvato il file e spostatelo (drag and drop) nella cartella di RaspberryPi (Fig. 36).

Vedremo molto spesso file di questo tipo nel mondo Linux.

Di che cosa si tratta?

Il suffisso "gz" è l'equivalente del formato .zip in Windows: un formato compresso per diminuire lo spazio del file su disco e velocizzare i trasferimenti.

La prima operazione, quindi, è scompattare il file per portarlo al suo formato e alla dimensione originale. Si utilizza il comando:

```
gunzip RPi.GPIO-0.2.0.tar.gz
```

che ci restituisce un file con suffisso .tar. I file in formato .tar sono file di antica tradizione informatica che permettevano di raggruppare un insieme di file distinti in uno unico, per esempio per archivarli su nastro (tar sta per tape archive). Spacchettando un file .tar si ottiene una cartella dello stesso nome con tutti i file separati al

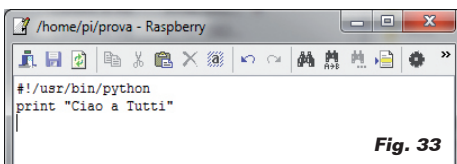


Fig. 33

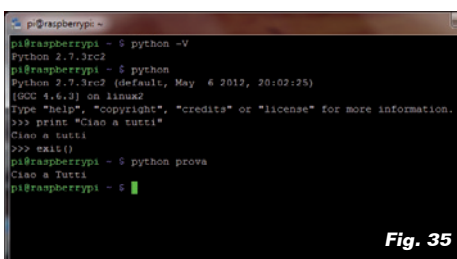


Fig. 35

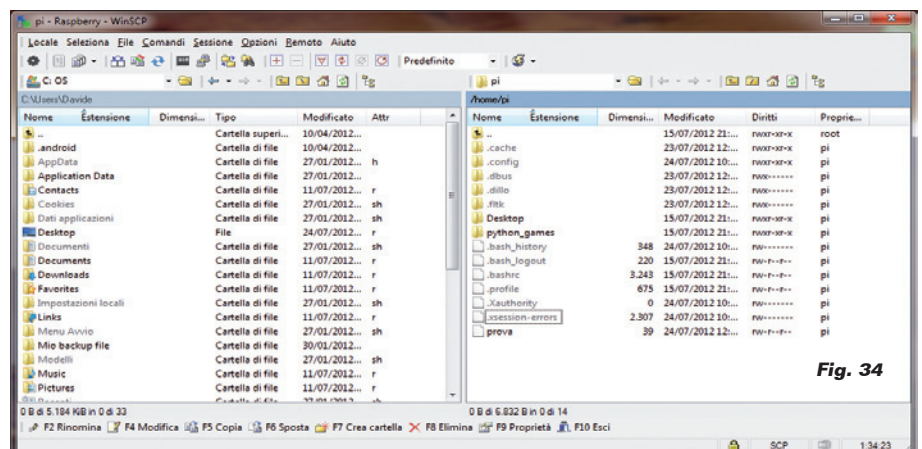
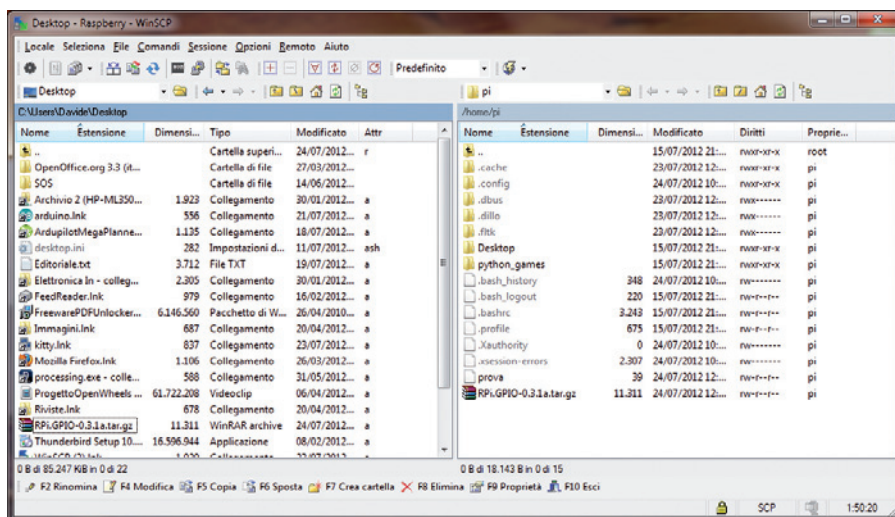


Fig. 34

Fig. 36



di errori nell'elenco di comandi che scorre sul terminale, siamo a posto. Apriamo il file "prova" e scriviamo

```
#!/usr/bin/python
```

```
import RPi.GPIO as GPIO
GPIO.setup (7, GPIO.OUT)
GPIO.output (7, True)
```

Notiamo alcune cose.

L'istruzione *import* serve a includere la libreria *RPi.GPIO* nel programma ed a dargli un nome di riferimento che verrà utilizzato nel resto del programma.

La *GPIO.setup* inizializza il device e ne imposta la direzione (di fatto è la somma dei comandi di export e di impostazione della direzione che abbiamo utilizzato da riga di comando).

Infine l'istruzione *GPIO.output* imposta il livello di segnale sul piedino desiderato.

Qui possiamo notare un fenomeno che capita spesso e fa impazzire chi deve sviluppare applicazioni.

Nell'interfaccia a comando, chi ha creato i comandi ha deciso di utilizzare i nomi dei pin per potervi accedere; il nostro LED in quel caso era riconosciuto come device virtuale GPIO4.

Chi ha scritto la libreria *GPIO* per Python ha invece deciso di utilizzare come riferimenti la posizione fisica del piedino che, per il nostro LED, corrisponde al 7. Non c'è rimedio per queste differenze e sta a noi ricordare le convenzioni di ciascun linguaggio. Chiudiamo il file salvandolo e lanciamolo con

```
sudo python prova
```

ricordatevi che il *GPIO* richiede privilegi di root per essere gestito, quindi utilizzate *sudo*. Acceso il LED?

per spegnerlo modificate l'ultima riga del programma in

```
GPIO.output (7, False)
```

salvate e rilanciate.

Proseguiamo complicando un poco il programma.

Inseriamo un ciclo *while* e facciamo accendere e spegnere il LED a intervalli.

```
#!/usr/bin/python
```

```
import time
import RPi.GPIO as GPIO
GPIO.setup (7, GPIO.OUT)
```

```
while True:
    GPIO.output (7, True)
    time.sleep (2)
    GPIO.output (7, False)
    time.sleep (2)
```

Notiamo ancora alcune cose: abbiamo incluso il richiamo alla libreria *time* per usare la funzione *sleep* in modo da temporizzare il lampeggio del LED, poi nella sintassi del linguaggio PYTHON non ci sono raggruppamenti di istruzioni.

Tutto è dettato dalle indentazioni: niente parentesi graffe o istruzioni *end* ed *endif* e niente ";" alla fine delle istruzioni.

Non consideratelo per questo un linguaggio dalle poche prestazioni: ha delle potenzialità enormi. Adesso tocca a voi: che ne dite

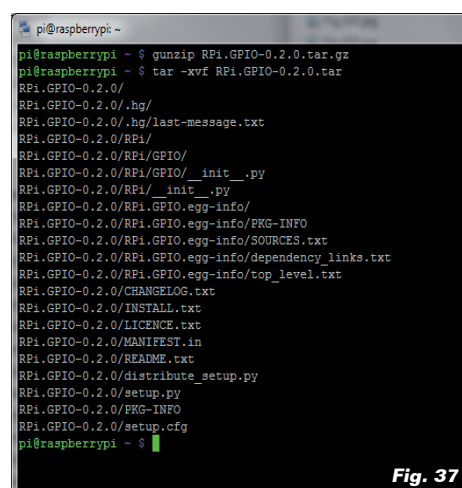


Fig. 37

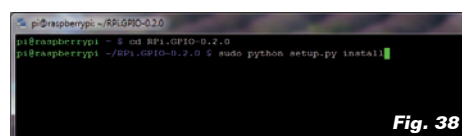


Fig. 38

di realizzare un semaforo con i tre LED che passano dal verde al giallo e al rosso, con la sequenza avviata dalla pressione di un pulsante? In fondo è il semaforo a richiesta dei passaggi pedonali. In più potrebbe restare attivo finché qualcuno impegna il passaggio, rilevandolo ad esempio con un sensore PIR.

Nel prossimo appuntamento prenderemo confidenza con l'installazione e la configurazione di pacchetti e servizi per attivare le funzionalità Web e Database.



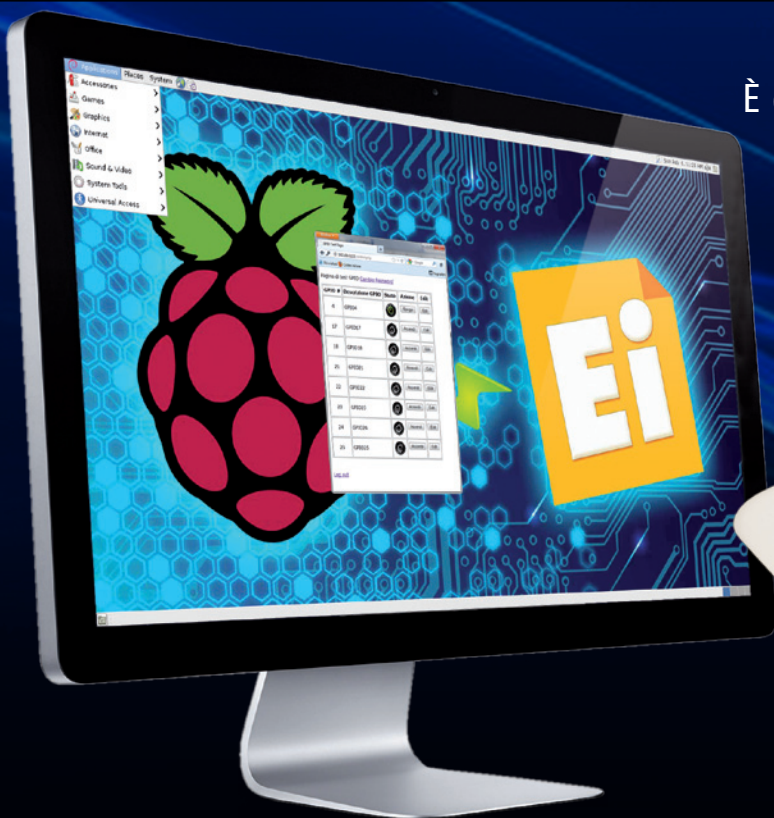
Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

# Elettronica In

[www.elettronica.in.it](http://www.elettronica.in.it)

oltre l'elettronica





È ora di fare il salto di qualità, trasformando la RaspberryPi, in un sistema completo di gestione del GPIO, sia da remoto, che a fronte di eventi. In altre parole, controlliamo il nostro LED dalla rete.



# DOTIAMO *RASPBERRYPI* DI DATABASE E SERVER WEB

di MARCO MAGAGNIN

**N**el numero 168 di Luglio/Agosto, nel box di approfondimento abbiamo descritto brevemente il significato dei pacchetti in Linux e la loro modalità di gestione. Questo articolo può essere visto come una "palestra" per allenarvi sulla tecnica di installazione e gestione dei pacchetti.

Fin dal primo appuntamento ci siamo ripromessi di costruire assieme a voi un'architettura in grado di sfruttare molte delle possibilità del mondo embedded, utilizzando come filo conduttore la gestione di un LED.

Abbiamo visto la possibilità di interfacciare il GPIO dapprima dalla linea di comando e poi con

un piccolo programma in Python; ora è il momento di andare avanti ed estendere questa architettura per gestire il nostro LED via WEB, utilizzando un comune browser su un qualunque PC in Rete. Con questo approccio graduale stiamo costruendo un template, ovvero un progetto di esempio, che contiene tutti i mattoni necessari a realizzare gli obiettivi inizialmente dichiarati e che potrà essere esteso in modo da adattarsi ad una vasta serie di esigenze del mondo embedded reale. L'abbiamo più volte sottolineato, ma ci permettiamo di ripeterlo ancora: tutto ciò che stiamo sperimentando e realizzando non si applica solo a RaspberryPi, ma

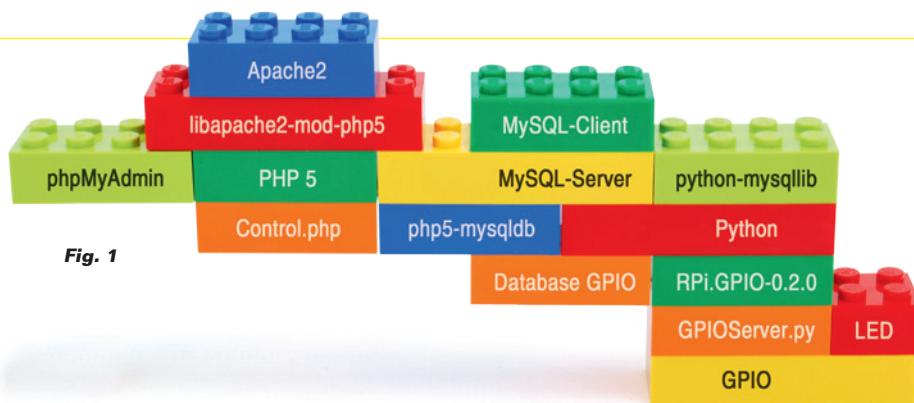


Fig. 1

si può utilizzare su piattaforme Linux embedded differenti, ed in alcuni casi anche su Windows, come vedremo già nel prossimo appuntamento.

In perfetto stile Linux, vediamo di compilare la "lista della spesa" con ciò che ci occorre per realizzare la nostra idea. In un progetto Linux embedded questa è la parte più difficile ed è anche dove si esprimono la bravura e l'esperienza dei progettisti Linux; in questa fase si deve conoscere l'esistenza dei pacchetti Linux che offrono determinate funzionalità, le loro dipendenze e la compatibilità con altri pacchetti che vorremmo includere nella nostra architettura. Per questo progetto abbiamo fatto una scelta - fra le molte possibili - tenendo conto dell'intento didattico di questi articoli e privilegiando questo aspetto rispetto, ad esempio, alle prestazioni. Dedicheremo spazio anche a questo più avanti. Torniamo ora alla "lista della spesa", che passa per quella che viene definita l'analisi del problema. Questo è il percorso che, a partire dalle funzionalità che si vogliono realizzare, porta ad anticipare le problematiche da risolvere, a trovare le soluzioni adatte a tutte le casistiche che si possono verificare ed infine a tracciare il disegno finale che verrà realizzato. Al posto di un'arida serie di schemi, vi proponiamo un approccio discorsivo, un po' come avviene nelle riunioni preliminari di progetto definite con il termine "brainstorming". Innanzitutto, visto che vogliamo avere la gestione del nostro LED

via web ci serve un web-server, cioè un'applicazione che resti costantemente in esecuzione sul nostro sistema in attesa di chiamate dall'esterno e sia in grado di rispondere con pagine web; in queste pagine possono essere contenute richieste di azioni che il nostro web server deve essere in grado di recepire ed eseguire direttamente o passando l'incarico a qualche altra applicazione. Per il web-server abbiamo scelto Apache2: un web server professionale utilizzato da moltissime imprese per siti di grandi dimensioni; imparare a conoscerlo sarà tempo ben speso. Ma gira sulla RaspberryPi? La risposta è sì. Una volta scelto il web server, dobbiamo scegliere qualcosa che ci permetta di scrivere e gestire le pagine web in modo tale da non gestire unicamente una struttura di pagine grafiche HTML, ma che dia la possibilità di intercettare e gestire le richieste di azioni. Si tratta quindi di qualcosa di programmabile, ovvero un linguaggio di programmazione adatto ad essere integrato in una struttura di pagine web. Abbiamo scelto il PHP, un linguaggio di scripting professionale che è particolarmente indicato per lo sviluppo web e può essere incorporato nell'HTML. Il significato iniziale di PHP è "Personal Home Page", poi trasformato in "PHP: Hypertext Preprocessor" (Preprocessore di Ipertesti). PHP è un linguaggio di scripting interpretato, ovvero che non necessita di compilazione, che viene utilizzato principalmente

per sviluppare applicazioni web lato server, ma che può anche essere usato anche per scrivere script a riga di comando o applicazioni stand-alone con interfaccia grafica. Lo scopo di questo linguaggio è di permettere agli sviluppatori web di scrivere velocemente pagine generate dinamicamente ed è utilizzato dai giganti del web come Facebook: sicuramente un'altra buona esperienza da citare nel nostro curriculum. Gira anche PHP su RaspberryPi? Ancora Sì. È difficile da imparare? Secondo noi meno di quanto possa sembrare, ma dipende comunque dalle capacità di ciascuno. Noi proporremo degli esempi che potrete modificare per aggiungere o cambiare delle funzionalità, per non farvi partire da zero.

Ora è necessario ragionare sulle funzionalità che vogliamo implementare e qui incontriamo un primo problema. Ammettiamo di avere realizzato già tutto e di potere fare accendere il nostro LED per mezzo di un pulsante su una pagina web. Chiamiamo la pagina e premiamo l'apposito bottone: il LED si accende e rimane in questo stato fino a quando decideremo di spegnerlo. Giusto? Purtroppo no, o meglio, dipende. Finché RaspberryPi rimane acceso e in funzione, il LED rimane acceso, ma se per un motivo qualsiasi la scheda viene spenta o riavviata, il LED si spegne. Abbiamo infatti visto che i pin del GPIO, alla partenza di RaspberryPi vengono inizializzati come "out", con livello impostato a LOW, lasciando spento il nostro LED. Questo è il problema da risolvere, anche perché noi siamo ancora convinti che sia rimasto

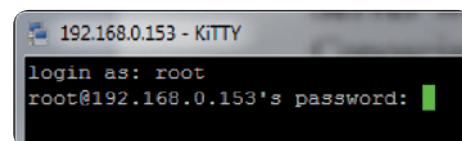


Fig. 2



```

login as: root
root@192.168.0.153's password:
Linux raspberrypi 3.1.9+ #168 PREEMPT Sat Jul 14 18:56:31 BST 2012 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Type 'startx' to launch a graphical session

Last login: Mon Jul 30 14:37:52 2012 from 192.168.0.166
root@raspberrypi:~# apt-get update

```

Fig. 3

Questa soluzione penalizza un po' le prestazioni ma offre diversi vantaggi: permette di centralizzare la gestione del GPIO, includendo anche il trattamento di funzioni collegate ad eventi, come ad esempio la pressione di un pulsante per attivare una sequenza di LED stile semaforo, che non potrebbe essere intercettata da una struttura solamente basata sul web. Accentrando la funzione in un programma permanentemente in esecuzione, invece, potremo attivare l'evento sia da pulsante fisico che da web. Questa soluzione, che disaccoppia i moduli web, il database e il programma di gestione permette infine di rendere l'intera architettura scalabile. Architettura scalabile significa che quando un contenitore hardware non è più in grado di reggere il carico di una applicazione, i moduli che costituiscono l'architettura stessa possono essere distribuiti su hardware dedicati, agendo unicamente su parametri di configurazione. Ad esempio, nel nostro progetto potremmo spostare il database su un altro computer, lasciando il web server ed il programma di gestione del GPIO su RaspberryPi e tutto continuerà a funzionare tranquillamente, come vi dimo-

accesso non avendo informazioni sul fatto che la scheda è stata riavviata. Per mantenerlo acceso dovremmo utilizzare la pagina web di comando ripetutamente nel tempo, metodo molto poco deterministico, oppure ... trovare un'altra soluzione. In un microcontrollore avremmo adottato la soluzione di scrivere lo stato del LED in EEPROM; in caso di reset il programma sarebbe andato a leggere i valori in EEPROM per ripristinarli alla condizione antecedente il reset. In un sistema embedded generalmente non abbiamo una EEPROM; al suo posto potremmo usare uno o più file - ad esempio uno per ogni pin di I/O - da scrivere ogni volta che da una pagina web arriva un comando e da leggere in caso di reboot per ripristinare le condizioni di pre-riavvio. Il problema con i file è l'impossibilità di accesso multiplo in scrittura: se due utilizzatori, ciascuno dalla pagina di comando, volessero modificare più o meno contemporaneamente lo stato del LED, uno dei due causerebbe un errore. Abbiamo bisogno di una soluzione più robusta e in casi come questi la soluzione robusta, tipicamente, passa per l'utilizzo di un database. Anche noi, quindi, utilizzeremo un database. Scegliremo quello professionale e di fatto quasi standard, naturalmente, e cioè MySQL: il database relazionale principe in Linux, utilizzato dalle principali aziende, ricco di funzionalità e dalle prestazioni eccezionali. Gira su RaspberryPi?

La risposta è ancora sì. È compatibile con il web server Apache2? sì. E con il linguaggio PHP? Anche, ma con un'apposita libreria di estensione, ricordiamocene! È difficile da usare? Anche questa volta dipende; secondo noi meno di quello che sembra e lo affronteremo un passo alla volta, con degli esempi già preconfezionati. Chi usa ad esempio Access per Windows si troverà molto avvantaggiato. Adesso dobbiamo immaginare il programma che dovrebbe partire al riavvio per ripristinare lo stato dei pin alla situazione precedente allo spegnimento. Nel nostro caso, visto che vogliamo realizzare un progetto che sia un modello per applicazioni future reali, adotteremo una soluzione più sofisticata, ovvero un vero e proprio programma sempre funzionante che faccia da ponte tra le richieste web e la gestione fisica del GPIO.

```

Hit http://mirrordirector.raspbian.org wheezy/main armhf Packages
Hit http://archive.raspberrypi.org wheezy InRelease
Hit http://mirrordirector.raspbian.org wheezy/contrib armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/non-free armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/rpi armhf Packages
Hit http://archive.raspberrypi.org wheezy/main armhf Packages
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/contrib Translation-en
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/main Translation-en
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/non-free Translation-en
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en_GB
Ign http://mirrordirector.raspbian.org wheezy/rpi Translation-en
Ign http://archive.raspberrypi.org wheezy/main Translation-en_GB
Ign http://archive.raspberrypi.org wheezy/main Translation-en
Reading package lists... Done
root@raspberrypi:~#
root@raspberrypi:~#
root@raspberrypi:~#
root@raspberrypi:~#
root@raspberrypi:~#
root@raspberrypi:~#
root@raspberrypi:~# apt-get install apache2 php5 libapache2-mod-php5

```

Fig. 4



```

root@raspberrypi:~#
root@raspberrypi:~#
root@raspberrypi:~#
root@raspberrypi:~#
root@raspberrypi:~#
root@raspberrypi:~# apt-get install apache2 php5 libapache2-mod-php5
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2-mpm-prefork apache2-utils apache2.2-bin apache2.2-common libapr1
  libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libonig2 libqdbm14
  php5-cli php5-common ssl-cert
Suggested packages:
  apache2-doc apache2-suexec apache2-suexec-custom php-pear openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-mpm-prefork apache2-utils apache2.2-bin apache2.2-common
  libapache2-mod-php5 libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libonig2 libqdbm14 php5 php5-cli php5-common ssl-cert
0 upgraded, 16 newly installed, 0 to remove and 15 not upgraded.
Need to get 4,893 kB/7,743 kB of archives.
After this operation, 21.3 MB of additional disk space will be used.
Do you want to continue [Y/n]? y

```

Fig. 5

```

Enabling module authz user.
Enabling module authz groupfile.
Enabling module authn file.
Enabling module authz host.
Enabling module reqtimeout.
Setting up apache2-mpm-prefork (2.2.22-9) ...
[....] Starting web server: apache2apache2: Could not reliably determine the ser
ver's fully qualified domain name, using 127.0.1.1 for ServerName
. ok
Setting up apache2 (2.2.22-9) ...
Setting up libonig2 (5.9.1-1) ...
Setting up libqdbm14 (1.0.78-2) ...
Setting up libapache2-mod-php5 (5.4.4-2) ...

Creating config file /etc/php5/apache2/php.ini with new version
[....] Restarting web server: apache2apache2: Could not reliably determine the s
erver's fully qualified domain name, using 127.0.1.1 for ServerName
... waiting apache2: Could not reliably determine the server's fully qualified
domain name, using 127.0.1.1 for ServerName
. ok
Setting up php5 (5.4.4-2) ...
Setting up php5-cli (5.4.4-2) ...

Creating config file /etc/php5/cli/php.ini with new version
update-alternatives: using /usr/bin/php5 to provide /usr/bin/php (php) in auto m
ode.
Setting up ssl-cert (1.0.31) ...
root@raspberrypi:~#

```

Fig. 6

sterremo in pratica. Magia? No, solo una combinazione di LINUX, mondo embedded e buona pratica di progettazione.

**INSTALLIAMO I PACCHETTI PRESCELTI**  
Server WEB. Conosciamo già

l'applicazione APT e la sua funzionalità apt-get, ed è ciò che utilizzeremo per la installazione della nostra architettura applicativa. Alla lista precedente, dopo avere verificato le dipendenze tra pacchetti, dobbiamo aggiungere un paio di pacchetti di ricordo.

Il primo crea il ponte tra il server web Apache2 e l'interprete del linguaggio PHP5; si chiama *libapache2-mod-php5*. Per utilizzare il database MySQL con il linguaggio PHP è necessario un secondo pacchetto aggiuntivo, *php5-mysql*, che fornisce un

## Database relazionali

La storia dei database comincia da lontano, da prima degli anni '60, con la ricerca di un modello di rappresentazione dei dati che fosse standard e, per quanto possibile, indipendente dai linguaggi di programmazione. In realtà si assistette alla proliferazione di prodotti proprietari ed eterogenei. Verso la fine degli anni '60 si tentò una standardizzazione tramite il consorzio Codasyl, che in diverse riprese propose un modello di rappresentazione dati al quale aderirono alcuni produttori, più o meno rigorosamente. Edgar F. Codd lavorava alla sede californiana di IBM come ricercatore sulla nascente tecnologia degli hard disk quando osservò l'inefficienza dell'approccio Codasyl con la nuova modalità di memorizzazione dei dati, inefficienza principalmente dovuta all'assenza di una efficace e veloce funzione di ricerca. Nel 1970 cominciò a produrre diversi documenti che schematizzavano un nuovo approccio alla costruzione delle basi di dati, culminati nel "Modello relazionale per Basi di dati condivise" ("A Relational Model of Data for Large Shared Data Banks").

In questo articolo Codd descrisse un nuovo sistema per archiviare e modificare grandi quantità di dati. Invece di utilizzare delle "righe" ("record", in informatica) collegate tra di loro attraverso un qualche tipo di struttura "ad albero", come in Codasyl, ritenne di utilizzare una "tabella" di righe a lunghezza fissa. Questo metodo sarebbe stato molto inefficiente nell'archiviazione di dati "sparsi", in cui la tabella avrebbe potuto avere diverse "celle" vuote; tale errore di impostazione fu corretto suddividendo i dati in diverse tabelle, in cui gli elementi opzionali venivano spostati, anziché sprecare spazio nella tabella principale. Ad oggi i database relazionali sono il tipo di database attualmente

più diffuso. I motivi di questo successo sono fondamentalmente due:

1. forniscono sistemi semplici ed efficienti per rappresentare e manipolare i dati
2. si basano su un modello, quello relazionale, con solide basi teoriche

La struttura fondamentale del modello relazionale è appunto la "relazione", cioè una tabella bidimensionale costituita da righe (tuple) e colonne (attributi). Le relazioni rappresentano le entità che si ritiene essere interessanti nel database. Ogni istanza dell'entità troverà posto in una tupla della relazione, mentre gli attributi della relazione rappresenteranno le proprietà dell'entità. Ad esempio, se nel database si dovranno rappresentare delle persone, si potrà definire una relazione chiamata "Persone", i cui attributi descrivono le caratteristiche delle persone (Figura A). Ciascuna tupla della relazione "Persone" rappresenterà una particolare persona.

Una rappresentazione della definizione di tale relazione potrebbe essere la seguente:

Persone (nome, cognome, data\_nascita, sesso, stato\_civile)

Le tuple in una relazione sono un insieme nel senso matematico del termine, cioè una collezione non ordinata di elementi differenti. Per distinguere una tupla da un'altra si ricorre al concetto di "chiave primaria", cioè ad un insieme di attributi che permettono di identificare univocamente una tupla in una relazione. Naturalmente in una relazione possono esserci più combinazioni di attributi che permettono di identificare univocamente una tupla, ma fra queste ne verrà scelta una sola da uti-

modulo per le connessioni ad un database MySQL direttamente da script PHP.

Alla fine della lunga introduzione e delle ulteriori specifiche appena descritte andremo a configurare una architettura strutturata come quella rappresentata nella Fig. 1, utilizzando la metafora dei mattoncini.

Accendiamo RaspberryPi come di consueto, e colleghiamoci in remoto utilizzando Kitty. Al login entriamo come utente root con relativa password e poi iniziamo l'installazione di tutti i pacchetti che abbiamo elencato con le tre sole righe di comando che vi indichiamo qui di seguito. Diamo prima un comando di refresh del catalogo dei pacchetti (Fig. 2):

```
apt-get update
```

Poi, con un unico comando, installiamo il server web, il linguaggio PHP5 e il modulo di raccordo (Fig. 3):

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

Diamo "Y" alla richiesta di conferma della quantità di spazio che verrà utilizzato (Fig. 4).

Dopo una lunga sequenza di comunicazioni di log, l'installazione termina. Possiamo scorrere il log e, ad esempio, verificare che sono stati installati molti più pacchetti di quelli da noi richiesti, come visibile nel frammento di log della Fig. 6; l'applicazione apt infatti ricerca ed installa automaticamente tutte le dipendenze richieste

Fig. 7

```
root@raspberrypi:~# service apache2 status
Apache2 is running (pid 2918).
root@raspberrypi:~#
```

dai pacchetti principali per poter funzionare.

Alla fine di questo processo il server web è stato installato e potete verificare la cosa con il comando (Fig.7):

```
service apache2 status
```

Ovviamente, visto che Apache viene installato con una pagina HTML di test predefinita, per verificare il buon funzionamento del server potremmo anche aprire un browser e puntare l'indirizzo di RaspberryPi.

Proviamo e otteniamo il risultato visibile nella Fig. 8. Siamo a buon punto. Ora, per i nostri scopi futuri, dobbiamo modificare una

lizzare come chiave primaria. Gli attributi della chiave primaria non possono assumere il valore null (che significa un valore non determinato), in quanto non permetterebbero più di identificare una particolare tupla in una relazione. Questa proprietà delle relazioni e delle loro chiavi primarie va sotto il nome di integrità delle entità.

Spesso per ottenere una chiave primaria "economica", cioè composta da pochi attributi facilmente manipolabili, si introducono uno o più attributi fittizi, che conterranno dei codici identificativi univoci per ogni tupla della relazione, in genere un campo ad incremento automatico. Ogni attributo di una relazione è caratterizzato da un nome e da un dominio. Il dominio indica quali valori possono essere assunti da una colonna della relazione. Spesso un dominio viene definito attraverso la dichiarazione di un tipo per l'attributo (ad esempio dicendo che è una stringa di dieci caratteri), ma è anche possibile definire domini più complessi e precisi. Ad esempio per l'attributo "sesso" della nostra relazione "Persone" possiamo definire un dominio per cui gli unici valori validi sono 'M' e 'F'; oppure per l'attributo "data\_nascita" potremmo definire un dominio per cui vengono considerate valide solo le date di nascita dopo il primo gennaio del 1960, se nel nostro database non è previsto che ci siano persone con data di nascita antecedente a quella. Il DBMS si occuperà di controllare che negli attributi delle relazioni vengano inseriti solo i valori permessi dai loro domini. Caratteristica fondamentale dei domini di un database relazionale è che siano "atomici", cioè che i valori contenuti nelle colonne non possano essere separati in valori di domini più semplici. Più formalmente si dice che non è possibile avere attributi multivalore.

| Nome     | Cognome  | Data_nascita | Sesso | Stato_civile |
|----------|----------|--------------|-------|--------------|
| Emiliano | Pandolfi | 03/06/1973   | M     | Celibe       |
| Alessio  | Borgogni | 29/12/1974   | M     | Coniugato    |
| Claudia  | Ciolfi   | 12/08/1989   | F     | Nubile       |

Ad esempio, se una caratteristica delle persone nel nostro database fosse anche quella di avere uno o più figli, non sarebbe possibile scrivere la relazione Persone nel seguente modo: Persone (nome, cognome, data\_nascita, sesso, stato\_civile, figli), in quanto non sarebbe possibile determinare a priori quanti "campi" riservare ai figli, oltre a perdere la caratteristica della atomicità degli attributi. In pratica non sono ammessi array nelle tuple.

Ciascuna relazione trova corrispondenza fisica in una "tabella" di database, alla quale può essere associati tutta una serie di elementi accessori, come indici di ricerca e collegamenti tra tabelle, per mezzo delle chiavi, per ottimizzarne l'integrità e l'efficienza di accesso ai dati. Per integrità si intende la capacità della base di dati di contenere unicamente dati formalmente corretti e congruenti con tutti gli altri dati contenuti nelle diverse tabelle. L'insieme di tutte le tabelle, gli elementi accessori, i privilegi di accesso degli utenti ed ora anche moduli software di stretta gestione dei dati va sotto il nome di database. Per la creazione, l'amministrazione, l'accesso ed in genere l'utilizzo dei database relazionali è comunemente usato il linguaggio SQL (Structured Query Language) disponibile sia come linguaggio di scripting da terminale, sia integrato in pressoché tutti i linguaggi di programmazione che permette la tanto ricercata indipendenza delle strutture dati dai linguaggi di programmazione.



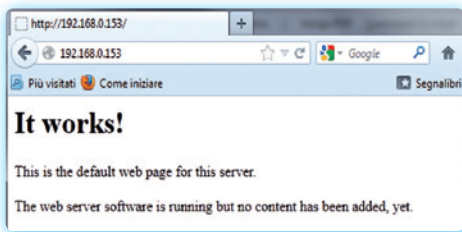


Fig. 8

indicato fino al file 000-default (Fig. 9) ed apriamolo con un click del mouse. Modifichiamo la riga contrassegnata in blu nella Fig. 10 cambiandone il contenuto da:

Riavviamo il server Apache 2 con il comando

```
service apache2 restart
```

Ora passiamo ad installare il database MySQL sia server che client e il modulo di connessione con il linguaggio PHP, il tutto con il comando (Fig. 12):

```
apt-get install mysql-server mysql-client php5-mysql
```

Ed anche qui rispondiamo “y” alla richiesta di conferma dell’installazione con relativa comunicazione dello spazio che verrà occupato (Fig. 13). Dopo la solita ricerca e recupero delle dipendenze ed un tempo

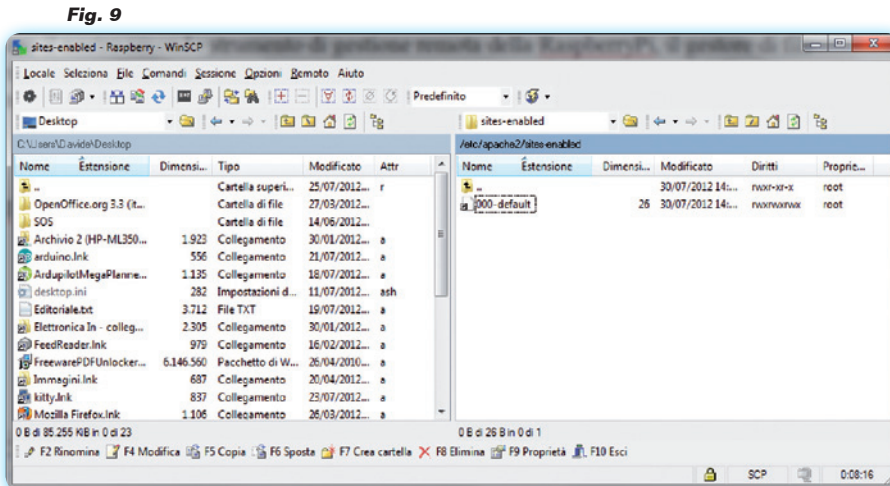


Fig. 9

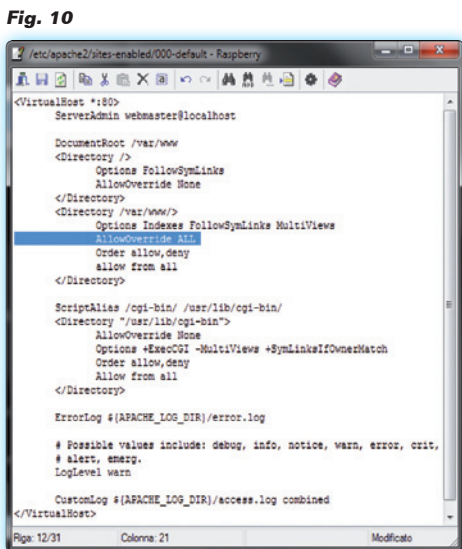


Fig. 10

Fig. 11

```
root@raspberrypi:~# service apache2 restart
[....] Restarting web server: apache2apache2: Could not reliably determine the
server's fully qualified domain name, using 127.0.1.1 for ServerName
... waiting apache2: Could not reliably determine the server's fully qualified
domain name, using 127.0.1.1 for ServerName
. ok
root@raspberrypi:~#
```

Fig. 12

```
root@raspberrypi:~# apt-get install mysql-server mysql-client php5-mysql
```

Fig. 13

```
root@raspberrypi:~# apt-get install mysql-server mysql-client php5-mysql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  heirloom-mailx libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysql
  libnet-daemon-perl liblwp-perl mysql-client-5.5 mysql-common mysql-server-5.5 mys
Suggested packages:
  exim4 mail-transport-agent libipc-sharedcache-perl libterm-readkey-perl tinyc
Recommended packages:
  mailx
The following NEW packages will be installed:
  heirloom-mailx libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysql
  libnet-daemon-perl liblwp-perl mysql-client mysql-client-5.5 mysql-common mysql-s
  mysql-server-core-5.5 php5-mysql
0 upgraded, 16 newly installed, 0 to remove and 15 not upgraded.
Need to get 29.1 MB of archives.
After this operation, 95.2 MB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

AllowOverride None

a:

AllowOverride ALL

Salviamo il file e chiudiamo.

abbastanza lungo, inizia l’installazione vera e propria che si interromperà ad un certo punto con la richiesta della password di amministrazione per l’ambiente di database in installazione (Fig. 14).

configurazione, lo facciamo subito, così non dovremo ricordarcene più avanti. Dobbiamo editare e modificare il file `/etc/apache2/sites-enabled/000-default` e per farlo usiamo il nostro secondo strumento di gestione remota di RaspberryPi, il gestore di file WinScp. Apriamolo e collegiamoci anche qui come utente “root” e relativa password, i file di configurazione di Apache e degli altri pacchetti richiedono i privilegi di root. Seguiamo il percorso

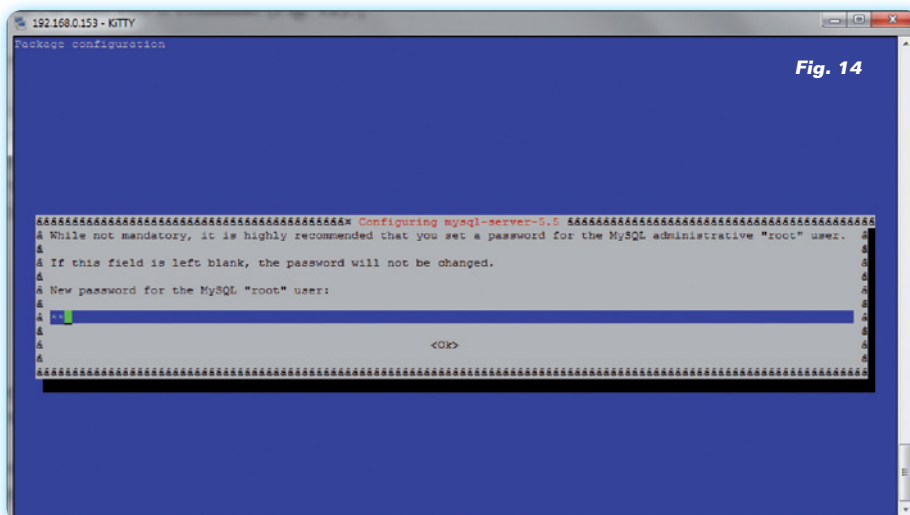


Fig. 14

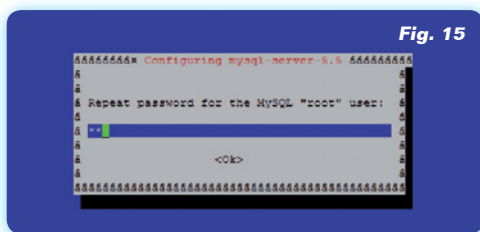


Fig. 15

Impostiamo la password, selezioniamo "OK" col tasto "Tab" e diamo "Invio". Alla richiesta di riconferma della password la digiteremo nuovamente, ci posizioneremo su "OK" e daremo invio (Fig. 15). Non verremo più disturbati fino al termine dell'installazione. Anche qui possiamo sincerarci del buon esito dell'installazione con il comando (Fig. 16):

```
service mysql status
```

se otteniamo una pagina come quella della Fig. 16 significa che il server database è attivo e funzionante e che noi, passo dopo passo stiamo trasformando RaspberryPi in una vera e propria installazione

server aziendale, conosciuta anche con il nome di LAMPP dalle iniziali di Linux Apache MySQL PHP Perl; in effetti ci mancherebbe il Perl, altro linguaggio di programmazione, ma per ora possiamo accontentarci.

Per amministrare, gestire e, nel nostro caso, anche imparare a conoscere il mondo dei database con una interfaccia grafica adatta allo scopo, aggiungiamo alla nostra lista anche il pacchetto phpMyAdmin, un add on in PHP, scritto appositamente per facilitare l'amministrazione e la manutenzione del gestore di database MySQL. Anche in questo caso l'installazione è molto semplice: scarichiamo il pacchetto, salvandolo per esempio nel desktop di Windows, dal sito <http://www.phpmyadmin.net>.

In WinSCP andiamo alla cartella /var/www di RaspberryPi e creiamo una nuova cartella (Fig. 17) "phpmyadmin", facendo attenzione a scrivere tutto minuscolo (Fig. 18).

Adesso apriamo il file compresso che abbiamo appena scaricato, entriamo nella cartella principale "phpMyAdmin-3.5.2-all-languages", selezioniamo tutti i contenuti e spostiamoli nella cartella di "phpmyadmin" di RaspberryPi (Fig. 20).

Il risultato è visibile nella Fig. 21). Per verificare immediatamente il funzionamento di PhpMyAdmin, apriamo il browser e digitiamo l'Url:

```
<indirizzo-IP-della-RaspberryPi>/  
phpmyadmin
```

Se otteniamo la videata visibile in Fig. 22 siamo a un buon punto; inseriamo l'utente root e la password che abbiamo scelto in fase di installazione di MySQL e premiamo "Esegui" (Fig. 23)

Dopo qualche secondo, se otteniamo la videata visibile in Fig. 24 abbiamo la conferma che tutto il processo di installazione eseguito fino a questo punto è andato a buon fine, e che la nostra scheda RaspberryPi non ha niente da invidiare ai servizi di un vero provider, prestazioni a parte, ovviamente.

Fin qui, abbiamo costruito l'ossatura del nostro progetto installando e configurando in modo opportuno i pacchetti standard disponibili per l'ambiente Linux. Ora dobbiamo personalizzare la struttura realizzata con l'applicazione che ci permetterà di accendere il nostro LED da remoto, via WEB. Per questa applicazione abbiamo modificato, personalizzando, i componenti del progetto Open Source "raspberrypi-gpio" che, se volete, potete scaricare dal sito: <https://code.google.com/p/raspberrypi-gpio/downloads/list>. I codici di esempio che abbiamo modificato li trovate come sempre all'indirizzo della rivista [www.elettronica.in](http://www.elettronica.in).

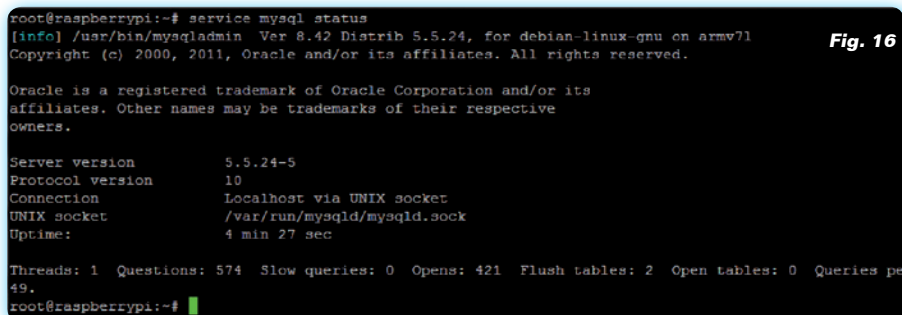


Fig. 16



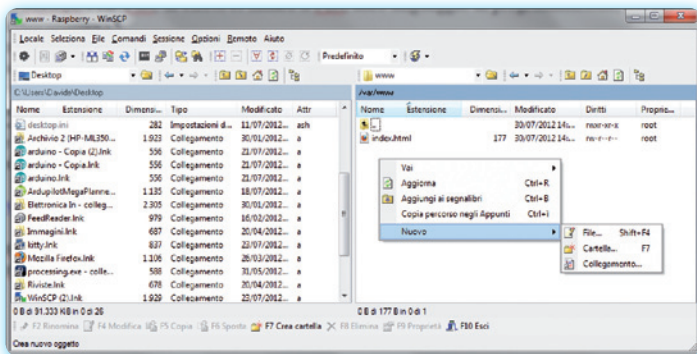


Fig. 17

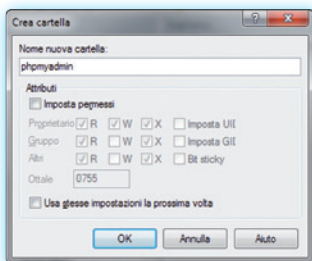


Fig. 18

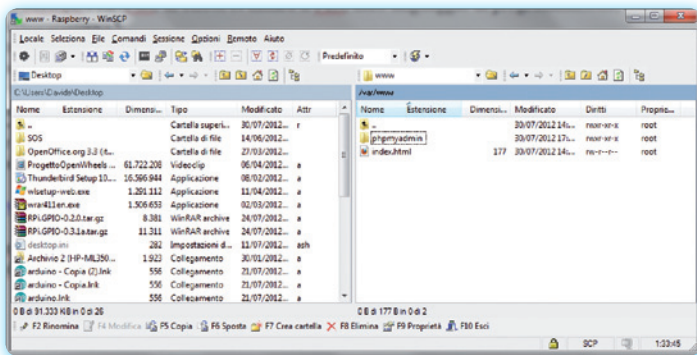


Fig. 19

Fig. 20

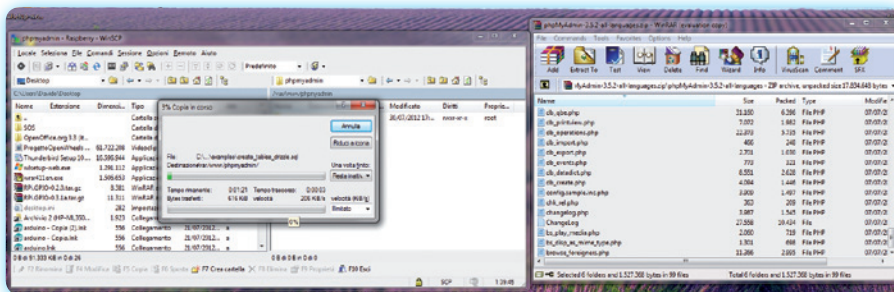
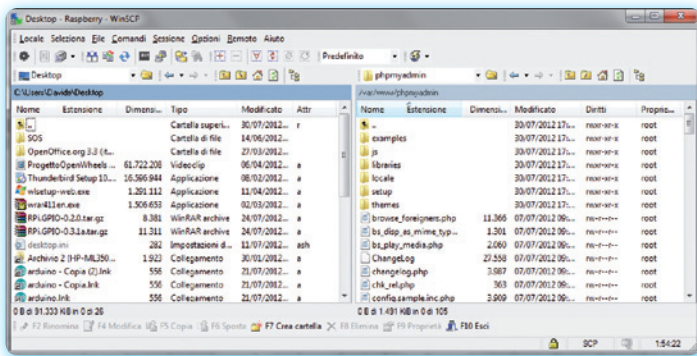


Fig. 21



Il progetto si compone di tre moduli:

- Il primo "gpio.sql" è un file di testo che contiene le istruzioni per generare il database di supporto in ambiente MySQL, con tutte le tabelle ed i dati necessari per la nostra applicazione. Dovrà essere utilizzato una volta sola in fase di configurazione per generare il database;
- Il secondo "control.php" è sempre un file di testo che contiene le istruzioni in linguaggio PHP per gestire le pagine WEB della nostra applicazione con tutte le azioni previste che dettigheremo in seguito; è il programma che verrà richiamato all'interno del server Web quando digiteremo l'Url della nostra applicazione;
- Il terzo programma del progetto originale è GPIOServer.sh, il programma sempre funzionante in memoria che mantiene la corrispondenza tra lo stato elettrico dei pin del GPIO e la loro rappresentazione nelle tabelle del database. Questo programma è uno "Shell script", ovvero un file di testo che contiene un elenco di comandi come quelli che si utilizzano a linea di comando. Nel nostro caso abbiamo preferito sostituirlo con un programma Python - più significativo dal punto di vista didattico - che ci farà da base per tutti gli esempi che proporremo in futuro e che potrà essere "portato" anche su piattaforme Linux embedded diverse da RaspberryPi. Cosa che speriamo di proporvi molto presto;

- Gli altri due file che trovate sia nel sito originale che nel nostro sono due immagini Jpeg che indicano i pin in stato ON e OFF e che verranno visualizzate nelle pagine Web della nostra applicazione.

Con le nostre modifiche abbiamo comunque fatto in modo che GPIOServer.sh possa continuare

a funzionare nella nostra applicazione, così avete la possibilità di sperimentare le diverse versioni. Scaricate e scompattate il file con i moduli della nostra applicazione in una cartella sul vostro PC. Cominciamo con il creare il database per la nostra applicazione in ambiente MySQL. Torniamo al browser, che abbiamo lasciato con aperta la videata visibile in Fig. 24 e selezioniamo la funzione "Importa" dalla fascia orizzontale in alto della finestra di MySQL (Fig. 25). Selezioniamo "Sfoglia" e nella finestra di dialogo ci portiamo nella cartella con i file scaricati, quindi



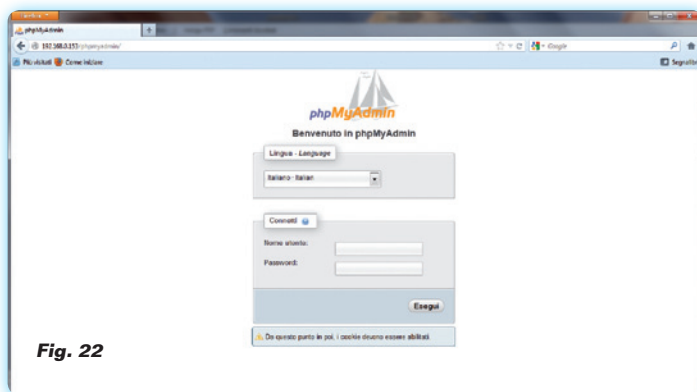


Fig. 22

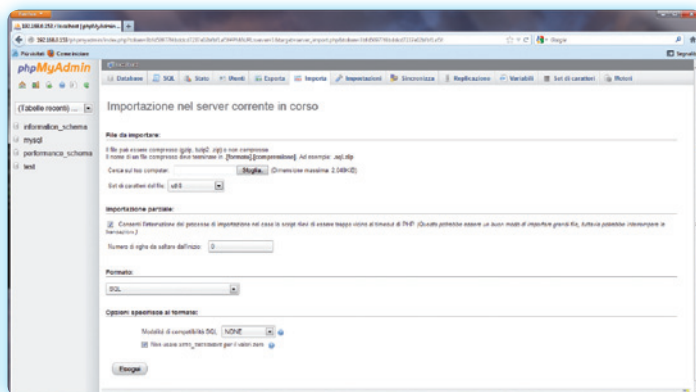


Fig. 25

Fig. 23



scegliamo "gpio.sql" (Fig. 26) e poi "Apri". Premiamo il tasto "Esegui" per fare eseguire il file di istruzioni SQL che creerà il database, tutte le tabelle e le popolerà con i dati che ci serviranno di esempio (Fig. 27).

Se tutto ha funzionato correttamente ci troveremo come risultato il messaggio di Fig. 28 che ci conferma che il nostro database "gpio" è stato creato correttamente. Per poter eseguire la connessione e poter utilizzare il database "gpio" da parte di programmi applicativi è necessario definire uno o più utenti che abbiano i privilegi necessari per eseguire tutte le operazioni richieste dalle applicazioni.

È buona pratica per la sicurezza e per evitare il rischio di operazioni indesiderate su database differenti, definire un utente specifico per ciascun database, cosa che faremo anche nel nostro caso. Selezioniamo la funzione "Utenti" - sempre dalla fascia orizzontale - e nella scheda che si apre (Fig. 29) scegliamo, in basso "Aggiungi utente" e compiliamo la videata che appare come nella Fig. 30: come Nome utente abbiamo usato "gpio", come Host "localhost" e password "pi", da inserire

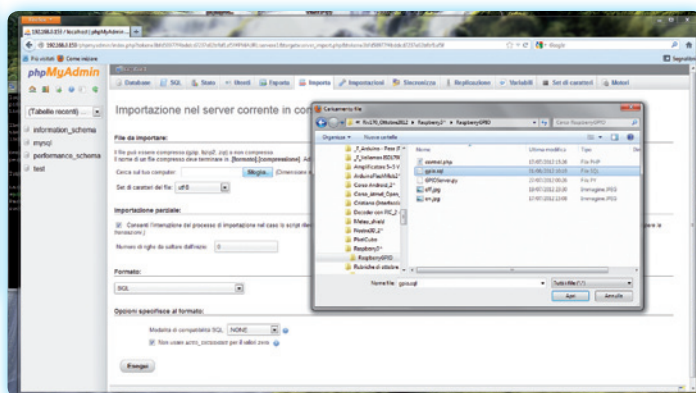


Fig. 26

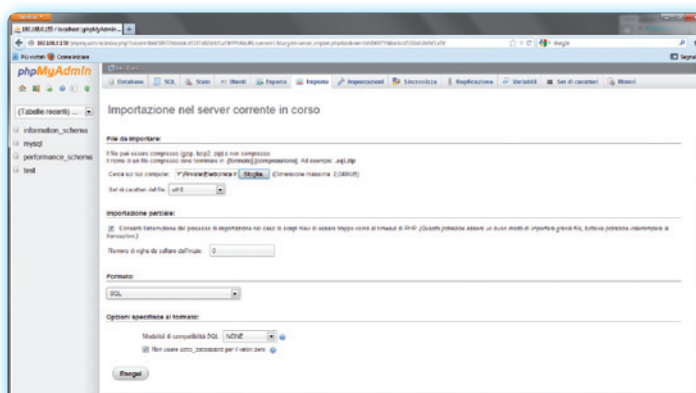


Fig. 27

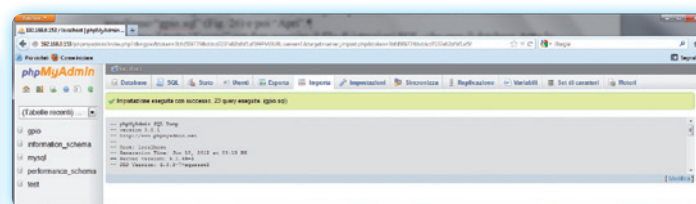


Fig. 28

due volte. Confermiamo il tutto premendo il tasto in basso "Aggiungi utente". Ci si ripresenterà la pagina con l'elenco degli utenti, questa volta con l'utente "gpio" che è entrato nella lista (Fig. 31). Ora dobbiamo dargli i privilegi sul nostro databa-

Fig. 24

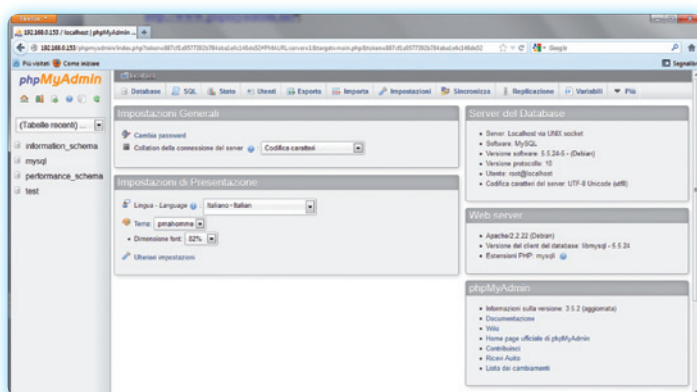


Fig. 29

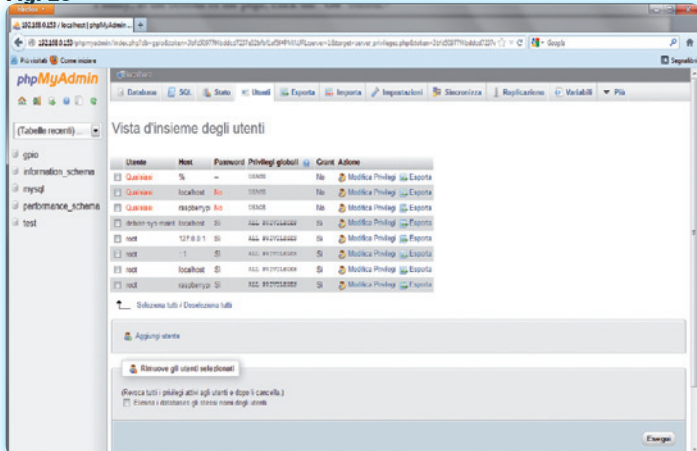


Fig. 32

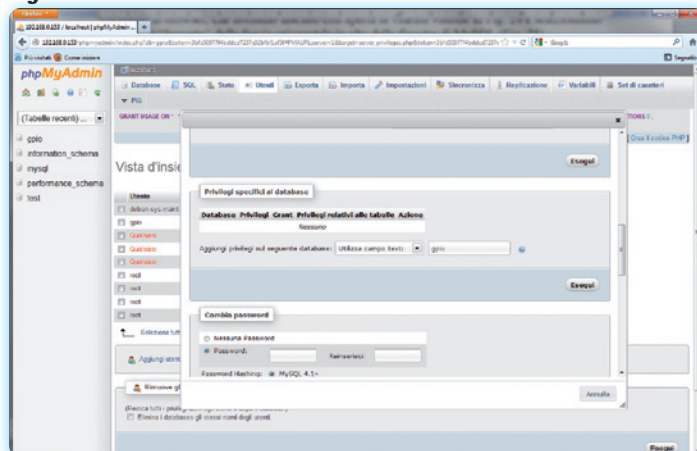


Fig. 30

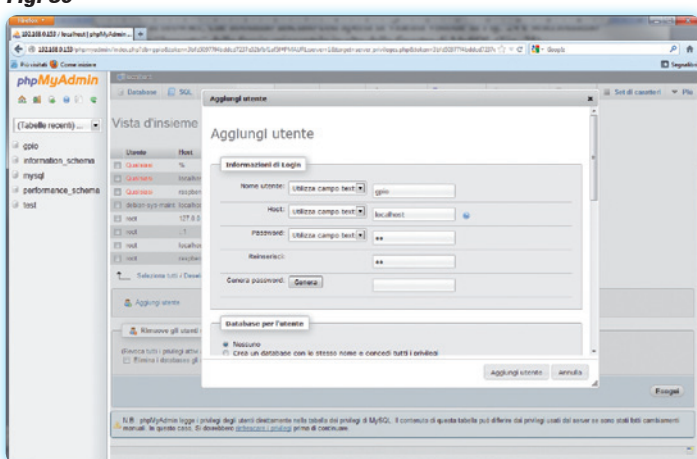


Fig. 33

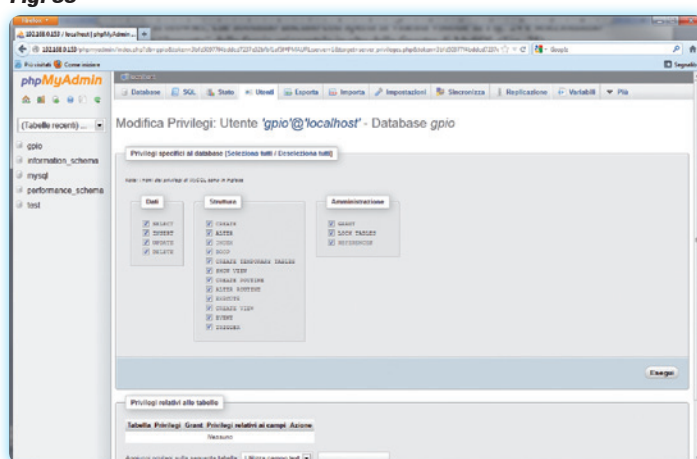


Fig. 31

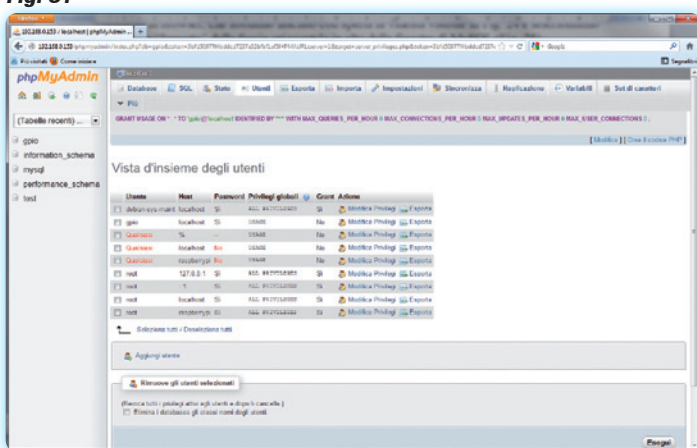
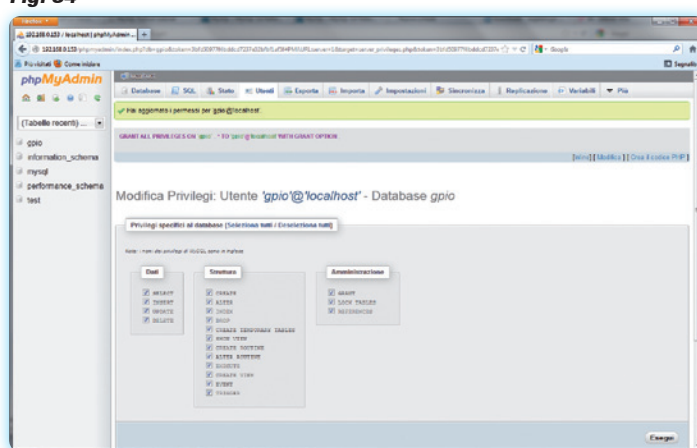


Fig. 34



se. Sulla riga in corrispondenza dell'utente "gpio" selezioniamo "Modifica Privilegi" e alla pagina successiva, nella sezione "Privilegi specifici al database", nel campo "Aggiungi privilegi sul seguente database" inseriamo "gpio" come in Fig. 32. Premiamo il pulsante "Esegui" nella

stessa sezione (attenzione!) e nella pagina successiva (Fig. 33) selezioniamo tutti i privilegi con "Seleziona tutti", lasciamo invariato tutto il resto e premiamo ancora una volta "Esegui", nella stessa sezione. Otterremo la pagina visibile in Fig. 34, a conferma di tutto il nostro

lavoro. Complimenti per essere arrivati fino a questo punto: avete creato e amministrato un database MySQL completo. Per uscire (logout) da phpMyAdmin, dovete selezionare l'icona in alto a sinistra di fianco alla casetta, che rappresenta una porta con una freccia verde di uscita,

Fig. 35

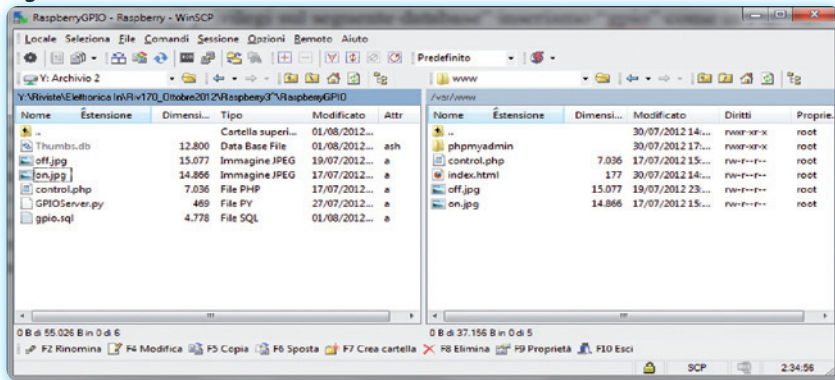


Fig. 36

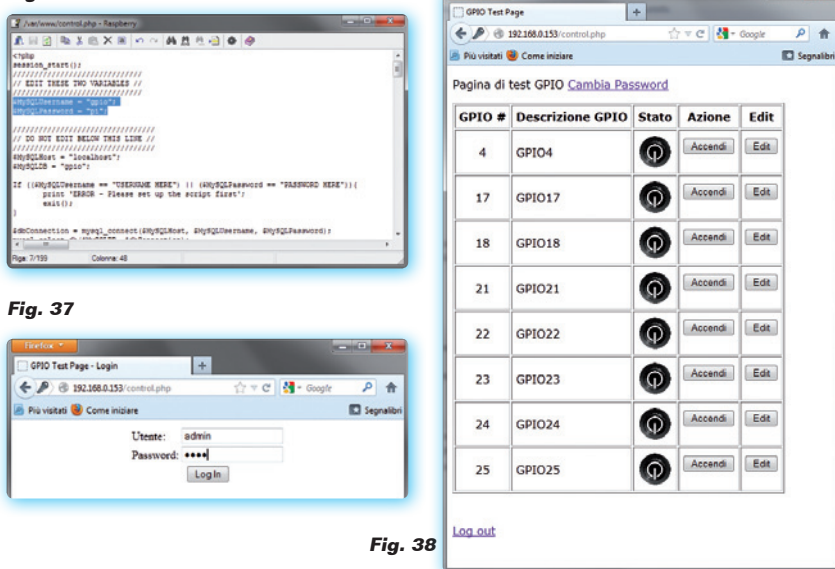


Fig. 37

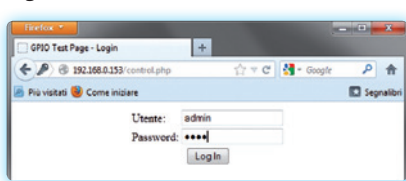


Fig. 38

root@raspberrypi:/home# apt-get install python-mysqldb

Fig. 39

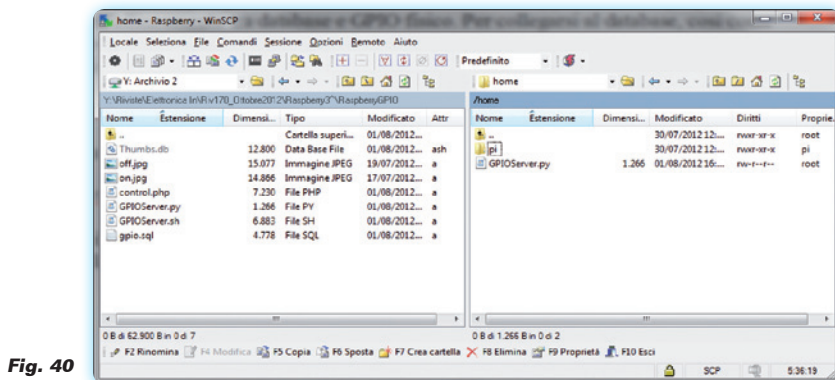


Fig. 40

Disconnetti, appunto. è l'unica via di uscita, non propriamente semplice da trovare. Ora passiamo alle impostazioni del Web server e spostiamo il programma PHP di gestione delle nostre pagine nella apposita cartella di RaspberryPi. La cartella è /var/www, la stessa

dove abbiamo installato precedentemente phpMyAdmin. Sempre da WinSCP, controllando di essere connessi con l'utente root, spostiamo il file control.php e le due immagini jpg nella cartella www, come in Fig. 35. Ora apriamo il file control.php con un click e modifichiamo le due ri-

ghe in blu della Fig. 36, per configurare l'utente e la password che dovranno essere utilizzati dal programma per accedere al database "gpio", inseriremo utente e password che abbiamo appena configurato.

```
$MySQLUsername = "gpio";
$MySQLPassword = "pi";
```

Salviamo e chiudiamo. Possiamo già fare un primo test delle pagine di gestione del GPIO aprendo il browser e digitando l'indirizzo

<indirizzo-IP-della-RaspberryPi>/control.php

Ci apparirà la prima pagina della nostra applicazione con la richiesta di un Utente e una Password, digitiamo rispettivamente "admin" e "gpio" come in Fig. 37 e poi premiamo "Login", otterremo la pagina visibile in Fig. 38. Chiaramente se utilizziamo la funzione accendi sul nostro LED non succederà nulla perché manca ancora l'ultimo mattoncino alla nostra struttura.

Ora vediamo di aggiungere quest'ultimo mattoncino: il programma Python di gestione dell'allineamento tra database e GPIO fisico. Per collegarsi al database, così come abbiamo fatto nel numero scorso per il collegamento al GPIO, Python richiede un modulo aggiuntivo, di raccordo tra Python e MySQL. Il suo nome è

python-mysqldb

ed essendo un pacchetto autoinstallante utilizzeremo il comando consueto (Fig. 39):

```
apt-get install python-mysqldb
```

ora portiamo il programma Python di raccordo tra database e hardware GPIO dalla cartella



```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import time
import RPi.GPIO as GPIO
import MySQLdb as mdb
import sys

DBServer = "localhost"
DBUser = "gpio"
DBPassword = "pi"
GPIOPinFoe = 99
GPIOPinStatus = 0
GPIOPinDirection = "out"

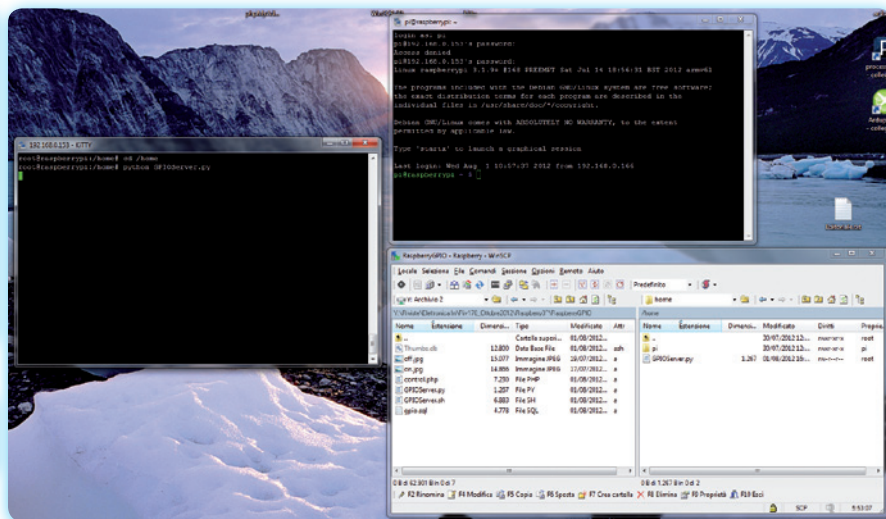
while True:
    con = mdb.connect(DBServer, DBUser, DBPassword, 'gpio')
    cur = cur.execute('SELECT * FROM gpio')
```

Fig. 41

```
root@raspberrypi:/home# cd /home
root@raspberrypi:/home# python GPIOServer.py
```

Fig. 42

Fig. 43



nel nostro PC nella cartella di RaspberryPi /home/, come visibile in Fig. 40, apriamolo e modifichiamo i due campi in blu nella Fig. 35, per configurare correttamente l'utente e la password necessari a connettersi al database "gpio", quindi salviamo e chiudiamo.

Questo tipo di programmi, che vengono lanciati e fatti funzionare permanentemente in memoria, richiedono una "shell" dedicata: in pratica richiedono una finestra terminale a loro dedicata. Possiamo tranquillamente usare la nostra finestra di terminale

sul PC, assicuriamoci di esserci autenticati come utente "root", perché l'accesso al GPIO richiede questi privilegi, quindi posizioniamoci nella cartella "home" con l'eventuale comando

*cd /home*

e diamo il comando (Fig. 41):

*python GPIOServer.py*

Il programma GPIOServer.py entrerà in esecuzione tenendo la finestra di terminale impegnata; in questa finestra non sarà più possibile impartire comandi finché lo stesso è in esecuzione. Se vogliamo, possiamo comunque aprire ulteriori finestre come in Fig. 42 per eseguire altre operazioni, in concomitanza con i programmi già in esecuzione.

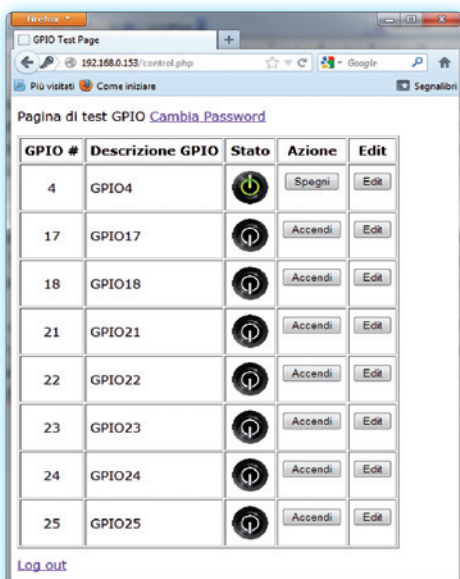
Questo a riprova, se ne fosse necessario, che Linux è un vero sistema operativo multiprocesso e che in Linux, "tutto è un file". In questo momento, a parte i processi di servizio del sistema operativo, su RaspberryPi stanno "girando" il server SSH con tre sessioni aperte, quelle sul terminale del PC, il server web Apache, il server database MySQL, e il nostro programma

GPIOServer.py. Scusate se è poco, e c'è spazio per altro ancora. Proviamo a verificare il funzionamento di tutta l'architettura. Torniamo al browser con la pagina di test GPIO della Fig. 38 e premiamo il pulsante "Accendi" in corrispondenza della riga "GPIO4". Dovremmo trovarci con la pagina di risposta visibile in Fig. 44 e con il LED sul GPIO di RaspberryPi acceso. Il pulsante cambia di stato e diventa "Spegni" e l'immagine cambia a rappresentare una condizione di "Acceso".

Per interrompere l'esecuzione di quello che ormai possiamo chiamare il nostro "Server GPIO" basta posizionarsi sulla finestra terminale e premere contemporaneamente il tasto "Ctrl" e "c", per tornare al prompt della linea di comando.

Nei prossimi appuntamenti, oltre a qualche accattivante estensione all'architettura che abbiamo realizzato, analizzeremo più in dettaglio i listati dei programmi proposti, dando le informazioni che vi permetteranno di comprenderli e modificarli a vostro piacimento. Inoltre approfondiremo anche le modalità di gestione diretta del database, utilizzando l'applicazione phpMyAdmin.

Fig. 44



Dopo avere reso possibile il controllo via web degli ingressi e delle uscite digitali facciamo in modo che RaspberryPi notifichi in locale, utilizzando messaggi in viva voce, se qualcuno invia dei comandi da remoto.



# DIAMO VOCE A RASPBERRYPI

di MARCO MAGAGNIN

**D**opo una lunga maturazione nell'ambiente esclusivo di sistemisti, hacker e professionisti informatici, il sistema operativo Linux (GNU-Linux per la precisione) sta vivendo un momento di crescente popolarità tra gli utilizzatori comuni, fino ad ora pressoché totalmente orientati ad utilizzare i sistemi operativi di Microsoft ed Apple. I primi segnali di questo fenomeno si sono intravisti quando alcuni utilizzatori istituzionali come amministrazioni statali e locali, soprattutto all'estero, ma anche aziende grandi e meno grandi hanno deciso di eleggere il sistema operativo Linux, e più

in generale il software open source, come l'architettura strategica di riferimento per il supporto alle proprie esigenze gestionali, informative ed informatiche. La prima motivazione di questa scelta è dettata dalla concreta esigenza di protezione degli investimenti in tecnologie informatiche delle istituzioni. E questa è maggiormente garantita se tutto il codice che costituisce il patrimonio informatico è disponibile in formato sorgente e quindi ispezionabile, riutilizzabile, modificabile e personalizzabile. Caratteristiche che sicuramente non appartengono ai sistemi "proprietary" e che di conseguenza sono



# eSpeak

eSpeak è un software di sintesi vocale che permette di generare, a partire da stringhe di testo multilingua, sia messaggi vocali sulla uscita audio del computer che di registrare lo stesso tipo di output su file di tipo audio.

La sintassi di base del comando è:

```
espeak [opzioni] [<testo>]
```

| Opzioni                         |                                                                                                                                                                                                            |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -h                              | Mostra a terminale un help con l'elenco delle opzioni disponibili.                                                                                                                                         |
| -f<file di testo>               | Permette di specificare il nome del file di testo che contiene il messaggio da sintetizzare                                                                                                                |
| --stdin                         | Legge il messaggio da sintetizzare dallo stdin invece che da un file, ad esempio per concatenare direttamente l'uscita in formato testo di un programma all'ingresso del sintetizzatore vocale             |
| -q                              | Modalità silenziosa, non produce alcun suono sull'uscita audio, può essere utile con il parametro -x                                                                                                       |
| -a <intero>                     | Livello audio in uscita da 0 to 20, il valore predefinito è 10                                                                                                                                             |
| -p <intero>                     | Aggiustamento dell'intonazione della voce da 0 to 99, predefinito 50                                                                                                                                       |
| -s <intero>                     | Velocità della dizione in numero di parole pronunciate al minuto, il valore predefinito è 160                                                                                                              |
| -v <file vocale di riferimento> | Il parametro richiede di utilizzare uno specifico file di impostazione vocale, per esempio it. La lista delle "voci" disponibili si ottiene col comando <code>espeak --voices</code>                       |
| -m                              | Specifica che il testo da sintetizzare contiene dei tag come SSML, XML o HTML come nel caso si vogliano leggere automaticamente pagine web. Il sintetizzatore adatta la dizione in base ai tag incontrati. |
| -w <nome del file wave>         | Scriva il risultato della sintesi su un file audio, al posto della pronuncia diretta.                                                                                                                      |
| -x                              | Indirizza verso stdout i fonemi mnemonici relativi al testo da sintetizzare                                                                                                                                |
| --stdout                        | Indirizza verso stdout il messaggio sintetizzato                                                                                                                                                           |
| -punct="<caratteri>"            | I caratteri di punteggiatura specificati vengono pronunciati per nome es: "virgola". Se non viene specificato il parametro =<caratteri> tutti i caratteri di punteggiatura vengono pronunciati per nome    |
| -k <valore>                     | Specifica il modo di pronunciare le lettere maiuscole 1=pronuncia normale, 2=con la parola "Maiuscolo", valori maggiori di 2 con un incremento del valore di intonazione (provare -k20).                   |
| --voices[=<codice lingua>]      | Produce una lista di tutti i codici voce disponibili. Se è presente il parametro =<codice lingua> vengono presentati solo i codici voci disponibili per la lingua specificata.                             |

legati all'esistenza della azienda produttrice e contemporaneamente legano l'utilizzatore verso determinati fornitori. Per semplificare il concetto di protezione dell'investimento prendiamo il caso di un impianto industriale come una raffineria, costruita con centinaia di componenti meccanici, idraulici, termici, di lattoneria etc. Tutti i disegni e la documentazione sono in possesso della azienda, così che, quando si manifesta un guasto oppure in occasione della manutenzione periodica, operatori qualificati, non necessariamente

appartenenti all'impresa che ha costruito l'impianto, possono intervenire riparando, sostituendo e, se necessario, addirittura ricostruendo parte dell'impianto. Ipotizziamo ora che questo impianto sia gestito da un sistema automatizzato di controllo di processo e quindi basato su una architettura hardware e software. Se questo software è proprietario e l'azienda, a fronte di una modifica all'impianto, ha necessità anche di modificare parti del software proprietario, l'unica possibilità a sua disposizione è quella di rivolgersi al produttore

originario, sperando che sia ancora in attività e possenga ancora le competenze e la conoscenza del sistema da modificare. Se queste condizioni non si dovessero verificare, la nostra azienda si troverebbe ad affrontare un problema che potrebbe comprometterne l'efficienza, la sicurezza, la competitività sul mercato e quindi, nelle conseguenze più estreme, la sua stessa sopravvivenza. Nel caso di utilizzo di sistemi open source, invece, si troverebbe, come nel caso del resto dell'impianto, possedendo tutti gli schemi, la documentazione ed il codice sorgente di tutto il proprio patrimonio software e quindi nella condizione di poter contare su una vasta comunità di professionisti indipendenti per manutenzioni e modifiche. Tutta questa premessa, rafforzata anche dall'entrata in vigore in Italia della legge n. 134 datata 7 agosto 2012 che impone alla Pubblica Amministrazione di privilegiare le soluzioni basate su open source nelle proprie scelte di architetture, sistemi e applicazioni software, per invitarvi a sperimentare ed approfondire l'ambiente operativo Linux e le applicazioni disponibili in questo ambito, oltre che nella configurazione embedded che caratterizza RaspberryPi, anche come sistema operativo per il vostro PC personale. Se inizialmente non volete dedicare un PC allo scopo, potete partire utilizzando una distribuzione *live*, ad esempio Debian, la stessa della RaspberryPi, su chiavetta USB, dalla quale è possibile fare il boot direttamente e utilizzare Linux senza toccare un solo bit della vostra installazione originale. In questo modo potete utilizzare e configurare Linux, i pacchetti applicativi già disponibili ed installarne di nuovi. Se



invece volete dedicare da subito una parte del vostro PC a sperimentare Linux potete orientarvi verso la distribuzione Ubuntu, una distribuzione destinata a far conoscere ed apprezzare il mondo "Linux" agli utenti comuni, anche per porsi come possibile alternativa in occasione dell'uscita delle nuove versioni dei sistemi operativi di Microsoft e Apple. Anche sul fronte RaspberryPi si susseguono le novità, rappresentate dalla disponibilità di nuove distribuzioni del sistema operativo. Citiamo un ulteriore importante upgrade a Raspbian, che permette di utilizzare appieno le potenzialità hardware della RaspberryPi, sfruttando anche in modo automatico l'overclocking del processore quando necessario; vedremo in seguito quanto è semplice l'upgrade a questa versione. Le altre due distribuzioni che stanno giungendo a maturazione sono la Raspbmc, una personalizzazione di Raspbian dedicata a chi vuole impiegare la RaspberryPi come gestore del proprio media center domestico, ed una distribuzione messa a punto da Adafruit, molto specialistica e destinata agli sviluppatori di sistemi embedded, che include tutti i driver ed i moduli necessari a gestire il GPIO ed il colloquio con dispositivi esterni che supportano i bus SPI, I2C, one wire e seriale. Entrambe le distribuzioni sono basate su Raspbian ed in particolare la seconda è totalmente compatibile con le realizzazioni che vi stiamo proponendo in questi articoli.

## AGGIORNIAMO RASPBERRYPI

Tornando su un terreno più concreto, riaccendiamo Raspberry e vediamo di espandere le sue funzionalità dandogli la possibilità di parlare. Adotteremo la funzionalità text-to-speech, che

è la capacità di sintetizzare una stringa di testo in un messaggio vocale generato dall'altoparlante collegato all'uscita audio. Useremo questa funzionalità aggiuntiva per arricchire l'applicazione che ci permette di attivare gli ingressi e le uscite del GPIO da remoto dando la possibilità di avvisare in viva voce chi è in prossimità della scheda, allorché qualcuno modifichi da remoto lo stato dei pin del GPIO.

Prima di cominciare ci teniamo a ricordare RaspberryPi è un prodotto in continua evoluzione, sia dal punto di vista hardware che dello sviluppo software a corredo. Dal punto di vista hardware la scheda nella sua versione B, quella con tutti i connettori esterni già montati sulla scheda stessa, viene fornita con 512 Mb di memoria RAM al posto di 256 Mb. Dal punto di vista del sistema operativo sono stati ottimizzati ancora di più i moduli del kernel per sfruttare appieno le caratteristiche hardware dei processori e della memoria ed è stata introdotta una nuova funzionalità di gestione dell'overclocking, personalizzabile dall'utilizzatore ed ottimizzata per evitare danni alla scheda. È decisamente conveniente aggiornare il sistema operativo all'ultima versione, anche per chi possiede la versione da 256 Mb di memoria. Per chi ha ancora la versione Debian Squeeze è necessario scaricare la nuova immagine e seguire le istruzioni descritte nei numeri precedenti. Per chi invece ha già installato una qualsiasi versione di Raspbian è sufficiente dare i soliti comandi dal prompt della finestra terminale:

```
sudo apt-get clean
```

per liberare lo spazio occupato sulla SD-Card da eventuali aggiornamenti precedenti

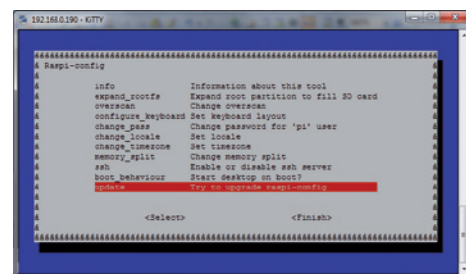


Fig. 1

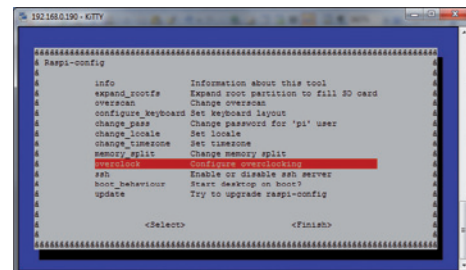


Fig. 2

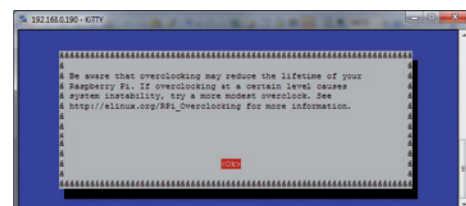


Fig. 3

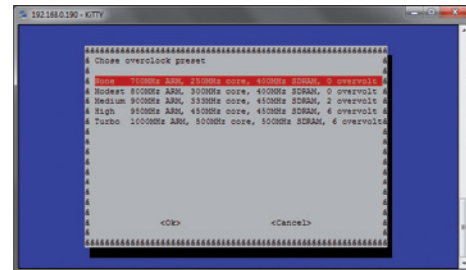


Fig. 4

```
sudo apt-get update
```

per aggiornare l'elenco dei pacchetti con le versioni correnti ed infine

```
sudo apt-get upgrade
```

per eseguire l'aggiornamento vero e proprio; questa operazione dura qualche decina di minuti. Terminato l'upgrade richiamiamo il tool di configurazione con il comando

```
raspi-config
```

selezioniamo la voce Update

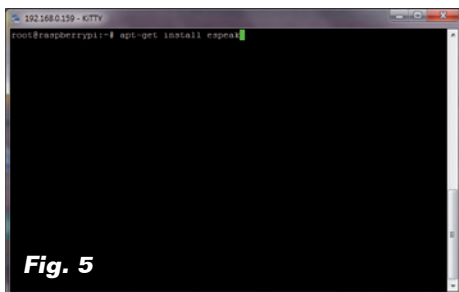


Fig. 5

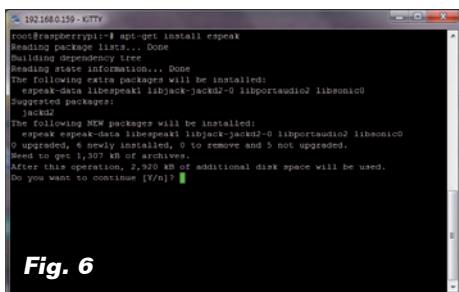


Fig. 6

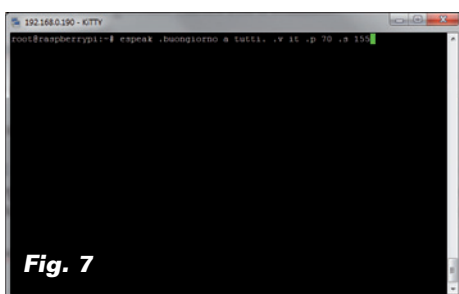


Fig. 7

(Fig. 1) e diamo invio, per aggiornare il tool di configurazione stesso (Fig. 2).

Al termine dell'aggiornamento vedremo il menu delle configurazioni arricchito con la scelta *overclock*, se la selezioniamo verremo prima allertati con il messaggio di avvertimento della Fig. 3 e poi potremo visualizzare le opzioni di overclocking possibili, con le quali sperimentare diverse configurazioni (Fig. 4).

Le modalità di overclocking sono state implementate in modo da essere attivate solo quando le applicazioni in esecuzione lo richiedono, per migliorare le prestazioni, ed anche in questo caso solo se la CPU non è surriscaldata o comunque non rischia di danneggiarsi.

Dopo aver aggiornato il sistema operativo ed il tool di configurazione, installiamo il software

necessario a produrre i messaggi vocali sintetizzati a partire da stringhe di testo.

Anche in questo caso si tratta di scegliere tra i pacchetti software disponibili nei repository della distribuzione Raspian e di installarli con i comandi dell'ambiente *apt-get*.

Abbiamo scelto il pacchetto *eSpeak* in quanto è il più semplice da installare e integrare nella nostra applicazione, oltre ad essere il meno pesante dal punto di vista delle risorse; questo al prezzo di una qualità di dizione non eccellente, ma comunque accettabile per i nostri scopi. Il pacchetto *eSpeak* è in grado di sintetizzare in forma vocale frasi da testi scritti in differenti lingue adattando la dizione alla pronuncia delle diverse lingue. Se si desidera una qualità vocale elevata, è possibile integrare il sintetizzatore *eSpeak* con il pacchetto *mbrola*, che permette di estendere la gamma di voci producibili, sia in termini di lingue che di tonalità, come maschile o femminile. Ora installiamo il nostro pacchetto *eSpeak* con i comandi:

```
apt-get update
apt-get install espeak (Fig. 5)
```

Rispondiamo "Y" alla richiesta di continuare dopo l'informazione sullo spazio che verrà occupato sulla SDCard a fronte dell'installazione (Fig. 6).

Per pilotare l'uscita sonora, il sintetizzatore *eSpeak* si appoggia al pacchetto ALSA (Advanced Linux Sound Architecture) che è il componente del kernel di Linux che si occupa, tra l'altro, della gestione automatica dei driver per le schede audio più comuni, della gestione hardware del MIDI e del mixing tra canali multipli.

Una volta terminata l'installazio-

ne, per essere certi di instradare l'audio sull'uscita che desideriamo, configuriamo ALSA per forzare l'uscita del mixer audio verso l'uscita cuffie di RaspberryPi, con il comando:

```
amixer cset numid=3 1
```

dove il valore dell'ultimo parametro assume i significati:

0=auto, 1=analog, 2=hdmi

Collegiamo all'uscita cuffie di RaspberryPi un altoparlante amplificato del tipo destinato ai lettori mp3, smartphone e tablet. Fatto questo possiamo provare il funzionamento del tutto, dando il comando (Fig. 7):

```
espeak 'buongiorno a tutti' -v it -p 70 -s 155
```

Nel BOX dedicato ad *eSpeak* viene indicata la sintassi del comando ed il significato dei vari parametri; in questo comando *-v* indica il nome del file relativo al tipo di voce che si vuole ottenere, nel nostro caso una voce italiana generica; *-p* permette di controllare l'intonazione della voce, mentre *-s* la velocità del parlato, espresso in numero di parole al minuto.

Potete provare a variare i vari parametri e verificare gli effetti che si ottengono.

## FACCIAMO PARLARE RASPBERRY DA REMOTO

Passiamo ora ad integrare la sintesi vocale nella applicazione che gestisce le entrate e uscite digitali del GPIO da remoto, via Web.

Il risultato che vogliamo ottenere è di essere avvisati per mezzo di un messaggio vocale quando lo stato di una uscita digitale del GPIO viene modificata via

web, da un utilizzatore remoto dell'applicazione. Come ulteriore specifica vogliamo che l'avviso venga emesso solo per alcune delle uscite, e vogliamo anche che sia personalizzabile senza necessità di modificare il codice di un programma. Per fare questo adattiamo il database GPIO per supportare i dati necessari alle nuove funzionalità ed i programmi per gestirle in PHP e Python. Per quanto riguarda le modifiche ai programmi abbiamo preferito crearne una copia prima delle modifiche, in modo da conservare entrambe le versioni. I programmi modificati per la gestione della sintesi vocale si chiamano **control2.php** per la gestione delle pagine web e **GPIOServer2.py** per la gestione fisica del GPIO.

#### SCHEMA DI RIFERIMENTO DELLA APPLICAZIONE

Nello schema di **Fig. 8** è visibile il funzionamento dinamico dell'applicazione. Come descritto nel numero 170 della rivista, dopo l'autenticazione dell'utente con *userid* e *password*, che ricordiamo essere "admin" e "gpio", la pagina Web presenta l'elenco dei pin GPIO che è possibile gestire da remoto, la loro descrizione, lo stato corrente, indicato dall'immagine di un interruttore, ed i pulsanti virtuali per cambiare lo stato di ciascun pin ed eventualmente la sua descrizione. Quando un utente cambia lo stato di un pin l'azione viene recepita dal programma **control2.php** che modifica lo stato del pin interessato nella tabella **pinStatus**, nel database **gpio** (gestito da MySQL), rimandando al browser la pagina con la nuova configurazione. Il programma Python **GPIOServer.py**, che gira permanentemente in memoria, periodicamente interroga lo

stato dei pin sulla stessa tabella **pinStatus** e provvede ad inviare al driver del GPIO le istruzioni per configurare i pin fisici con gli stati presenti nella tabella. In questo modo si accende e si spegne fisicamente il LED collegato al connettore GPIO della nostra RaspberryPi.

Ora dobbiamo modificare la logica di funzionamento dell'applicazione, presentata nel numero 170, per introdurre la funzionalità di avviso vocale associato alle modifiche di stato dei pin operato da remoto.

Il programma che si occupa di produrre il messaggio vocale è il demone **GPIOServer2.py**, copia di **GPIOServer.php**, che deve "accorgersi" che lo stato di un pin è stato modificato e quindi, oltre all'azione fisica sul pin del GPIO deve anche reperire lo specifico messaggio da sintetizzare, e trasmetterlo all'uscita cuffie tramite il sintetizzatore *eSpeak*.

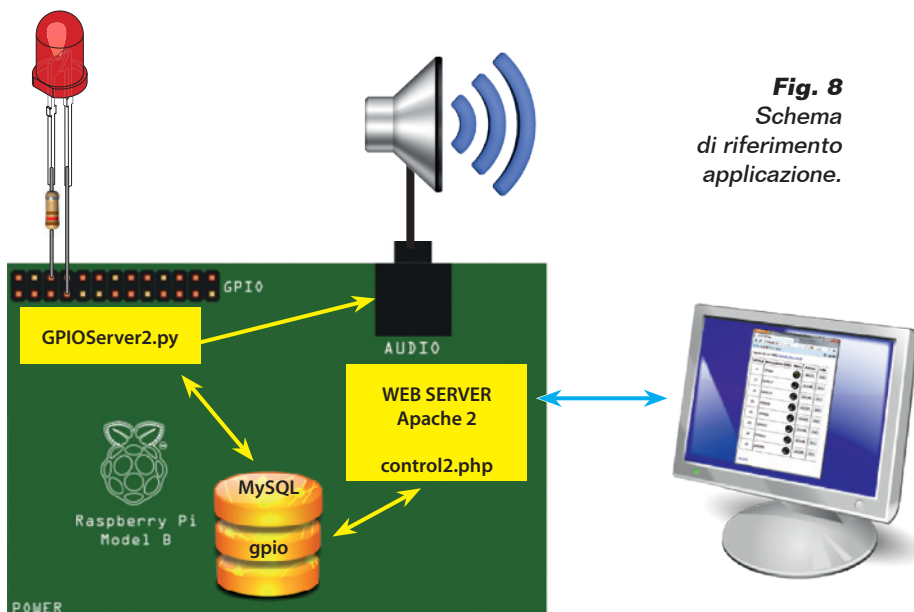
Abbiamo deciso di accollare al programma PHP di gestione delle pagine web, il compito di avvisare il GPIOServer della variazione di stato di un pin, modificandolo in modo che, al variare

di un pin, provveda anche ad "accendere" un indicatore sul database, che notifichi l'avvenuta variazione al GPIOServer, oltre che scrivere sulla tabella **pinStatus** il nuovo stato del pin.

Questa soluzione impiega una sorta di semaforo o indicatore di stato ed è comunemente utilizzata nei sistemi informativi per comunicare variazioni di stato tra processi che girano in modo asincrono e indipendenti tra di loro, come le macchine a stati.

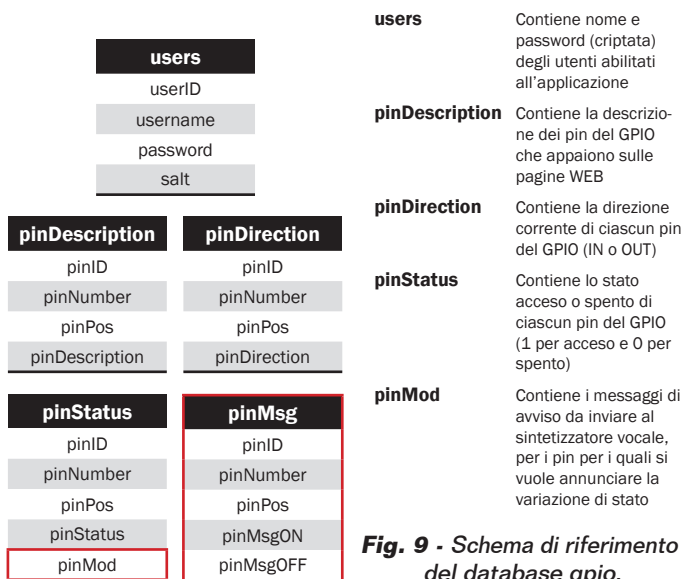
Quando il demone **GPIOServer2.py** eseguirà il suo "giro" di allineamento dei pin fisici del GPIO allo stato presente sulla tabella **pinStatus**, nel caso dei pin con l'indicatore di variazione acceso (valore = "1"), provvederà a produrre anche il messaggio di avviso e poi a spegnere l'indicatore, in modo da produrre il messaggio di avviso una sola volta. Notate che il sistema funziona anche nel caso in cui RaspberryPi venga spenta e riaccesa mentre il processo è ancora in corso.

Vediamo di approfondire prima le modifiche al database e poi ai programmi di gestione.



**Fig. 8**  
Schema di riferimento applicazione.





## DATABASE GPIO

Partiamo ad analizzare la struttura del database **gpio**, per il quale abbiamo descritto nel numero 170 della rivista le operazioni di installazione e configurazione. Nel database **gpio** sono presenti le tabelle che contengono le relazioni tra i dati necessari alla gestione del sistema, strutturate come visibile nello schema di **Fig. 9**.

Per tabella si intende una relazione tra una entità e gli attributi (informazioni) che la caratterizzano. Come esempio grossolano si potrebbe immaginare un foglio di calcolo, composto da tante righe, dove alcune colonne sono utilizzabili come campi di ricerca per selezionare una specifica riga e le altre ne contengono le informazioni descrittive. Ciascuna tabella è caratterizzata da una chiave primaria che nel nostro caso è un identificativo numerico. Alcuni

**users** Contiene nome e password (criptata) degli utenti abilitati all'applicazione

**pinDescription** Contiene la descrizione dei pin del GPIO che appaiono sulle pagine WEB

**pinDirection** Contiene la direzione corrente di ciascun pin del GPIO (IN o OUT)

**pinStatus** Contiene lo stato acceso o spento di ciascun pin del GPIO (1 per acceso e 0 per spento)

**pinMod** Contiene i messaggi di avviso da inviare al sintetizzatore vocale, per i pin per i quali si vuole annunciare la variazione di stato

attributi particolarmente significativi per la ricerca di un singolo elemento della tabella sono impostati come unici (non possono essere presenti nella tabella due righe con lo stesso valore) e non nulli (devono contenere dei valori significativi); su questi attributi sono costruiti gli indici per velocizzare le operazioni di ricerca e di aggiornamento di singole righe delle tabelle da parte dei programmi o delle applicazioni di gestione del database. Di fatto si configurano come chiavi di ricerca secondarie.

Per la tabella **users**, che contiene l'elenco degli utenti autorizzati all'uso della applicazione e le relative password, la chiave di ricerca è l'attributo (campo) **username** mentre per tutte le altre tabelle le chiavi di ricerca sono gli attributi **pinNumber** e **pinPos**, questa doppia impostazione è dovuta al diverso modo di riconoscere ciascun pin del GPIO da parte dei linguaggi PHP e Python. Il linguaggio PHP, per indirizzare un pin del GPIO, richiede l'identificativo GPIO, per esempio, nel caso del nostro LED, 4 (GPIO4), mentre il linguaggio Python indirizza i pin in base alla loro posizione fisica sul connettore (pin 7 nel nostro caso). Di conseguenza il programma PHP, che gestisce le pagine WEB dell'applicazione utilizza come campo di ricerca nelle tabelle

l'attributo **pinNumber** mentre il programma Python che gestisce fisicamente il GPIO utilizza l'attributo **pinPos**.

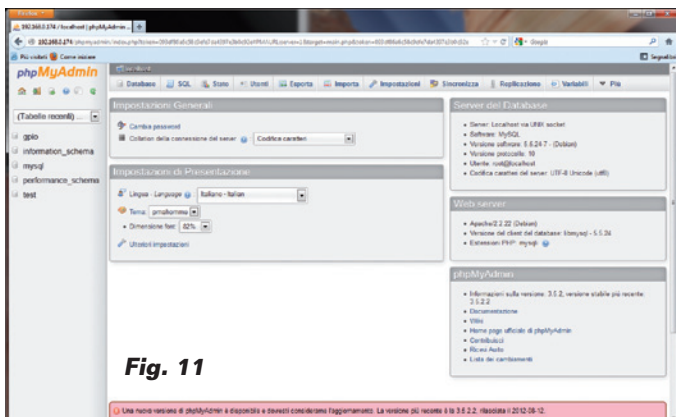
Sempre nella **Fig. 9** è visibile la struttura delle tabelle del database **gpio**, con la descrizione dei contenuti di ciascuna; le modifiche che dovranno essere apportate sono evidenziate con i rettangoli rossi.

## AGGIORNAMENTO DATABASE

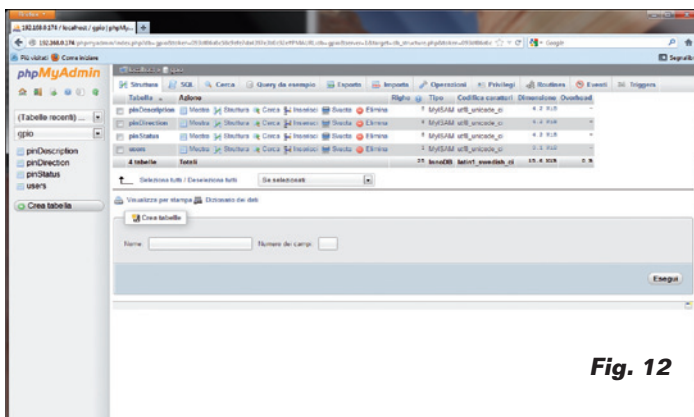
Con riferimento alla **Fig. 9** dovremo integrare la struttura del



**Fig. 10**



**Fig. 11**



**Fig. 12**

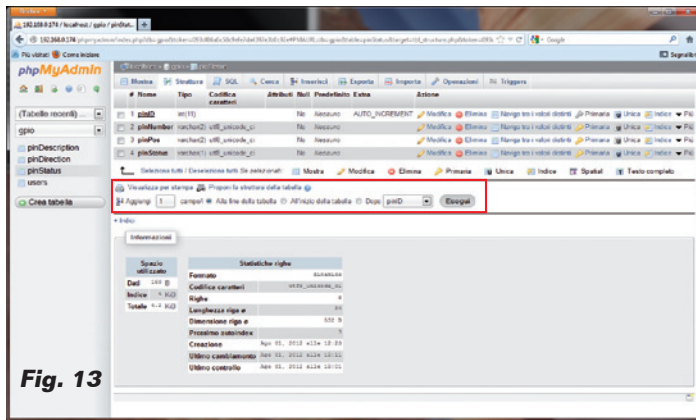


Fig. 13

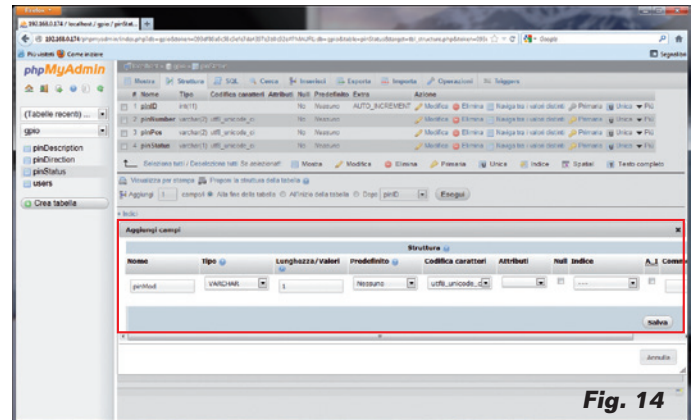


Fig. 14

database per portarlo alla configurazione che comprende anche il nuovo campo e la nuova tabella, evidenziati dai rettangoli rossi. In pratica si tratta di aggiungere alla tabella **pinStatus** un nuovo attributo, **pinMod**, che sarà utilizzato come indicatore di variazione dello stato del pin nel modo descritto in precedenza, ed una nuova tabella **pinMsg** che conterrà i testi dei messaggi di avviso che dovranno essere emessi in caso di variazione di stato di ciascun pin, differenziati sia per pin che in funzione della condizione di accesso o spento.

Eseguiamo passo passo il processo di modifica del database **gpio**, utilizzando l'applicazione di gestione database **phpMyAdmin**.

Entriamo in phpMyAdmin digitando l'indirizzo IP della scheda RaspberryPi seguito da /phpmyadmin come per esempio <http://192.168.0.190/phpmyadmin>.

Inseriamo userid e password e diamo Invio come in Fig. 10. Otteniamo la pagina di Fig. 11 sulla quale selezioniamo con doppio click il database **gpio** ottenendo la pagina di Fig. 12 che elenca le tabelle presenti nel database.

Clicchiamo sulla voce **Struttura** in corrispondenza della riga della tabella **pinStatus**, che apre la pagina di dettaglio della struttura della tabella stessa (Fig. 13). La nostra intenzione è di inserire un nuovo campo nella tabella e quindi configuriamo la richiesta di aggiunta campi come indicato nel rettangolo rosso e premiamo il pulsante **Esegui**. Nella pagina di Fig. 14 impostiamo le caratteristiche del nuovo campo **pinMod** che sarà utilizzato come indicatore di cambio di stato del pin corrispondente, con le scelte visibili nel rettangolo rosso. Alla fine confermiamo il tutto con il pulsante **Salva**. La pagina di Fig. 15 ci conferma che il campo **pinMod** è stato aggiunto alla tabella **pinPos**. Ora selezioniamo il tab **Mostra** per ottenere una griglia (Fig. 16) dove è possibile modificare il contenuto dei campi della tabella. Dato che nella creazione abbiamo specificato il campo come NOT NULL,

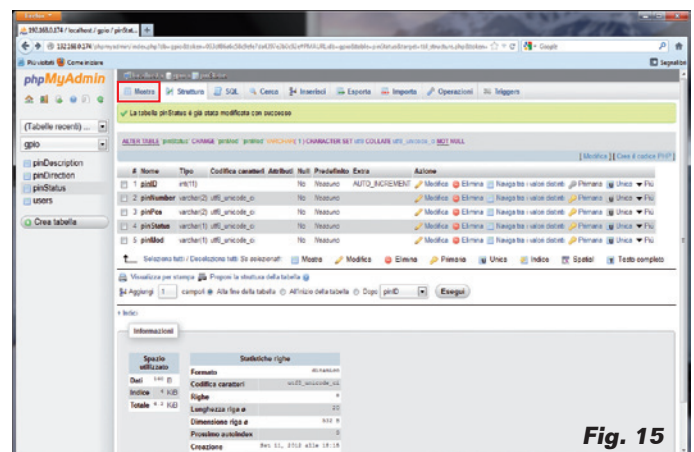


Fig. 15

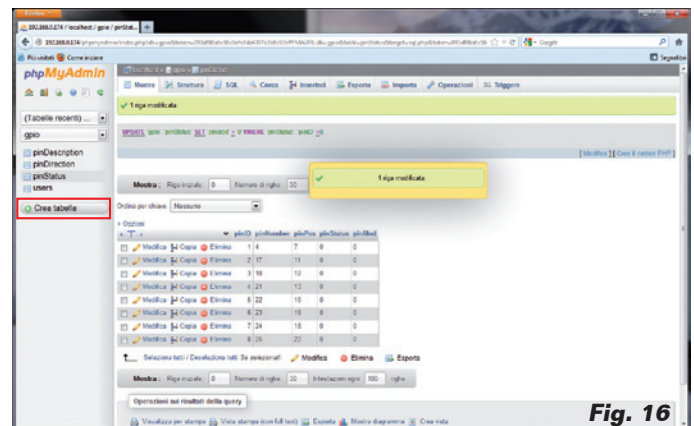


Fig. 16

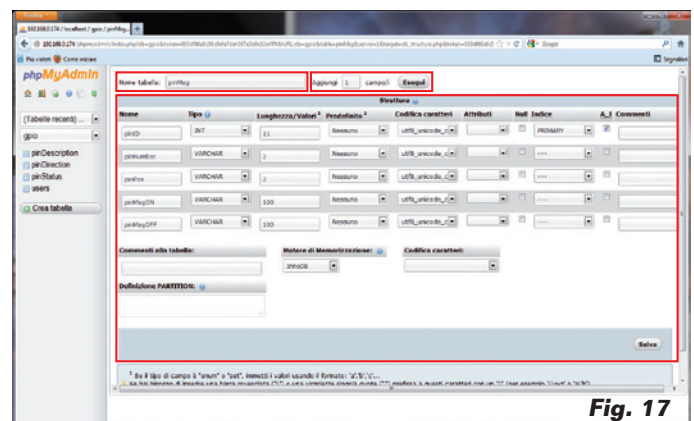


Fig. 17



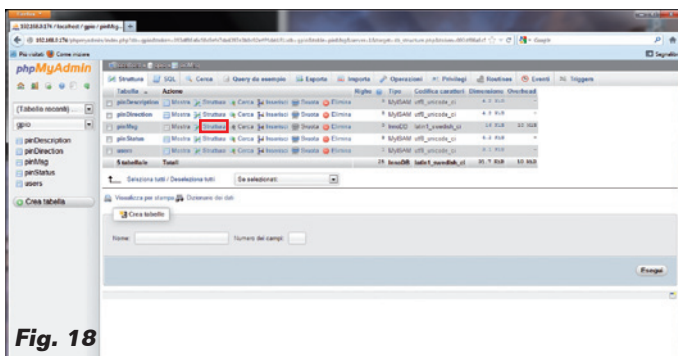


Fig. 18

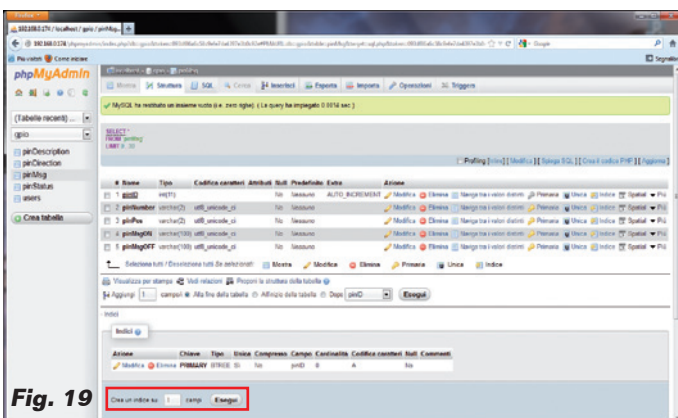


Fig. 19

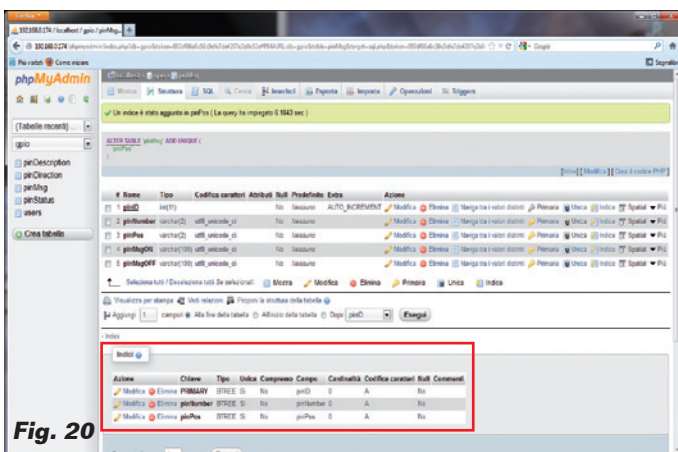


Fig. 20

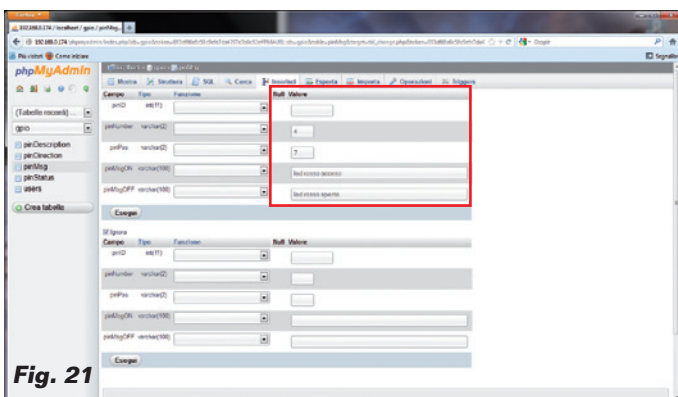


Fig. 21

inseriamo il valore 0 nel campo **pinMod** di ciascuna riga della tabella. Quando passiamo da una riga della tabella alla successiva o quando diamo *Invio* il campo nella tabella viene aggiornato al valore desiderato. Alla fine otteniamo il risultato visibile nella **Fig. 16**.

Creiamo ora la tabella che contiene i messaggi di avviso di cambiamento di stato dei pin. Sulla sinistra della pagina di **Fig. 16** selezioniamo l'ultimo pulsante in basso *Crea Tabella* e riempiamo i campi della pagina di **Fig. 17** come indicato nella stessa. Nel caso ci serva una riga in più inseriamola cliccando sul pulsante *Esegui* della funzione in alto. Aggiungiamo 1 campo/i sempre nella pagina di **Fig. 17**. Clicchiamo sul pulsante *Salva* in fondo a destra nella pagina e otteniamo nuovamente la pagina di **Fig. 18** con elencate 5 tabelle, clicchiamo su *Struttura* della tabella **pinMsg** per entrare nella pagina di **Fig. 19** che useremo per aggiungere gli indici, che ci serviranno per velocizzare le ricerche gli aggiornamenti sulla tabella stessa. Aggiungiamo un indice sul campo **pinNumber** ed uno sul campo **pinPos**.

Ora che è stata creata la tabella per contenere il testo dei messaggi che dovranno avvisare delle variazioni di stato dei piedini del GPIO, è il momento di popolarla.

Clicchiamo sulla tab *Mostra* in alto a sinistra della pagina web e otteniamo la pagina visibile in **Fig. 21**. Compiliamola con i contenuti visibili nella pagina stessa; i campi **pinMsgON** e **pinMsgOFF** contengono il testo dei messaggi che verranno sintetizzati al verificarsi dei corrispondenti eventi. Chiaramente potete inserire ciò che desiderate. Clicchiamo sul pulsante *Esegui* per confermare l'inserimento della riga nella tabella **pinMsg**.

A questo proposito, tra le caratteristiche del progetto, dobbiamo decidere se rendere obbligatoria la comunicazione di un messaggio per ciascun pin che

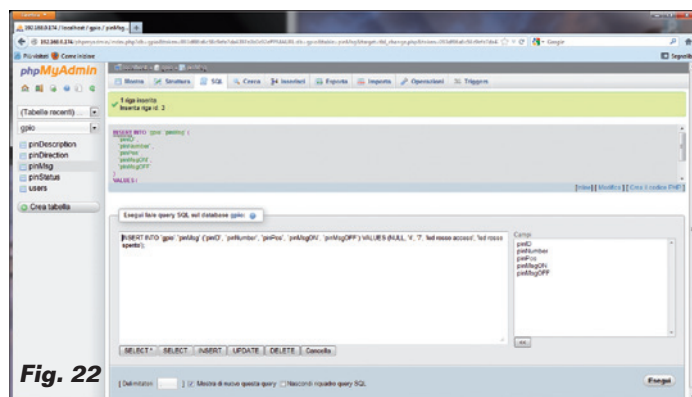
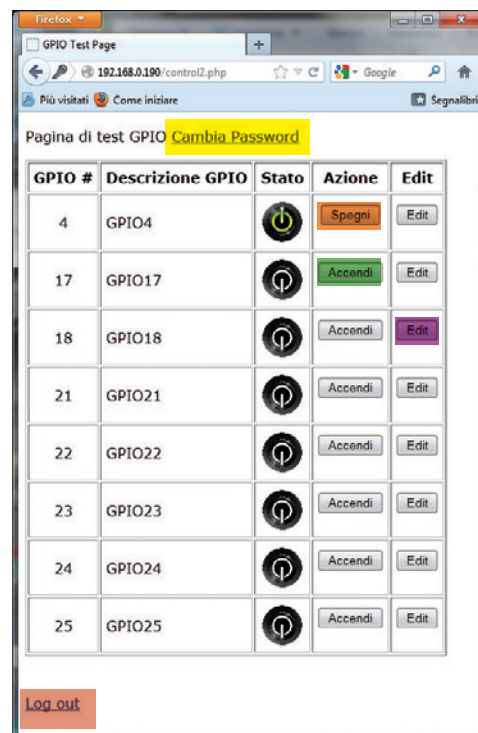
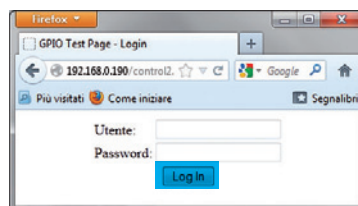
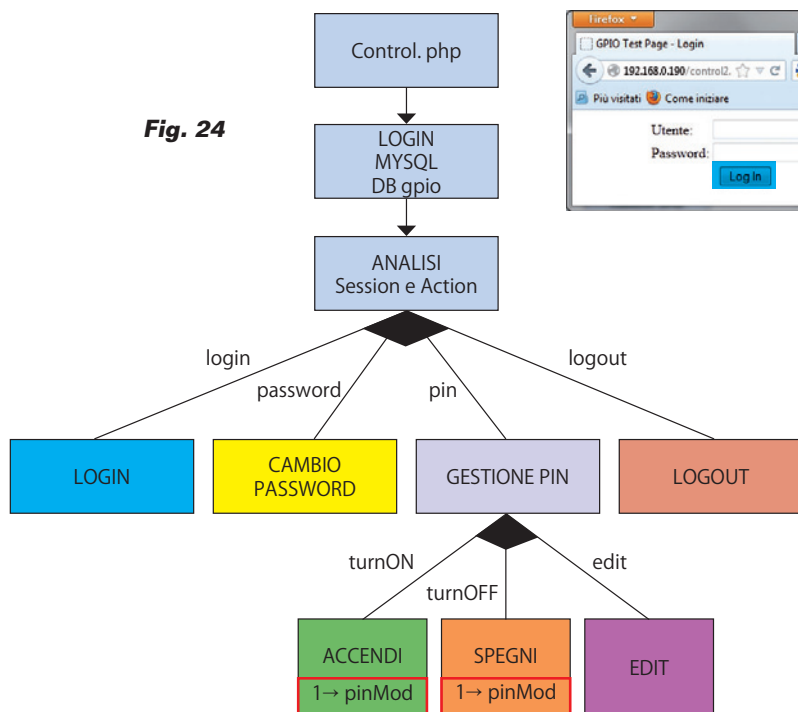


Fig. 22

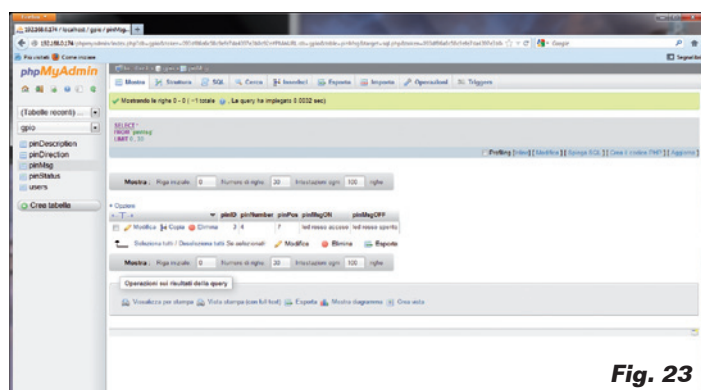




può essere variato da remoto, e di conseguenza inserire una riga nella tabella **pinMsg** per ciascun pin, oppure lasciare decidere all'utilizzatore se generare un messaggio oppure no quando uno specifico pin cambia di stato. Abbiamo optato per la seconda possibilità e quindi inseriamo solo le righe relative ai pin per i quali desideriamo un avviso, tralasciando le altre.

Nell'adattare il software di gestione dovremo poi tenere conto della presenza o meno di un messaggio da produrre, adeguando il comportamento del programma.

Dopo avere confermato l'inserimento della riga con il pulsante *Esegui* otterremo la pagina di conferma visibile in **Fig. 22**. Clicchiamo sul tab *Mostra* in alto a sinistra sulla pagina e otterremo la visualizzazione delle righe contenute nella tabella (**Fig. 23**).



Per il momento abbiamo terminato con le modifiche al database e possiamo ad analizzare e modificare i programmi di gestione delle pagine web e dello stato fisico del pin del GPIO.

## PROGRAMMA GESTIONE PAGINE WEB

Come descritto nel numero 170, per la gestione delle pagine web dell'applicazione abbiamo adottato il server web Apache2, che è universalmente utilizzato per questa funzionalità. Per la gestione delle pagine web, che nel nostro caso sono dinamiche e vengono generate ex novo ad ogni richiesta utente, abbiamo scelto il linguaggio PHP, molto diffuso per questo genere di utilizzo, facile da apprendere e con una notevole valenza didattica.

L'acronimo PHP che un tempo stava per Personal Home Page è stato in seguito aggiornato in Hypertext Preprocessor, per sottolineare la sua vocazione di generatore di pagine HTML dinamiche. Sempre nel numero 170 abbiamo presentato il programma **control.php** per la gestione delle pagine web del nostro progetto, derivato dall'iniziativa open source raspberrypi-gpio che potete trovare all'indirizzo <https://code.google.com/p/raspberrypi-gpio/>.

In questo programma dobbiamo inserire le istruzioni necessarie a portare il campo **pinMod** della tabella **pinStatus** al valore "1", quando un pin viene cambiato di stato.

Senza volere entrare nei dettagli del linguaggio PHP e della logica di produzione delle pagine HTML dinamiche, che esula dagli scopi della nostra

## Listato 1

```
$action = $_GET['action'];
$pin = mysql_real_escape_string($_GET['pin']);
$setMod = "1";
if ($action == "turnOn"){
    $setting = "1";
    mysql_query("UPDATE pinStatus SET pinStatus='$setting', pinMod='1' WHERE pinNumber='$pin'");
    mysql_close();
    header('Location: control2.php');
} else if ($action == "turnOff"){
    $setting = "0";
    mysql_query("UPDATE pinStatus SET pinStatus='$setting', pinMod='1' WHERE pinNumber='$pin'");
    mysql_close();
    header('Location: control2.php');
```

rivista, vi proponiamo invece nella **Fig. 24** lo schema della struttura funzionale del programma. Nello schema strutturale, le losanghe indicano blocchi eseguiti selettivamente a seconda dei valori contenuti nelle variabili di controllo. Nella Figura 24 i rettangoli colorati che evidenziano le singole funzionalità nelle immagini delle pagine web corrispondono ai colori dei blocchi nel diagramma di struttura che elaborano le funzionalità stesse, e che trovano riscontro nelle corrispondenti sezioni del listato. Quando un utilizzatore esterno richiede la pagina, come prima operazione il programma si connette al database **gpio** ospitato dal server MySQL. Poi verifica se l'utente si collega per la prima volta, in tal caso restituisce la pagina di login con la richiesta di Userid e Password.

Dopo il buon esito del login viene presentata la pagina di gestione dei pin. Questa pagina permette la richiesta di cambio stato di uno dei pin evidenziati nonché la possibilità di cambiare la password e di uscire dalla applicazione con logout (rispettivamente rami *pin*, *password* e *logout* nello schema). Per ciascuna riga che si riferisce ad un singolo pin è possibile variare lo stato in Acceso o Spento, a seconda della stato corrente, nei quali

casi vengono attivati rispettivamente i rami *turnOn* e *turnOff* dello schema, oppure modificarne la descrizione (ramo *edit*). Con riferimento al diagramma dovremo intervenire sul comportamento dei blocchi ACCENDI PIN (ramo *turnOn*) e SPEGNI PIN (ramo *turnOff*) in modo che sulla tabella **pinStatus** oltre al campo **pinStatus** sia modificato anche il campo **pinMod** portandolo al valore 1.

Nel **Listato 1** le istruzioni da modificare nel programma `control2.php`, ottenuto come copia del programma `control.php` sono state evidenziate con il rettangolo rosso. All'interno del listato del programma `control2.php` devono anche essere modificate tutte le occorrenze di `control.php` in `control2.php`.

Come informazione aggiuntiva (**Listato 2**) riportiamo un estratto condensato della sequenza di istruzioni necessarie a collegarsi al database MySQL da un programma PHP; per approfondire l'argomento potete visitare il sito <http://www.html.it/guide/guida-php-e-mysql-pratica/>, mentre per approfondire il linguaggio PHP potete visitare il sito <http://php.net/manual/it/index.php>.

### PROGRAMMA GESTIONE FISICA GPIO

Sempre nel numero 170 abbiamo anticipato la scelta del linguaggio Python per la realizzazione di programmi di gestione lato server, destinati ad essere eseguiti come processi permanenti o richiamati come processi secondari per eseguire specifiche

## Listato 2

```
$MySQLHost = "localhost";
$MySQLDB = "gpio";
$MySQLUsername = "gpio";
$MySQLPassword = "pi";

$dbConnection = mysql_connect($MySQLHost, $MySQLUsername, $MySQLPassword);
mysql_select_db($MySQLDB, $dbConnection);
...
$query = mysql_query("SELECT pinDescription FROM pinDescription WHERE
pinNumber='$pin'");
...
mysql_query("UPDATE pinStatus SET pinStatus='$setting', pinMod='1' WHERE
pinNumber='$pin'");
...
mysql_close();
```

funzioni al verificarsi di eventi particolari. Sono ovviamente disponibili altri linguaggi per la scrittura di processi lato server come il C, il Perl, l'assembler e molti altri, ma il Python si presta particolarmente per la sua, sempre relativa, facilità di apprendimento e semplicità nella scrittura dei programmi; ad esempio non è necessario dichiarare le variabili prima di utilizzarle e non sono necessari terminatori per le istruzioni ed i blocchi.

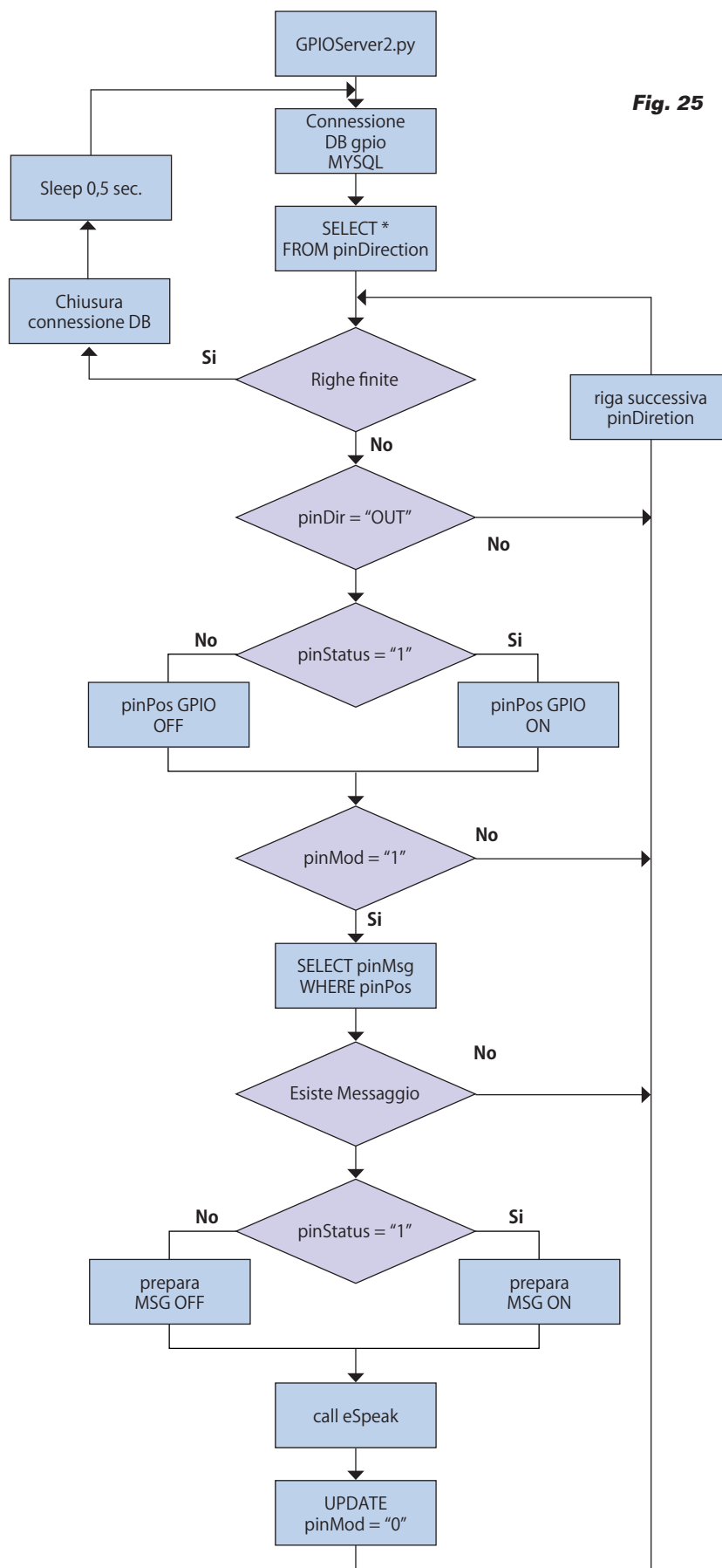
La struttura logica del programma la si imposta gestendo correttamente la indentazione delle istruzioni sia singole che in blocchi strutturati. In Python deve essere prestata particolare attenzione all'indentazione dei blocchi di istruzioni in modo da rispettare la logica del programma che si sta realizzando.

Vediamo ora come adattare il programma **GPiOServer.py** che abbiamo presentato nel numero 170 per fargli riconoscere la variazione di stato di un pin e attivare il sintetizzatore vocale trasferendogli contestualmente il messaggio corretto da produrre. Anche in questo caso proponiamo il listato del programma, approfittandone per dare delle indicazioni iniziali sulle caratteristiche del linguaggio. I commenti sono indicati con il carattere "#".

Le istruzioni **import** servono per includere le librerie necessarie al funzionamento del programma; nel nostro caso la libreria **time** per la gestione del ritardo tra un ciclo di esecuzione del programma ed il successivo; la libreria **RPi.GPIO** per la gestione dei pin del connettore GPIO; la libreria **MySQLdb** per l'interfacciamento con il database MySQL.

L'import della funzione **call** dalla libreria **subprocess** è necessaria in quanto abbiamo deciso di ri-

**Fig. 25**





## Listato 3

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import time
import RPi.GPIO as GPIO
import MySQLdb as mdb
import sys
from subprocess import call

DbServer = "localhost"
DbUser = "gpio"
DbPasswd = "pi"

X_pinPos = 99
X_pinStatus = 0
X_pinDirection = "zzz"
X_pinMod = 0
X_pinZero = 0

while True:

    # con = mdb.connect('localhost', 'gpio', 'pi', 'gpio');
    con = mdb.connect(DbServer, DbUser, DbPasswd, 'gpio');

    with con:

        curl = con.cursor(mdb.cursors.DictCursor)
        curl.execute("SELECT * FROM pinStatus")

        rows = curl.fetchall()
        for row in rows:

            X_pinPos = row["pinPos"]
            X_pinStatus = row["pinStatus"]
            X_pinMod = row["pinMod"]

            cur2 = con.cursor(mdb.cursors.DictCursor)
            cur2.execute("SELECT * FROM pinDirection WHERE pinPos = %s", (X_pinPos))
            rows2 = cur2.fetchall()
            for row2 in rows2:
                X_pinDirection = row2["pinDirection"]

            if X_pinDirection == "out":
                GPIO.setup(X_pinPos, GPIO.OUT)
                if X_pinStatus == "1":
                    GPIO.output(X_pinPos, True)
                else:
                    GPIO.output(X_pinPos, False)

            if X_pinMod == "1":
                cur3 = con.cursor(mdb.cursors.DictCursor)
                cur3.execute("SELECT * FROM pinMsg WHERE pinPos = %s", (X_pinPos))

                if cur3.rowcount > 0:
                    rows3 = cur3.fetchall()

                    for row3 in rows3:

                        if X_pinStatus == "1":
                            parla=row3["pinMsgON"]
                        else:
                            parla=row3["pinMsgOFF"]

                        call(["espeak", parla, "-v", "it", "-p", "70", "-s", "155"])

                cur4 = con.cursor(mdb.cursors.DictCursor)
                cur4.execute("UPDATE pinStatus SET pinMod=%s
WHERE pinPos=%s", (X_pinZero, X_pinPos))

        if con:
            con.close()

        time.sleep(0.5)
```

chiamare il sintetizzatore di voce *eSpeak*, quando richiesto, come processo esterno. Ciò significa che ogni volta che è necessario eseguire *eSpeak* per sintetizzare un messaggio, il programma Python richiede l'apertura di un nuovo processo al quale passa il comando di esecuzione di *eSpeak* con la stessa sintassi di una riga di comando di terminale:

```
call(["espeak", parla, "-v",
"it", "-p", "70", "-s",
"155"])
```

l'uso dei doppi apici permette di trasferire correttamente le variabili e le costanti dall'argomento della **call** alla riga di comando del processo esterno.

In questo modo diventa semplice eseguire dei test da riga di comando del terminale prima di modificare le istruzioni all'interno del programma o inserire i messaggi nella tabella **pinMsg** del database.

Nella **Fig. 25** è visibile il diagramma a blocchi del programma che ci permetterà di interpretarne meglio il listato.

Approfondiamo ora alcune particolarità riguardo l'interfacciamento con il database MySQL. Il linguaggio di riferimento per la manipolazione di database relazionali è l'SQL (Structured Query Language), che può essere utilizzato come linguaggio di comandi oppure integrato in un altro linguaggio ospite, come Python, nel nostro caso.

Per interrogare i dati contenuti in una tabella utilizziamo l'istruzione **SELECT** con la sua struttura sintattica ed i suoi parametri. Il risultato di una istruzione **SELECT** è una lista di righe che soddisfano i criteri di selezione della **SELECT** stessa. La lista può essere vuota, in caso nessuna *tupla* soddisfi i criteri

di ricerca, oppure composta da una o più *tuple* soddisfacenti i criteri di selezione.

In un linguaggio come Python è necessario elaborare il risultato della selezione ottenuta una riga alla volta. È necessario quindi utilizzare un costrutto intermedio che permetta di indicizzare le singole righe della lista ottenuta.

Il costrutto in oggetto è il cursore (**cursor**) che fornisce sia un'area dati strutturata per contenere il risultato della selezione che il meccanismo di puntamento per poter scorrere l'area dati una riga alla volta.

Un cursore deve essere utilizzato secondo una corretta sequenza di istruzioni che riportiamo di seguito stralciandole dal Listato 2. Come in PHP la prima operazione da eseguire è connettersi al database.

```
DbServer = "localhost"
DbUser = "gpio"
DbPasswd = "pi"
```

```
con = mdb.connect(DbServer,
DbUser, DbPasswd, 'gpio');
```

Dobbiamo poi definire un cursore associato alla connessione dandogli un nome univoco all'interno del programma

```
curl = con.cursor(mdb.cursors.DictCursor)
```

Il metodo **execute** permette il popolamento dell'area strutturata con il risultato della istruzione **SELECT**.

```
curl.execute("SELECT *
FROM pinStatus")
```

Se dobbiamo verificare che il numero di righe ottenute dalla selezione sia maggiore di zero si può usare l'istruzione:

```
if curl.rowcount > 0:
```

Per circoscrivere il sottoinsieme della selezione che vogliamo elaborare all'interno del programma usiamo l'istruzione:

```
rows = curl.fetchall()
```

Con **fetchall()** vengono reperite tutte le righe che soddisfano i criteri della **SELECT**, con **fetchone()** viene reperita solo la prima delle righe che soddisfano i criteri della select mentre con **fetchmany(num)** vengono reperite un numero di righe uguali a num.

```
for row in rows:
```

L'istruzione **for** permette di scorrere una riga alla volta il sottoinsieme impostato con le fetch; i campi contenuti in ciascuna riga possono essere reperiti con l'istruzione

```
X_pinPos = row["pinPos"]
```

Alla fine della elaborazione chiudiamo la connessione al database, liberando tutte le risorse impegnate, con l'istruzione:

```
con.close()
```

Il costrutto **cursor** viene utilizzato anche per eseguire istruzioni SQL diverse dalla **SELECT**, per esempio, come nel nostro caso, per azzerare il campo **pinMod** nella tabella **pinStatus** al fine di evitare di ripetere il messaggio più volte:

```
cur4.execute("""UPDATE
pinStatus SET pinMod=%s
WHERE pinPos=%s""", (X_
pinZero, X_pinPos))
```

Le modifiche da apportare al programma originario, per includere la funzionalità di sintesi dei mes-

## Guarda RaspberryPi in azione su YouTube!



**Vuoi sentire parlare Raspberry? Nulla di più semplice: collegati alla pagina <http://goo.gl/PzCDB>. Potrai anche vedere i video di moltissimi altri nostri progetti proposti in passato.**

saggi vocali sono evidenziate con i rettangoli rossi nel **Listato 3**. Nel caso non vogliate eseguire manualmente le modifiche descritte in questo articolo, come sempre potete scaricare la versione aggiornata dei programmi dal sito della rivista ([www.elettronica-in.it](http://www.elettronica-in.it)).

Abbiamo anche aggiunto un nuovo file **gpio.sql** per creare da zero un nuovo database **gpio** con la nuova configurazione. Per provare i nuovi programmi valgono le istruzioni descritte nel numero 170, facendo attenzione al fatto che abbiamo modificato i nomi dei programmi, in modo da mantenere nella RaspberryPi entrambe le versioni funzionanti contemporaneamente.

Quindi da web bisogna richiamare **control2.php**

<http://indirizzo-ip/control2.php>

e come programma lato server bisogna lanciare **GPIOServer2.py** dall'utente **root**.

```
cd /home
python GPIOServer2.php
```

In un prossimo articolo approfondiremo ulteriormente le possibilità di integrazione delle periferiche GPIO nelle applicazioni. ■



Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

# Elettronica In

[www.elettronica.in.it](http://www.elettronica.in.it)

oltre l'elettronica



# STAZIONE METEO CON RASPBERRYPI

di MARCO MAGAGNIN

Approfondiamo nuovi aspetti di Linux assemblando una applicazione che rileva i dati meteo dalla centralina La Crosse WS2355 e li pubblica sul sito meteo Weather Underground senza scrivere una riga di codice. Se RaspberryPi tarda ad arrivare l'applicazione funziona senza modifiche su MK802.



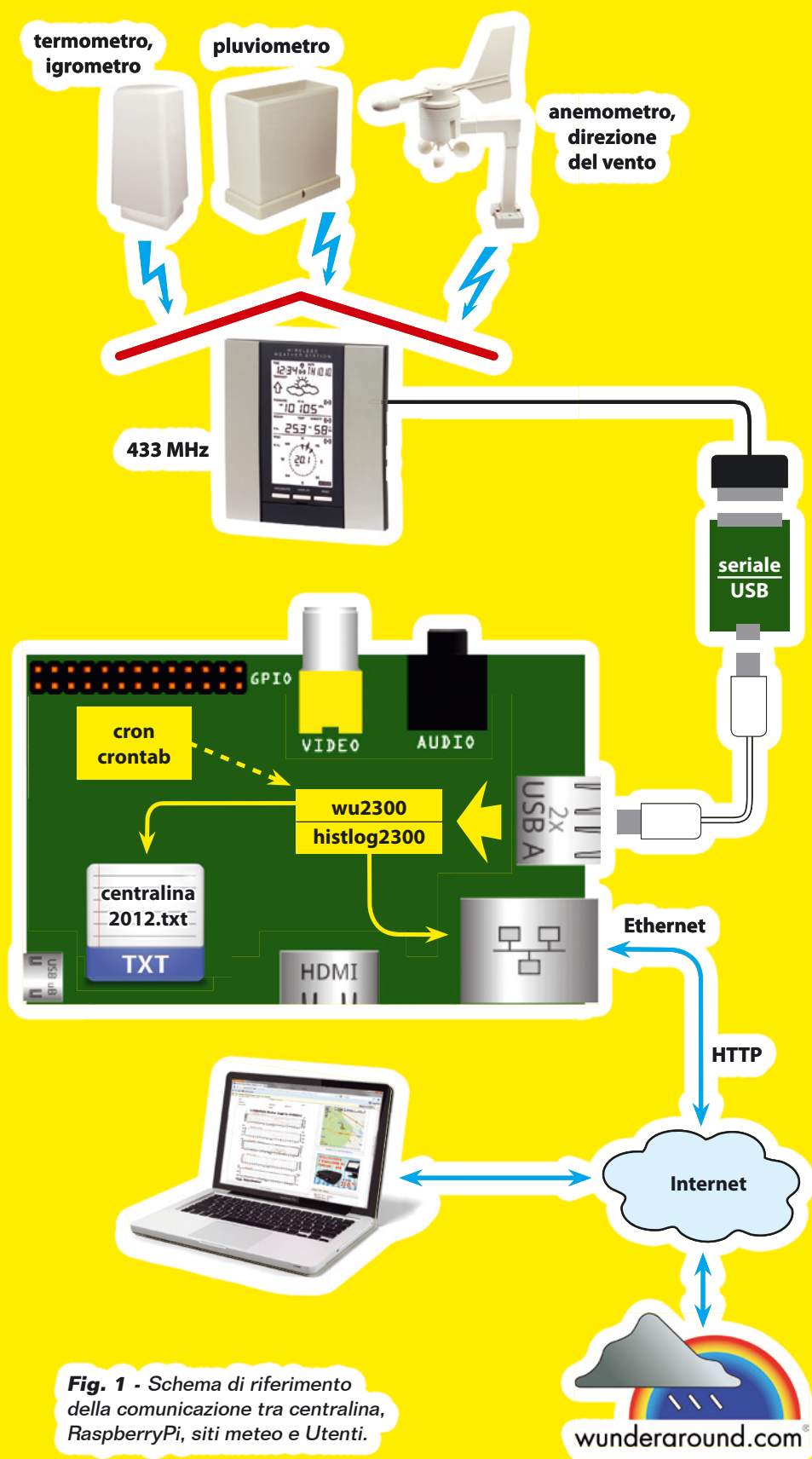
**D**i mese in mese continua la crescita di GNU-Linux, il sistema operativo di riferimento e di supporto a un numero sempre maggiore di dispositivi entrati a far parte delle nostre abitudini quotidiane. A prima vista l'affermazione può apparire azzardata, soprattutto se ci limitiamo ad osservare i PC che utilizziamo al lavoro e a casa, dato che sono per lo più dotati di sistemi operativi Micro-

soft Windows. Se però ci soffermiamo un attimo a considerare il campo di applicazione dei sistemi operativi nella sua interezza, possiamo constatare che le cose stanno diversamente. Prendiamo ad esempio gli smartphone e i tablet, dei quali ormai siamo quasi tutti dotati: quelli con Android utilizzano il kernel GNU-Linux, così come molti dei dispositivi NAS e Smart TV. Senza tenere conto che



servizi come Google, Facebook e social network oltre che banche, aziende ed istituzioni pubbliche e private di tutto il mondo sono erogati da migliaia e migliaia di server con

sistema operativo GNU-Linux. Dai supercomputer, ai sistemi embedded, alle schede micro PC, il coro fa ormai sentire la sua voce. Soprattutto per i mi-



**Fig. 1** - Schema di riferimento della comunicazione tra centralina, RaspberryPi, siti meteo e Utenti.

cro PC di nostro interesse quali RaspberryPi, MK802, Aria G25, Hackberry e simili concordiamo con Linus Torwald - il creatore del kernel di Linux e tuttora "curatore" della sua manutenzione ed evoluzione - che afferma che gli attuali micro PC, per contenere i prezzi, soffrono di una carenza di prestazioni dovuta all'adozione di chip SoC già un po' datati. Le prossime versioni, con molta probabilità, avranno maggiore potenza e potranno essere impiegate in applicazioni più complesse e onerose computazionalmente rispetto alle attuali.

Questa premessa è per sollecitarvi ancora ad approfondire gli ambienti Linux ai vari livelli, anche per motivi, perché no, professionali.

Tornando ai nostri appuntamenti, dopo la panoramica presentata nei numeri scorsi, volutamente più pratica che teorica sui principali argomenti di gestione ed utilizzo del sistema Linux, passando anche per l'accesso da remoto, il web server, il database MySQL, il GPIO e la gestione della sintesi vocale, per questa puntata abbiamo pensato di proporvi un'applicazione utile sia a mettere in pratica tutti gli argomenti toccati finora, sia a introdurre di nuovi; anche in questo caso senza praticamente dover scrivere una riga di codice. Nella scelta dell'argomento da proporvi, siccome è un'abitudine ormai consolidata dell'uomo moderno iniziare la giornata parlando del tempo, abbiamo deciso di adeguarci. Descriveremo le modalità di interfacciamento a RaspberryPi di una centralina meteorologica, quindi utilizzeremo i dati ottenuti per gli scopi più vari, iniziando con la pubblicazione sui siti di raccolta e divulgazione centralizzata dei dati meteo - come Weather Underground e Citizens Weather - per poi arricchire l'applicazione



con funzioni personalizzate che vi presenteremo nei prossimi numeri. Il semplice interesse o la necessità di conoscere dati meteo in tempo reale e storicizzati porta molte persone ad acquistare ed installare una centralina meteo dotata di sensori esterni ed interni con la quale viene effettuata la misura di intensità e direzione del vento, precipitazioni, temperature, umidità. Se si vogliono acquisire ed archiviare i dati storici per utilizzo proprio e/o per caricarli su siti centralizzati di raccolta, si rende necessario collegare la centralina meteo ad un PC, che deve rimanere sempre acceso ed a sua volta essere collegato alla rete Internet per la trasmissione e/o la consultazione dei dati. Dato il tipo di impiego, questi PC sono per lo più totalmente dedicati, ed il costo di acquisto iniziale, sommato al consumo di energia nel tempo per tenerli in funzione, rappresenta un impegno economico non indifferente.

Se moltiplichiamo questi costi per il numero di centraline collegate ai siti di dati meteo la cifra totale diventa davvero consistente. Sostituire i PC dedicati con hardware di basso costo e di basso consumo può contribuire molto a mantenere in vita questi servizi, di notevole utilità e per lo più gestiti su base volontaria.

Circa l'utilità dei dati raccolti, oltre all'utilizzo meteorologico classico e alla documentazione delle condizioni che hanno caratterizzato eventi eccezionali se non catastrofici, questi sono di grande utilità nelle attività di ricerca legate, ad esempio, all'agricoltura ed all'allevamento.

L'andamento nel tempo di temperature, umidità e precipitazioni permette di determinare la maturazione di determinati prodotti agricoli, così come il determinare se le condizioni meteo sono favo-

revoli allo sviluppo di determinate specie parassite permette di approntare, per tempo, appropriate misure di contrasto, o ancora, determinare le conseguenze sulla presenza a determinate quote alpine di specie vegetali e/o animali al variare delle condizioni climatiche di lungo periodo.

Per queste rilevazioni molte centraline sono installate in quota presso rifugi di montagna o installazioni di dighe, in condizioni che un PC normale non è in grado di sopportare. Anche qui un RaspberryPi potrebbe spesso sostituire un ben più costoso PC.

Ma torniamo alla nostra applicazione su RaspberryPi. Come abbiamo già detto negli articoli precedenti, prima di iniziare a scrivere programmi per applicazioni in ambiente Linux è bene verificare cosa esiste di già disponibile; per fare questo è necessario costruirsi uno schema di riferimento del progetto per poter decidere per quali componenti sono disponibili soluzioni open source e per quali altri componenti è necessario scrivere il proprio codice. Lo schema di riferimento è visibile in Fig. 1.

### CENTRALINA

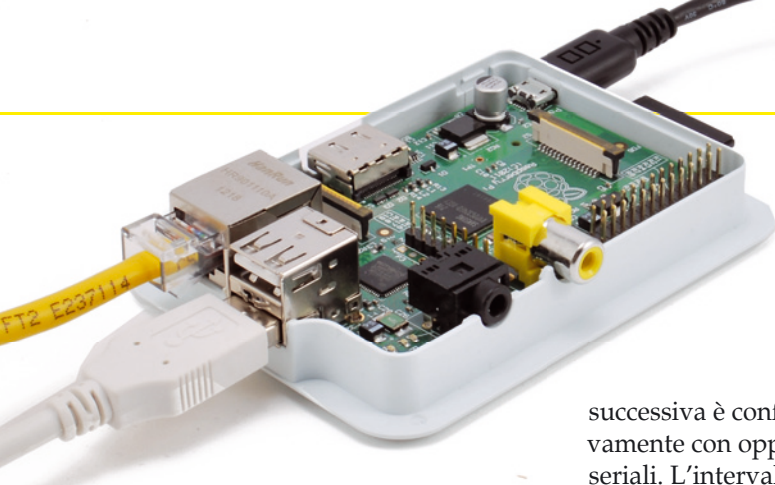
Iniziamo ad analizzare lo schema partendo dalla centralina, per la quale abbiamo scelto il modello WS2350. È prodotta dalla francese La Crosse Technology ed è reperibile presso Futura Elettronica con il codice B220-WS2355. I motivi della scelta sono: la larghissima diffusione di questa centralina - o di centraline della stessa famiglia - presso le stazioni di rilevamento che alimentano i siti meteo amatoriali e non, la presenza del collegamento seriale per l'accesso ai dati memorizzati nella centralina e la disponibilità della mappa secondo la quale i dati sono archiviati nella memoria della centralina stessa, la disponibilità di software open

## CARATTERISTICHE TECNICHE

- Ora e data radiocontrollati tramite segnale DCF77
- Visualizzazione dei dati meteo con funzioni di allarme programmabili, nonché la registrazione di tutti i valori minimi e massimi con ora e data di registrazione
- Temperatura interna ed esterna in °C o °F e visualizzazione dell'umidità relativa
- Visualizzazione dell'umidità interna e esterna
- Visualizzazione del giorno, data, ora (12/24h), mese e fuso orario
- Velocità del vento in mph, km/s, m/s, nodi o Beaufort
- Visualizzazione della direzione del vento con bussola LCD
- Visualizzazione temperatura wind chill
- Visualizzazione punto di rugiada
- Previsioni meteo con icone (soleggiato, nuvoloso, piovoso)
- Indicatore di tendenza meteo
- Allarme temporali
- Retroilluminazione a LED
- Visualizzazione simultanea di tutti i dati meteo con le impostazioni dei singoli utenti
- Dispone di porta COM per connessione al PC
- Tutti i rilevamenti meteo registrati dalla stazione di base (fino a 175 serie di dati possibili) sono trasmessi e aggiornati grazie al software (intervalli di misurazione impostabili a scelta dell'utilizzatore)
- Requisiti di sistema: Windows® XP, Windows® Vista (non con Windows® 7); con le indicazioni di questo articolo anche Linux in genere, RaspberryPi e MK802 in particolare.







source con le funzioni di base utili per la gestione.

Le caratteristiche della centralina sono descritte nel riquadro dedicato. Chiaramente, i sensori esterni che misurano la quantità di pioggia, la temperatura esterna, la pressione, la direzione e la velocità del vento dovranno essere installati, come descritto nel manuale, in modo da garantire le migliori condizioni per le misurazioni.

I sensori e la centralina comunicano alla frequenza di 433 Mhz e quindi è anche necessario non superare le distanze indicate tra i sensori e la centralina per garantire una buona comunicazione.

I dati ricevuti dai sensori sono visualizzati sul display della centralina e memorizzati a intervalli regolari nella memoria interna, dalla quale possono successivamente essere letti via software attraverso il collegamento seriale.

La centralina meteo possiede una porta seriale che permette di collegarla ad un PC o, come nel nostro caso, a RaspberryPi utilizzando il cavo seriale ed il convertitore seriale/USB inclusi nella confezione. I dati meteo vengono letti dal collegamento seriale ad intervalli regolari con programmi che provvedono ad elaborarli e/o a memorizzarli. Una volta memorizzati i dati possono essere pubblicati su uno o più siti di dati meteo centralizzati, oppure utilizzati in applicazioni di nostra utilità o ancora essere elaborati ed analizzati per fini statistici o per ulteriori applicazioni.

L'intervallo tra una memorizzazione di dati e la

successiva è configurato interattivamente con opportuni comandi seriali. L'intervallo configurato determina anche la copertura temporale delle registrazioni. In memoria sono disponibili 175 locazioni per la memorizzazione dei dati; se impostiamo un intervallo di un'ora tra una registrazione e la successiva, avremo una copertura di 175 ore di informazioni memorizzate; dopo le prime 175 ore, i dati successivi vengono sovrascritti su quelli più vecchi. Questo richiede di riportare la quantità di dati memorizzati con la frequenza di scaricamento da parte del PC, in base alle necessità effettive di frequenza di campionamento e di utilizzo dei dati meteo, tenendo conto che più frequenti sono le memorizzazioni nel tempo, più lunghi saranno i tempi di scarico, maggiori i volumi di dati da archiviare e più lunghi i tempi di elaborazione, ovviamente a parità di intervallo temporale di lettura. Se vogliamo conservare i dati per utilizzi storici e statistici è necessario prelevare i dati dalla memoria della centralina con una frequenza superiore al tempo di sovrascrittura. Quindi è necessario realizzare un modulo software che venga eseguito a tempo, che scarichi i dati storici dalla memoria della centralina e li archivi su RaspberryPi; visto che dobbiamo scaricare i dati per inviarli al centro meteo di raccolta e pensiamo di farlo ogni quarto d'ora, nello stesso momento scaricheremo anche i dati da storicizzare.

Le centraline della famiglia WS2300 vengono vendute corredate da un software di gestione sviluppato in ambiente Windows. Già negli anni scorsi era emersa

l'esigenza di interfacciare la centralina con hardware più economici ed in particolare i LinuxBox, da qui il nascere delle discussioni nei forum sul modo di documentare il protocollo di comunicazione seriale, proprietario e mai pubblicato dal produttore. Analisi sistematiche delle comunicazioni hanno portato alla ricostruzione di buona parte del protocollo e alla nascita dei primi programmi di interfacciamento in ambiente Linux, poi convogliati nel progetto open source Open2300. Utilizzeremo questo progetto come base per la comunicazione tra la centralina e RaspberryPi e lo potete trovare all'indirizzo <http://www.lavrsen.dk/foswiki/bin/view/Open2300/WebHome>.

Il progetto mette a disposizione una collezione di programmi che permettono le operazioni di comunicazione con la centralina sia richiamandoli da linea di comando, sia da altri programmi; le funzionalità disponibili sono:

- Configurazione della centralina;
- Azzeramento dei contatori di accumulo e statistici;
- Richiesta dei dati correnti con visualizzazione a terminale, oppure salvati su file sequenziali o su una tabella MySQL;
- Richiesta dei dati storici con memorizzazione su file sequenziali o su database MySQL;
- Invio dei dati della centralina ai servizi web di raccolta e divulgazione di dati meteo Weather Underground e/o Citizens Weather, oppure ad altri siti, scrivendo i programmi relativi.

Nel riquadro dedicato ai moduli inclusi nel progetto Open2300 sono descritte le funzioni messe a disposizione dai moduli stessi, con la descrizione del loro utilizzo e la sintassi dei comandi.

Nel riquadro "Listato 1 - Dati meteo dalla WS2355" sono elencati

tutti i dati che possono essere prelevati dalla centralina; il formato è identico sia per i dati correnti, sia per i dati storici. Nel riquadro è visibile il risultato dell'esecuzione del comando "fetch2300", che restituisce i dati correnti in formato collezione, ovvero una serie di righe con due elementi per ciascuna riga, il nome della variabile misurata ed il suo valore, alle quali abbiamo aggiunto una descrizione. Nei prossimi progetti vedremo come utilizzare il formato collezione (o dizionario) con i programmi PHP e Python, che offrono delle istruzioni specifiche per semplificarne molto l'utilizzo. Notiamo che dalla centralina ci vengono restituiti sia dati puntuali, sia valori di accumulo in determinati intervalli di tempo, come le precipitazioni nell'ultima ora, nella giornata e nell'anno, valori minimi, massimi e medie di periodo.

## CONSIDERAZIONI APPLICATIVE

Analizziamo ora le applicazioni che possono essere realizzate attorno alla centralina meteo ed ai dati che possiamo ottenere. Stiamo usando il plurale perché RaspberryPi è dotato di un sistema operativo che permette il controllo e l'esecuzione in contemporanea di diversi programmi sia applicativi, sia di servizio, anche se non propriamente in parallelo. Questo offre l'opportunità di utilizzare i dati della centralina per diversi scopi come la pubblicazione sui siti meteo che presentiamo in questo articolo, oppure per decidere se attivare o meno l'impianto di irrigazione del giardino, o chiudere il lucernario della mansarda se minaccia pioggia o ancora per essere svegliati al mattino con una musicchetta e la successiva comunicazione della temperature e delle condizioni meteo esterne e, perché no, l'accensione automatica della macchina del caffè.

```
Date 2012-Nov-19 - Data delle rilevazione
Time 13:22:01 - Ora della rilevazione
Ti 21.9 - Temperatura nel locale dove è posta la centralina
Tmin 13.5 - Temperatura minima nel locale interno
Tmax 31.3 - Temperatura massima nel locale interno
TTmin 05:55 - Ora rilevamento temperatura minima interna
DTmin 2012-11-19 - Data rilevamento temperatura minima interna
TTmax 00:07 - Ora rilevamento temperatura massima interna
DTmax 2012-08-20 - Data rilevamento temperatura massima interna
To 12.7 - Temperatura esterna
Tmin -0.2 - Temperatura minima esterna
Tmax 39.9 - Temperatura massima esterna
TTmin 06:58 - Ora rilevamento temperatura minima esterna
DTmin 2012-10-29 - Data rilevamento temperatura minima esterna
TTmax 10:20 - Ora rilevamento temperatura massima esterna
DTmax 2012-08-20 - Data rilevamento temperatura massima esterna
DP 6.3 - Temperatura punto di rugiada
Dmin -5.1 - Temperatura minima punto di rugiada
Dmax 22.3 - Temperatura massima punto di rugiada
TDPmin 04:20 - Ora temperatura minima punto di rugiada
DDPmin 2012-11-07 - Data temperatura minima punto di rugiada
TDPmax 09:01 - Ora temperatura massima punto di rugiada
DDPmax 2012-08-19 - Data temperatura massima punto di rugiada
RHi 41 - Umidità relativa locale interno dove è posta la centralina
RHmin 34 - Umidità relativa interna minima
RHmax 66 - Umidità relativa interna massima
TRHmin 16:51 - Ora rilevazione umidità relativa interna minima
DRHmin 2012-09-14 - Data rilevazione umidità relativa interna minima
TRHmax 18:58 - Ora rilevazione umidità relativa interna massima
DRHmax 2012-10-27 - Data rilevazione umidità relativa interna massima
RHo 65 - Umidità relativa esterna
RHmin 15 - Umidità relativa esterna minima
RHmax 97 - Umidità relativa esterna massima
TRHmin 15:44 - Ora rilevazione umidità relativa minima
DRHmin 2012-08-26 - Data rilevazione umidità relativa minima
TRHmax 02:57 - Ora rilevazione umidità relativa massima
DRHmax 2012-11-19 - Data rilevazione umidità relativa massima
WS 0.0 - Velocità del vento
DIRtext SW - Descrizione attuale direzione del vento
DIR0 225.0 - Direzione corrente del vento in gradi
DIR1 247.5 - Ultime cinque variazioni di direzione del vento in gradi
DIR2 270.0 -
DIR3 270.0 -
DIR4 247.5 -
DIR5 202.5 -
WC 12.7 - Wind chill
WCmin -9.1 - Wind chill minimo
WCmax 39.9 - Wind chill massimo
TWCmin 02:55 - Ora rilevazione wind chill minimo
DWCmin 2012-10-28 - Data rilevazione wind chill minimo
TWCmax 10:20 - Ora rilevazione wind chill massimo
DWCmax 2012-08-20 - Data rilevazione wind chill massimo
WSmin 0.0 - Velocità del vento minima
WSmax 2.4 - Velocità del vento massima
TWSmin 13:20 - Ora rilevazione velocità del vento minima
DWSmin 2012-11-19 - Data rilevazione velocità del vento minima
TWSmax 15:26 - Ora rilevazione velocità del vento massima
DWSmax 2012-11-14 - Data rilevazione velocità del vento massima
Rlh 0.00 - Rilevazione accumulo pioggia in un'ora
Rlhmax 4.66 - Accumulo massimo pioggia in un'ora
TRlhmax 17:20 - Ora rilevazione accumulo pioggia massimo in un'ora
DRlhmax 2012-09-26 - Data rilevazione accumulo pioggia massimo in un'ora
R24h 0.00 - Rilevazione accumulo pioggia in 24 ore
R24hmax 15.02 - Accumulo massimo pioggia in 24 ore
TR24hmax 12:53 - Ora rilevazione accumulo pioggia massimo in 24 ore
DR24hmax 2012-09-14 - Data rilevazione accumulo pioggia massimo in 24 ore
Rtot 48.17 - Rilevazione accumulo pioggia totale
TRtot 00:10 - Ora inizio accumulo dati pioggia
DRtot 2000-00-00 - Data inizio accumulo dati pioggia
RP 985.200 - Pressione barometrica
RPMmin 955.600 - Pressione barometrica minima
RPMmax 998.700 - Pressione barometrica massima
TRPmin 17:33 - Ora rilevazione pressione barometrica minima
DRPmin 2012-10-27 - Data rilevazione pressione barometrica minima
TRPmax 23:57 - Ora rilevazione pressione barometrica massima
DRPmax 2012-11-13 - Data rilevazione pressione barometrica massima
Tendency Falling - Tendenza della pressione barometrica
Forecast Rainy - Previsione condizioni meteo
```

La chiave che ci permette di realizzare tutte le applicazioni che abbiamo descritto, e molte altre, è il software Open2300 che ci fornisce le funzionalità di base di comunicazione, di lettura e di gestione dei dati della centralina.

## SCHEDULATORE CRON / CRONTAB

Abbiamo visto in precedenza che la lettura dei dati della centralina ed il loro trattamento avviene ad intervalli regolari. Dal punto di vista pratico significa che ad ogni intervallo di tempo prefissato deve essere attivato un programma che esegue il prelievo e la memorizzazione dei dati, per poi restare inattivo fino al risveglio successivo.

Questa esigenza ci permette di introdurre un servizio fondamentale nei computer dotati di sistema operativo; stiamo parlando dello schedulatore, un processo che gira in background (dietro le quinte, rispetto alla visibilità dell'utilizzatore) e che è in grado di lanciare programmi e comandi ad orari prestabiliti, in base alle impostazioni inserite nel suo file di configurazione. Vedremo nella descrizione dell'implementazione come gestire lo schedulatore in Linux ai fini di questo progetto e di tutti gli altri che vorrete realizzare sulla stessa falsa riga. Esiste una differenza fondamentale tra uno schedulatore ed un programma temporizzato; per esempio nei programmi GPIOServer e GPIOServer2, presentati nei numeri precedenti, abbiamo utilizzato l'istruzione "time.sleep(0.5)" per sospendere l'esecuzione del programma per un certo periodo; con questo metodo l'esecuzione del programma viene effettivamente sospesa ma il programma rimane comunque in memoria, togliendo risorse agli altri processi in esecuzione. I programmi richiamati dallo schedulatore, invece, al termine del processo si chiudono liberando totalmente le risorse di elaborazione fino al richiamo successivo. Come scelta applicativa si tende ad utilizzare lo schedulatore per gestire l'esecuzione di programmi a distanze di tempo abbastanza lunghe ed in momenti prefissati, ad esempio il primo giorno di ogni anno alle ore 00:00:00, mentre la tecnica temporizzata si utilizza quando si vuole campionare una risorsa con una frequenza prestabilita. Esiste una terza tecnica che vedremo nei prossimi articoli che è il polling di condizioni di interrupt, che in ambiente Linux non sono necessariamente cambi di livello su un pin di I/O, ma anche una operazione su un file o un messaggio in arrivo su una porta di comunicazione. Quest'ultima tecnica si utilizza quando si vuole tenere sotto controllo situazioni che dipendono da cause esterne al sistema e che possono accadere in qualsiasi momento.

## REALIZZAZIONE PRATICA

Lasciamo a ciascun utilizzatore e alle raccomandazioni contenute nel manuale d'uso della centralina la scelta di dove e come collocare i sensori esterni alla centralina, raccomandando, ancora una volta, di verificare la buona ricezione dei segnali di comunicazione dati a 433 Mhz.

Collegiamo la scheda RaspberryPi come al solito, con il cavo di rete, l'alimentazione e un hub USB con alimentazione separata - ad esempio quello reperibile presso Futura Elettronica con il codice 8220-PCUSB34 - ricordando che per l'alimentatore è meglio utilizzare un modello da due ampere, con due uscite in formato USB, una delle quali andrà all'alimentazione della scheda RaspberryPi mediante un cavetto microUSB e l'altra sarà utilizzata per alimentare l'hub USB.

Ora che RaspberryPi è in tensione colleghiamoci come abbiamo visto nei precedenti articoli con il

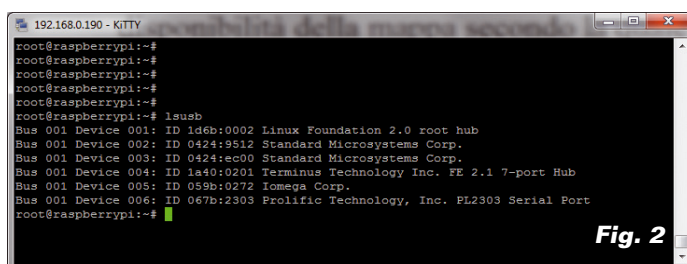


Fig. 2

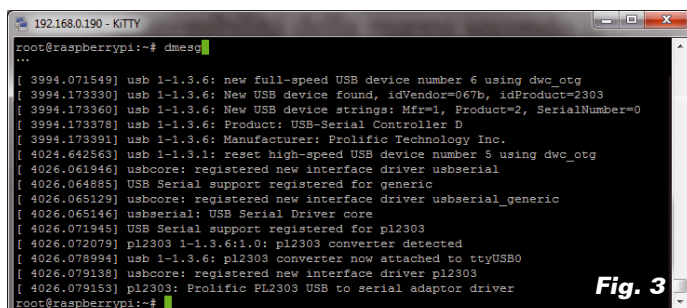


Fig. 3

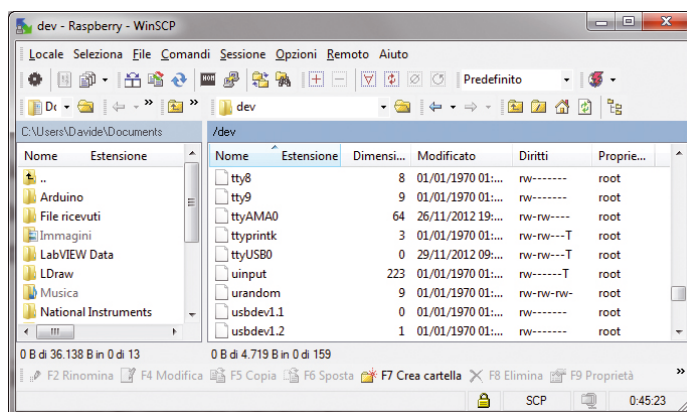


Fig. 4



server SSH utilizzando Kitty o Putty come terminale e WinSCP come explorer per i file; effettuiamo il login con utente root e la password che avete scelto a suo tempo. La centralina, se fornita con il convertitore seriale/USB va collegata, tramite l'apposito cavo seriale, al convertitore e quest'ultimo ha il cavetto USB adatto al collegamento con RaspberryPI. Diversamente fate riferimento al box "Collegamento seriale della centralina". Per curiosità potete verificare che il convertitore seriale/USB appena inserito sia stato correttamente riconosciuto con il comando

*lsusb*

Nella Fig. 2 è possibile vedere che il convertitore è stato riconosciuto come Prolific Technology con chip-set PL2302. Con il comando

*dmesg*

potete verificare in Fig. 3 che al convertitore è stata assegnata la porta seriale virtuale ttyUSB0. Come ulteriore conferma andate con WinSCP nella cartella /dev dove dovrete trovare il file rappresentativo della periferica "ttyUSB0" che corrisponde al nome della porta seriale virtuale assegnata al convertitore e da usare per comunicare con la centralina (un po' come COMxx in Windows) (Fig. 4). Ora, sempre per il motivo che il sistema operativo Raspbian è in continua evoluzione, aggiorniamolo alla versione più recente con i comandi ormai abituali:

*apt-get clean*

*apt-get update*

*apt-get upgrade*

Rispondete "Y" alla eventuale richiesta di utilizzo dello spazio per l'upgrade.



Fig. 5

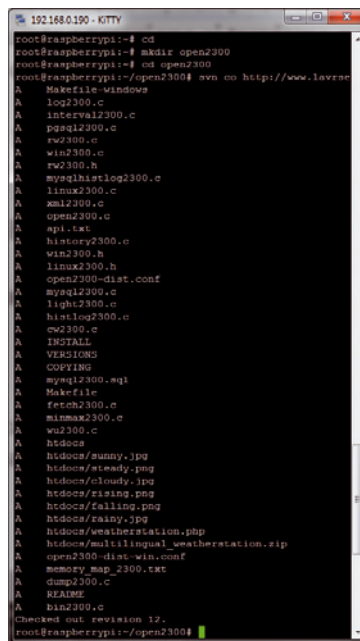
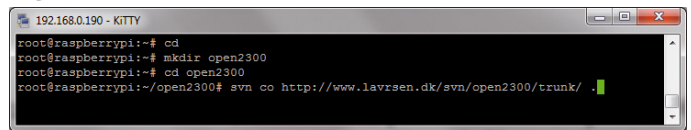


Fig. 6



## INSTALLAZIONE DI OPEN2300

A sistema "fresco" procediamo all'installazione del pacchetto Open2300. Di questo non esiste una versione installabile con il comando apt-get install, con tutti i programmi già compilati, ma esiste unicamente una raccolta dei moduli sorgenti dell'applicazione nel repository svn dell'applicazione raggiungibile all'indirizzo <http://www.lavrsen.dk/svn/open2300/trunk/>.

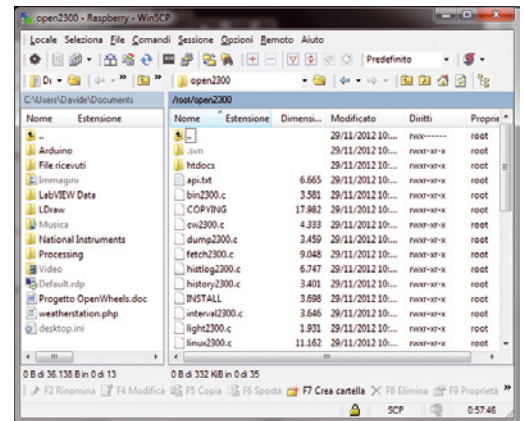


Fig. 7

Fig. 8

Un repository SVN (Subversion) è un sistema di gestione e controllo delle versioni di un'applicazione software con lo scopo di mettere a disposizione degli sviluppatori e degli utilizzatori un unico punto di archiviazione di tutti i moduli e le relative versioni che compongono un progetto.

Nel Subversion vengono archiviate tutte le versioni del software sviluppate nel tempo ed è possibile accedere e scaricare una qualsiasi versione dell'applicazione accedendo al progetto ed al numero di versione desiderato (Fig. 5).

Anche se è possibile scaricare in una propria cartella ciascun singolo modulo della lista, abbiamo preferito proporvi la strada classica, ovvero installare su RaspberryPI il client di Subversion e poi scaricare con quest'ultimo l'intero progetto nella sua ultima versione.

Subversion è un pacchetto che fa parte della distribuzione di Debian, e quindi di Raspbian, ed è installabile utilizzando i soliti comandi:

# Moduli inclusi nella collezione Open2300

Tutte le funzioni disponibili nel progetto Open2300 interagiscono con le registrazioni memorizzate nella centralina attraverso la comunicazione su porta seriale. Sono disponibili sia funzioni che permettono di leggere direttamente dati - come byte e nibble - dalla memoria della centralina, sia funzioni che restituiscono direttamente le informazioni in formato leggibile, un record alla volta, oppure a gruppi di record.

Il nibble è una sequenza composta da quattro bit e corrisponde a mezzo byte; è molto utilizzato nella gestione di registri e di dati binari in linguaggio assembler.

Per avere un esempio delle prime funzioni descritte, riportiamo un frammento della mappa di memoria della centralina meteo; potete consultare per intero la mappa, che è nel file `memory_map_2300.txt` incluso nel percorso di download, interrogandola con le funzioni `open2300`, `dump2300` e `bin2300`.

```
---
TEMPERATURA INTERNA
0346 0    Current Indoor Temperature: BCD offset 30 0.01s
[C]
0347 6    Current Indoor Temperature: BCD offset 30 0.1s [C]
0348 8    Current Indoor Temperature: BCD offset 30 1s [C]
0349 5    Current Indoor Temperature: BCD offset 30 10s [C]
034A 0
034B 0    Minimum Indoor Temperature: BCD offset 30 0.01s
[C]
034C 5    Minimum Indoor Temperature: BCD offset 30 0.1s [C]
034D 5    Minimum Indoor Temperature: BCD offset 30 1s [C]
034E 4    Minimum Indoor Temperature: BCD offset 30 10s [C]
---
0354 0    Time min Indoor Temperature:, minutes BCD 1s
0355 2    Time min Indoor Temperature:, minutes BCD 10s
0356 2    Time min Indoor Temperature:, hours BCD 1s
0357 1    Time min Indoor Temperature:, hours BCD 10s
0358 1    Date min Indoor Temperature:, BCD day 1s
0359 0    Date min Indoor Temperature:, BCD day 10s
035A 1    Date min Indoor Temperature:, BCD month 1s
035B 0    Date min Indoor Temperature:, BCD month 10s
035C 1    Date min Indoor Temperature:, BCD year 1s
035D 0    Date min Indoor Temperature:, BCD year 10s
---
```

A sinistra è indicato l'indirizzo della locazione di memoria, nella seconda colonna il valore rilevato (i dati ricevuti dalla centralina) seguiti dalla descrizione del formato (BCD, ovvero il valore del byte corrisponde alla cifra che rappresenta) e dalla posizione (decine o unità o decimi ...); come potete vedere, i valori vanno letti dall'indirizzo più basso a quello più alto, applicando, nel caso delle temperature il relativo offset adottato per evitare i valori negativi. Iniziamo a leggere la temperatura interna al locale dove è posizionata la centralina, a partire dalla locazione di memoria 0349 (in esadecimale). Qui leggiamo il valore 5. Proseguendo verso l'alto troviamo i valori 8, 6 e 0, cioè 58,60 dal quale dobbiamo togliere l'offset di 30 gradi, ottenendo la temperatura di 28,6 gradi. Le registrazioni diverse dalle temperature non hanno offset; con queste regole siete quindi in grado di leggere e decodificare l'intera mappa di memoria della centralina.

Vediamo ora le funzioni disponibili:

**open2300** - permette di leggere e scrivere da e nelle locazioni di memoria della centralina, gli indirizzi sono in notazione esadecimale come 349 o 35D.

`open2300 address_hex r umber_of_bytes` Legge i valori delle locazioni di memoria da un indirizzo dato per una data lunghezza in byte

`open2300 address_hex w text_hex_string` Scrive un nibble nella locazione di memoria indicata

`open2300 address_hex s bit_number` Setta a 1 il bit nella posizione indicata della locazione di memoria specificata

`open2300 address_hex c bit_number` Setta a 0 il bit nella posizione indicata della locazione di memoria specificata

**dump2300** - Scrive i valori esadecimali, sul file specificato, della porzione di mappa compresa tra gli indirizzi esadecimali indicati nel comando

`dump2300 filename start_address end_address`

**bin2300** - Scrive i valori esadecimali, sul file specificato, della porzione di mappa compresa tra gli indirizzi esadecimali indicati nel comando

`bin2300 filename start_address end_address`

**history2300** - Scrive i dati storici, sul file specificato, nell'intervallo specificato di record con valori esadecimali. Per esempio i valori 1B 3A indicano che si vogliono reperire i dati dal record 27 al 58

`history2300 filename start_record end_record`

**log2300** - Scrive i dati correnti, sul file di log specificato, in formato testo; ogni scrittura viene accodata alle precedenti. Se il file di configurazione è posizionato nella directory indicata all'interno dell'articolo, non è necessario specificarlo nel comando

`log2300 filename config_filename`

Il file di log ha il seguente tracciato:  
Timestamp Data Ora Temp\_interna Temp\_esterna Punto\_di\_rugiada  
Umidità\_relativa\_interna Umidità\_relativa\_esterna Velocità\_del\_vento  
Direzione\_del\_vento\_gradi Direzione\_del\_vento\_testo Wind\_chill  
Pioggia\_1h Pioggia\_24h Pioggia\_total Pressione Tendenza Previsione

*apt-get clean*

*apt-get update*

*apt-get install subversion*

Rispondiamo sempre "Y" alla domanda di continuare e ... aspettiamo. Ora siamo in grado di utilizzare Subversion per scaricare Open2300. Prima creiamo una cartella dove raccogliere i file scaricati, e questa volta utilizziamo la riga di comando dando i seguenti comandi (Fig. 6):

Esempio:

```
20051014081004 2005-Oct-14 08:10:04 27.6 10.2 7.2
35 82 0.0 90.0 E 10.2 0.00 0.00 125.35 1024.600
Falling Cloudy
```

**fetch2300** – Scrive i dati correnti in formato collezione sullo standard output, ovvero in un file se specificato o su video se non viene indicato nulla. La “collezione” è un array a due colonne dove la prima colonna contiene il nome del dato e la seconda colonna il valore corrispondente.

```
fetch2300 config_filename
```

**wu2300** – Interroga la centralina per reperire i dati correnti e li invia al sito Weather Underground.  
<http://www.wunderground.com/weatherstation/index.asp>  
Il sito WU richiede un account per poter accettare i dati inviati. L'identificativo utente e la password devono essere inseriti nel file di configurazione open2300.conf

```
wu2300 config_filename
```

**cw2300** – Interroga la centralina per reperire i dati correnti e li invia al sito di raccolta dell'organizzazione Citizen Weather Observer Program (CWOP).  
<http://www.wxqa.com/>  
CWOP richiede un account per poter accettare i dati inviati. L'identificativo utente e la password ed ulteriori dati per specificare i server di upload da utilizzare devono essere inseriti nel file di configurazione open2300.conf

```
cw2300 config_filename
```

**xml2300** – Scrive i dati correnti in formato XML sullo standard output, ovvero su un file se specificato o su video se non viene indicato nulla.

```
xml2300 xml-filename config_filename
```

**mysql2300** – Scrive i dati correnti sul database MySQL. Richiede di avere installato il server MySQL e configurato il database per raccogliere i dati.

```
mysql2300 config_filename
```

**pgsql2300** – Scrive i dati correnti sul database PostgreSQL.

```
pgsql2300 config_filename
```

**light2300** – Accende e spegne la retroilluminazione dello schermo della centralina

```
light2300 off config_filename    Spegne la retroilluminazione
light2300 on config_filename     Accende la retroilluminazione
```

**histlog2300** – Scrive i dati storici reperiti dalla memoria della

centralina, sul file di log specificato, in formato testo; ogni scrittura viene accodata alle precedenti in modo intelligente e senza ripetizioni. Il file può essere lo stesso del comando log2300 in quanto i dati prodotti hanno il medesimo formato.

```
histlog2300 log_filename config_filename
```

**interval2300** – Legge o imposta l'intervallo di tempo - in minuti - tra una registrazione cronologica e la successiva. Se il valore specificato è 0, il parametro viene letto e non modificato. Fate attenzione al fatto che se specificate un valore diverso da zero cancellerete la cronologia registrata fino a quel momento.

```
interval2300 interval_minutes next_    Imposta l'intervallo
dataset_minutes config_filename
interval2300 0 0 config_filename       Legge l'intervallo corrente
```

**minmax2300** – Reseta i contatori di accumulo e di memorizzazione dei minimi e dei massimi della centralina; alcuni dei comandi hanno l'opzione per azzerare solamente il valore minimo, il valore massimo o entrambi.

```
minmax2300 dailymax config_filename
Azzerare i valori massimi giornalieri (Temp, Umidità, WinChill, Pressione)
```

```
minmax2300 dailymin config_filename
Azzerare i valori minimi giornalieri (Temp, Umidità, WinChill, Pressione)
```

```
minmax2300 timax/timin/tiboth config_filename
Azzerare Temperatura Interna Max/Min/Both
```

```
minmax2300 tomox/tomin/toboth config_filename
Azzerare Temperatura Esterna Max/Min/Both
```

```
minmax2300 dpmax/dpmin/dpboth config_filename
Azzerare Punto di rugiada Max/Min/Both
```

```
minmax2300 wcmox/wcmin/wcboth config_filename
Azzerare Windchill Max/Min/Both
```

```
minmax2300 wmax/wmin/wboth config_filename
Azzerare Velocità del vento Max/Min/Both
```

```
minmax2300 himax/himin/hiboth config_filename
Azzerare umidità interna Max/Min/Both
```

```
minmax2300 homox/homin/hoboth config_filename
Azzerare Umidità Esterna Max/Min/Both
```

```
minmax2300 pmax/pmin/pboth config_filename
Azzerare Pressione Max/Min/Both
```

```
minmax2300 rlmax/r24max config_filename
Azzerare accumulo pioggia max 1h/24h
```

```
minmax2300 r1/r24/rtotal config_filename
Azzerare accumulo pioggia 1h/24h/Total
```

```
cd          si posiziona nella cartella di root
mkdir open2300 crea la cartella open2300 sotto la cartella di root
cd open2300 si posiziona nella cartella open2300 che diventa la cartella corrente
```

Ora scarichiamo il pacchetto Open2300 dal reposi-

tory con il comando subversion:  
`svn co http://www.lavrsen.dk/svn/open2300/trunk/`.  
(attenzione al punto finale).  
Il parametro “co” indica che vogliamo eseguire una operazione di check-out, cioè vogliamo prelevare i moduli



Fig. 9

```

192.168.0.190 - KTTY
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o history2300.o history2300.c
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o history2300.o history2300.o rw2300.o linux2300.o win2300.o -lm
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o histlog2300.o histlog2300.c
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o histlog2300.o histlog2300.o rw2300.o linux2300.o win2300.o -lm
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o bin2300.o bin2300.c
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o bin2300.o bin2300.o linux2300.o win2300.o -lm
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o xml2300.o xml2300.c
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o xml2300.o xml2300.o linux2300.o win2300.o -lm
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o light2300.o light2300.c
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o light2300.o light2300.o linux2300.o win2300.o -lm
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o interval2300.o interval2300.c
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o interval2300.o interval2300.o rw2300.o linux2300.o win2300.o -lm
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o minmax2300.o minmax2300.c
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o minmax2300.o minmax2300.o rw2300.o linux2300.o win2300.o -lm
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o mysql2300.o mysql2300.c rw2300.o linux2300.o -lm -I/usr/include/mysql
-I/usr/lib/mysql -lmysqldclient
gcc -Wall -O3 -DVERSION=\"1.11\" -c -o mysqlhistlog2300.o mysqlhistlog2300.c rw2300.o linux2300.o -lm -I/usr/
include/mysql -I/usr/lib/mysql -lmysqldclient
root@raspberrypi:~/open2300#

```

Fig. 10

```

192.168.0.190 - KTTY
root@raspberrypi:~/open2300# make install
mkdir -p /usr/local/bin
install open2300 /usr/local/bin
install dump2300 /usr/local/bin
install log2300 /usr/local/bin
install fetch2300 /usr/local/bin
install wu2300 /usr/local/bin
install cw2300 /usr/local/bin
install histlog2300 /usr/local/bin
install xml2300 /usr/local/bin
install light2300 /usr/local/bin
install interval2300 /usr/local/bin
install minmax2300 /usr/local/bin
install mysql2300 /usr/local/bin
install mysqlhistlog2300 /usr/local/bin
root@raspberrypi:~/open2300#

```

dall'indirizzo indicato verso il nostro PC. Al termine di questa operazione, come visibile in Fig. 7 e Fig. 8 ci troviamo nella cartella open2300, tutti i file sorgenti in linguaggio C, i file di configurazione, di documentazione ed accessori appena scaricati. Per compilare i programmi che interfacciano il database MySQL è richiesta la libreria di sviluppo per la realizzazione di programmi client MySQL.

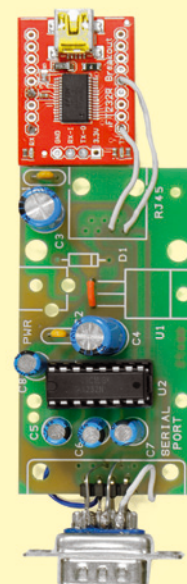
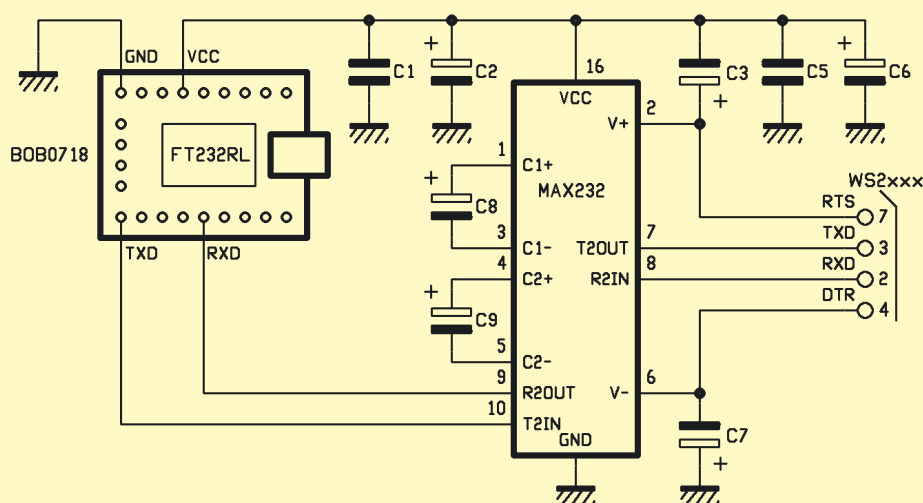
La installiamo nel solito modo:

*apt-get install libmysqlclient-dev*

Al termine dell'installazione, sempre rimanendo nella cartella open2300, diamo il comando:

*make*

## Collegamento seriale della centralina



Come già descritto in altri nostri articoli, le centraline della famiglia WS2300 sono dotate di una porta seriale dalle caratteristiche elettriche non standard. Si nota immediatamente che manca il collegamento di massa e quindi i segnali non lo possono usare come riferimento per la determinazione dei livelli e il produttore non ha rilasciato alcuna documentazione ufficiale su come effettuare il collegamento, tanto che il problema di questo collegamento è un argomento di discussione assai presente nei forum dedicati ed è anche oggetto della domanda di un lettore nella rubrica "Elettronica

Insieme" del numero scorso. La mancanza della massa fa ipotizzare che le due linee DTR e RTS siano prese come riferimento dei livelli logici utilizzati nella comunicazione. In altre parole, nello standard RS232 i livelli logici sulle linee dati sono +12 V ("space", corrispondente allo zero logico) e -12 V ("mark", corrispondente all'1 logico); questo vale per la cosiddetta "RS232 a logica invertita", che è quella normalmente utilizzata nei personal computer. Per quanto riguarda le linee di controllo (DTR, RTS, CTS, DSR ecc.) la logica è al contrario, ossia al livello alto corrispondono +12

V e a quello basso -12 V. Questi livelli di tensione sono riferiti a massa, perciò abbiamo a che fare con una linea di comunicazione di tipo sbilanciato. Oltre a questo, le linee DTR ed RTS consentono, se opportunamente usate, di raddoppiare l'ampiezza delle tensioni della linea dati, aumentando così l'immunità ai disturbi captati dai cavi. Infatti, se TXD della centralina va a +12 V ed RXD va a -12 V, l'ampiezza relativa dell'impulso diventa di 24 V. Questa soluzione per il collegamento seriale è stata probabilmente voluta sia per rendere proprietaria la connessione, sia

per incrementare l'immunità al rumore e poter usare cavi di collegamento più lunghi di quelli consentiti dallo standard RS232, almeno secondo noi. In conclusione, nella centralina La Crosse, affinché la comunicazione con le porte COM standard avvenga correttamente, la linea DTR deve presentare una tensione negativa, mentre la RTS deve essere positiva. Chiaramente, utilizzando con un pc Windows i cavi inclusi nella confezione della centralina e il software in dotazione, in genere non si presentano problemi, mentre cercando di interfacciare la centralina a sistemi diversi,

Fig. 11

che esegue gli script di compilazione contenuti nel file Makefile; il risultato è visibile in Fig. 9. Se non si verificano errori, per ogni file sorgente .c troveremo il corrispondente programma compilato (senza suffisso). Ora installiamo i moduli nella cartella appropriata con il comando:

*make install*

che esegue lo script contenuto nel file Makefile con label

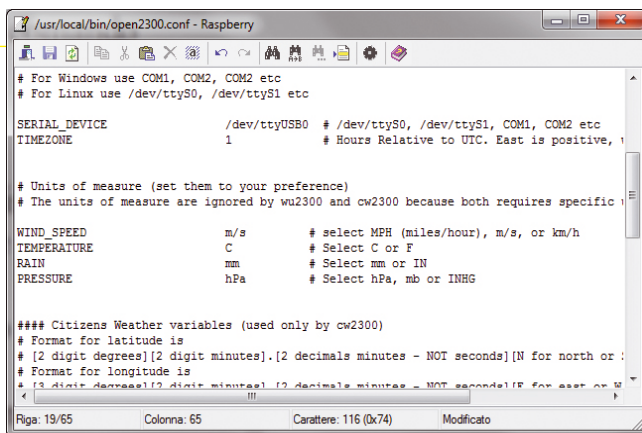


Fig. 12

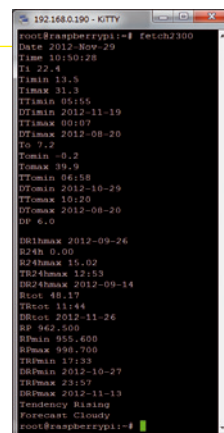
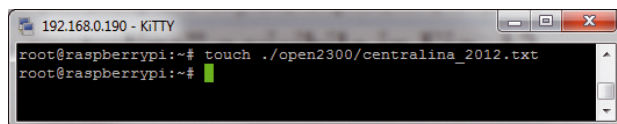


Fig. 13



come Linux e open2300, la situazione si complica. I prodotti più recenti dispongono di convertitori seriale/USB che funzionano perfettamente anche nella nostra configurazione. I prodotti più vecchi, invece, possono avere nella confezione un convertitore o meno, ma comunque non sono in grado di funzionare con una soluzione basata su un normale convertitore da Seriale a USB (anche quello fornito). Per risolvere alla radice il problema di questi prodotti, abbiamo realizzato un convertitore a "doppio stadio" per soddisfare le esigenze sia lato centralina, sia lato Linux. Dal lato Linux, nel kernel di Raspbian e di Debian in generale è disponibile il driver ftdi\_sio, dedicato ai convertitori USB/FTDI con chip FT232RL, quindi abbiamo realizzato un

convertitore ad-hoc composto dal convertitore USB/FTDI di Sparkfun BOB0718, cablato in serie al convertitore TTL/RS232 ottenuto con l'hacking del Kit FT475 (entrambi acquistabili presso Futura Elettronica con i codici 7300-BOB0718 e 7100-FT475).

Come visibile dallo schema elettrico, per soddisfare le caratteristiche di connessione dalla centralina abbiamo prelevato la tensione di circa +10 V e -10 dai piedini di uscita dei servomotori, presenti nell'integrato MAX232, e li abbiamo portati rispettivamente ai piedini 7 e 2 del connettore seriale maschio, collegato allo stampato FT475. Questa soluzione si è dimostrata funzionare perfettamente con centraline vecchie e nuove (Vedi Fig. A e Fig. B).

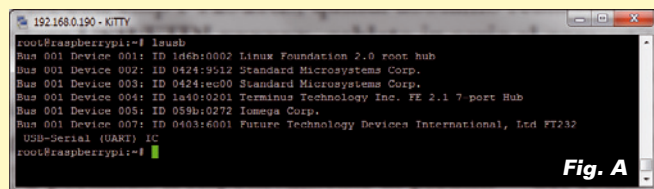


Fig. A

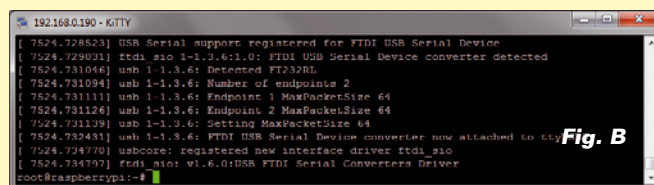


Fig. B

install; il risultato è visibile in Fig. 10.

Vedrete che i moduli compilati vengono copiati nella cartella /usr/local/bin, che è la cartella da dove verranno richiamati al momento dell'esecuzione. La maggior parte dei programmi necessitano, per poter essere eseguiti, delle informazioni contenute nel file di configurazione open2300.conf. Inizialmente questo file non esiste e lo dobbiamo creare copiando il file open2300-dist.conf nella cartella /usr/bin/local, con il nome open2300.conf; per fare questo utilizziamo il comando:

*cp open2300-dist.conf /usr/local/bin/open2300.conf*

che copia il file nella cartella di destinazione cambiandogli contemporaneamente il nome.

Ora apriamo il file /usr/local/bin/open2300.conf con doppio click sul nome del file in WinSCP e modifichiamo la riga "SERIAL\_DEVICE" di definizione della porta seriale, sostituendo il riferimento alla porta seriale con ttyUSB0 come visibile in Fig. 11. Salviamo, richiudiamo e passiamo a fare qualche prova. Iniziamo col comando

*fetch2300*

che ci restituisce a linea di comando i dati correnti della centralina, come descritto nel riquadro "Listato 1 - Dati meteo dalla WS2355" e visibile in Fig. 12. Ora creiamo un file, dove accumulare i dati storici scaricati dalla centralina, nella cartella open2300 e che chiameremo centralina\_2012.txt con il comando di Fig. 13:

*touch ./open2300/centralina\_2012.txt*

Possiamo utilizzare il comando:

*log2300 ./open2300/centralina\_2012.txt*

Fig. 14

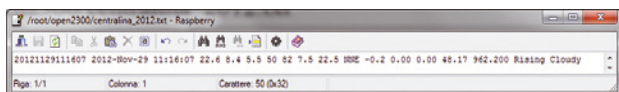


Fig. 15

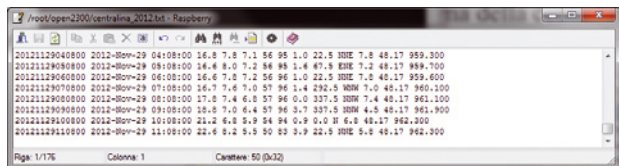


Fig. 16

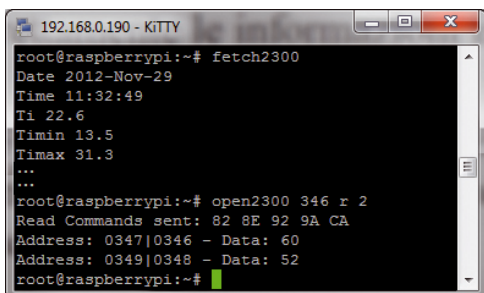


Fig. 17

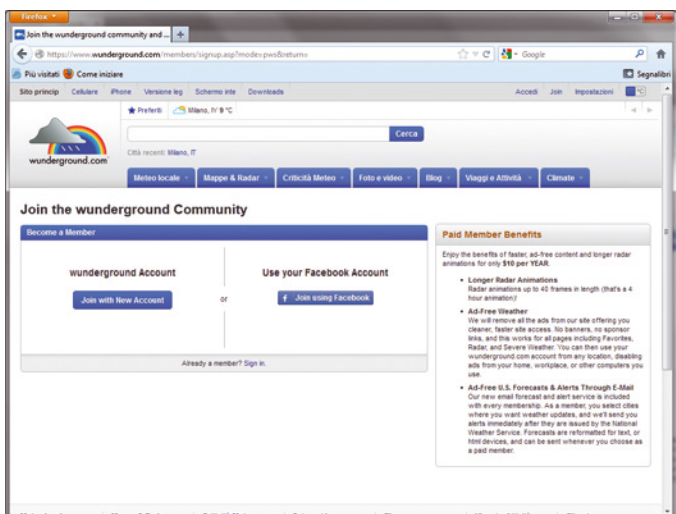
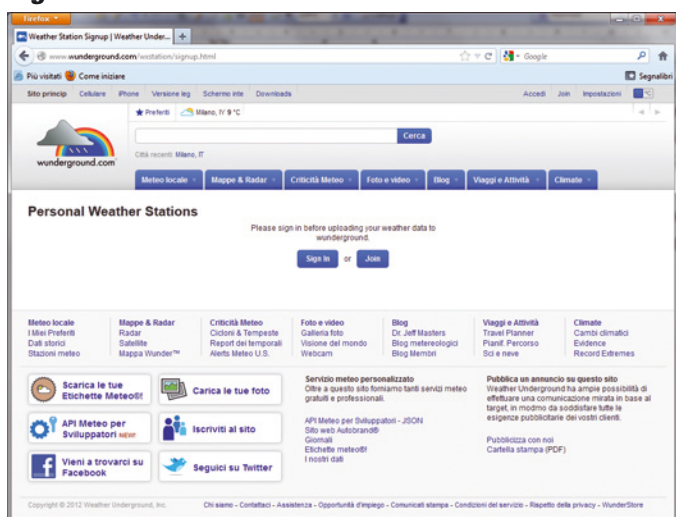


Fig. 18

per trasferire i dati correnti della centralina nel file indicato, con il risultato visibile in Fig. 14. Il comando

`histlog2300 ./open2300/centralina_2012.txt`

trasferisce nello stesso file i dati storici della centralina in modo intelligente; la prima volta che viene richiamato, questo scarica tutti i dati presenti nella memoria, mentre nei richiami successivi scarica solo i dati nuovi rispetto a quelli contenuti nel file. Un frammento del risultato è visibile in Fig. 15.

Date le prestazioni della scheda RaspberryPi, non è consigliabile utilizzare i moduli che trasferiscono i dati su database MySQL, se non adottando degli accorgimenti che vi proporremo in un prossimo articolo. Infine, possiamo leggere i contenuti della memoria interna della centralina con comandi come:

`open2300 346 r 2`

che significa leggere due byte a partire dall'indirizzo 346, che noi sappiamo essere quello relativo alla misura corrente della temperatura interna con un offset di 30 gradi. L'offset per le misure di temperatura è stato scelto come soluzione per poter gestire valori fino a 30 gradi sotto lo zero senza ricorrere a un segno. Se guardiamo la Fig. 16 vediamo la lettura dei byte confrontata con la lettura "in chiaro" dei dati della centralina, dalla lettura in byte si vede il valore 52 per i gradi e 60 per i decimali. Sottraendo l'offset di 30 gradi si ottiene esattamente il valore di  $T_i = 22,6$ .

Con lo stesso metodo e le informazioni fornite nel riquadro "Moduli inclusi nella collezione open2300" potete analizzare tutta la mappa di memoria della centralina.

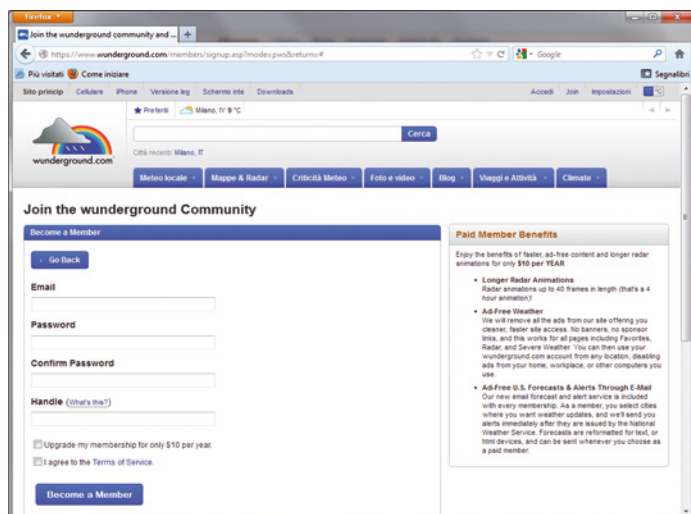


Fig. 19



**Personal Weather Stations**

Le tue stazioni ufficiali

ID stazione: Localizzazione: Link alla fonte: Attiva

ILOMBARD54 Via Adige, 11, Gallarate (VA), LOMBARDY, ITALY Futura elettronica Yes Modifica Cancellazione

ILOMBARD54 Status Message: No message - Edit

Nota: Nel caso tu voglia spostare la tua stazione meteo di qualche isolato rispetto alla posizione attuale, dovrai richiedere un nuovo ID-Stazione anziché modificare le informazioni associate all'ID già esistente.

Nuova stazione meteo - Upload an image of Your Station

**Aggiungi una stazione**

STEP 1. Inserisci le tue informazioni per trovare la tua posizione nella mappa sulla destra. Fields in bold are required to be filled in.

Nome del proprietario: Bona Landoni

Indirizzo: Via Adige, 11

Città: Gallarate (VA)

Organizzazione: Futura Elettronica srl

Vicinanze:

Zona oraria stazione: Europa/Roma

Provincia/Stat: LOMBARDY

Paese: ITALY

Codice postale: 21013

Altitudine (piedi): 791

Traccia località

Latitude: 45.8480054 (N o S 37.775 or N 37.45 36.375)

Longitude: 8.01104374 (E o W 122.391 or W 122.23)

Carica la mappa per l'ID

ID MACID: (If this station is a MACID station, please include the MACID of here)

Link a testo: Futura Elettronica (the title or name of your website)

Link a URL: www.futura-shop.it (the URL of your website)

Tipi di stazione: RaspberryPi

Associated WebCam:

Receive PWS notifications via E-mail: ☐

Receive PWS community newsletter via E-mail: ☒

STEP 3. Dopo aver controllato la cartina intorno alla località della tua stazione, ed esserti assicurato che tutte le informazioni geografiche sono corrette, puoi inviare i dati cliccando sul pulsante "Invia stazione".

Invia / Annulla

**What Happens After Signup?**

Once you have a station ID and password you need to use them in your weather station software. The password for each of your stations is the password you use to log in to our site.

Please Note: Make sure you enter the ID in all capitals, and the password exactly as you choose it, both fields are case sensitive.

Make sure that the software you are using supports the feature.

Even if you already have a supported package, make sure that you have a new enough version that supports the feature.

**Detailed Instructions**

- Raspberry Pi 100 Setup
- WeatherStation
- Virtual Weather Station
- Weather Display
- WeatherView 32
- WxView32
- WxViewLink
- WxView

Upload a Station Image

You can now upload an image of each station you own and have that image appear on your station's history pages.

- Click by Site Instructions
- Upload Your Image

**Troubleshooting**

My Data On The Forecast Page?

I have about 1 day for our database to update all of our web servers and post your station on your local forecast page. Also, if we don't have current data from your station, that can be caused by a number of things. Let us know if we get the next update. Also, make sure you have correctly entered your ID - all capitals. The password is case sensitive too.

I'm Still Not Seeing My Data!

Make sure that your computer's system clock and your weather station's clock are set to the correct time in the current time zone. If the time is incorrect you will be sending wrong data instead of live data. Otherwise contact support.

This Page Doesn't Work!

If you're having trouble with the new sign up page, you should contact support.

Fig. 20

## PUBBLICAZIONE DATI SU WEATHER UNDERGROUND

Per poter trasferire i dati della nostra centralina ad un sito di raccolta e pubblicazione dati meteo dobbiamo creare un account sul sito stesso, descrivere la posizione della nostra centralina e rispettare il protocollo di comunicazione richiesto dal sito per ricevere ed utilizzare i dati in arrivo dalla nostra centralina. Ecco la procedura per registrarsi ed utilizzare il sito Weather Underground.

La prima operazione è creare un'utenza nel sito di Weather Underground andando all'indirizzo <http://www.wunderground.com/wxstation/signup.html> e selezionando l'opzione Sign Up, come visibile in Fig. 17. Si ottiene la pagina di Fig. 18 dove scegliamo "Join with New Account". Nella pagina di Fig. 19 inseriamo un nostro indirizzo mail e una password che utilizzeremo anche in futuro per accedere alla centralina. Dopo aver inserito i dati e aver seguito le istruzioni per la registrazione ci verrà presentata la pagina "Personal Weather Stations" dove potremo inserire tutti i dati relativi alla nostra o alle nostre centraline. Date "Invio" per confermare i dati, ottenendo una pagina simile a quella di Fig. 20. Da questa pagina dovete prendere e annotare il nome che viene assegnato alla

```

/usr/local/bin/open2300.conf - Raspberry

APRS_SERVER rotate.aprs.net 14580 # These are the APRS servers and ports for
APRS_SERVER first.aprs.net 14580 # Citizens Weather reporting.
APRS_SERVER second.aprs.net 14580 # They they are tried in the entered order
APRS_SERVER third.aprs.net 14580 # you may enter up to 5 alternate servers

#### WEATHER UNDERGROUND variables (used only by wu2300)

WEATHER_UNDERGROUND_ID ILOMBARD54 # ID received from Weather Underground
WEATHER_UNDERGROUND_PASSWORD ***** # Password for Weather Underground

### MySQL Settings (only used by mysql2300)

MYSQL_HOST localhost # Localhost or IP address/host name
MYSQL_USERNAME open2300 # Name of the MySQL user that has access to t
MYSQL_PASSWORD muen12300 # Password for the MySQL user

Riga: 52/65 Colonna: 41 Carattere: 32 (lx20)

```

Fig. 21

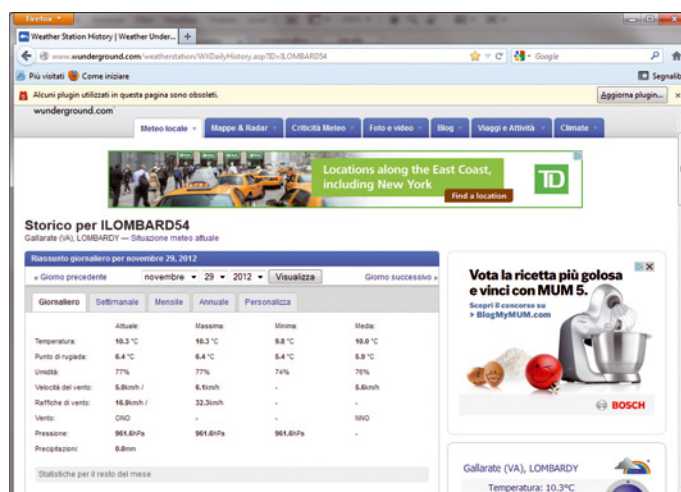


Fig. 22

centralina perché dovrà essere utilizzato come uno dei parametri del messaggio di invio dei dati dalla centralina stessa. Ora possiamo uscire dalla sessione con il sito di Weather Underground e configurare il software su RaspberryPI per comunicare con esso. La configurazione consiste nell'aprire una seconda volta il file open2300.conf e configurare la voce WEATHER\_UNDERGROUND\_ID con il nome assegnato alla centralina (nel nostro caso ILOMBARD54) e il campo WEATHER\_UNDERGROUND\_PASSWORD con la password che abbiamo scelto per la registrazione al sito, come visibile in Fig. 21. Come prova lanciamo dalla linea di comando il programma

wu2300

e, al termine, andiamo a vedere il risultato sul sito di Weather Underground all'indirizzo <http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=ILOMBARD54> sostituendo il nome della nostra centralina ILOMBARD54, dopo la variabile ID=, con il nome della vostra. Dovreste trovare i dati appena inviati, come visibili in Fig. 22.

Il programma wu2300, del quale potete analizzare il

# Installazione sul mini PC MK802 con Linux

Per esercizio e per dimostrare la portabilità delle applicazioni sulle diverse piattaforme, purché abbiano installato il sistema operativo Linux, abbiamo applicato tutto ciò che abbiamo descritto nell'articolo anche su un Mini PC MK802.

Per far partire Linux su MK802 è sufficiente caricare l'immagine della distribuzione Linux su una scheda SDCard tramite un computer e poi inserirla nello slot SDCard di MK802.

Il firmare del Mini PC MK802 è predisposto per effettuare il boot dalla memoria Flash interna su cui risiede Android, oppure, nel caso sia inserita una SDCard, il boot viene eseguito da quest'ultima e può determinare vari risultati, fra cui l'aggiornamento di Android.

Nel nostro caso abbiamo utilizzato una distribuzione minimale di Debian Wheezy (lo stesso di RaspberryPi) compilata per il processore Allwinner A10. Questa è reperibile nel sito <http://romanrm.ru/en/a10>

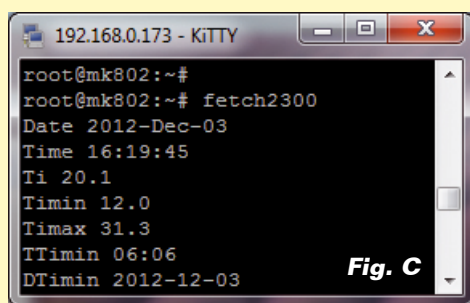
Abbiamo in questo caso una distribuzione headless, cioè senza interfaccia grafica, ma predisposta al collegamento con Kitty e WinSCP mediante il collegamento SSH. La versione è stata arricchita con i moduli server, sempre reperibili nello stesso sito. In pratica è configurata in modo quasi identico al sistema operativo di RaspberryPi.

L'unico elemento aggiuntivo è un convertitore USB/Ethernet per collegare il Mini PC MK802 alla rete locale direttamente via cavo, in quanto MK802 è dotato solo di collegamento WiFi.

Noi abbiamo utilizzato il convertitore Velleman "Interface 2.0 USB Lan" disponibile da Futura Elettronica, e lo abbiamo collegato mediante un Hub USB con alimentazione autonoma.

Per interfacciarsi con la centralina abbiamo usato il convertitore "casalingo" descritto nel riquadro "Collegamento seriale alla centralina", collegandolo ancora all'Hub USB.

A questo punto basta ripetere tutte le operazioni descritte in questo articolo per ottenere esattamente lo stesso risultato, come visibile in Fig. C.



```
192.168.0.173 - KITTY
root@mk802:~#
root@mk802:~# fetch2300
Date 2012-Dec-03
Time 16:19:45
Ti 20.1
Tmin 12.0
Tmax 31.3
TTmin 06:06
DTmin 2012-12-03
```

Fig. C

sorgente come per tutti gli altri programmi, per inviare i dati al sito Weather Underground, prepara una stringa di dati che contiene tutte le informazioni da inviare al sito nel formato richiesto, come nell'esempio visibile in Fig. 23.

Il formato è quello standard delle request HTTP, ovvero una sequenza di coppie "variabile=valore" concatenate dal carattere "&". L'invio avviene con la creazione di una connessione al sito di destinazione (socket) che utilizza il nome specificato nella riga

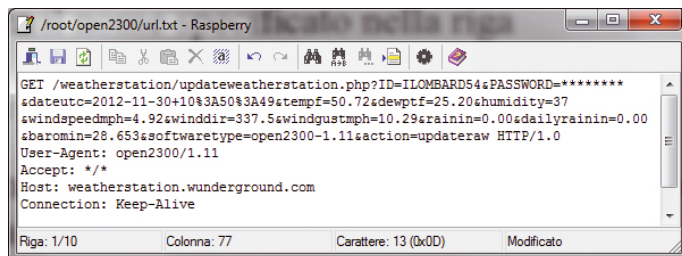


Fig. 23

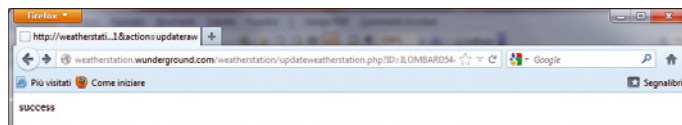


Fig. 24

Host: *weatherstation.wunderground.com*

Una volta stabilita la connessione viene inviata l'intera stringa con il metodo GET, ovvero come se fosse scritta di seguito al nome del sito nella riga di comando di un browser normale.

In pratica, il metodo di invio simula l'invio da un browser dove nel campo URL viene inserito prima il nome host (campo in giallo 1 nella Fig. 23) e poi la stringa di richiesta (campo in giallo 2 nella Fig. 23). Se componete nel browser la riga in questo modo otterrete in risposta "success" e l'aggiornamento dei dati nel sito (Fig. 24).

## SCHEDULAZIONE AGGIORNAMENTI A TEMPO

Introduciamo ora il servizio di schedulazione cron che ci permetterà di inviare una nuova serie di dati al sito Weather Underground agli orari stabiliti.

Il servizio cron è il servizio predefinito in Linux per la gestione dell'esecuzione di programmi e comandi a tempo.

Per funzionare, cron si appoggia a una tabella che

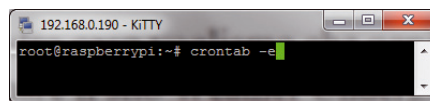


Fig. 25

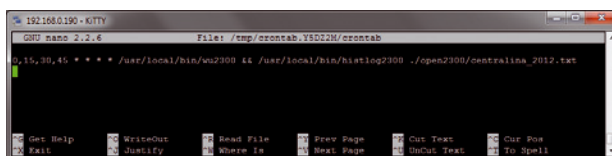


Fig. 26

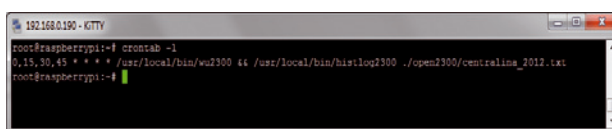


Fig. 27

contiene le "istruzioni" circa i programmi da eseguire e in quali istanti lanciaarli. La tabella utilizzata da cron si gestisce con la funzionalità crontab che ha la sintassi:

*crontab <opzione>* 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 in Fig. 25:

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 es: 0,3,7,9 come un intervallo es: 3-7 oppure un misto dei due es: 0,3,4-6,9.

Nel nostro caso vogliamo che ogni quarto d'ora vengano eseguiti i programmi wu2300 per aggiornare il sito Weather Underground ed il programma hystlog2300 per scaricare i dati storici della centralina e accodarli al file centralina\_2012.txt che abbiamo creato in precedenza. Nella finestra che ci mostra il contenuto del file cancelliamo tutto il contenuto e scriviamo in una sola riga come in Fig. 26:

0,15,30,45 \* \* \* \* /usr/local/bin/wu2300 && /usr/local/bin/histlog2300 ./open2300/centralina\_2012.txt

che significa: far eseguire i due programmi in sequenza allo scadere di ogni quarto d'ora di ogni ora, giorno e mese. 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 quarto d'ora verranno lanciati i due programmi con gli aggiornamenti conseguenti. Per modificare le impostazioni di cron richiamiamo la funzionalità

*crontab -e*

Per vedere le impostazioni utilizziamo l'opzione (Fig. 27):

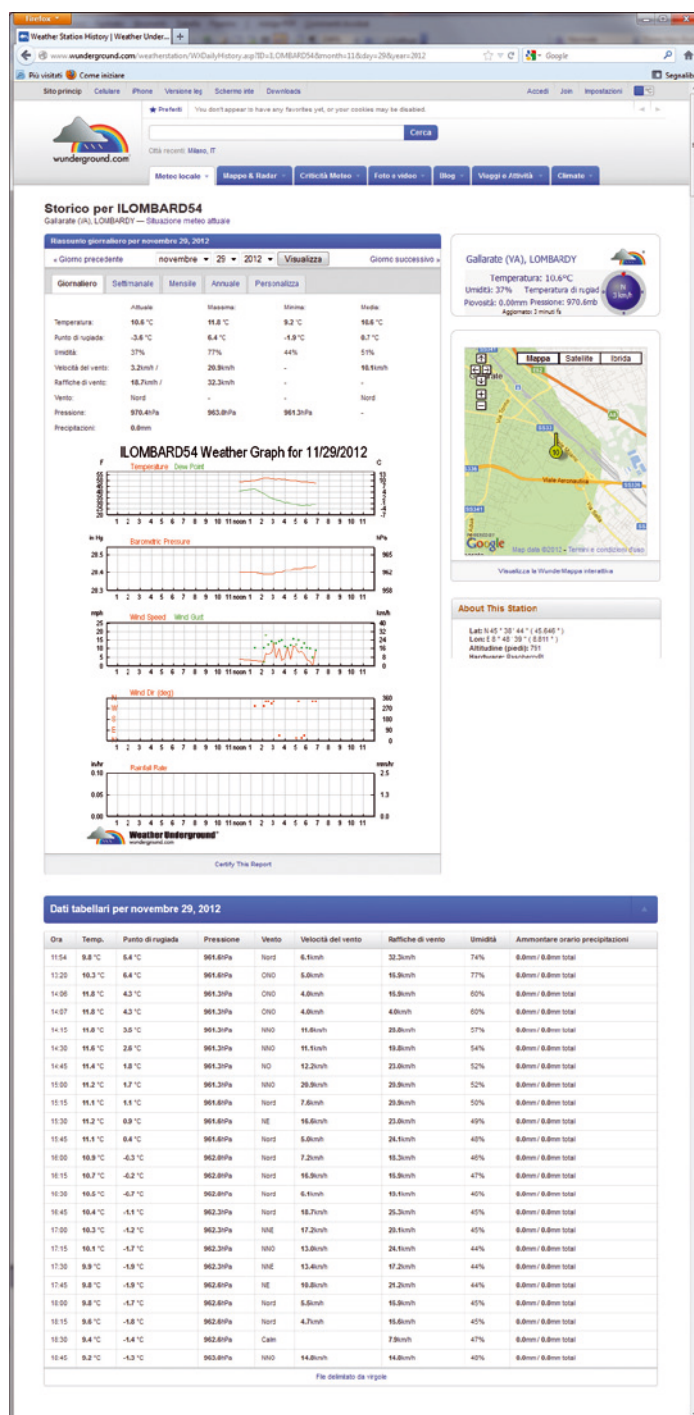


Fig. 28

*crontab -l*

mentre per interrompere l'esecuzione automatica digitiamo

*crontab -r*

Nella Fig. 28 è visibile la pagina del sito di Weather Underground dopo un certo numero di ore di funzionamento automatico.



Mensile di elettronica applicata, attualità scientifica, novità tecnologiche.

# Elettronica In

[www.elettronica.in.it](http://www.elettronica.in.it)

oltre l'elettronica



# SHIELD DI ESPANSIONE ADC PER RASPBERRYPI COMPATIBILE CON ARDUINO

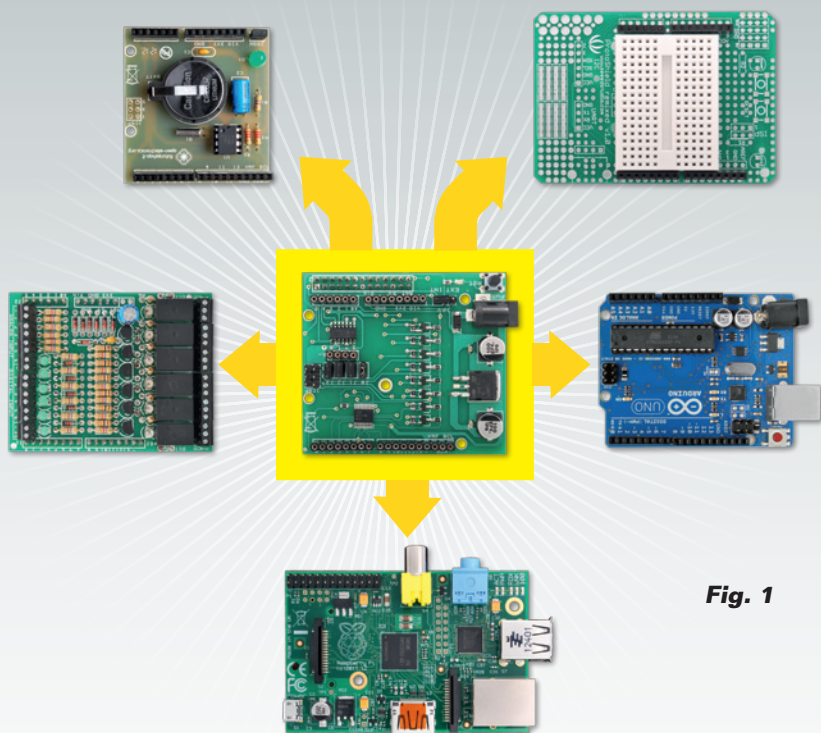


di MARCO MAGAGNIN

Esandiamo le funzionalità di RaspberryPi con uno shield compatibile con la piedinatura di Arduino e con a bordo un convertitore analogico digitale a 16 bit di risoluzione. Ulteriori espansioni che presenteremo in futuro permetteranno di aggiungere un convertitore DAC, una serie di I/O digitali aggiuntivi e la possibilità di collegarsi anche a schede e PC sprovvisti di GPIO.

**C**ontinuiamo la nostra serie di appuntamenti legati al mondo RaspberryPi ed alle sue applicazioni tornando ad occuparci più approfonditamente delle funzionalità della periferica GPIO. In questo e nei prossimi articoli vi proponiamo una serie di dispositivi che permettono diverse possibilità di integrare periferiche e dispositivi hardware sia collegandoli al GPIO della scheda RaspberryPi, sia ad altri microPC non dotati di GPIO ed anche ai comuni PC portatili e desktop.

Nella Fig. 1 potete vedere un'immagine d'insieme delle integrazioni possibili con la scheda che vi presentiamo in questo articolo, uno shield di espansione ADC che permette di aggiungere a RaspberryPi la possibilità di eseguire letture da sensori con uscita analogica. Come vedremo nel prosieguo dell'articolo, questo shield può essere alimentato sia a 12V che a 5V ed esegue



**Fig. 1**

la conversione dei livelli di tensione da 3,3V a 5V per i pin digitali del GPIO e per le linee di comunicazione dei bus i2C, SPI e seriale. Inoltre questo shield possiede sia un connettore adatto ad inserirsi nel connettore GPIO di RaspberryPi sia le strip che corrispondono ai connettori tipici dei microcontrollori della famiglia Arduino e dei relativi shield di espansione.

Con questo shield potremo utilizzare con RaspberryPi molti degli shield già disponibili per Arduino, ed anche per mettere in comunicazione la scheda RaspberryPi con lo stesso microcontrollore Arduino. Nel prossimo futuro vi proporremo un ulteriore shield con a bordo un integrato DAC ed un integrato di espansione I/O con sedici pin, entrambi pilotati via i2C.

Per permettere anche ai dispositivi Linux non dotati di GPIO come i microPC Mk802, Mk802 III ed anche normali PC, di utilizzare periferiche di ingresso/uscita analogiche e digitali

vi proporremo dei convertitori USB/i2C e USB/SPI che potranno essere utilizzati con questo shield.

Chiaramente completeremo questa panoramica con esempi software per utilizzare questa architettura nei vari linguaggi e con i diversi dispositivi Linux.

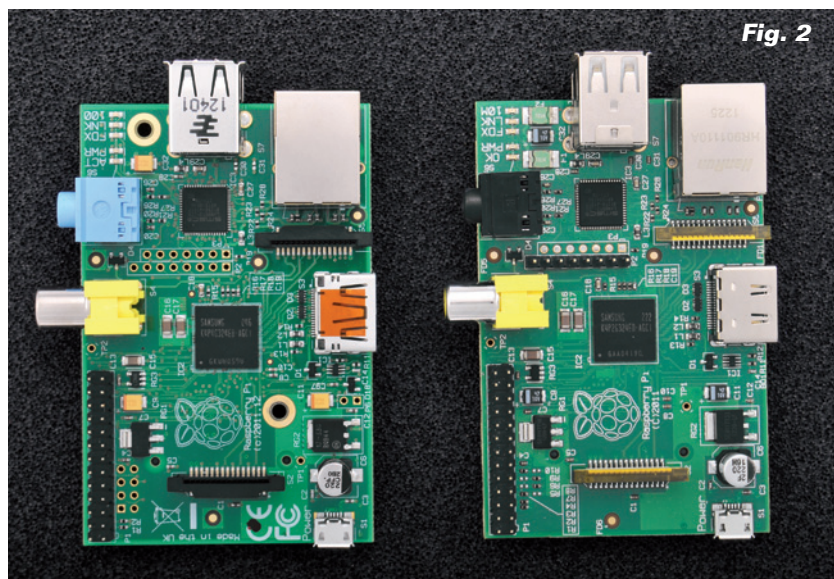
Prima di iniziare la descrizione dello schema elettrico dello shield è però necessario

tenere conto ancora una volta delle modifiche e degli aggiornamenti che si susseguono in un mondo in evoluzione accelerata come quello legato al sistema operativo Linux e che non risparmia la nostra RaspberryPi.

Quella attualmente in vendita ed in distribuzione è la rev. 2 della versione B di RaspberryPi, caratterizzata principalmente, rispetto alla rev. 1, dall'aumento della memoria RAM da 256 a 512 Mb e da alcune variazioni ai riferimenti di I/O del connettore GPIO. Per verificare il modello di RaspberryPi in vostro possesso potete fare riferimento alla **Fig. 2**; a destra la scheda rev. 1, a sinistra la scheda rev. 2, riconoscibile, tra l'altro, dai due evidenti fori di fissaggio.

Nella **Fig. 3** sono visibili le piedinature del connettore GPIO nelle due versioni della scheda; le principali differenze sono i riferimenti agli I/O digitali per i pin 3, 5, 7, 13 e per i pin del bus i2C che nella rev. 1 riportano le linee del bus(0) e nella rev. 2 quelle del bus(1).

Vediamo ora la descrizione del



**Fig. 2**



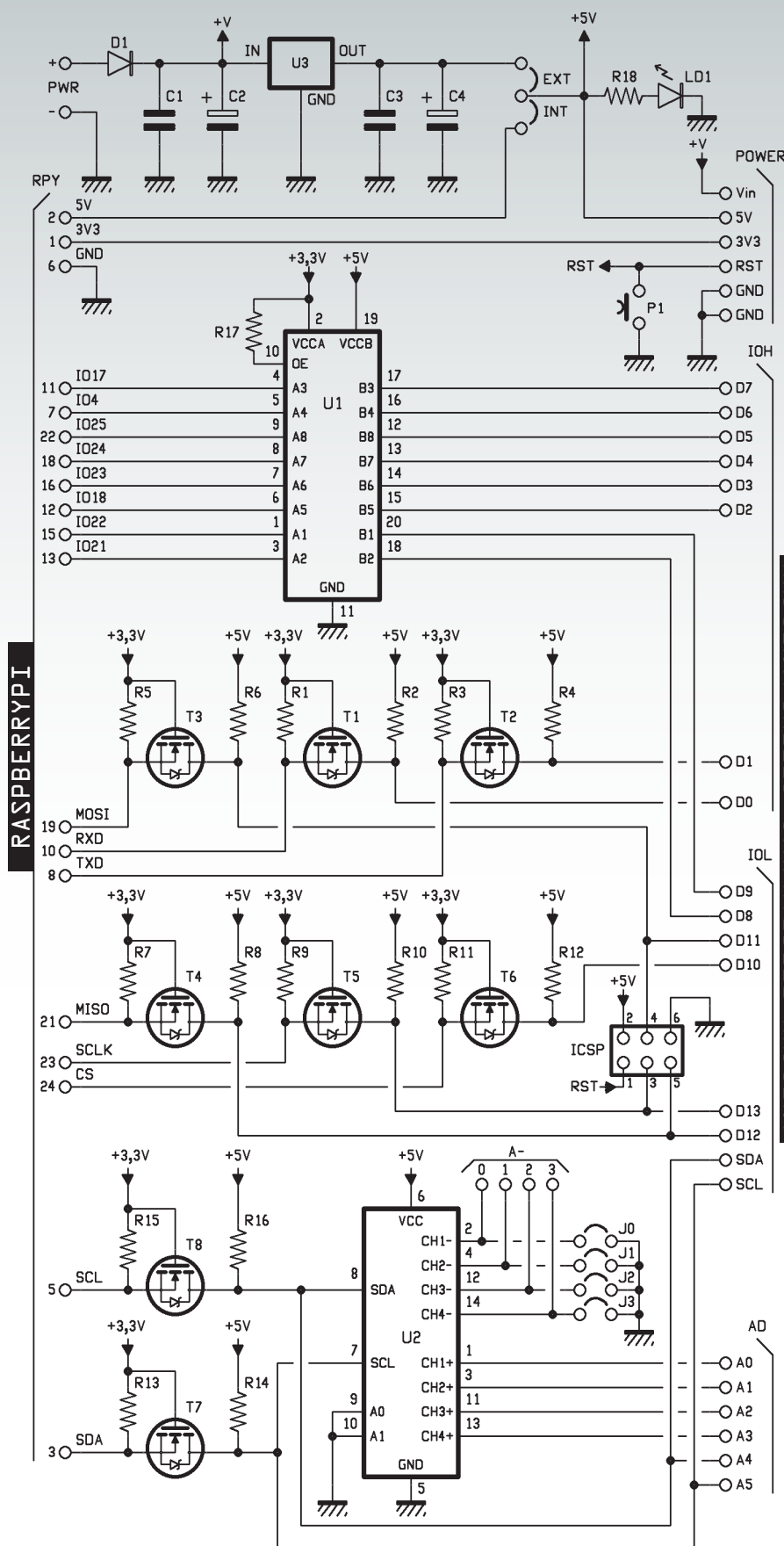
circuito elettrico dello shield e poi gli aspetti software per l'utilizzo dello stesso.

### SCHEMA ELETTRICO

Lo shield di integrazione del GPIO è stato progettato per offrire le seguenti possibilità:

- Estendere le funzionalità del GPIO con un convertitore ADC a quattro canali singoli o differenziali, direttamente sulla basetta e, con un ulteriore shield, un'espansione a ulteriori 16 I/O digitali ed un convertitore DAC a 4096 passi, entrambi su bus i2C;
- Alimentazione diretta dello shield prelevando i 5V da RaspberryPi oppure con alimentatore esterno sia a 5V sia a 12V per compatibilità con gli shield di Arduino;
- Conversione dei livelli da 3,3V a 5V, e viceversa, degli I/O digitali, dei bus i2C, SPI e della porta seriale, per rispettare sia i requisiti elettrici di RaspberryPi che dei sensori e dispositivi esterni;
- Possibilità di utilizzare il convertitore ADC sia con gli ingressi in modalità differenziale che lineare;
- Connettore a 26 pin per il collegamento con il connettore GPIO di RaspberryPi;
- Connettori per il collegamento diretto ad una scheda Arduino e agli shield previsti per quest'ultima;
- Possibilità di collegarsi a convertitori aggiuntivi esterni USB/i2C e USB/SPI.

Analizziamo lo schema elettrico per verificare come i requisiti posti sono stati realizzati in pratica. Iniziamo dalla alimentazione costituita dalla classica configurazione centrata sul regolatore 7805. I condensatori filtrano e stabilizzano la tensio-





**Fig. 3**

ne in uscita. I ponticelli EXT e INT permettono di scegliere se ricevere l'alimentazione a 5V dal pin 2 del connettore GPIO oppure dall'alimentazione esterna tramite il regolatore 7805. In questo modo è possibile evitare un'alimentazione esterna, quando non sia necessario. Il pin 2 di RaspberryPi è in grado di erogare 500 mA nella rev. 1 e 300 mA nella rev. 2. Per shield e sensori con assorbimenti maggiori o per shield che richiedono tensioni maggiori come lo shield relè, è necessario utilizzare un alimentatore esterno. Veniamo ora ai circuiti di conversione dei livelli, che sono realizzati con due approcci diversi. La conversione dei livelli dei pin di I/O digitale è realizzata utilizzando l'integrato TBX0108 un traslatore di livelli ad 8 bit con due linee di alimentazione, del quale trovate la descrizione nel riquadro dedicato. Sul lato A dell'integrato sono collegate le linee provenienti dal connettore GPIO di RaspberryPi indicato nello schema con RPY, sul lato B le linee collegate ai connettori IOL, IOH, POWER e AD, che mappano i rispettivi connettori I/O di Arduino. Il terminale GND è collegato a massa, i terminali  $V_{CCA}$  e  $V_{CCB}$  sono i riferimenti di tensione per la conversione dei livelli e sono collegati alle linee di tensione, rispettivamente a 3,3V e 5V, il terminale OE (Enable), quando è posto a livello alto, abilita l'operatività dell'integrato ed è collegato al livello  $V_{CCA}$  (tensione 3,3V). Nella tabella di **Fig. 4** sono riportate le corrispondenze tra i pin del GPIO lato RaspberryPi ed i corrispondenti pin sui connettori Arduino; le due sezioni della tabella riportano le corrispondenze sia per la

configurazione della rev. 1 sia della rev. 2. Nell'integrato TBX0108 la conversione dei livelli funziona in entrambe le direzioni in modo automatico, in quanto l'integrato stesso è in grado di determinare la direzione del flusso dati per ciascun pin di I/O (autosensing). Questa caratteristica, utile per la conversione dei livelli dei pin di I/O digitale rende poco affidabile l'impiego dello stesso integrato per la conversione dei livelli delle linee del bus i2C, in quanto dalla parte di RaspberryPi le linee del bus i2C sono dotate di resistenze di pull up, che interferiscono con la funzione di autosensing dell'integrato TBX0108 (vedi riquadro dedicato). Per le linee del bus i2C, del bus SPI e della porta seriale è stata adottata una soluzione basata sui MOS-FET BSS138, transistor ad effetto di campo a canale N enhancement mode con una soglia VGS di 1,3V.

Il circuito di conversione di livello è identico per ciascuna linea di segnale. Come esempio ci riferiamo alla linea SDA del bus i2C. Il gate (g) del MOS-FET BSS138 (T7) è collegato alla linea di alimentazione a 3,3V, il source (s) alla linea di segnale a livello inferiore (3,3V) ed il drain (d) alla linea di segnale a livello più alto (5V). Tra il source ed i 3,3V è collegata la resistenza di pull up R13 mentre tra il drain ed i 5V è collegata la resistenza di pull up R14.

Vediamone il funzionamento nelle diverse condizioni:

1. Nessuna comunicazione in corso. - Nessun dispositivo porta le linee di segnale a livello basso e di conseguenza la linea di segnale a 3,3V è mantenuta a livello alto dalla resistenza R13 di pull-up. Il gate ed il source del MOS-FET sono entrambi a 3,3V, di conseguenza la VGS (gate-to-source voltage) è al di sotto della soglia di innesco ed il MOS-FET non conduce. In questa condizione la linea di segnale a tensione maggiore (5V) è mantenuta a livello

**Fig. 4**

| Arduino | RPI rev. 1 | RPI rev. 2 |
|---------|------------|------------|
| D2      | GPIO18     | GPIO18     |
| D3      | GPIO23     | GPIO23     |
| D4      | GPIO24     | GPIO24     |
| D5      | GPIO25     | GPIO25     |
| D6      | GPIO4      | GPIO4      |
| D7      | GPIO17     | GPIO17     |
| D8      | GPIO21     | GPIO27     |
| D9      | GPIO22     | GPIO22     |

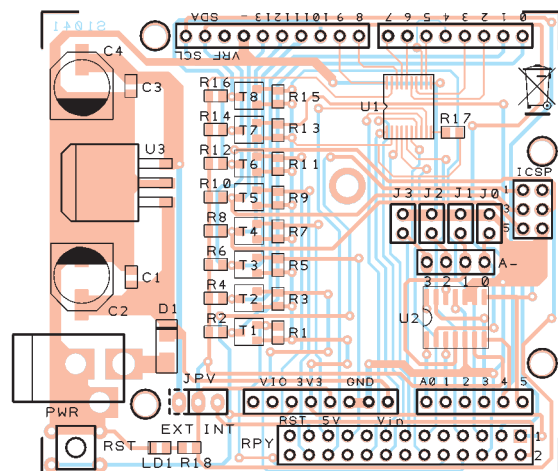
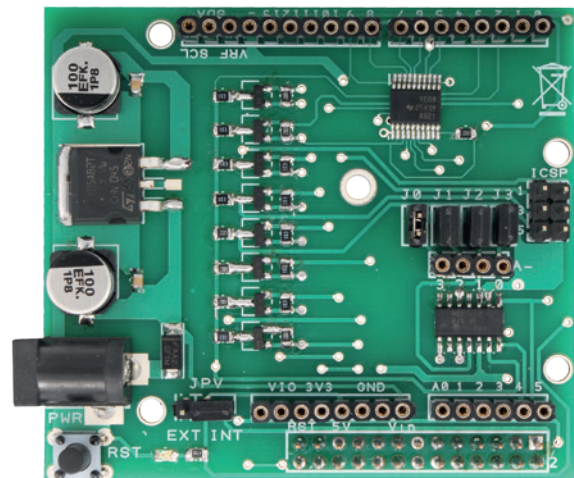
alto dalla resistenza di pull-up R14. Entrambe le sezioni della linea di segnale sono a livello alto, ma ai rispettivi livelli di tensione.

2. Comunicazione proveniente dal dispositivo a 3,3V. - Un dispositivo collegato alla sezione a 3,3V della linea di segnale porta la linea stessa a livello basso. Il source del MOS-FET T7 viene portato anch'esso a livello basso, mentre il gate si mantiene al livello di 3,3V. La VGS viene a trovarsi al di sopra della soglia di innesco ed il MOS-FET entra in conduzione. In questa condizione la sezione della linea di segnale a livello maggiore viene portata anch'essa a livello basso, pilotata dal dispositivo a 3,3V e dal MOS-FET in conduzione. In conclusione entrambe le sezioni della linea di segnale vengono a trovarsi a livello basso.
3. Comunicazione proveniente dal dispositivo a 5V. - Un dispositivo collegato alla sezione a 5V della linea di segnale porta la linea stessa a livello basso. In questa condizione la sezione a 3,3V viene portata a livello basso dal diodo drain-substrate finchè la VGS supera la soglia di innesco ed il MOS-FET T7 entra in conduzione. Ora la sezione a 3,3V della linea di segnale viene tenuta a livello basso pilotata dal dispositivo a 5V e dal MOS-FET in conduzione. In conclusione anche in questo caso entrambe le sezioni della linea di segnale vengono a trovarsi a livello basso.

In questo modo i livelli logici sono convertiti e trasferiti in entrambe le direzioni, indipendentemente dal device che guida la comunicazione. Le condizioni 2 e 3 realizzano la funzione "wired AND" tra le due sezioni della linea di segnale come richiesto dalle specifiche i2C, e non sono affette dalle resistenze di pull up presenti sul bus stesso.

Veniamo ora alla conversione ADC per la quale è stato adottato l'integrato MCP3428 della Microchip (U2).

Nella conversione ADC si è dovuto accettare un piccolo compromesso nella mappatura dei pin sui connettori che accolgono gli shield Arduino. Arduino possiede sei ingressi ADC ma due di questi, attestati sui pin A4 e A5 sono condivisi con gli ingressi i2C. In ambiente Arduino, le diverse destinazioni dei pin sono configurabili a programma, in base alle esigenze della applicazione. Questa possibilità non è riproducibile con il nostro shield che impiega in permanenza i pin del bus i2C per colloquiare con il convertitore



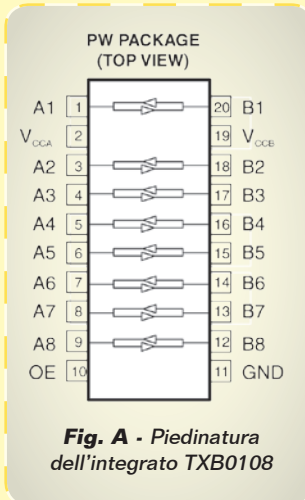
### Elenco Componenti:

|                                       |                                  |
|---------------------------------------|----------------------------------|
| R1 ÷ R17: 10 kohm (0805)              | T6: BSS138W-7-F                  |
| R18: 470 ohm (0805)                   | T7: BSS138W-7-F                  |
| C1: 100 nF ceramico (0805)            | T8: BSS138W-7-F                  |
| C2: 100 µF 25 VL<br>elettrolitico (E) | D1: GF1M-E3/67A                  |
| C3: 100 nF ceramico (0805)            | RST: Microswitch                 |
| C4: 100 µF 25 VL<br>elettrolitico (E) |                                  |
| U1: TXB0108PWR                        | Varie:                           |
| U2: MCP3428-E/SL                      | - Plu alimentazione              |
| U3: MC7805ABD2T<br>(D2PAK)            | - Strip Maschio 2 vie (4 pz.)    |
| LD1: LED verde (0805)                 | - Strip maschio 3 vie (3 pz.)    |
| T1: BSS138W-7-F                       | - Strip maschio 4 vie            |
| T2: BSS138W-7-F                       | - Strip femmina 2x13 vie (1 pz.) |
| T3: BSS138W-7-F                       | - Strip femmina 6 poli           |
| T5: BSS138W-7-F                       | - Strip femmina 8 poli (2 pz.)   |
|                                       | - Strip femmina 10 poli          |
|                                       | - Jumper (5 pz.)                 |
|                                       | - Circuito stampato              |



# L'integrato TXB0108

L'integrato TXB0108 (**Fig. A**) è un traslatore di livelli ad otto bit con due linee di alimentazione configurabili, e consente di far comunicare dispositivi e sistemi che presentano interfacce digitali caratterizzate da livelli di segnale



a tensioni differenti, per esempio 3,3V e 5V. L'architettura del TXB0108, della quale in **Fig. B** è visibile una sezione relativa ad una linea di segnale, è in grado di determinare automaticamente la direzione del flusso di dati, e quindi non richiede un segnale di controllo per impostare a priori la direzione del flusso di dati da un device all'altro. Quando una sezione della linea è attivata per traslare dati in una direzione, i driver di output dell'IC mantengono tutta la linea al livello impostato, alto o basso. Data la bassa corrente di mantenimento del livello, prevista allo scopo in progettazione, un device esterno che impegna la linea in direzione opposta con un livello di corrente che supera quello di mantenimento dell'IC, è in grado di provocare l'inversione di direzione del flusso di dati. Per questa ragione i device che pilotano le linee di I/O dell'integrato devono poter mantenere una

corrente di almeno  $\pm 2$  mA. Quando un dispositivo impegna la linea in una direzione, il circuito one-shot rileva il fronte di salita o di discesa, sulle porte A o B a seconda della provenienza del flusso dati. Durante il rilevamento del fronte di salita il circuito one-shot mette in conduzione i transistor PMOS (T1, T3) per un breve periodo, in modo da velocizzare la transizione dal livello basso a quello alto. In modo simile, durante il rilevamento di un fronte discendente il circuito one-shot mette in conduzione i transistor NMOS (T2, T4) per un breve periodo, velocizzando la transizione dal livello alto a quello basso. Il TXB0108 è progettato per pilotare carichi capacitivi fino a 70pF. I driver di output dell'IC sono caratterizzati da una corrente di pilotaggio molto bassa. Se si collegano resistenze di pull up o pull down esterne sulle linee di I/O il loro valore deve essere superiore ai 50 kOhm per essere certi che non contrastino i driver di output

dell'IC. Per la stessa ragione è consigliabile non utilizzare il TXB0108 in applicazioni che richiedono la conversione di livelli dei bus i2C o 1 wire dove di solito i driver collegati alle linee di I/O sono in configurazione open drain con resistenze di pull-up che interferiscono con il meccanismo di auto sensing dell'IC.

La porta A è pilotata dalla tensione presente sul pin  $V_{CCA}$  ed accetta livelli di tensione compresi tra 1,2 V e 3,6 V. La porta B è pilotata dalla tensione presente sul pin  $V_{CCB}$  ed accetta livelli di tensione compresi tra 1,65 V e 5,5 V. La tensione presente sul pin  $V_{CCA}$  non deve mai essere maggiore di quella presente sul pin  $V_{CCB}$ . Quando il terminale output-enable (OE) è al livello basso tutte le uscite sono poste nello stato di alta impedenza e la conversione di livello viene inibita. In funzionamento normale il terminale OE deve essere mantenuto a livello alto, collegato alla linea di alimentazione  $V_{CCA}$ .

ADC, impegnando i relativi pin e lasciandone solo quattro pin liberi per gli ingressi ADC. In compenso l'integrato MCP3428 permette conversioni sino a 16 bit di precisione sia di segnali lineari che differenziali.

I terminali da CH1+ a CH4+ sono collegati rispettivamente ai pin da A0 ad A3 sul connettore AD di Arduino. I terminali da CH1- a CH4- sono attestati sullo strip A-, e possono essere posti a massa singolarmente mediante i jumper J0, J1, J2 e J3. In questo modo è possibile configurare ciascun pin per acquisire segnali analogici sia lineari sia differenziali. I pin SDA e SCL dell'integrato sono collegati da un lato ai pin corrispondenti sul connettore Arduino e dall'altra al drain dei transistor T7 e T8. I pin ADR0 e ADR1 permettono di assegnare diversi indirizzi all'integrato a seconda della combinazione dei livelli alto e basso assegnati ai pin stessi. Nel nostro caso teniamo entrambi i pin a livello basso collegandoli a massa in modo da impostare l'indirizzo 0x68. Ovviamente il pin  $V_{CC}$  è collegato ai 5V ed il pin  $V_{SS}$  a massa. Le linee del bus SPI, MISO, MOSI, SCLK e CS vanno dai piedini 19, 21, 23 e 24 del connettore GPIO ai pin SPI del connettore Arduino ed anche al connettore ICSP. Le linee del bus seriale provenienti dai pin 8 e 10 del connettore GPIO vanno anch'essi ai corrispettivi circuiti a transistor di conversione dei livelli e poi ai pin TXD e RXD del connettore Arduino.

## UTILIZZIAMO LO SHIELD SU RASPBERRYPI

Per procedere nelle operazioni successive, utilizziamo sempre RaspberryPi, modalità terminale, le due finestre SSH ed SCP aperte e login come utente root, come descritto negli articoli precedenti dedicati a RaspberryPi.

Per prima cosa, se RaspberryPi è in funzione, spegniamola con il comando

```
shutdown -h now
```

oppure

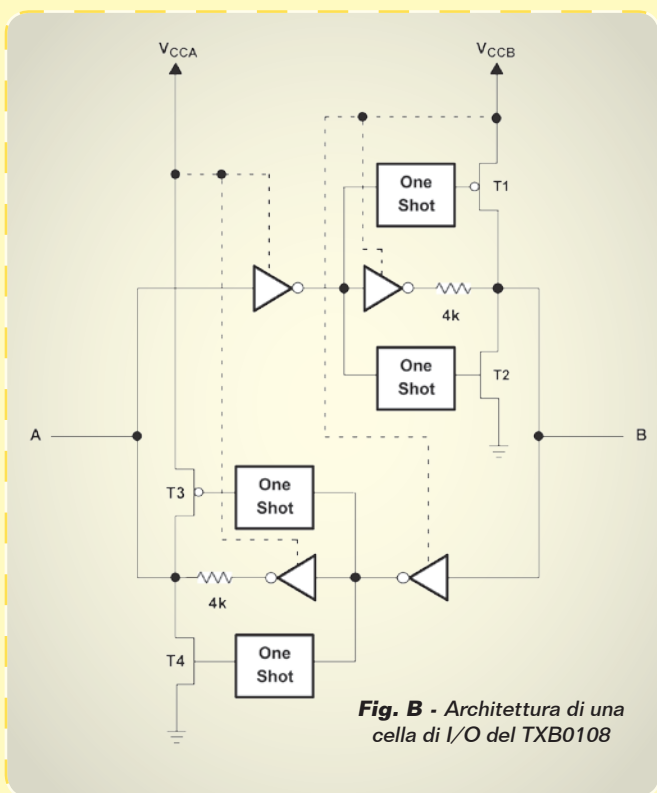
```
halt
```

e poi togliamo l'alimentazione. Colleghiamo il connettore RPY dello shield sul connettore GPIO di RaspberryPi e ridiamo tensione.

Ora dobbiamo attivare i moduli di gestione (nel mondo Windows chiamati driver) i2C ed SPI oppure uno solo dei due, in base alle vostre esigenze.

A tal proposito vi è da sottolineare la diversità di approccio ai moduli di gestione delle periferiche esterne nei mondi Windows e Linux che illustriamo nel riquadro di approfondimento dedicato.

Come potete verificare nel riquadro di approfondimento i moduli di gestione dei bus i2C ed SPI sono compilati nel kernel di Raspbian come



**Fig. B** - Architettura di una cella di I/O del TXB0108

moduli esterni e non facenti parte integrale del kernel. Oltre a questo, nella distribuzione di Raspbian che utilizziamo per i nostri articoli sono anche blacklisted ovvero oscurati all'utilizzo normale. Questo perché le linee dei bus condividono alcuni pin digitali del GPIO che vengono privilegiati nella configurazione predefinita. Per ... resuscitare e rendere utilizzabili uno o entrambi i moduli di gestione dei bus i2C e SPI è necessario toglierli dalla blacklist e poi "aggiungerli" all'insieme di moduli conosciuti dal kernel.

Visto che i moduli di gestione dei bus i2C ed SPI sono stati introdotti nelle versioni più recenti della distribuzione Raspbian, iniziamo sempre con l'aggiornamento della distribuzione presente sulla nostra scheda con i comandi:

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

Ora apriamo il file di configurazione che contiene l'elenco dei moduli blacklisted (oscurati), con il comando :

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

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

Eliminiamo i moduli i2C ed SPI dalla blacklist cancellando le righe o, come abbiamo preferito noi, commentandole con un "#" (Fig. 5). Premiamo Ctrl-X e poi Y alla richiesta di salvare il file dopo le modifiche.

Ora dobbiamo fare in modo che i due moduli "liberati" vengano caricati e diventino parte integrante del kernel.

Per questa operazione abbiamo due possibilità. La prima ci permette di caricare i moduli a comando, ed ha validità per tutto il tempo nel quale RaspberryPi rimane acceso. Al boot successivo i moduli dovranno essere ricaricati a comando.

La seconda ci permette di caricare i moduli direttamente al boot del sistema operativo, e renderli disponibili alle applicazioni immediatamente dopo il boot, condizione indispensabile in un sistema server unattended.

La prima possibilità richiede l'utilizzo del comando modprobe. Scriviamo (Fig. 6).

**Fig. 5**

**Fig. 6**

| Module         | Size   | Used by                           |
|----------------|--------|-----------------------------------|
| i2c_dev        | 5587   | 0                                 |
| bneq           | 10514  | 2                                 |
| rftcomm        | 33663  | 0                                 |
| bluetooth      | 157711 | 10 rftcomm, bneq                  |
| snd_bcm2835    | 12808  | 0                                 |
| snd_pcm        | 74834  | 1 snd_bcm2835                     |
| snd_timer      | 19698  | 1 snd_pcm                         |
| snd            | 52489  | 3 snd_timer, snd_pcm, snd_bcm2835 |
| snd_page_alloc | 4951   | 1 snd_pcm                         |
| spidev         | 5136   | 0                                 |
| i2c_bcm2708    | 3681   | 0                                 |
| spi_bcm2708    | 4401   | 0                                 |

**Fig. 7**

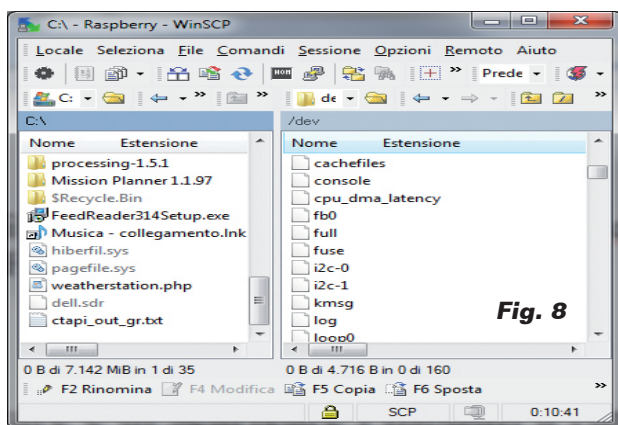


Fig. 8

*modprobe i2c-dev*

Possiamo vedere il buon esito della attivazione dei driver con il comando che mostra la lista di tutti i moduli installati (Fig. 7):

*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. 8). 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 nostri moduli dovranno essere ricaricati manualmente. Questa condizione non è adatta a funzionare con una 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, per esempio:

*modprobe -r i2c-dev*

Se si desidera che i moduli vengano caricati alla accensione di RaspberryPi è necessario abilitare il caricamento permanente del driver i2C, che si realizza modificando opportunamente il file di configurazione /etc/modules, che contiene la lista dei driver da caricare al momento del boot. Per modificare il file possiamo usare il comando:

*nano /etc/modules*

ed aggiungere una nuova linea al file di configurazione (Fig. 9)

*i2c-dev*

L'integrato MCP3428 è un convertitore analogico-digitale Delta-Sigma a basso rumore che può raggiungere una risoluzione fino a 16 bit. Sono disponibili quattro canali di ingresso differenziali; il layout del dispositivo è visibile in Fig. C.

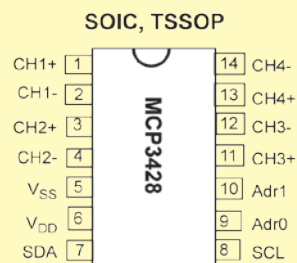


Fig. C

La frequenza di campionamento dipende dalla risoluzione che si vuole ottenere; con una risoluzione di 16 bit si possono ottenere 15 misurazioni al secondo, 60 misurazioni

al secondo alla risoluzione di 14 bit e 240 misurazioni alla risoluzione di 12 bit. Il dispositivo si calibra automaticamente eliminando gli errori di offset e di guadagno dovuti alle variazioni di temperatura e di tensione di alimentazione. Come visibile nella Fig. D, la tensione di riferimento di 2,048V è gestita internamente al dispositivo, il che permette un intervallo di tensioni di ingresso di  $\pm 2.048V$  differenziali. Il segnale in ingresso può essere amplificato per mezzo dell'amplificatore di guadagno interno (PGA nella Fig. D), programmabile dall'utente per ottenere fattori di guadagno di 1, 2, 4, o 8 volte. È anche possibile programmare la modalità di conversione, che può essere a campionamento continuo o singolo. In modalità continua il dispositivo esegue letture in continuazione ed

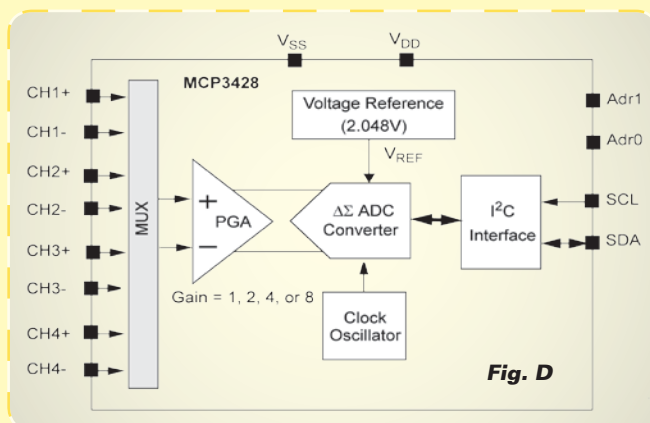


Fig. D

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:

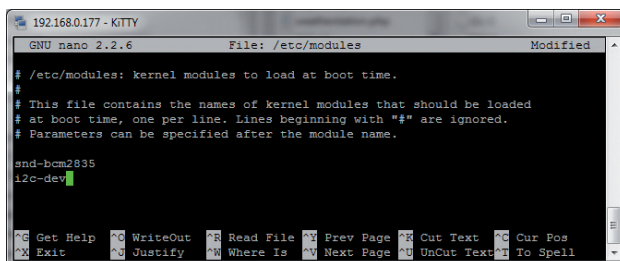


Fig. 9



| I <sup>2</sup> C Device Address Bits |    |    | Logic Status of Address Selection Pins |               |
|--------------------------------------|----|----|----------------------------------------|---------------|
| A2                                   | A1 | A0 | Adr0 Pin                               | Adr1 Pin      |
| 0                                    | 0  | 0  | 0 (Addr_Low)                           | 0 (Addr_Low)  |
| 0                                    | 0  | 1  | 0 (Addr_Low)                           | Float         |
| 0                                    | 1  | 0  | 0 (Addr_Low)                           | 1 (Addr_High) |
| 1                                    | 0  | 0  | 1 (Addr_High)                          | 0 (Addr_Low)  |
| 1                                    | 0  | 1  | 1 (Addr_High)                          | Float         |
| 1                                    | 1  | 0  | 1 (Addr_High)                          | 1 (Addr_High) |
| 0                                    | 1  | 1  | Float                                  | 0 (Addr_Low)  |
| 1                                    | 1  | 1  | Float                                  | 1 (Addr_High) |
| 0                                    | 0  | 0  | Float                                  | Float         |

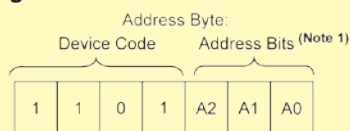
**Fig. E**

aggiorna il buffer con l'ultima lettura eseguita. In modalità singola la lettura viene eseguita a seguito di un comando ricevuto e poi il device entra in standby, in attesa della richiesta successiva. Quest'ultima modalità permette di ridurre considerevolmente i consumi di corrente.

La comunicazione con l'integrato avviene sul bus I<sup>2</sup>C con l'invio dei registri di indirizzo e di comando, di due byte ciascuno.

## Registro di indirizzo

Il registro di indirizzo serve ad identificare quale dispositivo si vuole raggiungere sul bus I<sup>2</sup>C. È composto da 7 bit i primi 4 dei quali sono fissi, rappresentano il codice device univoco assegnato al produttore ed assumono la configurazione 1101. Gli ultimi tre sono determinati da



come configuriamo i due pin di indirizzo del dispositivo, la tabella in **Fig. E** contiene le diverse configurazioni dei tre bit a seconda dello stato dei pin di address. Nel nostro caso abbiamo collegato entrambi i pin a massa e di conseguenza i tre bit assumono valore 000. La configurazione completa del byte di indirizzo diventa 1101000 che equivale a 0x68 in esadecimale.

## Registro di configurazione

Un po' più complessa è la gestione del registro di configurazione, che è utilizzato per impostare le modalità di funzionamento del convertitore ADC e per richiedere la lettura

del buffer di uno specifico canale. Il byte di configurazione è il risultato del concatenamento di diverse impostazioni espresse in formato binario come visibile nella **Fig. F**.

## bit 7 RDY: Bit di Ready

Il bit costituisce l'indicatore che i dati nel buffer sono pronti da leggere. Nella modalità one-shot configurare a "1" il bit istruisce il device ad iniziare una nuova lettura.

In lettura il bit RDY assume i seguenti significati:

- 1 = Il buffer di output non è stato aggiornato.
- 0 = Il buffer di output è stato aggiornato con l'ultima lettura.

Significato del bit RDY in scrittura:

In modalità di conversione continua viene ignorato. In modalità di conversione a campionamento singolo:

- 1 = Inizia una nuova conversione.
- 0 = Nessun significato.

## bit 6-5 C1-C0: Bit di selezione del canale di lettura

- 00 = Canale 1
- 01 = Canale 2
- 10 = Canale 3
- 11 = Canale 4

## bit 4 O/C: Bit che indica la modalità di conversione

- 1 = Modalità di conversione a campionamento continuo.
- 0 = Modalità di conversione a campionamento singolo.

## bit 3-2 S1-S0: Bit di configurazione della risoluzione della conversione

- 00 = 12 bit con 240 conversioni al secondo
- 01 = 14 bit con 60 conversioni al secondo
- 10 = 16 bit con 15 conversioni al secondo

## bit 1-0 G1-G0: PGA Bit di configurazione del fattore di guadagno

- 00 = x1
- 01 = x2
- 10 = x4
- 11 = x8

Se vogliamo leggere il canale 1 in modalità one-shot, con risoluzione di 16 bit e guadagno x1 dovremo comporre il registro con la seguente sequenza:

1-00-0-10-00

che in esadecimale corrisponde al valore 0x88 che corrisponde al registro usato nel nostro programma.

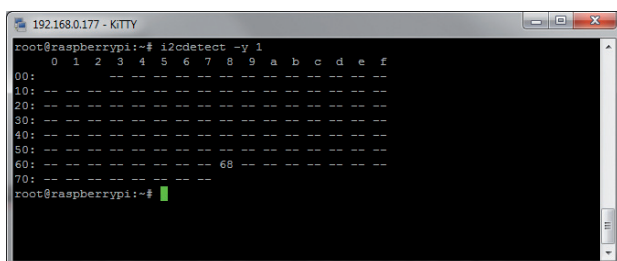
**Fig. F**

| R/W-1 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| RDY   | C1    | C0    | O/C   | S1    | S0    | G1    | G0    |
| 1 *   | 0 *   | 0 *   | 1 *   | 0 *   | 0 *   | 0 *   | 0 *   |
| bit 7 |       |       |       | bit 0 |       |       |       |

`apt-get install i2c-tools`

Aggiungiamo il nostro utente pi al gruppo i2c:

`adduser pi i2c`



**Fig. 10**

Eseguiamo il reboot di RaspberryPi per attivare le nuove configurazioni con il comando

`reboot`

Dopo che RaspberryPi si è riavviata e vi siete riconnessi con Putty o Kitty verificate se sul bus I<sup>2</sup>C è visibile il convertitore ADC con il comando

`i2cdetect -y 0` per RaspberryPi rev. 1 oppure  
`i2cdetect -y 1` per RaspberryPi rev. 2

Dovrete ottenere un risultato simile a quello visibile in **Fig. 10** dove l'indirizzo 0x68 identifica il

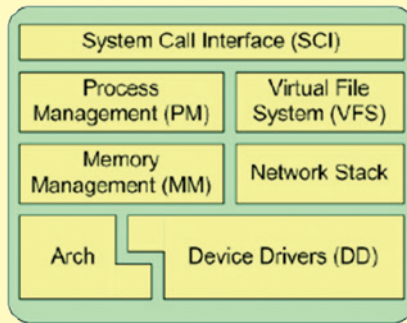
# Gestione moduli in Linux

Quello di GNU-Linux è un mondo immenso, ed una trattazione approfondita delle sue logiche di funzionamento, oltre a non essere nel campo di interesse di questa rivista, richiederebbe centinaia e centinaia di pagine e di schemi. Tuttavia pensiamo sia opportuno approfondire alcuni aspetti che possono facilitare la comprensione del perché si eseguono certe operazioni o che possono dare una impostazione alla risoluzione di problemi che possono presentarsi. Prendiamo spunto dalla necessità di attivare i moduli di gestione dei bus i2C ed SPI per approfondire maggiormente gli aspetti reconditi della gestione dei Device Driver in Linux.

Ci ricollegiamo a quanto descritto nel box di approfondimento sulla architettura del sistema operativo GNU-Linux presentata nel numero 168 di ElettronicaIn. Abbiamo descritto le caratteristiche del kernel, che ne hanno permesso la portabilità sui processori più disparati, e quindi il principale fattore della sua diffusione.

Se approfondiamo l'architettura del kernel possiamo considerarlo composto da un certo numero di sottosistemi come visibile nella **Fig. G**.

Sofferamoci per brevità su due sottosistemi particolari, quello denominato Arch nella **Fig. G** che può essere considerato il vero cuore del kernel, che si occupa dell'interfaccia verso il processore hardware specifico al quale è destinato e che deve essere riscritto, almeno in parte, per ogni nuovo processore: il cosiddetto processo di "porting". L'altro sottosistema, fortemente integrato con il primo è l'insieme dei Device Drivers, l'insieme di moduli che permettono di gestire le periferiche più disparate da parte del kernel stesso e delle applicazioni utente.



**Fig. G**

La maggior parte dei dieci milioni di righe di codice che costituiscono i sorgenti del kernel di GNU-Linux sono concentrati proprio nel sottosistema Device Drivers e sono compilati congiuntamente al kernel. Sono disponibili due opzioni di compilazione dei Device Driver, la prima consiste nel compilare i Device Driver all'interno del kernel in modo da ottenere un unico oggetto monolitico. Questa opzione è utilizzata per integrare nel kernel i Device Driver di utilizzo obbligatorio come console, terminali e dischi oppure per rendere accessibili automaticamente (plug and play) i dispositivi come dongle e periferiche USB, in modo da rendere queste operazioni totalmente trasparenti all'utilizzatore.

La seconda modalità permette di compilare i Device Driver come moduli caricabili nel kernel separatamente, e solo se necessario, come è il caso dei Device Driver per i bus i2C e SPI.

Come abbiamo già visto negli articoli scorsi le periferiche gestite dai Device Driver vengono poi presentate alle applicazioni come file nella cartella /dev (gpio4 o ttyUSB0 per fare degli esempi che abbiamo già incontrato).

Vediamo per sommi capi la sequenza di azioni e collegamenti che permettono di creare la catena di trasmissione di dati e comandi tra la periferica hardware reale, gestita dal kernel, e la sua rappresentazione applicativa, nella cartella /dev.

Quando si desidera compilare una nuova distribuzione di Linux si inizia col disporre i sorgenti che si desidera includere nella compilazione in una struttura di cartelle opportunamente organizzate.

Uno dei passi successivi più impegnativi è la definizione delle caratteristiche che dovrà assumere la distribuzione finale, obiettivo che si ottiene configurando le centinaia di opzioni e parametri previste all'interno dei sorgenti. Per fare questa operazione si utilizza un configuratore a menù del quale, nelle **Fig. H** e **Fig. I** si vedono due pagine della sezione dedicata ai Device Driver.

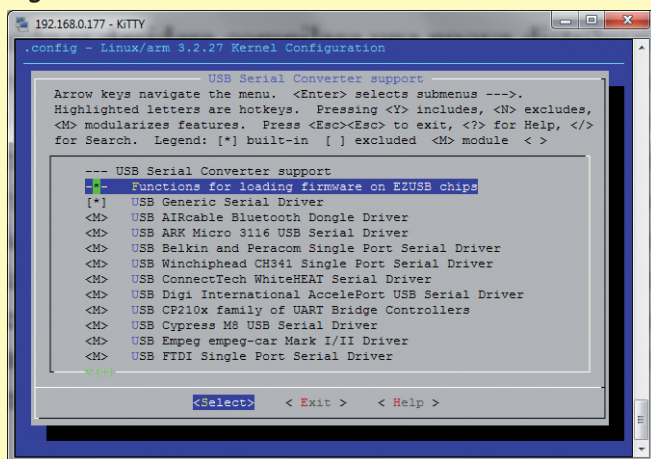
Nella **Fig. H** si può notare, per esempio, il driver FTDI (quello utilizzato per la Centralina meteo, mentre nella **Fig. I** si può vedere il Device Driver per il bus i2C. I moduli da includere nel corpo del kernel sono selezionati con "\*", quelli da compilare separatamente sono contrassegnati con una "M".

Dopo aver terminato, e salvato il file di configurazione, si esegue la compilazione effettiva del kernel utilizzando il compilatore "C" (pacchetto g++). Al termine si installano i componenti compilati del kernel nelle rispettive partizioni di boot e di root. Il kernel riconosce ciascun Device Driver per mezzo dei cosiddetti numeri di device major e minor.

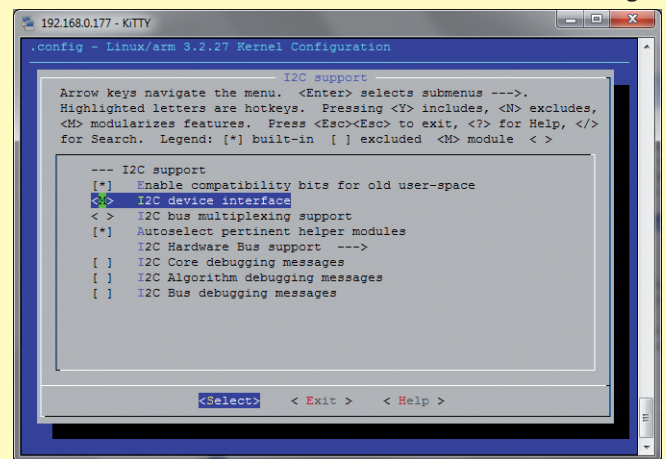
Questi numeri sono identificativi univoci per i Device Driver che sono definiti nella documentazione del kernel e sono cablati all'interno del codice dei driver per ciascun tipo di dispositivo.

Per avere un'idea dei driver disponibili nel

**Fig. H**



**Fig. I**



```

192.168.0.177 - KTTY
root@raspberrypi:~# cat /proc/devices
Character devices:
1 mem
4 /dev/vc/0
4 tty
5 /dev/tty
5 /dev/console
5 /dev/ptmx
5 ctyprintk
7 vcs
10 misc
13 input
14 sound
29 fb
89 i2c
116 alsa
128 ptm
136 pts
153 spi
162 raw
180 usb
189 usb device
204 ttyAMA
216 rfcomm
252 vchiq
253 vc-mem
254 rtc

```

Fig. J

```

192.168.0.177 - KTTY
root@raspberrypi:~# cd /dev
root@raspberrypi:/dev# ls -l
total 0
crw-rw-rw- 1 root root 10, 235 Jan 1 1970 autofs
drwxr-xr-x 2 root root 600 Jan 1 1970 block
crw-rw-rw- 1 root root 10, 234 Jan 1 1970 btrfs-control
drwxr-xr-x 3 root root 60 Jan 1 1970 bus
crw-rw-rw- 1 root root 10, 63 Jan 1 1970 cachefiles
drwxr-xr-x 2 root root 2300 Jan 18 10:33 char
crw-rw-rw- 1 root root 5, 1 Jan 18 08:59 console
crw-rw-rw- 1 root root 10, 62 Jan 1 1970 cpu_dma_latency
drwxr-xr-x 5 root root 100 Jan 1 1970 disk
crw-rw-rw- 1 root root 29, 0 Jan 1 1970 fb0
lrwxrwxrwx 1 root root 13 Jan 1 1970 fd -> /proc/self/fd
crw-rw-rw- 1 root root 1, 7 Jan 1 1970 full
crw-rw-rw- 1 root fuse 10, 229 Jan 1 1970 fuse
crw-rw-rw- 1 root root 89, 0 Jan 18 10:33 i2c-0
crw-rw-rw- 1 root root 89, 1 Jan 18 10:33 i2c-1
drwxr-xr-x 2 root root 60 Jan 1 1970 input
crw-rw-rw- 1 root root 1, 11 Jan 1 1970 kmsg
srw-rw-rw- 1 root root 0 Jan 18 08:59 log
brw-rw-rw- 1 root disk 7, 0 Jan 1 1970 loop0
brw-rw-rw- 1 root disk 7, 1 Jan 1 1970 loop1
brw-rw-rw- 1 root disk 7, 2 Jan 1 1970 loop2
brw-rw-rw- 1 root disk 7, 3 Jan 1 1970 loop3
brw-rw-rw- 1 root disk 7, 4 Jan 1 1970 loop4
brw-rw-rw- 1 root disk 7, 5 Jan 1 1970 loop5

```

Fig. K

kernel di RaspberryPi possiamo dare il comando

`cat /proc/devices`

con il quale otteniamo la lista di **Fig. J** che mostra i driver disponibili con il loro numero identificativo major. Potete notare che la lista è divisa tra dispositivi a carattere e dispositivi a blocco. L'associazione dei Device Driver del kernel con i file di device presenti nella cartella /dev avviene utilizzando proprio i numeri di device. Possiamo costatarlo andando nella cartella /dev e digitando il comando

`ls -l`

Otterremo il risultato visibile in **Fig. K**. Per ciascuna riga che identifica un device troviamo la prima lettera "c" o "b" che identifica se il dispositivo è a carattere o a blocco, seguono le autorizzazioni, poi ancora il "proprietario" del file. Seguono ancora due numeri separati da

una virgola; il primo è il numero di device major ed il secondo il minor. Dopo la data troviamo il nome del file che identifica il device per le applicazioni. In questo modo quando una applicazione richiede l'utilizzo di un certo device richiamandolo con il nome corrispondente in /dev, il sistema operativo è in grado, risalendo a ritroso la catena descritta, di associare il Device Driver corretto per la sua gestione. Quando si compila un modulo per un nuovo Device Driver l'associazione tra numeri di device ed ingressi in /dev viene creata utilizzando il comando `mknod`, come per esempio potrebbe essere nel caso del driver del bus i2C.

`mknod /dev/i2c-0 c 89 0`

Nel corpo dell'articolo è descritto come caricare nel kernel i Device Driver compilati come moduli, specificatamente per i moduli di gestione dei bus i2C ed SPI.

convertitore ADC. Dato che stiamo realizzando uno strumento di integrazione con gli shield di Arduino, per sperimentare il funzionamento del convertitore ADC, e del bus i2C in generale, abbiamo preparato un piccolo circuito, derivato dagli esempi presenti nel libro Arduino Uno - Programmazione avanzata e librerie di sistema e dall'eserciziario LabVIEW MyDAQ (entrambi reperibili presso Futura Elettronica), per misurare, rispettivamente, la temperatura con una resistenza NTC e la luminosità ambiente con una fotoresistenza.

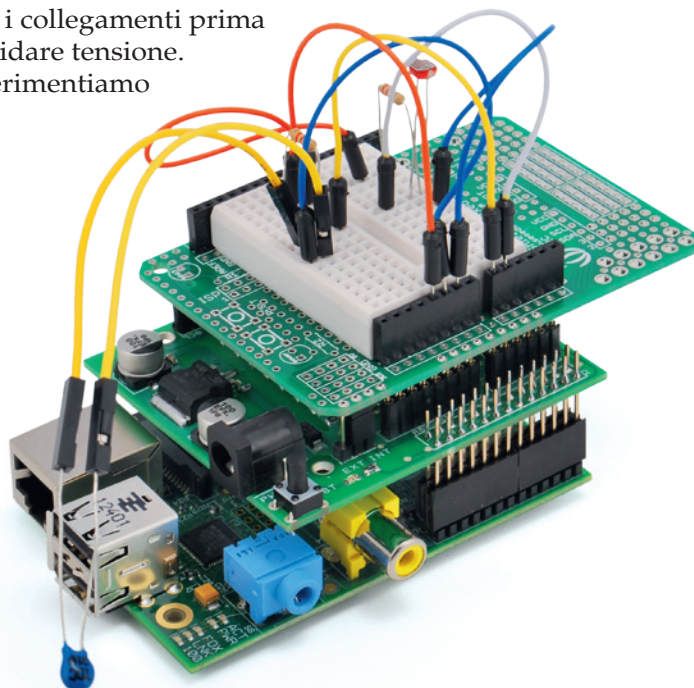
Lo schema del circuito è visibile in **Fig. 11**.

Le resistenze sono da 22kOhm.

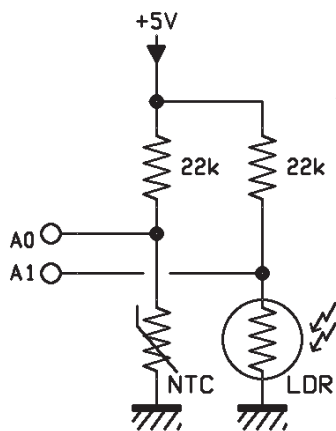
La NTC è una Vishay NTCLE-100E-3103 da 10kOhm.

La foto resistenza è una LDR04 o una GL5528 da 10kOhm.

Il punto centrale del partitore di tensione del quale fa parte la NTC è collegato al canale CH1+ (A0) del convertitore ADC, il punto centrale del partitore di tensione del quale fa parte la fotoresistenza è collegato al canale CH2+ (A1). I terminali CH1- e CH2- sono collegati a massa. Il circuito è realizzato con uno shield di prototipazione per Arduino al quale abbiamo incollato una bread board mille fori con nastro biadesivo. Lo shield di prototipazione è montato a pacchetto sullo shield di integrazione, che a sua volta è innestata sul connettore GPIO di RaspberryPi. Spegnete sempre RaspberryPi prima di eseguire interventi e verificate sempre i collegamenti prima di ridare tensione. Sperimentiamo







**Fig. 11**

ora il funzionamento del convertitore utilizzando la linea di comando da terminale. Usiamo il comando

*i2cget -y 0 0x68 0x88 w* per RaspberryPi rev. 1  
oppure  
*i2cget -y 1 0x68 0x88 w* per RaspberryPi rev. 2

per leggere il valore esadecimale (**Fig. 12**) dei due byte restituiti dal canale 1 del convertitore ADC che rappresenta il valore digitale della tensione misurata ai capi della resistenza NTC.

Il secondo parametro indica il bus i2C da interrogare, il terzo il registro che contiene l'indirizzo del dispositivo da leggere (0x68), il quarto il registro di comando che indica l'ingresso del quale desideriamo conoscere la lettura digitale e le modalità di campionamento, l'ultimo, il parametro w, che indica che desideriamo di ritorno entrambi i byte della misura. A seconda della temperatura otteniamo un valore di quattro

caratteri, che rappresentano la conversione in valore digitale della tensione misurata ai capi della resistenza NTC espressa da due byte esadecimali, nel nostro caso "5769" (**Fig. 12**). Ricordiamo sempre che il primo byte (low byte) è il meno significativo ed il secondo è il più significativo (high byte). Prima di utilizzarli, per esempio per convertirli in un valore decimale, dobbiamo invertirli, ottenendo, nel nostro caso, "6957". Per convertire il valore esadecimale nel corrispondente valore decimale possiamo usare il comando:

*echo 'ibase=16;obase=A;6957' | bc*

che ci restituirà il valore 26967 (**Fig. 13**).

Il comando stampa su console il risultato della conversione da base 16 a base 10 (A in esadecimale) del valore 6957 (è obbligatorio utilizzare le maiuscole per le lettere) e lo passa al programma "bc". "bc" è un generico programma di calcolo a riga di comando, che permette calcoli e conversioni in tutte le basi, val la pena di provarlo perché può risolvere molte situazioni. Per esempio per le conversioni da binario a esadecimale utilizzate il comando:

*echo 'ibase=2;obase=10000;bbbbbbbb' | bc*

sostituendo bbbbbbbb con il

## per il MATERIALE

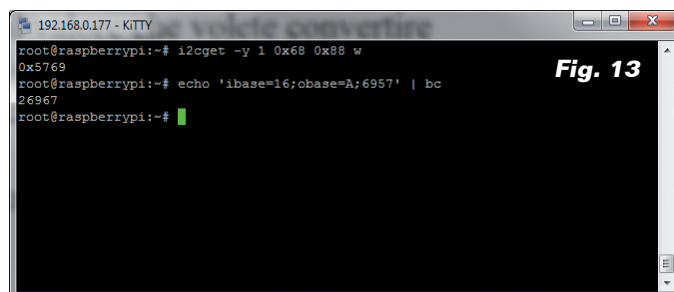
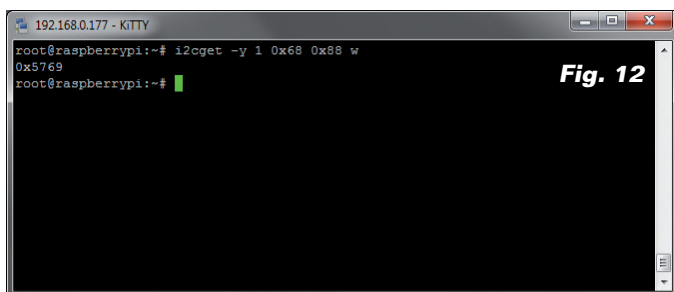
Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista così come il programma di esempio che verrà presentato nel prossimo numero. Di questo shield Futura Elettronica ha realizzato la versione montata e collaudata (cod. FT1041M) disponibile al prezzo di 27 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>

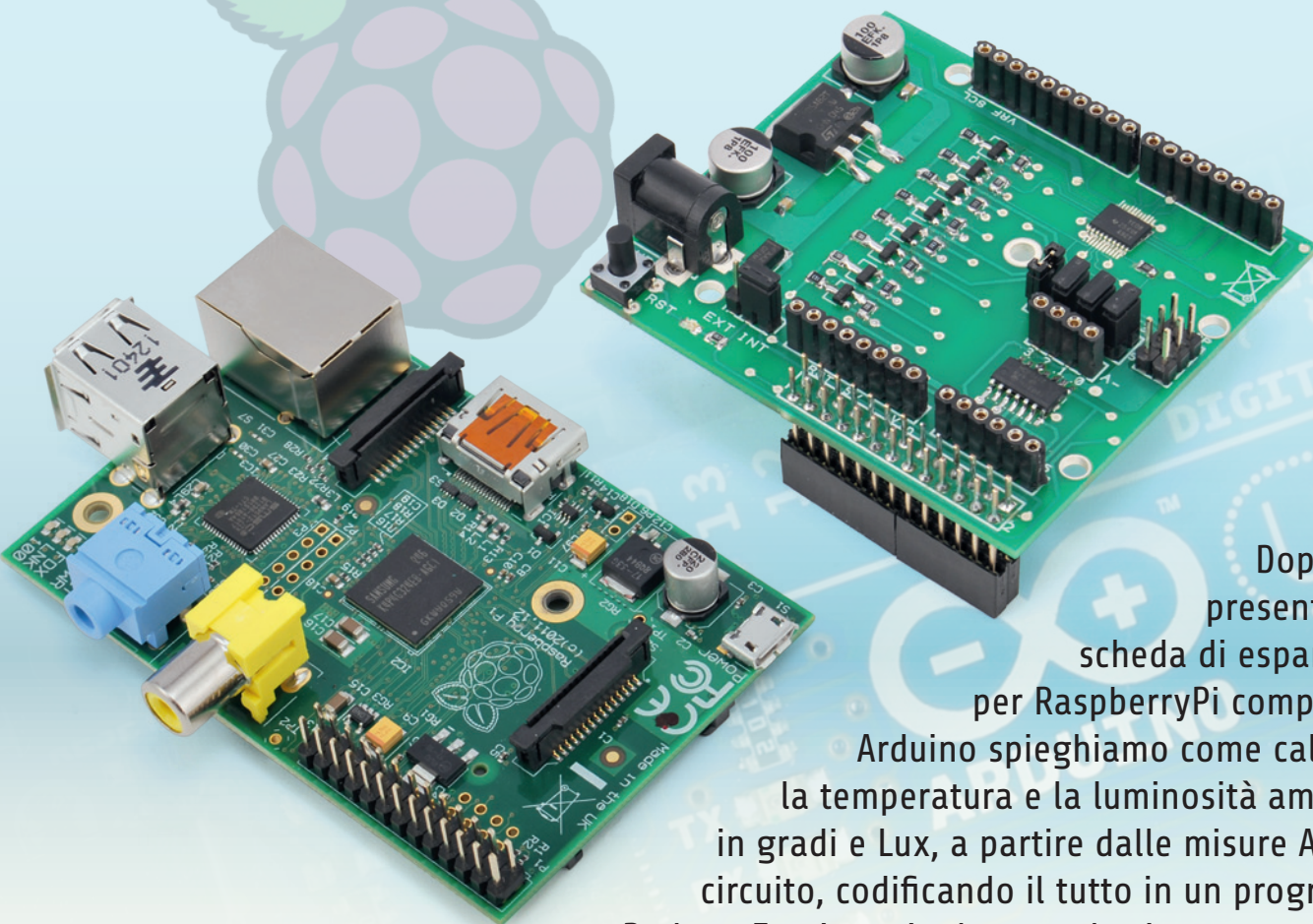
valore che volete convertire. Per convertire da esadecimale a binario utilizzate il comando:

*echo 'ibase=16;obase=2;hh' | bc*

sostituendo hh con il valore che volete convertire. Possiamo eseguire lo stesso test sul canale CH2 del convertitore ADC per sincerarci del corretto funzionamento del circuito della foto resistenza. Nel prossimo numero approfondiremo i calcoli di conversione dalla lettura digitale ai valori di temperatura e luminosità, con il relativo programma in python. ■



# SHIELD DI ESPANSIONE ADC PER RASPBERRYPI COMPATIBILE CON ARDUINO



Dopo aver presentato la scheda di espansione per RaspberryPi compatibile con Arduino spieghiamo come calcolare la temperatura e la luminosità ambiente in gradi e Lux, a partire dalle misure ADC del circuito, codificando il tutto in un programma Python. Facciamo inoltre un ulteriore passo avanti verso la compatibilità descrivendo come realizzare programmi che funzionino su entrambi i sistemi.

di MARCO MAGAGNIN

**D**opo avere descritto la scheda di espansione del GPIO di RaspberryPi compatibile con la piedinatura di Arduino ed il modo di comunicare in i2C con il convertitore ADC montato sulla scheda stessa, abbiamo utilizzato un piccolo circuito per misurare temperatura e luminosità ambiente utilizzando una NTC ed una fotoresistenza. Continuiamo con la descrizione dei calcoli necessari a passare dalle misure di tensione ai capi delle resistenze, lette dal convertitore

ADC, ai corrispondenti valori di temperatura e luminosità espressi in gradi centigradi e lux; questi calcoli li codifichiamo in un programma in Python che presenta i risultati a video, ma è già predisposto per essere adattato in futuro per un progetto di integrazione con interfaccia Web. Abbiamo poi dedicato un approfondimento alla possibilità di scrivere programmi che, utilizzando la scheda di espansione su RaspberryPi, possano girare indifferente su Raspberry o su

# Scriviamo programmi che funzionano sia su *RaspberryPi* che su *Arduino*

La nostra scheda di espansione è compatibile con la libreria ArduPi, una libreria per RaspberryPi, che “porta” molte delle funzioni disponibili nelle librerie standard di Arduino in ambiente RaspberryPi, mantenendone inalterata la sintassi.

La libreria è stata realizzata, in linguaggio C++, dalla Libelium Comunicaciones Distribuidas S.L. ed è un software open source distribuito secondo i termini della licenza GNU versione 3.

Questo approccio permettere di scrivere programmi che funzionano sia su Arduino, sia su RaspberryPi senza modifiche, anche se limitatamente

alle funzioni “compatibili”. Le funzioni disponibili nella libreria ArduPi sono elencate nel riquadro. Come potete vedere l'attenzione è posta all'utilizzo degli I/O digitali, all'uso della porta seriale, dei bus i2C (Wire) e SPI.

Sottolineiamo solamente la differenza di comportamento della funzione `attachInterrupt()`. Siccome RaspberryPi non supporta direttamente gli interrupt (e neanche potrebbe data la natura multithread e multi processo del sistema operativo), la funzione è stata realizzata in modo da attivare un thread separato, rispetto al programma principale, che esegue il polling del pin sul

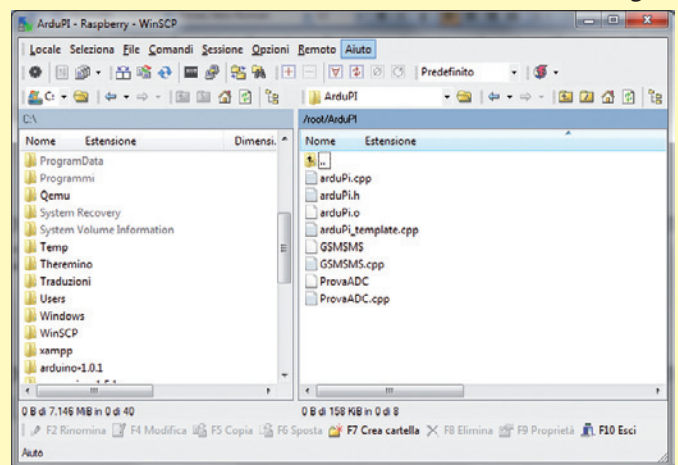


Fig. B

quale si vuole gestire l'inter-rup-t. Gli eventi possono essere RISING, FALLING, HIGH e LOW. Possono essere utilizzati i pin da D2 a D9.

Per esempio, se vogliamo controllare un interrupt sul fronte di salita del pin 6 possiamo specificare l'istruzione

```
attachInterrupt(6,funzione_
da_chiamare,RISING)
```

Per verificare il funzionamento della libreria ArduPi con la nostra scheda di espansione abbiamo riscritto il programma

Python di calcolo della temperatura e luminosità in stile Arduino; lo abbiamo compilato con l'IDE di Arduino (Fig A) e poi, dopo aver modificato il riferimento alla libreria, lo abbiamo compilato ed eseguito con RaspberryPi. 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\\_rev2\\_1-3](http://www.cooking-hacks.com/skin/frontend/default/cooking/images/catalog/documentation/raspberry_arduino_shield/arduPi_rev2_1-3).

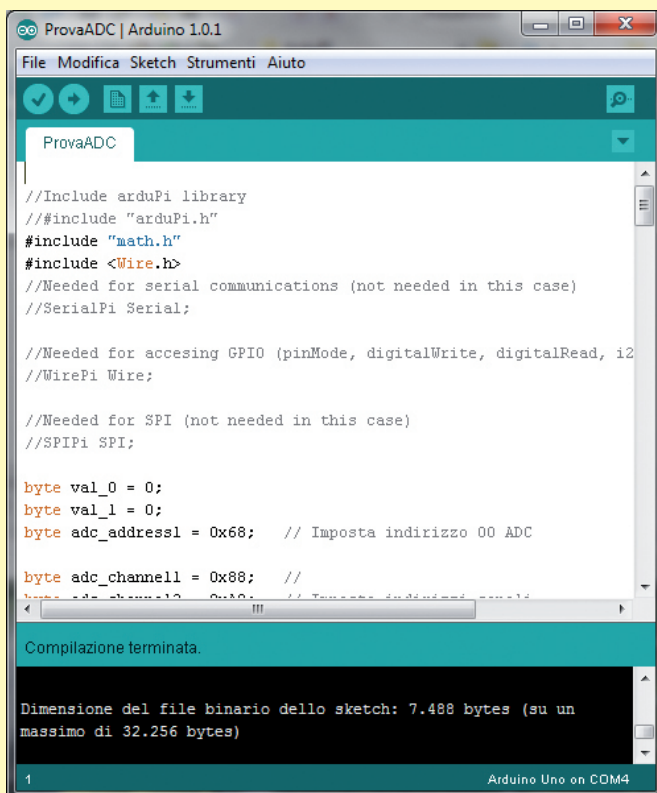


Fig. A

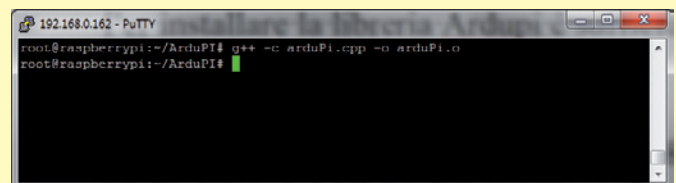


Fig. C

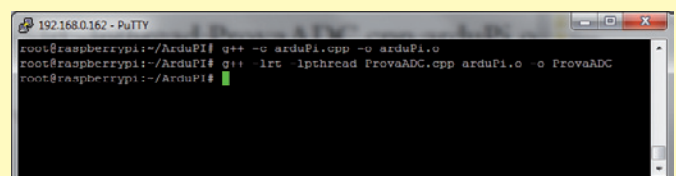


Fig. D

una scheda Arduino. Come esempio abbiamo duplicato in C il programma Python. Riportiamo in Fig. 1 il circuito dei sensori di temperatura e luminosità che abbiamo usato

negli esempi del numero scorso. Approfondiamo le caratteristiche dei sensori e le sequenze di calcoli, che ci permettono di ottenere valori significativi di temperatura e luminosità.



tar.gz per la relv. 2 e

[http://www.cooking-hacks.com/skin/frontend/default/cooking/images/catalog/documentation/raspberry\\_arduino\\_shield/arduino\\_rev1\\_1-3.tar.gz](http://www.cooking-hacks.com/skin/frontend/default/cooking/images/catalog/documentation/raspberry_arduino_shield/arduino_rev1_1-3.tar.gz) per la rev. 1.

La scompattiamo nella cartella root o dove vogliamo e, per comodità, gli modifichiamo il nome in ArduinoPi. Sempre per comodità metteremo nella stessa cartella i programmi che andremo a realizzare. Compiliamo la libreria ArduinoPi con il comando (**Fig. C**):

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

Dopo aver scritto il programma ProvaADC.cpp visibile nel **Listato 1** (potete come sempre scaricarlo dal sito di Elettronica In) lo potete compilare e linkare alla libreria ArduinoPi con il comando (**Fig. D**):

```
g++ -lrt -lthread ProvaADC.cpp arduinoPi.o -o ProvaADC
```

Per eseguirlo diamo il comando (**Fig. E**)

```
./ProvaADC
```

Nella stessa figura vediamo il risultato, identico a quello ottenuto con il programma Python. Vedremo come farne buon uso per integrare ancora di più i due ambienti.

```
192.168.0.162 - PuTTY
root@raspberrypi:~/ArduinoPi# g++ -c arduinoPi.cpp -o arduinoPi.o
root@raspberrypi:~/ArduinoPi# g++ -lrt -lthread ProvaADC.cpp arduinoPi.o -o ProvaADC
root@raspberrypi:~/ArduinoPi# ./ProvaADC
Steinhart - Celsius = 22.442598
parametro B - Celsius = 22.574796
Lux = 178.719807
root@raspberrypi:~/ArduinoPi#
```

**Fig. E**

**Funzioni generali:**

- delay()
- delayMicroseconds()
- pinMode()
- digitalWrite()
- digitalRead()
- shiftIn()
- shiftOut()
- attachInterrupt()
- detachInterrupt()

**Libreria Serial:**

- available()
- begin()
- end()
- flush()
- peek()
- print()
- println()
- read()
- readBytes()
- readBytesUntil()
- find()
- findUntil()
- parseInt()
- parseFloat()
- setTimeout()
- write()

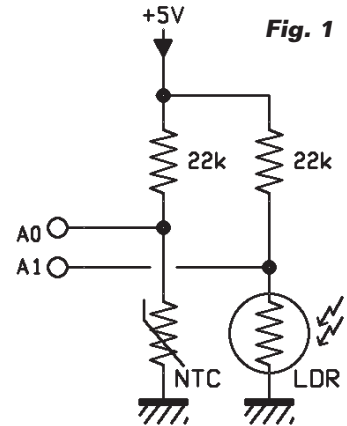
**Libreria Wire:**

- begin()
- requestFrom()
- beginTransmission()
- endTransmission()
- write()
- read()

**Libreria SPI:**

- begin()
- end()
- setBitOrder()
- setClockDivider()
- setDataMode()
- transfer()

**Fig. 1**



cient) è un componente che ha la caratteristica di variare il proprio valore resistivo al variare della temperatura, in particolare la sua resistenza decresce con l'aumentare della temperatura. Per questa caratteristica ben si adatta ad essere utilizzato come sensore di temperatura. La variazione del valore della resistenza non segue linearmente la variazione della temperatura, ma una curva che è possibile approssimare matematicamente, applicando due formule che possono essere scelte in base ai dati forniti nel datasheet del termistore stesso. La prima è la formula di Steinhart-Hart, un polinomio del terzo ordine in grado di interpolare i valori di temperatura rispetto alla resistenza con una precisione anche di 0,002 gradi.

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

Equazione di Steinhart-Hart

dove R è la resistenza del termistore nel momento della misura, Rref è la resistenza nominale del termistore alla temperatura di riferimento di 25 C che, nel nostro caso, vale 10kOhm.

I coefficienti A, B, C e D del polinomio si possono determinare, per un termistore, con alcune misurazioni a temperature conosciute e con una serie di calcoli. In genere, fortunatamente, questi valori sono già specificati in un'apposita tabella nel datasheet del termistore che utilizziamo, congiuntamente ai gradi di precisione della misura ottenibile nei diversi intervalli di temperatura.

Per il nostro termistore abbiamo utilizzato i seguenti valori, che danno il grado di precisione migliore nell'intorno dei 25 gradi centigradi:

A = 0.00335402  
B = 0.000256985  
C = 2.62013e-6  
D = 6.38309e-8

La misura di temperatura prodotta dalla formula di Steinhart-Hart è in gradi Kelvin e non tiene conto degli effetti del fattore di dissipazione K del termistore, cioè dell'autoriscaldamento del componente dovuto al passaggio della corrente

tà a partire dalle letture del convertitore ADC.

**TERMISTORE NTC**

Il termistore NTC (Negative Temperature Coeffi-

## Listato 1

```
//Include arduPi library
#include "arduPi.h"
#include "math.h"
// #include <Wire.h>
// Needed for serial communications (not needed in this case)
// SerialPi Serial;

// Needed for accessing GPIO (pinMode, digitalWrite, digitalRead, i2C functions)
WirePi Wire;

// Needed for SPI (not needed in this case)
// SPIPi SPI;

byte val_0 = 0;
byte val_1 = 0;
byte adc_address1 = 0x68; // Imposta indirizzo 00 ADC

byte adc_channel1 = 0x88; //
byte adc_channel2 = 0xA8; // Imposta indirizzi canali
byte adc_channel3 = 0xC8; // con risoluzione di 16 bit
byte adc_channel4 = 0xE8; //

double R1 = 22000.0; // Resistenza R1
double R2 = 22000.0; // Resistenza R2
double R_th0 = 10000.0; // Resistenza riferimento termistore a 25 c
double R_fr0 = 70000.0; // Resistenza fotoresistore con illuminamento IL unitario

double V_IN = 5.0; // Tensione riferimento misura ADC
double A = 0.00335402; // Steinhart-Hart Costante A
double B = 0.000256985; // Steinhart-Hart Costante B
double C = 2.62013e-6; // Steinhart-Hart Costante C
double D = 6.38309e-8; // Steinhart-Hart Costante D

double pB = 4100.0; // Costante parametro B

double K = 6; // Fattore dissipazione K 6mV C

double Pend = 0.7; // Valore Gamma fotoresistenza

int channel_0 = 0;
int channel_1 = 0;
int channel_2 = 0;
int channel_3 = 0;

double analog_0 = 0.0;
double analog_1 = 0.0;
double analog_2 = 0.0;
double analog_3 = 0.0;

// ===== VARIABILI DI SERVIZIO
double V = 0.0;
double R_th = 0.0;
double logR = 0.0;
double logR2 = 0.0;
double logR3 = 0.0;
double Stein = 0.0;
double Celsius = 0.0;
double dec_val = 0.0;

double RRo = 0.0;
double logRRo = 0.0;
double parB = 0.0;
double Celsius2 = 0.0;

double dec_val2 = 0.0;
double V2 = 0.0;
double R_lm = 0.0;
double Lux = 0.0;

// ===== VARIABILI DI SERVIZIO END

void setup()
{
    Wire.begin(); // join i2C bus (address optional for master)
}

void loop()
{
    // Inizio elaborazione

    // Lettura canale 1 ADC (channel 0) - due byte esadecimale
    // La prima istruzione imposta la richiesta di un singolo campionamento
    // La seconda istruzione attende per il tempo necessario ad ADC per campionare
    // La terza legge il buffer con il valore letto
```

e che alza, anche se di poco, la temperatura effettivamente misurata.

Il fattore di dissipazione è reperibile anch'esso nel datasheet e per il nostro termistore vale  $K = 6$

Il valore in gradi da sottrarre al risultato della formula di Steinhart-Hart è dato dalla formula

$$Td = V \cdot V / K \cdot R_{th}$$

Ancora una considerazione: le formule appena descritte utilizzano nei calcoli il valore della resistenza che il termistore assume alla temperatura di misurazione, mentre all'ingresso del convertitore ADC noi invece leggiamo il valore della tensione ai capi dell'NTC, che è funzione della resistenza che l'NTC assume in base alla temperatura.

Sapendo che ciascuno "scalino" del valore letto dal convertitore ADC a 16 bit di precisione corrisponde una tensione di 6,5 microvolt, possiamo calcolare il valore di tensione corrispondente al valore decimale della lettura.

$V = \text{Valore ADC in decimale per } 0,65 \text{ microvolt}$

Per passare dal valore di tensione alla resistenza utilizziamo la formula di trasferimento:

$$R_{th} = (R1 \cdot V) / (V_{IN} - V)$$

Nel **Listato 2** potete analizzare la conversione delle formule descritte nel programma in Python che descriviamo nel seguito dell'articolo. Nel caso in cui il datasheet del termistore in nostro possesso non ci fornisca i valori dei coefficienti da utilizzare con

(Continua)

## Listato 1 (seguito)

la formula di Steinhart-Hart, molto probabilmente ci fornirà i valori per utilizzare un secondo metodo che si basa sul parametro B del termistore, un valore costante che è possibile utilizzare con l'equazione:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \ln\left(\frac{R}{R_0}\right)$$

Il valore di temperatura ottenuto ha una precisione più grossolana rispetto al metodo precedente, e la formula produce risultati attendibili solo nell'intervallo di temperatura indicato nel datasheet. Il valore  $R_0$  è il valore di resistenza del termistore alla temperatura di riferimento  $T_0$ , in genere i soliti 25 gradi Centigradi. Anche questi valori si trovano nel datasheet, e nel nostro caso  $R_0 = 10\text{k}\Omega$  alla temperatura  $T_0$  di 298,15 K (25 C).

Nel **Listato 2** trovate la codifica in Python anche di questo metodo di calcolo.

### FOTORESISTENZA

Passiamo ora a descrivere come misurare la luminosità in Lux utilizzando la fotoresistenza.

La fotoresistenza è un componente elettronico la cui resistenza è inversamente proporzionale all'intensità della luce che riceve. Dalle specifiche fornite dal costruttore del sensore, la resistenza per un illuminamento di 0 lux ha un valore almeno di 1 M $\Omega$ , mentre a 10 lux varia da 8 a 20 k $\Omega$ . Per comprendere meglio questi dati è utile introdurre la definizione di lux. Il lux (simbolo lx) è l'unità di misura dell'illuminamento adottata dal Sistema Internazionale. Un lux è pari a un lumen su metro quadrato; il lumen è l'unità di misura del flusso luminoso. Di seguito sono elencati alcuni esempi di

```
Wire.beginTransaction(adc_address1);
Wire.write(adc_channel1);
Wire.endTransmission();
delay (700);
Wire.requestFrom(adc_address1,2.0);

val_0 = Wire.read(); // receive high byte (overwrites previous reading)
// delay (200);
val_1 = Wire.read();
channel_0 = int(val_0) * 255.0 + int(val_1);
dec_val = channel_0;

// Trasformazione in tensione del valore letto
V = (dec_val * 62.5) / 1000000.0;

// Calcolo della resistenza
R_th = (R1 * V) / (V_IN - V);

// Calcolo dei gradi Kelvin con la formula di Steinhart-Hart
logR = log(R_th / R_th0);
logR2 = pow(logR, 2.0);
logR3 = pow(logR, 3.0);
Stein = 1.0 / (A + B * logR + C * logR2 + D * logR3);

// Conversione in gradi Celsius e applicazione fattore di dispersione
Celsius = Stein - 273.15 - pow(V, 2.0) / (K * R_th);

// Stampa del risultato
printf("Steinhart - Celsius = %f\n", Celsius );

// Calcolo dei gradi Kelvin con la formula del parametro B
RRo = (R_th / 10000.0);
logRRo = log(RRo);
parB = 1.0 / ((1.0 / 298.15) + (logRRo / pB));

// Conversione in gradi Celsius e applicazione fattore di dispersione
Celsius2 = parB - 273.15 - pow(V, 2.0) / (K * R_th);

// Stampa del risultato
printf("parametro B - Celsius = %f\n", Celsius2 );

// Lettura canale 2 ADC - due byte esadecimale
// La prima istruzione imposta la richiesta di un singolo campionamento
// La seconda istruzione attende per il tempo necessario ad ADC per campionare
// La terza legge il buffer con il valore letto

Wire.beginTransaction(adc_address1);
Wire.write(adc_channel2);
Wire.endTransmission();
delay (700);
Wire.requestFrom(adc_address1,2.0);

val_0 = Wire.read(); // receive high byte (overwrites previous reading)
// delay (500);
val_1 = Wire.read();
channel_1 = int(val_0) * 255.0 + int(val_1);
dec_val2 = channel_1;
// Trasformazione in tensione del valore letto
V2 = (dec_val2 * 62.5) / 1000000.0;

// Calcolo della resistenza
R_lm = (R1 * V2) / (V_IN - V2);

// Calcolo della luminosità
Lux = pow((R_lm / R_fr0), (1.0/-Pend));

// Stampa del risultato
printf("Lux = %f\n", Lux );

}

int main () {
int x = 1;
  setup();
  while(x == 1) {
    loop();
    x = 0;
    break;
    delay(5000);
  }
  return (0);
}
```



# Istruzioni libreria Smbus

Di seguito il riassunto delle funzioni principali fornite dalla libreria smbus per il linguaggio Python.

All'inizio di ciascun programma che utilizza il bus i2C è necessario importare la libreria con l'istruzione.

```
from smbus import SMBus
```

Prima di configurare o accedere ad un device sul bus è necessario "aprire" il bus stesso creando un'istanza del bus con l'istruzione

```
bus = SMBus(1)
```

le istruzioni di comunicazione sul bus sono metodi (funzioni) richiamabili nella forma:

```
bus.read_byte_data(0x68,0x88)
```

Nella tabella che segue sono indicati i principali metodi utili per progetti come il nostro.

| Metodo                                     | Descrizione               | Parametri                  |
|--------------------------------------------|---------------------------|----------------------------|
| <code>read_byte(addr)</code>               | Legge un byte             | Indirizzo                  |
| <code>write_byte(addr,val)</code>          | Scrive registro config.   | Indirizzo, comando         |
| <code>read_byte_data(addr,cmd)</code>      | Legge un byte di dati     | Indirizzo, comando         |
| <code>write_byte_data(addr,cmd,val)</code> | Scrive un byte di dati    | Indirizzo, comando, Valore |
| <code>read_word_data(addr,cmd)</code>      | Legge una parola di dati  | Indirizzo, comando         |
| <code>write_word_data(addr,cmd,val)</code> | Scrive una parola di dati | Indirizzo, comando, Valore |

illuminamento espressi in lux:

- la luce del sole mediamente varia tra i 32.000 lx (32 klx) e i 100.000 lx (100 klx);
- sotto i riflettori degli studi televisivi si hanno circa 1.000 lux (1 klx);
- in un ufficio luminoso si hanno circa 400 lx;
- la luce della luna è pari a circa 1 lx.

Nel datasheet del sensore si trova il grafico che esprime l'andamento della resistenza, espressa in kΩ, in funzione dell'illuminamento, espresso in lux (**Fig. 2**). Prestate attenzione al fatto che il grafico è in scala logaritmica e quindi l'equazione che lega luminosità e resistenza è l'equazione di una "retta" in forma logaritmica:

$$\log R = -a \log IL + \log R1$$

In questa formula:

- R = resistenza del fotorestore;
- a = pendenza della retta chiamata valore Gamma (fornito dal costruttore e nel nostro caso uguale a 0,7);

- IL = illuminamento espresso in lux;
- R1 = resistenza del fotorestore in condizioni di illuminamento IL unitario (ricavabile dal grafico e comunque nel nostro caso uguale a 70kΩ).

Da questa equazione è semplice ricavare il valore di illuminamento IL espresso in lux, in funzione degli altri dati. Si ottiene così:

$$\log \frac{R}{R1} = \log IL^{-a} \rightarrow IL = \sqrt[-a]{\frac{R}{R1}}$$

Questa è la formula che abbiamo utilizzato nel programma Python per ricavarvi il valore di illuminamento IL. Per eseguire tutti questi calcoli abbiamo preparato il programma in linguaggio Python visibile nel **Listato 2** e scaricabile dal sito della rivista [www.elettronica.in.it](http://www.elettronica.in.it), che ci permette di introdurre sia la modalità di utilizzo del bus i2C in un linguaggio di programmazione, sia le istruzioni Python utili per gestire stringhe di dati, conversioni di valori ed esecuzione dei calcoli che realizzano le

formule descritte in precedenza, ma utilizzabili in tutti i casi di lettura di valori provenienti da sensori.

Prima di iniziare dobbiamo scaricare una libreria che ci fornisca le funzioni per poter comunicare con il bus i2C. Ne esistono diverse, noi abbiamo scelto smbus che è più diretta da usare, potete anche sperimentare con Quick2Wire che adotta un approccio orientato alle classi ed alla programmazione ad oggetti. Aggiorniamo il dizionario dei pacchetti con i soliti comandi (**Fig. 3**):

```
apt-get clean  
apt-get update
```

Installiamo la libreria smbus (**Fig. 4**):

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

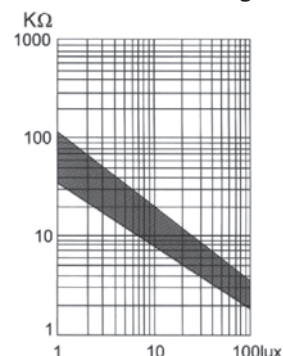
Guardando il **Listato 2**, all'inizio del programma troviamo gli include delle librerie che comprendono la libreria smbus per la gestione del bus i2C, la libreria time per la gestione degli intervalli di ritardo e la libreria math che contiene, tra le altre, la funzione logaritmo, necessaria per i nostri calcoli.

Nel listato abbiamo utilizzato il bus(1) che è il bus esposto sul connettore GPIO della rev. 2 di RaspberryPi. Se possedete RaspberryPi rev. 1, modificate l'istruzione

```
bus = SMBus(1)  
con  
bus = SMBus(0)
```

Il programma legge i valori di tensione in esadecimale dai canali analogici CH1 e CH2 ai

**Fig. 2**



## Listato 2

quali sono collegati rispettivamente il termistore NTC e la fotoresistenza. I valori letti sono convertiti in decimale e poi ne viene calcolato il valore di tensione corrispondente.

Dal valore di tensione si calcola la resistenza sia per l'NTC che per la fotoresistenza e poi si applicano le formule che abbiamo descritto in precedenza per ottenere i valori di temperatura e luminosità. Infine vengono stampati a video i risultati. Per la logica del programma vi rimandiamo al **Listato 2** e ai commenti che abbiamo inserito pressoché ad ogni istruzione. Approfondiamo solamente le istruzioni di gestione del bus i2C fornite dalla libreria smbus, al quale abbiamo dedicato il riquadro "Istruzioni libreria smbus" e la gestione delle stringhe.

Nel **Listato 2** potete individuare il seguente blocco di istruzioni.

```
hex_val = "00" + hex_val
hex_hi = hex_val[-2:]
hex_low = hex_val[-4:-2]
hex_raw = hex_hi + hex_low
dec_val = int(hex_raw,16)
```

Il loro scopo è di invertire i due byte ottenuti dalla lettura del canale ADC, in quanto il byte meno significativo (low), arriva sempre prima del byte più significativo (high). Una volta ricostruito il valore esadecimale con i byte nell'ordine corretto, viene calcolato il valore decimale equivalente.

La prima istruzione aggiunge due zeri in testa alla stringa esadecimale, per assicurarsi che la stringa stessa sia lunga almeno quattro caratteri (due byte esadecimali).

Le istruzioni successive dividono la stringa in porzioni, in particolare vengono estratti il byte high con l'operatore [pos\_i:pos\_f] che

```
#!/usr/bin/env python
# Import librerie
from smbus import SMBus
import math
import time
# Definizione costanti
#bus = SMBus(0) # RaspberryPi rev 1
bus = SMBus(1) # RaspberryPi rev 2
adc_address1 = 0x68 # Imposta indirizzo 00 ADC
adc_channel1 = 0x88 #
adc_channel2 = 0xA8 # Imposta indirizzi canali
adc_channel3 = 0xC8 # con risoluzione di 16 bit
adc_channel4 = 0xE8 #
R1 = 22000.0 # Resistenza R1
R2 = 22000.0 # Resistenza R2
R_th0 = 10000.0 # Resistenza riferimento termistore a 25 c
R_fr0 = 70000.0 # Resistenza fotoresistore con illuminamento IL unitario
V_IN = 5.0 # Tensione riferimento misura ADC
A = 0.00335402 # Steinhart-Hart Costante A
B = 0.000256985 # Steinhart-Hart Costante B
C = 2.62013e-6 # Steinhart-Hart Costante C
D = 6.38309e-8 # Steinhart-Hart Costante D
pB = 4100.0 # Costante parametro B
K = 6 # Fattore dissipazione K 6mV C
Pend = 0.7 # Valore Gamma fotoresistenza
# Inizio elaborazione
# Lettura canale 1 ADC - due byte esadecimale
# La prima istruzione imposta la richiesta di un singolo campionamento
# La seconda istruzione attende per il tempo necessario ad ADC per campionare
# La terza legge il buffer con il valore letto
bus.write_byte(adc_address1,adc_channel1)
time.sleep(0.07)
raw_val = bus.read_word_data(adc_address1,adc_channel1)
# Riordino del byte low e high del valore letto e trasf. In decimale
hex_val = hex(raw_val)[2:].rstrip('L')
hex_val = "00" + hex_val
hex_hi = hex_val[-2:]
hex_low = hex_val[-4:-2]
hex_raw = hex_hi + hex_low
dec_val = int(hex_raw,16)
# Trasformazione in tensione del valore letto
V = (dec_val * 62.5) / 1000000.0
# Calcolo della resistenza
R_th = (R1 * V) / (V_IN - V)
# Calcolo dei gradi Kelvin con la formula di Steinhart-Hart
logR = math.log(R_th / R_th0)
logR2 = logR**2
logR3 = logR**3
Stein = 1.0 / (A + B * logR + C * logR2 + D * logR3)
# Conversione in gradi Celsius e applicazione fattore di dispersione
Celsius = Stein - 273.15 - V**2 / (K * R_th)
# Stampa del risultato
print "Steinhart - Celsius = ",
print Celsius
# Calcolo dei gradi Kelvin con la formula del parametro B
RRo = (R_th / 10000.0)
logRRo = math.log(RRo)
parB = 1.0 / ((1.0 / 298.15) + (logRRo / pB))
# Conversione in gradi Celsius e applicazione fattore di dispersione
Celsius2 = parB - 273.15 - V**2 / (K * R_th)
# Stampa del risultato
print "parametro B - Celsius = ",
print Celsius2
# Lettura canale 2 ADC - due byte esadecimale
# La prima istruzione imposta la richiesta di un singolo campionamento
# La seconda istruzione attende per il tempo necessario ad ADC per campionare
# La terza legge il buffer con il valore letto
bus.write_byte(adc_address1,adc_channel2)
time.sleep(0.07)
raw_val2 = bus.read_word_data(adc_address1,adc_channel2)
# Riordino del byte low e high del valore letto e trasf. In decimale
hex_val2 = hex(raw_val2)[2:].rstrip('L')
hex_val2 = "00" + hex_val2
hex_hi2 = hex_val2[-2:]
hex_low2 = hex_val2[-4:-2]
hex_raw2 = hex_hi2 + hex_low2
dec_val2 = int(hex_raw2,16)
# Trasformazione in tensione del valore letto
V2 = (dec_val2 * 62.5) / 1000000
# Calcolo della resistenza
R_lm = (R1 * V2) / (V_IN - V2)
# Calcolo della luminosità
Lux = (R_lm / R_fr0)**(1.0/-Pend)
# Stampa del risultato
print "Lux = ",
print Lux
```

```

192.168.0.162 - PuTTY
root@raspberrypi:~# apt-get clean
root@raspberrypi:~# apt-get update

```

Fig. 3

```

192.168.0.162 - PuTTY
root@raspberrypi:~# apt-get install python-smbus

```

Fig. 4

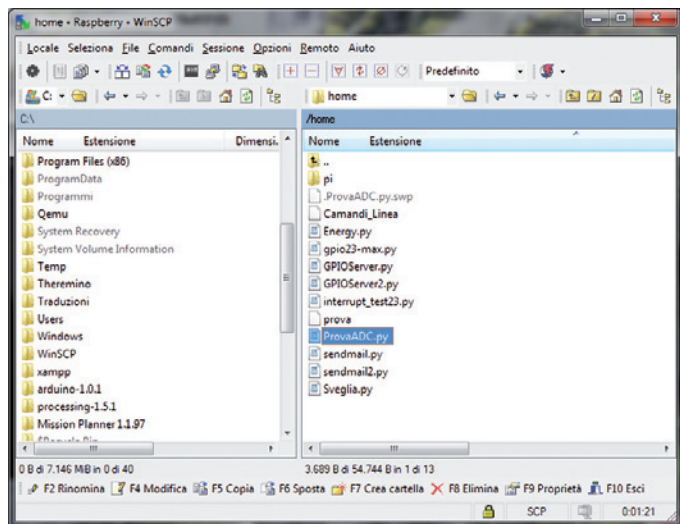


Fig. 5

```

192.168.0.162 - PuTTY
root@raspberrypi:~# modprobe i2c-dev
root@raspberrypi:~#

```

Fig. 6

```

192.168.0.162 - PuTTY
root@raspberrypi:~# python /home/ProvaADC.py
Steinhart Celcius - 22.5572726458
parametro B - Celsius - 22.6834833551
Lux - 175.952023202
root@raspberrypi:~#

```

Fig. 7

```

192.168.0.162 - PuTTY
root@raspberrypi:~# python /home/ProvaADC.py
Steinhart Celcius - 30.1969955354
parametro B - Celsius - 29.9397752651
Lux - 174.229899128
root@raspberrypi:~#

```

Fig. 8

permette di estrarre una porzione della stringa in base ai valori di posizione iniziale e finale della porzione desiderata. Per esempio l'istruzione della seconda riga estrae i caratteri dalla posizione -2 (dalla fine della stringa) alla fine, cioè gli ultimi due caratteri della stringa che rappresentano il byte più significativo della lettura (high). La terza riga estrae la porzione che rappresenta il byte meno significativo e che si trova davanti al precedente, cioè dalla posizione -4, sempre partendo dal fondo, alla posizione -2.

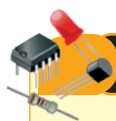
La quarta istruzione riordina i byte concatenando il byte high al byte low, mentre l'ultima istruzione converte il risultato finale nel corrispondente valore decimale. Per eseguire il programma basta copiarlo (o scriverlo) nella cartella /home (Fig. 5). Come ricordato nell'articolo precedente, se abbiamo appena acceso RaspberryPi dobbiamo dare anche il comando per caricare nel kernel il driver i2C (Fig. 6):

*modprobe i2c-dev*

Per eseguire il programma utilizziamo il comando (da root) (Fig. 7):

*python /home/ProvaADC.py*

per ottenere i risultati visibili sempre in Fig. 7. Dopo aver tenuto la resistenza NTC tra le dita per un po' di tempo proviamo a ridare il comando e vedremo la temperatura salire come in Fig. 8. Nei prossimi articoli vedremo come utilizzare programmi come questo per aggiornare un sito web, sempre ospitato su RaspberryPi, comunicando i valori letti via API HTTP.



## per il MATERIALE

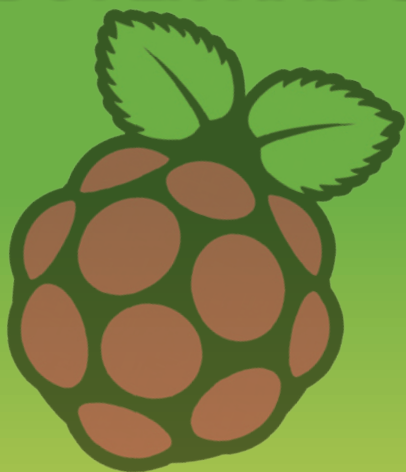
Tutti i componenti utilizzati in questo progetto sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista così come il programma di esempio che verrà presentato nel prossimo numero. Di questo shield Futura Elettronica ha realizzato la versione montata e collaudata (cod. FT1041M) disponibile al prezzo di 27 Euro, IVA compresa.

Il materiale va richiesto a:

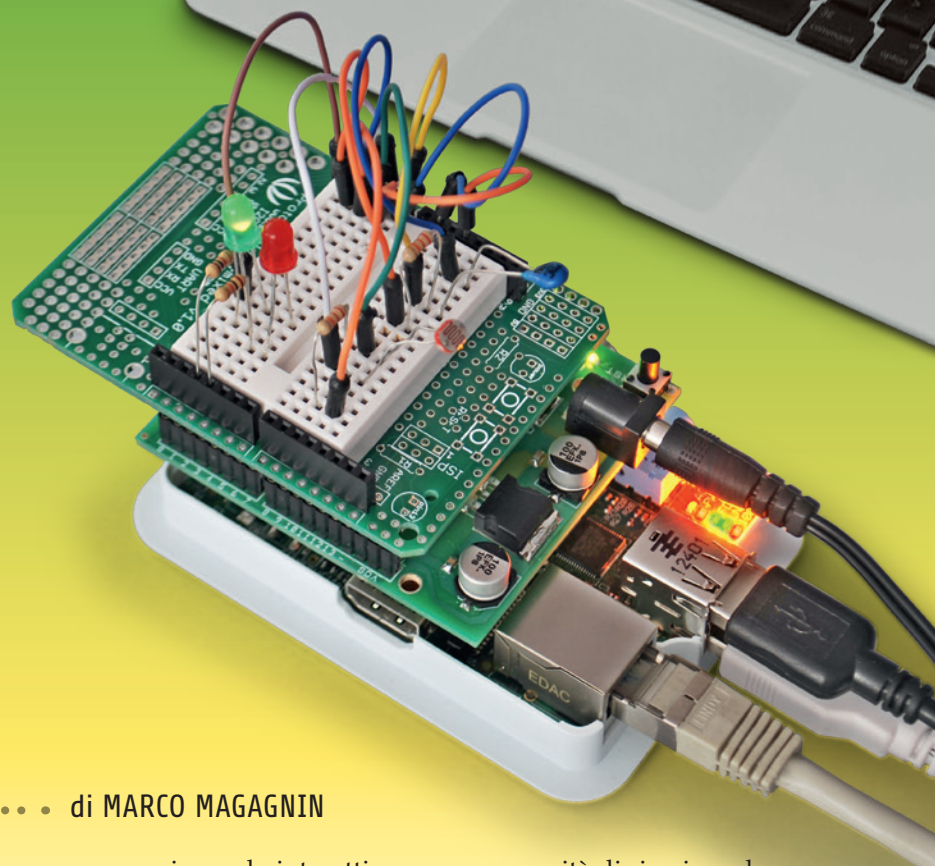
Futura Elettronica, Via Adige 11, 21013 Gallarate (VA)  
Tel: 0331-799775 - Fax: 0331-792287 - [www.futurashop.it](http://www.futurashop.it)



# TERMOSTATO WEB CON LO SHIELD DI ESPANSIONE ADC PER RASPBERRYPI



Completiamo la realizzazione del termostato web con la realizzazione della pagina HTML che include la dashboard emoncms e il pulsante per modificare le soglie di temperatura alle quali intervengono i sistemi che simulano il riscaldamento e il condizionamento.



..... di MARCO MAGAGNIN

**D**opo aver configurato il framework emoncms che gestisce gli indicatori grafici alimentati dalle misure dei sensori di temperatura e luminosità e lo stato della simulazione dei sistemi di riscaldamento e condizionamento, completiamo l'applicazione con i programmi di presentazione della pagina web.

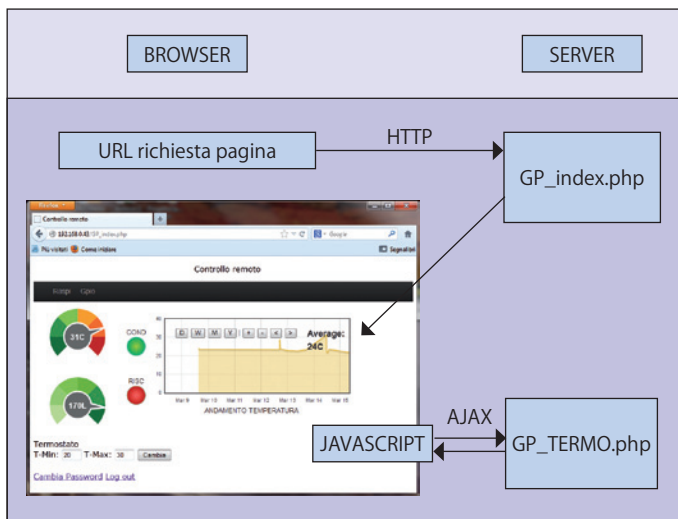
La particolarità principale è l'utilizzo della tecnica AJAX (Asynchronous JavaScript and XML) per gestire la configurazione delle soglie di temperatura

in modo interattivo, senza necessità di ricaricare la pagina ogni volta.

## PROGRAMMA DI GESTIONE DELLA PAGINA WEB

Alcune considerazioni sulla struttura dell'interfaccia web nel nostro progetto.

Vogliamo una pagina dove possano essere eseguiti comandi e impostazioni, la risposta ai quali non richieda di ricaricare ogni volta la pagina stessa. Il nostro obiettivo è di realizzare una pagina che, una



volta aperta se ne stia al suo posto accettando i nostri comandi e ritornando i risultati dopo la modifica interattiva dei soli campi interessati.

Prendiamo come riferimento lo schema di Fig. 1. Per ottenere questo risultato dobbiamo fare in modo che la pagina HTML possa eseguire delle azioni proprie, indipendenti dal server. Utilizzeremo il linguaggio Javascript che ci permette di inserire delle istruzioni (script) all'interno del codice HTML della pagina web, che verranno eseguite sul client a fronte di specifiche richieste. L'utilizzo di Javascript nelle pagine HTML è una delle tante tecniche che vanno sotto il nome di HTML dinamico (DHTML Dynamic HTML). Javascript permette di eseguire delle azioni all'interno della pagina web senza necessità di interessare il server. Questo significa che il browser che utilizziamo contiene al suo interno l'interprete Javascript per l'esecuzione delle istruzioni.

Per aggiornare i contenuti utilizziamo la tecnica AJAX che ci permette di gestire lo scambio di dati in sottofondo tra la pagina web ed il server, in modo asincrono.

In sintesi, per gestire la nostra pagina web dobbiamo adottare tecniche di pagine web dinamiche (PHP) coadiuvate da tecniche Javascript e di gestione interattiva dei contenuti (AJAX). Crediamo che basti.

Per restare il più semplici possibili utilizzeremo queste tecniche nel modo più elementare possibile, giusto per dare nota della loro esistenza. Comunque qui il terreno si fa un po' in salita.

Nello schema di Fig. 1 abbiamo visto la pagina GP\_index.html che viene richiamata per prima e che si colloca come coordinatore di tutte le altre funzionalità. L'azione "Cambia" viene eseguita dal programma GP\_TERMO.php separato rispetto al programma che gestisce la pagina principale, che viene richiamato ad ogni pressione del pulsante.

**Fig. 1**

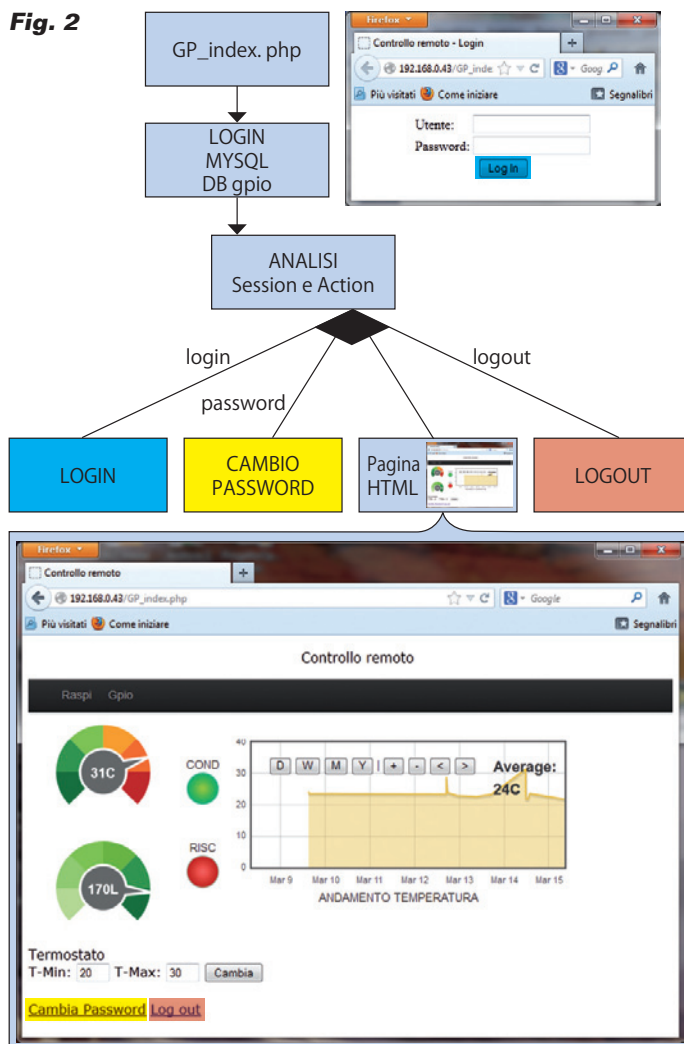
Vediamo di descrivere il funzionamento del programma GP\_index.php, almeno nelle sue linee essenziali.

Nella Fig. 2 i rettangoli colorati che evidenziano le singole funzionalità nelle immagini delle pagine web corrispondono ai colori dei blocchi nel diagramma di struttura, che elaborano le funzionalità stesse, e che trovano riscontro nelle corrispondenti sezioni del listato.

Quando un utilizzatore esterno richiede la pagina, come prima operazione il programma si connette al database "gpio" ospitato dal server MySQL. Poi verifica se l'utente si collega per la prima volta, in tal caso restituisce la pagina di login con la richiesta di Userid e Password. Ricordiamo che l'utente è "admin" e la password "gpio".

Dopo il buon esito del login viene presentata la pagina principale del progetto. Questa pagina è divisa in due aree, un'area di sola visualizzazio-

**Fig. 2**



ne con la rappresentazione grafica dello stato del sistema e un'area di "comando" con i campi e i pulsanti necessari ad attivare le diverse funzionalità. Sono presenti due collegamenti per cambiare la password e per eseguire il logout. Abbiamo lasciato come ultima riga un campo dove vengono riportati i messaggi di ritorno del programma attivato con la tecnica Ajax.

Tralasciamo la spiegazione dei rami di Primo collegamento, Cambio password e Logout per concentrarci sulle sezioni di programma che gestiscono il pulsante "Cambia" e i campi associati.

Una sola osservazione riguarda l'inclusione nella nostra pagina della pagina grafica di emoncms, che abbiamo risolto con l'inclusione della pagina mediante un tag `<iframe>`.

**"Iframe"** significa **"inline frame"**: in una pagina che non utilizza una struttura a frame è possibile creare un frame "al volo" in qualsiasi punto del codice HTML, grazie a questo tag.

Possiamo specificare la larghezza e l'altezza del tag, mentre gli attributi di visualizzazione sono gli stessi del tag `<frame>`: si tratta di una vera e propria finestra verso l'esterno contenuta in un documento ordinario. Come riferimento abbiamo utilizzato il link alla pagina di menu del nostro server emoncms, utilizzando l'APIKey di sola lettura per l'autorizzazione all'accesso, come abbiamo già visto alla fine della personalizzazione della pagina emoncms.

Ora analizziamo più approfonditamente il meccanismo AJAX associato ai comandi operativi.

Vediamo che nella sezione di programma che gestisce la pagina principale è presente un tag `<script type="text/javascript">` al quale segue un lungo elenco di istruzioni decisamente non PHP, terminato dal tag `</script>`. Le istruzioni contenute

nel blocco sono istruzioni Javascript, che si occupano della gestione della parte interattiva della pagina HTML.

Queste istruzioni vengono passate all'interno della pagina HTML per essere eseguite, a richiesta, direttamente all'interno del browser. Javascript è un linguaggio di scripting e quindi richiede un interprete per poter essere eseguito. Ovviamente l'interprete fa parte del codice dei browser più moderni.

Come potete notare i moduli javascript sono organizzati in funzioni. Ciascuna delle quali può prevedere o meno dei parametri.

Di seguito vediamo il frammento di codice HTML che realizza la funzionalità di modifica delle soglie di temperatura. Possiamo modificare i valori nei campi T-Min e T-Max e poi premere il pulsante "Cambia" per inviare i valori al server. Rispetto al codice di una normale pagina HTML notiamo i tag id all'interno delle sezioni di descrizione degli elementi, caselle di testo e pulsante, in più, nel tag dell'elemento "button" è presente la gestione dell'evento onclick, che innesca la funzione javascript "spedisci Termo", passandogli come parametri i valori dei campi tmin e tmax (**Listato 1**).

Alla pressione del pulsante "Cambia" viene attivata la funzione Javascript "spedisci Termo" che, ricordiamo, viene eseguita all'interno del browser, recepisce i due valori di temperatura e inserisce il messaggio di attesa nella casella "descrizione", ultima in fondo alla pagina, che riporta i messaggi di comunicazione tra gli "attori" AJAX e poi, in squisito stile programmazione ad oggetti istanzia una richiesta al server web di nome "richiesta", per la quale prenota l'esecuzione della funzione `updateasincronoTermo` per quando verranno ricevute delle risposte da parte del server, imposta una chiamata con metodo

## Listato 1

```
Termostato
<br>
T-Min: <input type="text" name="Tmin" id="tmin" value="" . $xMin . \' size="3"/>
T-Max: <input type="text" name="Tmax" id="tmax" value="" . $xMax . \' size="3"/>
      <input type="button" value="Cambia" id="buttonTermo"
      onclick="spedisciTermo(document.getElementById(\'tmin\').value,
      document.getElementById(\'tmax\').value);"/>
```

## Listato 2

```
function spedisciTermo(Tmin,Tmax)
{
    document.getElementById("descrizione").innerHTML = "--- Attendi ... ---";
    server = "AJ_TERMIO.php?action=Termo" + "&Tmin=" + Tmin + "&Tmax=" + Tmax;
    richiesta = new XMLHttpRequest();
    richiesta.onreadystatechange = updateasincronoTermo;
    richiesta.open("GET", server, true);
    richiesta.send(null);
}
```



### Listato 3

```
function updateasincronoTermo(){
    if ((richiesta.readyState == 4) && (richiesta.status == 200))
    {
        esito = richiesta.responseText;
        iparr = esito.split(",");
        Stato = iparr[0];
        Tmin = iparr[1];
        Tmax = iparr[2];
        document.getElementById("tmin").value = Tmin;
        document.getElementById("tmax").value = Tmax;
        document.getElementById("descrizione").innerHTML = esito;
    }
}
```

### Listato 4

```
$MySQLUsername = "gpio";
$MySQLPassword = "pi";
$MySQLHost = "localhost";
$MySQLDB = "gpio";

$dbConnection = mysql_connect($MySQLHost, $MySQLUsername, $MySQLPassword);
mysql_select_db($MySQLDB, $dbConnection);

$query3 = mysql_query("SELECT xMin, xMax FROM xTemp WHERE xLocale = 1;");
$tempRow = mysql_fetch_assoc($query3);
$xMin = $tempRow['xMin'];
$xMax = $tempRow['xMax'];

mysql_close();
```

### Listato 5

```
<iframe src="http://192.168.0.43/emoncms/admin?id=1&apikey=5a79c4482e414309e9b6b98dc8651f2a" width="800" height="300" frameborder="0" ></iframe>
```

GET e la invia. La stringa null nel metodo send è associata al fatto che abbiamo eseguito la chiamata con il metodo GET. Il metodo GET passa i parametri al server direttamente nell'URL, dopo il punto di domanda, mentre il metodo POST passa i parametri in una stringa separata, che può contenere molti più dati di quelli gestibili con il metodo GET. Nel caso avessimo usato il metodo POST, i dati associati alla chiamata andrebbero inseriti al posto del "null" (Listato 2).

A quest'ora il server avrà già risposto da tempo, infatti vedrete molto raramente la scritta ---Attendi---, nel qual caso viene attivata la funzione `updateasincronoTermo()` che, se non vi sono errori - cioè lo stato della risposta è uguale a 4 e lo stato della richiesta è uguale a 200 - intercetta il messaggio di ritorno, costituito da campi divisi da virgole (lo ab-

biamo costruito noi nel programma lato server, come vedremo presto), lo "spacchetta" in campi singoli e aggiorna i valori nei campi `tmin` e `tmax` con i valori ricevuti di ritorno. Poi riporta l'intero messaggio di ritorno nel campo "descrizione". Potete anche eliminare questa informazione scrivendo "" nel campo o quello che preferite. Con questo abbiamo finito e come avrete notato, durante tutto questo lavoro la pagina non viene assolutamente ricaricata, ma rimane fissa al suo posto (Listato 3). Vediamo nell'estratto di codice successivo (Listato 4) come reperire dei dati da un database MySQL in PHP. Nel programma `GP_index.php` ci serve accedere ai valori di soglia minimo e massimo impostati per la funzione termostato, per evidenziarli nella pagina web.

Iniziamo con la connessione al database, per la quale è ne-

cessario specificare i parametri dell'host dove risiede il database, il nome del database, l'utente e la password abilitati alla sua gestione.

Si esegue la connessione con il metodo `mysql_select_db` e poi possiamo eseguire tutti i comandi SQL necessari; nel nostro caso sono la SELECT nella tabella `xTemp` dei valori `xMin` e `xMax` per il locale 1. Per l'esecuzione dei comandi utilizziamo il metodo `mysql_query` che restituisce il risultato della query nella collezione `$query3`. Nel nostro caso viene restituita una sola riga, in caso di restituzione di più righe è necessario eseguire un ciclo for che esaurisca le righe di risposta. Per ciascuna riga si indirizza il "cursore" della query alla riga in esame con il metodo `fetch_assoc` che restituisce i campi della riga nell'array dizionario di nome `$tempRow` e da questo si leggono i valori dei campi indicizzandoli con il loro nome, `xMin` e `xMax`. Prima di terminare il programma si chiude la connessione al database.

Visto che stiamo parlando del programma `GP_index.php`, prima di passare ad un altro

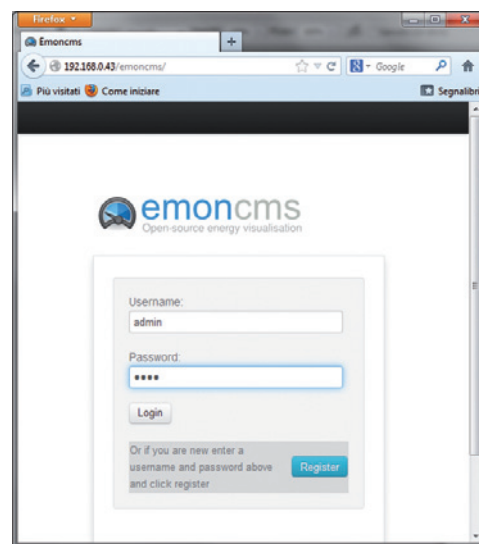


Fig. 3

argomento sistemiamo la configurazione dell'URL e dell'API-Key necessari a richiamare la dashboard emoncms nell'iframe del programma stesso.

Per fare questa operazione dobbiamo riaprire l'applicazione di configurazione di emoncms. Nel browser inseriamo l'URL <indirizzo-IP-RaspberryPi>/emoncms.

Nella finestra di **Fig. 3** inserite utente e password, noi vi avevamo suggerito utente "admin" e password "gpio". Selezionate il pannello "Dashboard" e poi cliccate sull'icona share (**Fig. 4**), copiate dal campo URL del browser il link alla pagina appena creata (**Fig. 5**) e inseritelo nel reference dell'iframe nel programma GP\_index.php, insieme all'APIkey di lettura (**Listato 5**).

## GESTIONE TERMOSTATO

Per la gestione del termostato abbiamo già descritto il programma, che scheduleremo a tempo e che legge i sensori collegati al convertitore analogico digitale, calcola i valori di temperatura e luminosità, li confronta con i valori di soglia sulla tabella xTemp del database e aggiorna il server emoncms con i valori letti e lo stato dei LED che rappresentano l'impianto di riscaldamento e di condizionamento.

Vedremo più avanti come utilizzare lo scheduler di sistema "cron" per eseguire il programma a intervalli determinati, risparmiando molte risorse rispetto all'esecuzione di cicli temporizzati con il programma sempre in esecuzione.

Dobbiamo ancora analizzare il programma PHP che realizza la funzione AJAX nella pagina web interattiva.

Il programma GP\_TERMO.php viene richiamato dalla richiesta in formato URL che proviene dalla

funzione javascript della pagina web, innescata dalla pressione del pulsante "Cambia" (**Listato 6**).

A questo punto sotto il controllo del server Apache2 viene eseguito il programma GP\_TERMO.php, visibile nel Listato 2, che esegue la connessione al database con le stesse modalità descritte nel paragrafo dedicato alla pagina GP\_index.php e poi esegue l'aggiornamento (UPDATE) del record nella tabella xTemp con i valori ricevuti nell'URL.

Alla fine viene restituito con l'istruzione print il semplice messaggio di ritorno

OK, 20, 28, Termo

con i campi separati da virgole che verranno intercettati dalla funzio-

ne javascript updateasincronoTermo() nella pagina web.

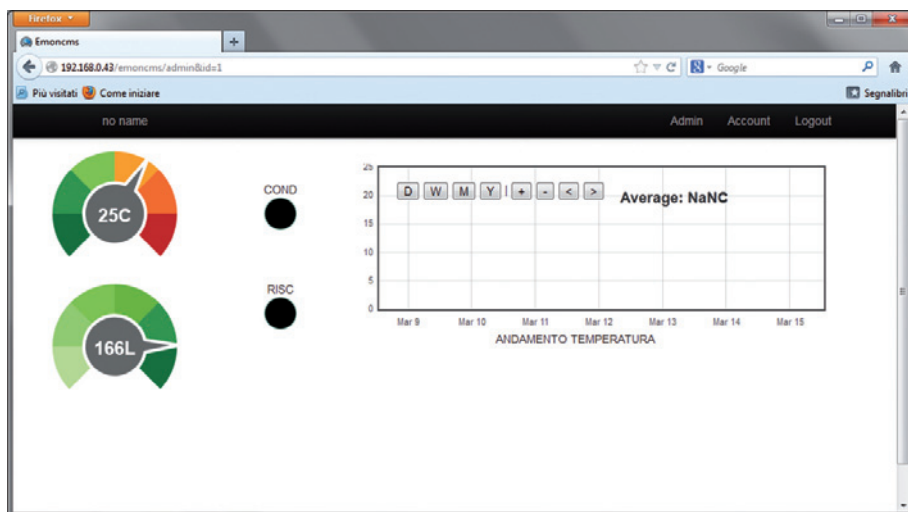
## SCHEDULATORE CRON

Possiamo lanciare a mano il programma ProvaADC2.py per fare qualche prova, poi conviene adottare una struttura più solida per eseguire il programma in modo automatico.

Introduciamo ora il servizio di schedulazione cron che ci permetterà di inviare una nuova serie di dati al nostro sito emoncms ad intervalli prestabiliti, per esempio ogni minuto.

Il servizio cron è il servizio predefinito in GNU/Linux per la gestione dell'esecuzione di programmi e comandi a tempo.

Per funzionare, cron si appoggia a una tabella che contiene le



**Fig. 4**

```
GNU nano 2.2.6      File: /tmp/crontab.pBXJ9s/crontab      Modified
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
*1 * * * * python /home/ProvaADC2.py > /dev/null
```

**Fig. 5**

## Listato 6

```
<?php
// GP_TERMO.php

$MySQLUsername = "gpio";
$MySQLPassword = "pi";

$MySQLHost = "localhost";
$MySQLDB = "gpio";

If (($MySQLUsername == "USERNAME HERE") || ($MySQLPassword == "PASSWORD HERE")){
    print 'ERROR - Please set up the script first';
    exit();
}

$dbConnection = mysql_connect($MySQLHost, $MySQLUsername, $MySQLPassword);
mysql_select_db($MySQLDB, $dbConnection);

$action = $_GET['action'];
$Tmin = mysql_real_escape_string($_GET['Tmin']);
$Tmax = mysql_real_escape_string($_GET['Tmax']);
if ($action == "Termo"){
    mysql_query("UPDATE xTemp SET xMin='$Tmin', xMax='$Tmax' WHERE xLocale=1;");
    mysql_close();
}
print 'OK, ' . $Tmin . ' , ' . $Tmax . ' , ' . $action;
?>
```

“istruzioni” circa i programmi da eseguire e in quali istanti lanciarli. La tabella utilizzata da cron si gestisce con la funzionalità “crontab” che ha la sintassi:

crontab <opzione> 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 in **Fig. 5**: Ogni riga del file di configurazione ha la struttura riportata nel **Comando 1**. Ciascun parametro può essere configurato come serie di valori es: 0,3,7,9 come un intervallo es: 3-7 oppure un misto dei due es: 0,3,4-6,9 oppure ancora come frequenza \*/5 (ogni 5 minuti, ore, etc.). Nel nostro caso vogliamo che il programma ProvaADC2.py venga eseguito ogni minuto, quindi digitiamo la linea di configura-

zione:

```
*1 *** python /home/ProvaADC2.py > /dev/null
```

che significa: esegui il programma che misura temperatura e luminosità una volta al minuto di ogni ora, giorno e mese, perdendo tutti gli eventuali messaggi di log che possano venire prodotti. 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 eseguito il programma ProvaADC2.py con gli aggiornamenti conseguenti.

Per modificare le impostazioni di cron richiamiamo la funzionalità

*crontab -e*

Per vedere le impostazioni utilizziamo l’opzione *crontab -l*

mentre per interrompere l’esecu-

zione automatica digitiamo

*crontab -r*

## LANCIAMO GPIOSERVER3.PY IN SOTTOFONDO

Avrete ovviamente notato che quando viene lanciato il programma GPIOServer3.py, questo tiene impegnata la finestra di terminale, impedendovi di eseguire altri comandi. A questo punto potremmo aprire più finestre per lanciare più programmi, che può essere una buona soluzione per il debug, ma ci costringe in ogni caso a tenere aperta la connessione SSH per far funzionare il programma. Una classica architettura “embedded” deve essere in grado di funzionare completamente abbandonata e senza connessioni di controllo. È ora di affrontare la gestione dei processi in background, ovvero senza interazione e senza necessità di una finestra di terminale aperta.

Descriviamo due modalità che agiscono, per così dire, a diverse profondità nel sistema operativo. Il primo metodo consiste nel consegnare l’esecuzione del programma ad un processo che, una volta lanciato, libera la linea di comando ma resta comunque collegato alla vita della finestra. Quando chiudiamo la finestra il processo viene interrotto. Per eseguire il programma in questa modalità si aggiunge una “&” alla fine del comando come, per esempio:

```
python /home/GPIOServer3.py &
```

Si dà invio e poi ancora invio: il programma resta in esecuzione in sottofondo e la linea di comando resta utilizzabile. Vediamo come interrompere il processo senza necessariamente chiudere la finestra “madre”. Quando abbiamo lanciato il

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

### Comando 1

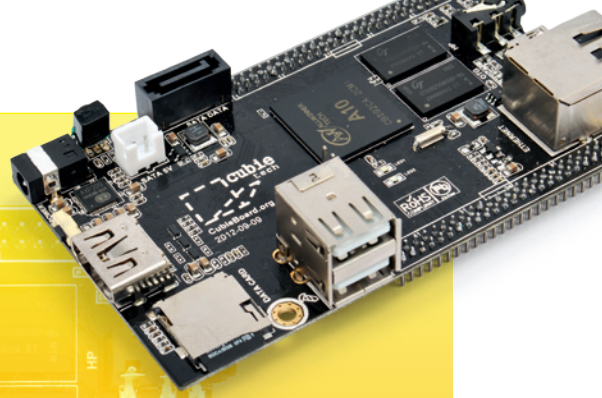


# CubieBoard, nuova scheda ARM Android/Linux

Continua l'uscita sul mercato di veri e propri "computer" con a bordo processori ARM RISC. La scheda Cubieboard, delle dimensioni di poco superiori a quelle di RaspberryPi (circa 10 x 6 cm), è dotata di un SoC Allwinner A10 che contiene un processore Cortex A8 - limitato ad 1 GHz per contenere i consumi - e una GPU Mali-400, 1 GB di RAM e 4GB di memoria Flash a bordo. Nasce con Android 4.0.4 già installato, ma è in grado di far girare diverse distribuzioni di GNU/Linux come Ubuntu 10.04 e 12.04, Arch Linux, Fedora ARM e Raspbian "wheezy" (sì, la stessa distribuzione di RaspberryPi) e quindi può essere utilizzato in diversi ambiti, dalla



smart TV, per la quale è previsto un sensore infrarosso in modo da poterlo comandare da un telecomando, fino a diventare una vera e propria pattaforma di sviluppo, coadiuvata dalla presenza di un connettore SATA per il collegamento di un disco fisso "vero". Completano la dotazione il connettore HDMI, lo slot SDCard, 2 USB, la porta Ethernet, ingresso e uscita audio e ben 96 pin di I/O che includono tutti i bus di utilizzo "elettronico" I2C, SPI, seriali, VGA e video composito. Non mancheremo di parlarne.



processo con il metodo indicato sopra abbiamo ricevuto in risposta un numero di processo. Tutti i processi sono identificati da un numero. Possiamo vedere l'elenco dei processi attivi con i relativi identificativi utilizzando il comando ps (process status):

```
ps -e
```

Per elencare solo i processi che ci interessano possiamo filtrarli concatenando il comando ps con il comando grep (general regular expression print) che ricerca in una lista di file le stringhe di testo che soddisfano l'espressione di filtro

```
ps -e | grep python
```

Troveremo elencati solo i processi eseguiti dall'interprete python, cioè i nostri.

Per controllare le caratteristiche di un singolo processo, ad esempio per verificare qual è il programma in esecuzione utilizziamo il comando

```
ps numero-processo
```

Per terminare intenzionalmente un processo che gira in sottofon-

do, dobbiamo conoscere il numero di processo che lo identifica ed utilizzare il comando

```
kill numero-processo
```

La seconda modalità ci permette di eseguire i programmi in processi ancora più profondi, totalmente svincolati dalla finestra di terminale. Lanciamo il programma con il comando nohup e il carattere "&" alla fine:

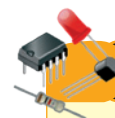
```
nohup python /home/GPIOServer3.py &
```

Con il programma lanciato in questo modo, possiamo tranquillamente chiudere la finestra di terminale e anche disconnetterci dalla sessione SSH con il programma che continuerà a funzionare. Nohup significa No Hang Up ed è nato ai tempi delle connessioni su rete commutata via modem, quando ci si connetteva in remoto utilizzando le normali linee telefoniche per amministrare i sistemi e lanciare processi che dovevano continuare a funzionare anche dopo avere chiuso la connessione (telefonata).

Anche in questo caso, per terminare intenzionalmente il proces-

so si utilizza il comando *kill numero-processo*

Se i processi che lanciamo utilizzano lo stdout, cioè la destinazione - per esempio - delle istruzioni print, che normalmente viene indirizzata sulla finestra terminale, quando li lanciamo in sottofondo possiamo oscurare lo stdout oppure reindirizzarlo verso un file, per esempio per poter leggere



## per il MATERIALE

Tutti i componenti utilizzati per realizzare lo shield di espansione per RaspberryPi utilizzato in questo progetto e descritto nel numero 173 sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista così come i programmi di esempio. Di questo shield Futura Elettronica ha realizzato la versione montata e collaudata (cod. FT1041M) disponibile al prezzo di 27 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>

**Fig. 6**

```

192.168.0.43 - KITTY
root@raspberrypi:~# nohup python /home/GPIOServer3.py > /dev/null &
[1] 4221
root@raspberrypi:~# nohup: ignoring input and redirecting stderr to stdout

root@raspberrypi:~#

```

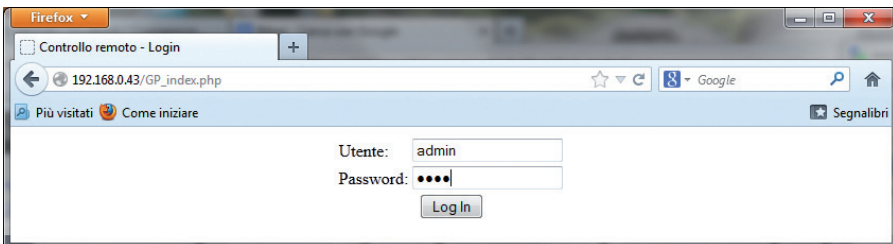
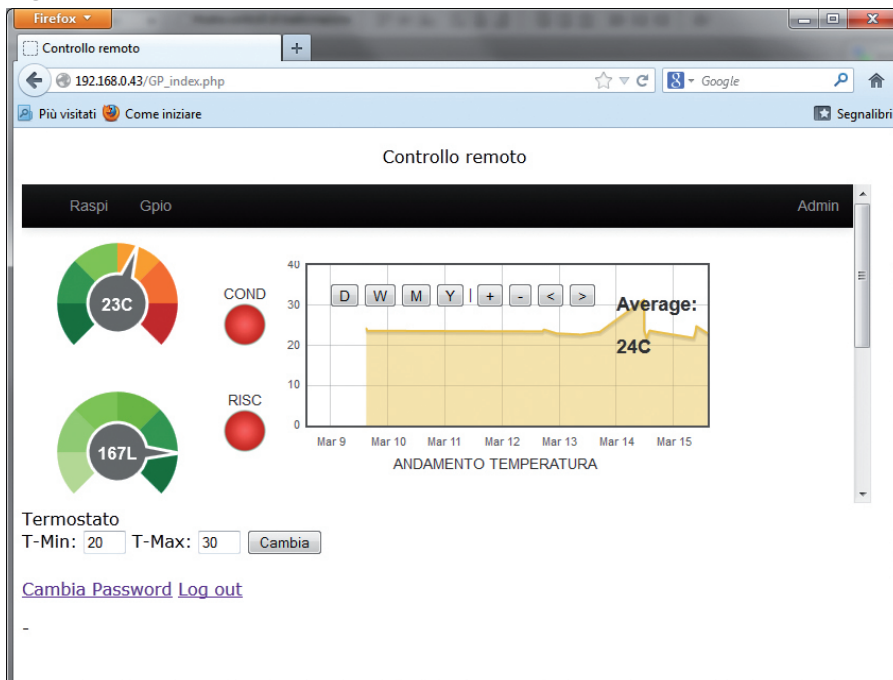
**Fig. 7**

```

192.168.0.43 - KITTY
root@raspberrypi:~# nohup python /home/GPIOServer3.py > /dev/null &
[1] 4221
root@raspberrypi:~# nohup: ignoring input and redirecting stderr to stdout

root@raspberrypi:~# ps -e | grep python
4221 pts/0    00:00:04 python
root@raspberrypi:~# ps 4221
  PID TTY          STAT       TIME COMMAND
 4221 pts/0    S          0:04      python /home/GPIOServer3.py
root@raspberrypi:~# kill 4221
  PID TTY          STAT       TIME COMMAND
[1]+  Terminated                  nohup python /home/GPIOServer3.py > /dev/null
root@raspberrypi:~#

```

**Fig. 8****Fig. 9**

i messaggi successivamente, a scopo di debug (Fig. 6 e Fig. 7). Nel primo caso usiamo il comando

```
nohup python /home/GPIOServer3.py > /dev/null &
```

nel secondo caso

```
nohup python /home/GPIOServer3.py > /home/GPIOServer.log &
```

Per provare l'applicazione usciamo da emoncms con "Logout", richiamiamo la pagina principale con l'URL

```
http://<IP-RaspberryPi/GP_index.php
```

alla pagina di login rispondiamo con "admin" e "gpio" (Fig. 8). Cambiando i valori di soglia della temperatura e scaldando la resistenza NTC tenendola tra le dita possiamo vedere attivarsi i LED veri e simulati a seconda delle condizioni di temperatura. Nel caso in cui l'aspetto grafico della pagina non vi soddisfi, riaprite emoncms in amministrazione, modificate il layout e riprovate (Fig. 9). Eccoci alla fine del percorso che ci permette di progettare e realizzare applicazioni embedded in ambiente GNU/Linux in generale, non soltanto utilizzando RaspberryPi. Chiaramente si è trattato solo di un assaggio, ma speriamo di essere riusciti a trasferire le informazioni necessarie e capire dove andare a guardare quando avremo necessità particolari. Ai tempi dei libri, quello che abbiamo cercato di descrivere stava nascosto all'interno di una "mezza libreria" di manuali. Oggi con internet a disposizione è molto più semplice approfondire architetture e tecniche, a patto di sapere dove cercare. Buona sperimentazione. ■





# CONTROLLIAMO UNA 3DRAG A DISTANZA TRAMITE RASPBERRY PI

Una soluzione Embedded Linux per stampare in remoto e controllare la vostra 3Drag senza tenere impegnato un personal computer.

di FRANCESCO DE FABIANI

**S**tufi di dover lasciare il personal computer acceso per ore ed ore in garage, senza contare l'inutile spreco di energia, per stampare con la vostra 3D printer?

La soluzione è utilizzare come sostituto del notebook o del pc una scheda RaspberryPi. Dapprima abbiamo ipotizzato d'installare la distribuzione del

software Repetier-Host per linux direttamente su RaspberryPi, poi ci siamo imbattuti in una versione specifica per la piccola scheda marchiat col lampone. Il suo nome è Repetier-Server e fa al caso nostro perché creando una pagina web all'indirizzo IP di Raspberry è possibile sia controllare a distanza la stampante, sia stampare dei modelli

semplicemente caricando il G-Code sul server, senza la necessità di tenere il pc acceso. Per installare Repetier-Server è necessario recarsi all'indirizzo <http://www.repetier.com/repetier-server-download/> scaricare il file 'Repetier-Server 0.24 for Raspberry Pi' e seguire le istruzioni che riportiamo nel box dedicato. Al termine dell'installazio-

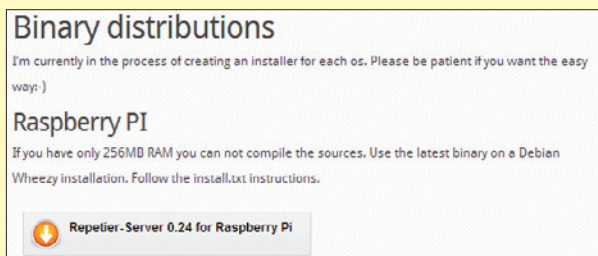
ne la pagina web sarà reperibile all'indirizzo IP di Raspberry alla porta 8080.

Dopo aver reso il tutto controllabile a distanza abbiamo riscontrato la necessità di visualizzare la stampa in corso per verificare che tutto funzioni, così abbiamo collegato una webcam alla seconda porta USB della scheda RaspberryPi.



# Installazione di Repetier-Server su Raspberry Pi

1. Recarsi all'indirizzo <http://www.repetier.com/repetier-server-download/>
2. Scaricare il file 'Repetier-Server 0.24 for Raspberry Pi'.



3. Spostarlo all'interno della directory di Raspberry `/home/pi` avvalendosi di WinSCP
4. Decomprimere il file con WinSCP:
  - 1) Cliccare con il tasto destro del mouse sul file appena spostato
  - 2) Spostarsi su *Comandi personalizzati*
  - 3) Selezionare *UnTar/GZip...*
  - 4) Confermare cliccando su *OK*
5. A questo punto aprire la cartella appena creata *Repetier-Server-Pi*
6. Nel file *install.txt* troverai le istruzioni per l'installazione di

Repetier-Server da riga di comando, è anche possibile effettuare i passaggi con WinSCP:

- 1) Modificare uno degli esempi di configurazione contenuti in *configs*:
  - *Version=1.0*
  - *Active=false*
  - *Name=3Drag*
  - *slugName=3Drag*
  - *device=/dev/serial/by-id/<troverete il nome del vostro dispositivo collegando la stampante e andando a vedere all'interno del percorso appena citato>*
  - *baudrate=250000*
  - *pingPong=false*
  - *protocol=0*
  - *okAfterResend=true*
  - Completare le altre opzioni con le proprie impostazioni e preferenze
  - *tempUpdateEvery=1*
  - Salvare il nuovo file con nome *3Drag.conf*
- 2) Copiare il file eseguibile *RepetierServer* nella cartella `/usr/bin`
- 3) Copiare il file *repetier-server.conf* in `/etc`
- 4) Creare la cartella *Repetier-Server* in `/var/lib`
- 5) Creare le cartelle *www*, *configs*, *storage* e *languages* all'interno della directory appena creata `/var/lib/Repetier-Server`

## TUTTO SOTT'OCCHIO CON UNA WEBCAM

Dopo aver realizzato tutto il sistema, appare evidente la necessità di poter verificare che tutto funzioni. Ancora una volta RaspberryPi ci viene incontro divenendo con poche modifiche un webcam server che permette di visualizzare l'immagine della stampa in tempo reale, direttamente dal pannello di controllo della pagina web di Repetier (**Fig. 1**).

Per prima cosa bisogna installare un'applicazione che acquisisca l'immagine dalla webcam collegata alla porta USB e la carichi su Repetier. Abbiamo scelto Motion perché, essendo nato come software per la videosorveglianza, è semplice da usare e possiede una vasta gamma di impostazioni personalizzabili. Per installare Motion il primo passo è come sempre quello di assicurarsi di avere installati sul proprio RaspberryPi le versioni più recenti dei software e

dei driver digitando nel terminale di RaspberryPi i comandi

```
sudo apt-get update
e
sudo apt-get upgrade
```

A questo punto è possibile installare Motion con il comando

```
apt-get install motion
```

Perché quest'ultimo svolga correttamente il suo compito di webcam server è necessario modificare alcune impostazioni. Digitando da terminale

```
sudo nano /etc/motion/motion.conf
```

sarà possibile configurarlo: trovare la voce

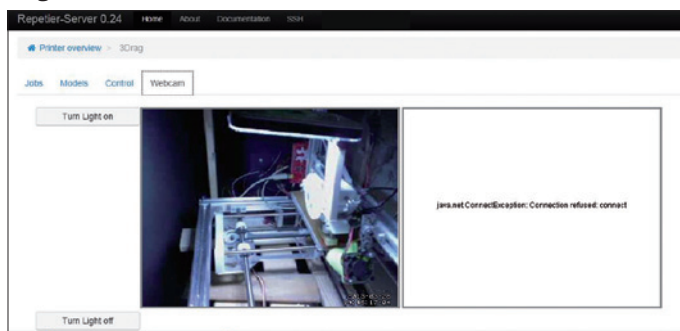
*Daemon = OFF*

e sostituire **OFF** con **ON**, questo consentirà a Motion di essere eseguito in modalità "demone", cioè in background; in seguito trovare la voce

*webcam\_localhost = ON*

e modificare **ON** con **OFF**, ciò permetterà anche agli altri computer connessi alla rete domestica di visuali-

**Fig. 1**



- 6) Copiare il nostro file *3Drag.conf* all'interno di */var/lib/Repetier-Server/configs*
- 7) Copiare il contenuto di *www* e *languages* all'interno delle rispettive cartelle in */var/lib/Repetier-Server*
- 8) Per fare in modo che il Server si avvii in automatico all'accensione di Raspberry:
  - Copiare il file *Repetier-Server.init* all'interno di */etc/init.d/Repetier-Server*
  - Dargli i permessi di esecuzione e lettura attraverso la digitazione del comando `sudo chmod 755 /etc/init.d/Repetier-Server` dal terminale
  - Attivare lo start script con il comando `sudo update-rc.d Repetier-Server defaults`
  - Per attivare\disattivare\riavviare il Server manualmente digitare i seguenti comandi  
`sudo /etc/init.d/Repetier-Server start`  
`sudo /etc/init.d/Repetier-Server stop`  
`sudo /etc/init.d/Repetier-Server restart`

7. L'installazione è terminata, ora è possibile accedere al Server con il proprio browser preferito all'indirizzo  
 <IP del Raspberry>:8080

È consigliabile impostare la scheda con indirizzo IP statico in modo che non cambi ad ogni reboot.

lizzare l'immagine dalla pagina web creata da Motion. Sempre con nano sarà possibile modificare molti altri parametri, come la frequenza di frame al secondo o la dimensione in pixel dell'immagine, ma che è consigliabile lasciare all'impostazione di default in modo da non appesantire troppo la CPU non molto potente di RaspberryPi. Per assicurarsi che all'avvio il motion service sia lanciato effettivamente come daemon si deve ancora modificare una configurazione in questo modo:

```
sudo nano /etc/default/motion
```

sostituire in  
 start\_motion\_daemon= **no** con **yes**.

Finalmente possiamo lanciare il programma con l'istruzione

```
sudo service motion start
```

e verificare che funzioni andando all'indirizzo IP di RaspberryPi, alla porta 8081. Per visualizzare l'immagine direttamente dalla pagina di controllo di Repetier bisogna apportare ancora alcune modifiche. Per prima cosa dovete copiare l'appli-

cazione Java Cambozola.jar, scaricabile dall'indirizzo <http://www.charliemouse.com/code/cambozola/>, in */var/lib/Repetier-Server/www*, in seguito aprire il file *printer.php* copiato al momento dell'installazione del server Repetier nella directory appena citata. Adesso basterà aggiungere all'interno del "Control Panel" all'interno del <body> del file php queste due righe di codice:

```
<applet code=com.charliemouse.cambozola.Viewer
      archive=cambozola.jar width="320"
height="240" style="border-width:1; border-
color:gray; border-style:solid;"> <param name=url
value="http://localhost:8081"> </applet>
```

e il gioco è fatto.

Così facendo apparirà nella pagina dei comandi manuali della stampante un riquadro con l'immagine aggiornata ogni secondo della webcam.

## CONTROLLARE VENTOLA E ILLUMINAZIONE DEL PIATTO DI STAMPA A DISTANZA

Proseguendo nella personalizzazione della postazione di stampa, sfruttiamo il nostro controllo remoto tramite pagina web per gestire la ventola di raffreddamento e l'illuminazione dell'area di stampa.

Per poter controllare ventola e illuminazione a distanza saranno sufficienti alcune semplici modifiche.

Iniziamo con il software residente sulla scheda di controllo. Per chi non l'avesse mai fatto, il firmware Marlin è reperibile alla sezione Software del sito ufficiale della stampante 3Drag, vale a dire

<http://3dprint.elettronica.in/>. Una

volta scaricato, per modificarlo, compilarlo e caricarlo sulla scheda Sanguinololu avremo bisogno della versione 0023 dell'IDE di Arduino, scaricabile dall'indirizzo seguente <http://www.arduino.cc/en/Main/Software>.

Cominciamo col modificare il file *pins.h*, spostandoci nella sezione relativa alla scheda Sanguinololu cerchiamo la voce

```
#define FAN_PIN
```

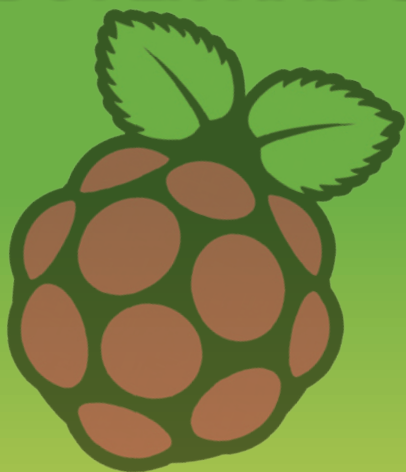
e sostituiamo -1 (che disabilita questa funzione) con 4 (il pin n°4 del microprocessore Atmel è l'unico tra quelli inutilizzati a poter generare un segnale PWM). In seguito aggiungiamo la dichiarazione del pin che



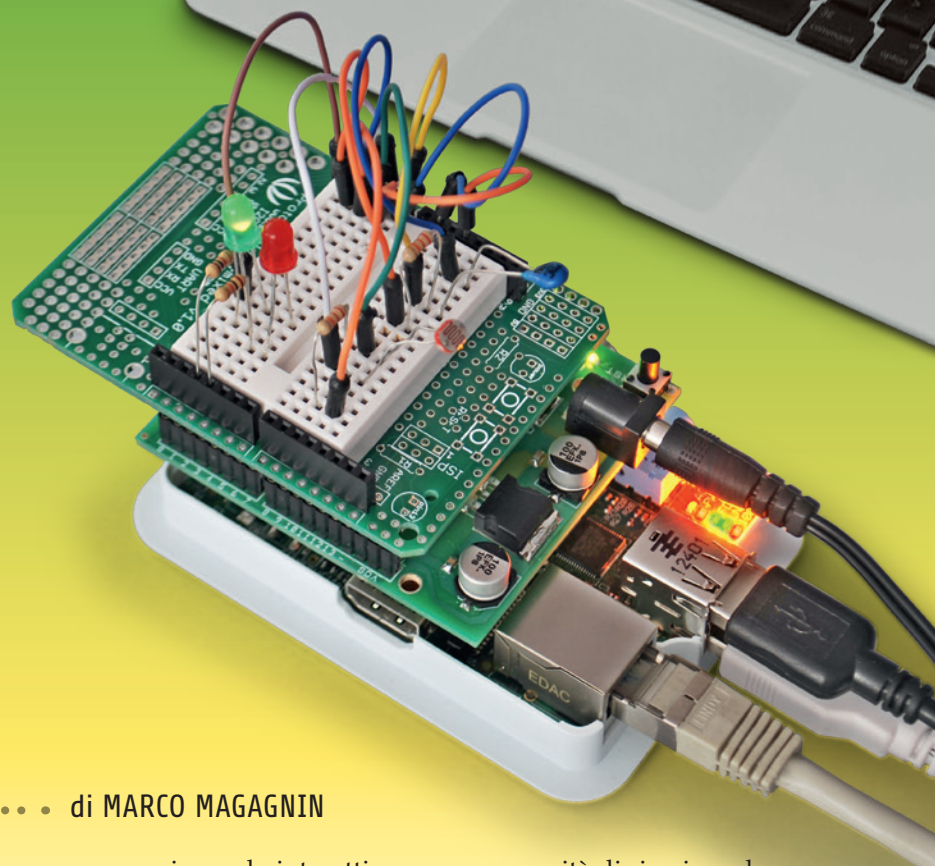




# TERMOSTATO WEB CON LO SHIELD DI ESPANSIONE ADC PER RASPBERRYPI



Completiamo la realizzazione del termostato web con la realizzazione della pagina HTML che include la dashboard emoncms e il pulsante per modificare le soglie di temperatura alle quali intervengono i sistemi che simulano il riscaldamento e il condizionamento.



..... di MARCO MAGAGNIN

**D**opo aver configurato il framework emoncms che gestisce gli indicatori grafici alimentati dalle misure dei sensori di temperatura e luminosità e lo stato della simulazione dei sistemi di riscaldamento e condizionamento, completiamo l'applicazione con i programmi di presentazione della pagina web.

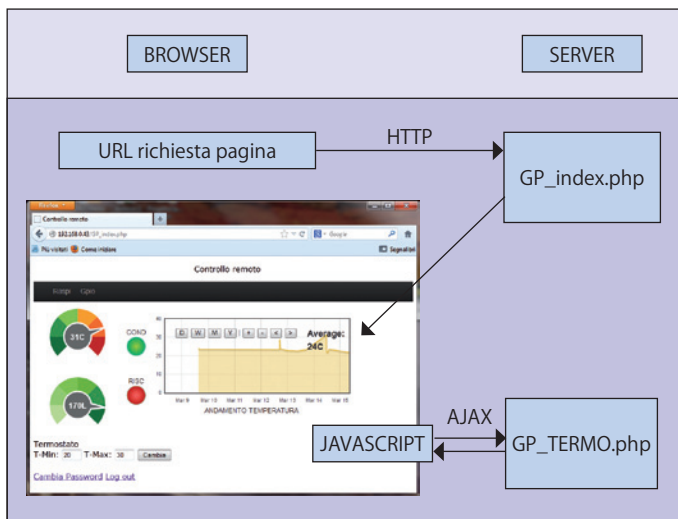
La particolarità principale è l'utilizzo della tecnica AJAX (Asynchronous JavaScript and XML) per gestire la configurazione delle soglie di temperatura

in modo interattivo, senza necessità di ricaricare la pagina ogni volta.

## PROGRAMMA DI GESTIONE DELLA PAGINA WEB

Alcune considerazioni sulla struttura dell'interfaccia web nel nostro progetto.

Vogliamo una pagina dove possano essere eseguiti comandi e impostazioni, la risposta ai quali non richieda di ricaricare ogni volta la pagina stessa. Il nostro obiettivo è di realizzare una pagina che, una



volta aperta se ne stia al suo posto accettando i nostri comandi e ritornando i risultati dopo la modifica interattiva dei soli campi interessati.

Prendiamo come riferimento lo schema di Fig. 1. Per ottenere questo risultato dobbiamo fare in modo che la pagina HTML possa eseguire delle azioni proprie, indipendenti dal server. Utilizzeremo il linguaggio Javascript che ci permette di inserire delle istruzioni (script) all'interno del codice HTML della pagina web, che verranno eseguite sul client a fronte di specifiche richieste. L'utilizzo di Javascript nelle pagine HTML è una delle tante tecniche che vanno sotto il nome di HTML dinamico (DHTML Dynamic HTML). Javascript permette di eseguire delle azioni all'interno della pagina web senza necessità di interessare il server. Questo significa che il browser che utilizziamo contiene al suo interno l'interprete Javascript per l'esecuzione delle istruzioni.

Per aggiornare i contenuti utilizziamo la tecnica AJAX che ci permette di gestire lo scambio di dati in sottofondo tra la pagina web ed il server, in modo asincrono.

In sintesi, per gestire la nostra pagina web dobbiamo adottare tecniche di pagine web dinamiche (PHP) coadiuvate da tecniche Javascript e di gestione interattiva dei contenuti (AJAX). Crediamo che basti.

Per restare il più semplici possibili utilizzeremo queste tecniche nel modo più elementare possibile, giusto per dare nota della loro esistenza. Comunque qui il terreno si fa un po' in salita.

Nello schema di Fig. 1 abbiamo visto la pagina GP\_index.html che viene richiamata per prima e che si colloca come coordinatore di tutte le altre funzionalità. L'azione "Cambia" viene eseguita dal programma GP\_TERMO.php separato rispetto al programma che gestisce la pagina principale, che viene richiamato ad ogni pressione del pulsante.

**Fig. 1**

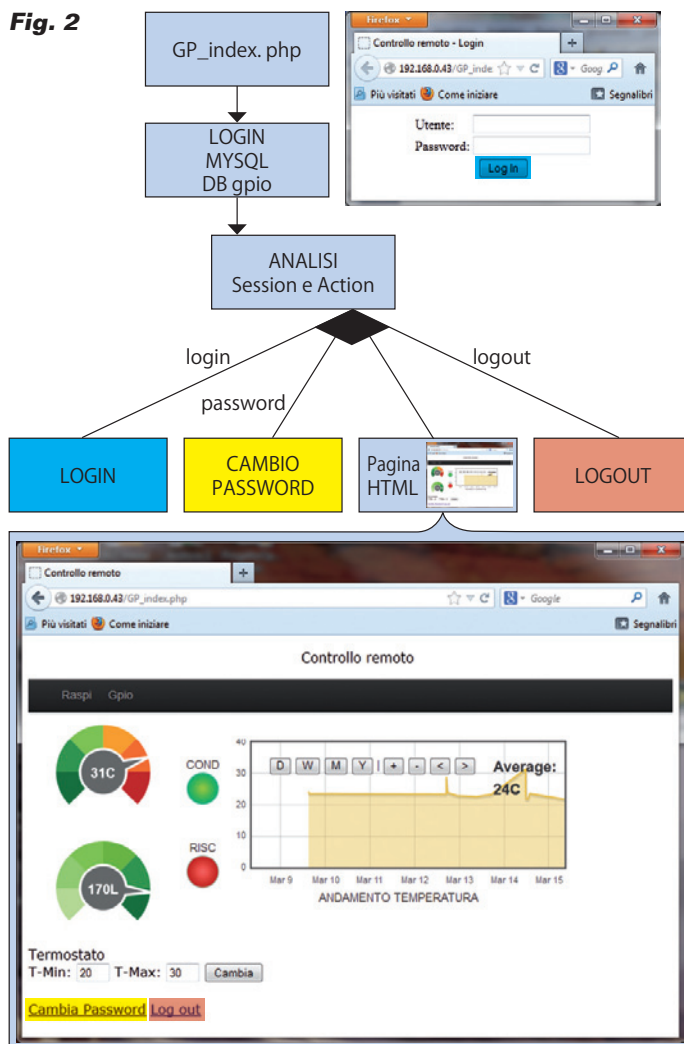
Vediamo di descrivere il funzionamento del programma GP\_index.php, almeno nelle sue linee essenziali.

Nella Fig. 2 i rettangoli colorati che evidenziano le singole funzionalità nelle immagini delle pagine web corrispondono ai colori dei blocchi nel diagramma di struttura, che elaborano le funzionalità stesse, e che trovano riscontro nelle corrispondenti sezioni del listato.

Quando un utilizzatore esterno richiede la pagina, come prima operazione il programma si connette al database "gpio" ospitato dal server MySQL. Poi verifica se l'utente si collega per la prima volta, in tal caso restituisce la pagina di login con la richiesta di Userid e Password. Ricordiamo che l'utente è "admin" e la password "gpio".

Dopo il buon esito del login viene presentata la pagina principale del progetto. Questa pagina è divisa in due aree, un'area di sola visualizzazio-

**Fig. 2**



ne con la rappresentazione grafica dello stato del sistema e un'area di "comando" con i campi e i pulsanti necessari ad attivare le diverse funzionalità. Sono presenti due collegamenti per cambiare la password e per eseguire il logout. Abbiamo lasciato come ultima riga un campo dove vengono riportati i messaggi di ritorno del programma attivato con la tecnica Ajax.

Tralasciamo la spiegazione dei rami di Primo collegamento, Cambio password e Logout per concentrarci sulle sezioni di programma che gestiscono il pulsante "Cambia" e i campi associati.

Una sola osservazione riguarda l'inclusione nella nostra pagina della pagina grafica di emoncms, che abbiamo risolto con l'inclusione della pagina mediante un tag `<iframe>`.

**"Iframe"** significa **"inline frame"**: in una pagina che non utilizza una struttura a frame è possibile creare un frame "al volo" in qualsiasi punto del codice HTML, grazie a questo tag.

Possiamo specificare la larghezza e l'altezza del tag, mentre gli attributi di visualizzazione sono gli stessi del tag `<frame>`: si tratta di una vera e propria finestra verso l'esterno contenuta in un documento ordinario. Come riferimento abbiamo utilizzato il link alla pagina di menu del nostro server emoncms, utilizzando l'APIKey di sola lettura per l'autorizzazione all'accesso, come abbiamo già visto alla fine della personalizzazione della pagina emoncms.

Ora analizziamo più approfonditamente il meccanismo AJAX associato ai comandi operativi.

Vediamo che nella sezione di programma che gestisce la pagina principale è presente un tag `<script type="text/javascript">` al quale segue un lungo elenco di istruzioni decisamente non PHP, terminato dal tag `</script>`. Le istruzioni contenute

nel blocco sono istruzioni Javascript, che si occupano della gestione della parte interattiva della pagina HTML.

Queste istruzioni vengono passate all'interno della pagina HTML per essere eseguite, a richiesta, direttamente all'interno del browser. Javascript è un linguaggio di scripting e quindi richiede un interprete per poter essere eseguito. Ovviamente l'interprete fa parte del codice dei browser più moderni.

Come potete notare i moduli javascript sono organizzati in funzioni. Ciascuna delle quali può prevedere o meno dei parametri.

Di seguito vediamo il frammento di codice HTML che realizza la funzionalità di modifica delle soglie di temperatura. Possiamo modificare i valori nei campi T-Min e T-Max e poi premere il pulsante "Cambia" per inviare i valori al server. Rispetto al codice di una normale pagina HTML notiamo i tag id all'interno delle sezioni di descrizione degli elementi, caselle di testo e pulsante, in più, nel tag dell'elemento "button" è presente la gestione dell'evento onclick, che innesca la funzione javascript "spedisci Termo", passandogli come parametri i valori dei campi tmin e tmax (**Listato 1**).

Alla pressione del pulsante "Cambia" viene attivata la funzione Javascript "spedisci Termo" che, ricordiamo, viene eseguita all'interno del browser, recepisce i due valori di temperatura e inserisce il messaggio di attesa nella casella "descrizione", ultima in fondo alla pagina, che riporta i messaggi di comunicazione tra gli "attori" AJAX e poi, in squisito stile programmazione ad oggetti istanzia una richiesta al server web di nome "richiesta", per la quale prenota l'esecuzione della funzione `updateasincronoTermo` per quando verranno ricevute delle risposte da parte del server, imposta una chiamata con metodo

## Listato 1

```
Termostato
<br>
T-Min: <input type="text" name="Tmin" id="tmin" value="" . $xMin . \' size="3"/>
T-Max: <input type="text" name="Tmax" id="tmax" value="" . $xMax . \' size="3"/>
      <input type="button" value="Cambia" id="buttonTermo"
      onclick="spedisciTermo(document.getElementById(\'tmin\').value,
      document.getElementById(\'tmax\').value);"/>
```

## Listato 2

```
function spedisciTermo(Tmin,Tmax)
{
    document.getElementById("descrizione").innerHTML = "--- Attendi ... ---";
    server = "AJ_TERM0.php?action=Termo" + "&Tmin=" + Tmin + "&Tmax=" + Tmax;
    richiesta = new XMLHttpRequest();
    richiesta.onreadystatechange = updateasincronoTermo;
    richiesta.open("GET", server, true);
    richiesta.send(null);
}
```



### Listato 3

```
function updateasincronoTermo(){
    if ((richiesta.readyState == 4) && (richiesta.status == 200))
    {
        esito = richiesta.responseText;
        iparr = esito.split(",");
        Stato = iparr[0];
        Tmin = iparr[1];
        Tmax = iparr[2];
        document.getElementById("tmin").value = Tmin;
        document.getElementById("tmax").value = Tmax;
        document.getElementById("descrizione").innerHTML = esito;
    }
}
```

### Listato 4

```
$MySQLUsername = "gpio";
$MySQLPassword = "pi";
$MySQLHost = "localhost";
$MySQLDB = "gpio";

$dbConnection = mysql_connect($MySQLHost, $MySQLUsername, $MySQLPassword);
mysql_select_db($MySQLDB, $dbConnection);

$query3 = mysql_query("SELECT xMin, xMax FROM xTemp WHERE xLocale = 1;");
$tempRow = mysql_fetch_assoc($query3);
$xMin = $tempRow['xMin'];
$xMax = $tempRow['xMax'];

mysql_close();
```

### Listato 5

```
<iframe src="http://192.168.0.43/emoncms/admin?id=1&apikey=5a79c4482e414309e9b6b98dc8651f2a" width="800" height="300" frameborder="0" ></iframe>
```

GET e la invia. La stringa null nel metodo send è associata al fatto che abbiamo eseguito la chiamata con il metodo GET. Il metodo GET passa i parametri al server direttamente nell'URL, dopo il punto di domanda, mentre il metodo POST passa i parametri in una stringa separata, che può contenere molti più dati di quelli gestibili con il metodo GET. Nel caso avessimo usato il metodo POST, i dati associati alla chiamata andrebbero inseriti al posto del "null" (**Listato 2**).

A quest'ora il server avrà già risposto da tempo, infatti vedrete molto raramente la scritta ---Attendi---, nel qual caso viene attivata la funzione `updateasincronoTermo()` che, se non vi sono errori - cioè lo stato della risposta è uguale a 4 e lo stato della richiesta è uguale a 200 - intercetta il messaggio di ritorno, costituito da campi divisi da virgole (lo ab-

biamo costruito noi nel programma lato server, come vedremo presto), lo "spacchetta" in campi singoli e aggiorna i valori nei campi `tmin` e `tmax` con i valori ricevuti di ritorno. Poi riporta l'intero messaggio di ritorno nel campo "descrizione". Potete anche eliminare questa informazione scrivendo "" nel campo o quello che preferite. Con questo abbiamo finito e come avrete notato, durante tutto questo lavoro la pagina non viene assolutamente ricaricata, ma rimane fissa al suo posto (**Listato 3**). Vediamo nell'estratto di codice successivo (**Listato 4**) come reperire dei dati da un database MySQL in PHP. Nel programma `GP_index.php` ci serve accedere ai valori di soglia minimo e massimo impostati per la funzione termostato, per evidenziarli nella pagina web. Iniziamo con la connessione al database, per la quale è ne-

cessario specificare i parametri dell'host dove risiede il database, il nome del database, l'utente e la password abilitati alla sua gestione.

Si esegue la connessione con il metodo `mysql_select_db` e poi possiamo eseguire tutti i comandi SQL necessari; nel nostro caso sono la SELECT nella tabella `xTemp` dei valori `xMin` e `xMax` per il locale 1. Per l'esecuzione dei comandi utilizziamo il metodo `mysql_query` che restituisce il risultato della query nella collezione `$query3`. Nel nostro caso viene restituita una sola riga, in caso di restituzione di più righe è necessario eseguire un ciclo for che esaurisca le righe di risposta. Per ciascuna riga si indirizza il "cursore" della query alla riga in esame con il metodo `fetch_assoc` che restituisce i campi della riga nell'array dizionario di nome `$tempRow` e da questo si leggono i valori dei campi indicizzandoli con il loro nome, `xMin` e `xMax`. Prima di terminare il programma si chiude la connessione al database.

Visto che stiamo parlando del programma `GP_index.php`, prima di passare ad un altro

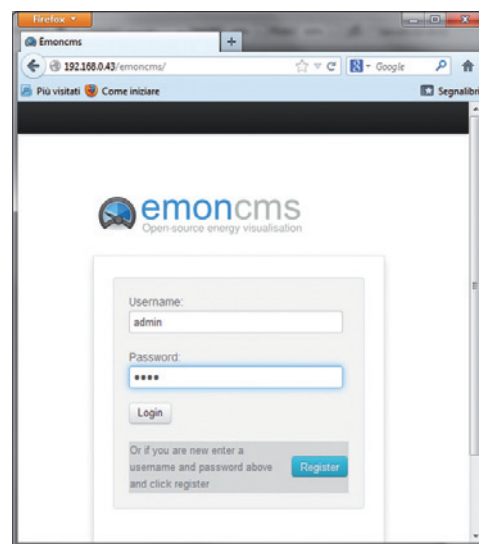


Fig. 3

argomento sistemiamo la configurazione dell'URL e dell'API-Key necessari a richiamare la dashboard emoncms nell'iframe del programma stesso.

Per fare questa operazione dobbiamo riaprire l'applicazione di configurazione di emoncms. Nel browser inseriamo l'URL <indirizzo-IP-RaspberryPi>/emoncms.

Nella finestra di **Fig. 3** inserite utente e password, noi vi avevamo suggerito utente "admin" e password "gpio". Selezionate il pannello "Dashboard" e poi cliccate sull'icona share (**Fig. 4**), copiate dal campo URL del browser il link alla pagina appena creata (**Fig. 5**) e inseritelo nel reference dell'iframe nel programma GP\_index.php, insieme all'APIkey di lettura (**Listato 5**).

## GESTIONE TERMOSTATO

Per la gestione del termostato abbiamo già descritto il programma, che scheduleremo a tempo e che legge i sensori collegati al convertitore analogico digitale, calcola i valori di temperatura e luminosità, li confronta con i valori di soglia sulla tabella xTemp del database e aggiorna il server emoncms con i valori letti e lo stato dei LED che rappresentano l'impianto di riscaldamento e di condizionamento.

Vedremo più avanti come utilizzare lo scheduler di sistema "cron" per eseguire il programma a intervalli determinati, risparmiando molte risorse rispetto all'esecuzione di cicli temporizzati con il programma sempre in esecuzione.

Dobbiamo ancora analizzare il programma PHP che realizza la funzione AJAX nella pagina web interattiva.

Il programma GP\_TERMO.php viene richiamato dalla richiesta in formato URL che proviene dalla

funzione javascript della pagina web, innescata dalla pressione del pulsante "Cambia" (**Listato 6**).

A questo punto sotto il controllo del server Apache2 viene eseguito il programma GP\_TERMO.php, visibile nel Listato 2, che esegue la connessione al database con le stesse modalità descritte nel paragrafo dedicato alla pagina GP\_index.php e poi esegue l'aggiornamento (UPDATE) del record nella tabella xTemp con i valori ricevuti nell'URL.

Alla fine viene restituito con l'istruzione print il semplice messaggio di ritorno

OK, 20, 28, Termo

con i campi separati da virgole che verranno intercettati dalla funzio-

ne javascript updateasincronoTermo() nella pagina web.

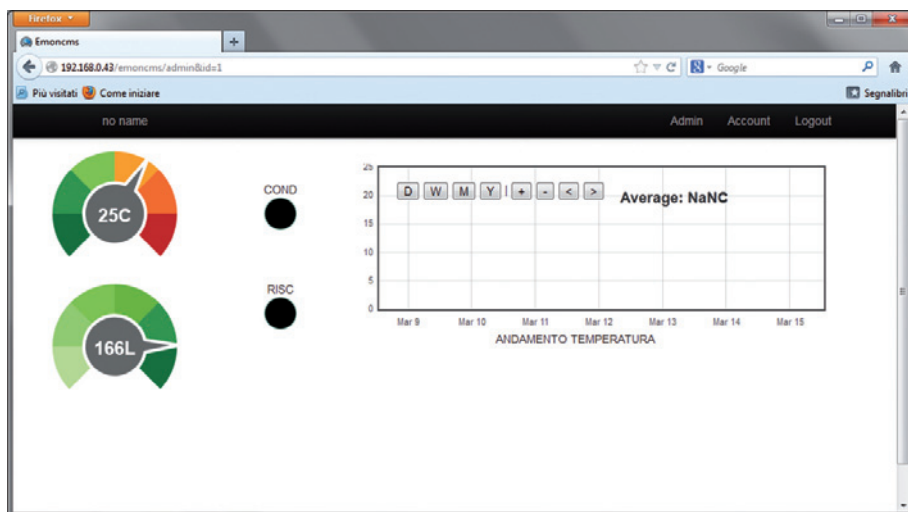
## SCHEDULATORE CRON

Possiamo lanciare a mano il programma ProvaADC2.py per fare qualche prova, poi conviene adottare una struttura più solida per eseguire il programma in modo automatico.

Introduciamo ora il servizio di schedulazione cron che ci permetterà di inviare una nuova serie di dati al nostro sito emoncms ad intervalli prestabiliti, per esempio ogni minuto.

Il servizio cron è il servizio predefinito in GNU/Linux per la gestione dell'esecuzione di programmi e comandi a tempo.

Per funzionare, cron si appoggia a una tabella che contiene le



**Fig. 4**

```
GNU nano 2.2.6      File: /tmp/crontab.pBXJ9s/crontab      Modified
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow  command
*1 * * * * python /home/ProvaADC2.py > /dev/null
```

**Fig. 5**

## Listato 6

```
<?php
// GP_TERMO.php

$MySQLUsername = "gpio";
$MySQLPassword = "pi";

$MySQLHost = "localhost";
$MySQLDB = "gpio";

If (($MySQLUsername == "USERNAME HERE") || ($MySQLPassword == "PASSWORD HERE")){
    print 'ERROR - Please set up the script first';
    exit();
}

$dbConnection = mysql_connect($MySQLHost, $MySQLUsername, $MySQLPassword);
mysql_select_db($MySQLDB, $dbConnection);

$action = $_GET['action'];
$Tmin = mysql_real_escape_string($_GET['Tmin']);
$Tmax = mysql_real_escape_string($_GET['Tmax']);
if ($action == "Termo"){
    mysql_query("UPDATE xTemp SET xMin='$Tmin', xMax='$Tmax' WHERE xLocale=1;");
    mysql_close();
}
print 'OK,' . $Tmin . ',' . $Tmax . ',' . $action;
?>
```

“istruzioni” circa i programmi da eseguire e in quali istanti lanciarli. La tabella utilizzata da cron si gestisce con la funzionalità “crontab” che ha la sintassi:

crontab <opzione> 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 in **Fig. 5**: Ogni riga del file di configurazione ha la struttura riportata nel **Comando 1**. Ciascun parametro può essere configurato come serie di valori es: 0,3,7,9 come un intervallo es: 3-7 oppure un misto dei due es: 0,3,4-6,9 oppure ancora come frequenza \*/5 (ogni 5 minuti, ore, etc.). Nel nostro caso vogliamo che il programma ProvaADC2.py venga eseguito ogni minuto, quindi digitiamo la linea di configura-

zione:

```
*1 *** python /home/ProvaADC2.py > /dev/null
```

che significa: esegui il programma che misura temperatura e luminosità una volta al minuto di ogni ora, giorno e mese, perdendo tutti gli eventuali messaggi di log che possano venire prodotti. 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 eseguito il programma ProvaADC2.py con gli aggiornamenti conseguenti.

Per modificare le impostazioni di cron richiamiamo la funzionalità

*crontab -e*

Per vedere le impostazioni utilizziamo l’opzione *crontab -l*

mentre per interrompere l’esecu-

zione automatica digitiamo

*crontab -r*

## LANCIAMO GPIOSERVER3.PY IN SOTTOFONDO

Avrete ovviamente notato che quando viene lanciato il programma GPIOServer3.py, questo tiene impegnata la finestra di terminale, impedendovi di eseguire altri comandi. A questo punto potremmo aprire più finestre per lanciare più programmi, che può essere una buona soluzione per il debug, ma ci costringe in ogni caso a tenere aperta la connessione SSH per far funzionare il programma. Una classica architettura “embedded” deve essere in grado di funzionare completamente abbandonata e senza connessioni di controllo. È ora di affrontare la gestione dei processi in background, ovvero senza interazione e senza necessità di una finestra di terminale aperta.

Descriviamo due modalità che agiscono, per così dire, a diverse profondità nel sistema operativo. Il primo metodo consiste nel consegnare l’esecuzione del programma ad un processo che, una volta lanciato, libera la linea di comando ma resta comunque collegato alla vita della finestra. Quando chiudiamo la finestra il processo viene interrotto. Per eseguire il programma in questa modalità si aggiunge una “&” alla fine del comando come, per esempio:

```
python /home/GPIOServer3.py &
```

Si dà invio e poi ancora invio: il programma resta in esecuzione in sottofondo e la linea di comando resta utilizzabile. Vediamo come interrompere il processo senza necessariamente chiudere la finestra “madre”. Quando abbiamo lanciato il

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

### Comando 1

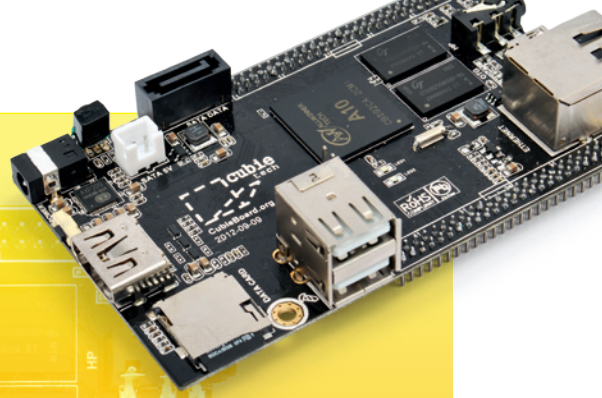


# CubieBoard, nuova scheda ARM Android/Linux

Continua l'uscita sul mercato di veri e propri "computer" con a bordo processori ARM RISC. La scheda Cubieboard, delle dimensioni di poco superiori a quelle di RaspberryPi (circa 10 x 6 cm), è dotata di un SoC Allwinner A10 che contiene un processore Cortex A8 - limitato ad 1 GHz per contenere i consumi - e una GPU Mali-400, 1 GB di RAM e 4GB di memoria Flash a bordo. Nasce con Android 4.0.4 già installato, ma è in grado di far girare diverse distribuzioni di GNU/Linux come Ubuntu 10.04 e 12.04, Arch Linux, Fedora ARM e Raspbian "wheezy" (sì, la stessa distribuzione di RaspberryPi) e quindi può essere utilizzato in diversi ambiti, dalla



smart TV, per la quale è previsto un sensore infrarosso in modo da poterlo comandare da un telecomando, fino a diventare una vera e propria pattaforma di sviluppo, coadiuvata dalla presenza di un connettore SATA per il collegamento di un disco fisso "vero". Completano la dotazione il connettore HDMI, lo slot SDCard, 2 USB, la porta Ethernet, ingresso e uscita audio e ben 96 pin di I/O che includono tutti i bus di utilizzo "elettronico" I2C, SPI, seriali, VGA e video composito. Non mancheremo di parlarne.



processo con il metodo indicato sopra abbiamo ricevuto in risposta un numero di processo. Tutti i processi sono identificati da un numero. Possiamo vedere l'elenco dei processi attivi con i relativi identificativi utilizzando il comando ps (process status):

```
ps -e
```

Per elencare solo i processi che ci interessano possiamo filtrarli concatenando il comando ps con il comando grep (general regular expression print) che ricerca in una lista di file le stringhe di testo che soddisfano l'espressione di filtro

```
ps -e | grep python
```

Troveremo elencati solo i processi eseguiti dall'interprete python, cioè i nostri.

Per controllare le caratteristiche di un singolo processo, ad esempio per verificare qual è il programma in esecuzione utilizziamo il comando

```
ps numero-processo
```

Per terminare intenzionalmente un processo che gira in sottofon-

do, dobbiamo conoscere il numero di processo che lo identifica ed utilizzare il comando

```
kill numero-processo
```

La seconda modalità ci permette di eseguire i programmi in processi ancora più profondi, totalmente svincolati dalla finestra di terminale. Lanciamo il programma con il comando nohup e il carattere "&" alla fine:

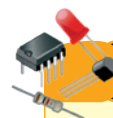
```
nohup python /home/GPIOServer3.py &
```

Con il programma lanciato in questo modo, possiamo tranquillamente chiudere la finestra di terminale e anche disconnetterci dalla sessione SSH con il programma che continuerà a funzionare. Nohup significa No Hang Up ed è nato ai tempi delle connessioni su rete commutata via modem, quando ci si connetteva in remoto utilizzando le normali linee telefoniche per amministrare i sistemi e lanciare processi che dovevano continuare a funzionare anche dopo avere chiuso la connessione (telefonata).

Anche in questo caso, per terminare intenzionalmente il proces-

so si utilizza il comando *kill numero-processo*

Se i processi che lanciamo utilizzano lo stdout, cioè la destinazione - per esempio - delle istruzioni print, che normalmente viene indirizzata sulla finestra terminale, quando li lanciamo in sottofondo possiamo oscurare lo stdout oppure reindirizzarlo verso un file, per esempio per poter leggere



## per il MATERIALE

Tutti i componenti utilizzati per realizzare lo shield di espansione per RaspberryPi utilizzato in questo progetto e descritto nel numero 173 sono di facile reperibilità. Il master del circuito stampato può essere scaricato dal sito della rivista così come i programmi di esempio. Di questo shield Futura Elettronica ha realizzato la versione montata e collaudata (cod. FT1041M) disponibile al prezzo di 27 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>

**Fig. 6**

```

192.168.0.43 - KITTYY
root@raspberrypi:~# nohup python /home/GPIOServer3.py > /dev/null &
[1] 4221
root@raspberrypi:~# nohup: ignoring input and redirecting stderr to stdout

root@raspberrypi:~#

```

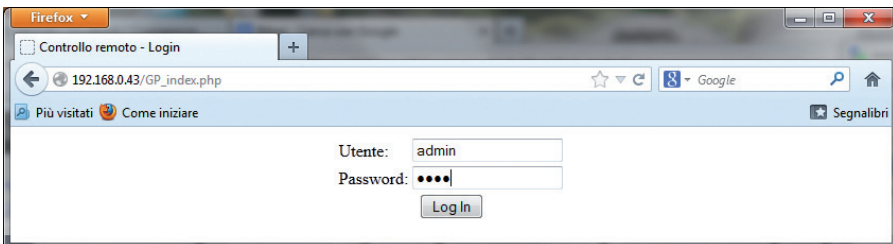
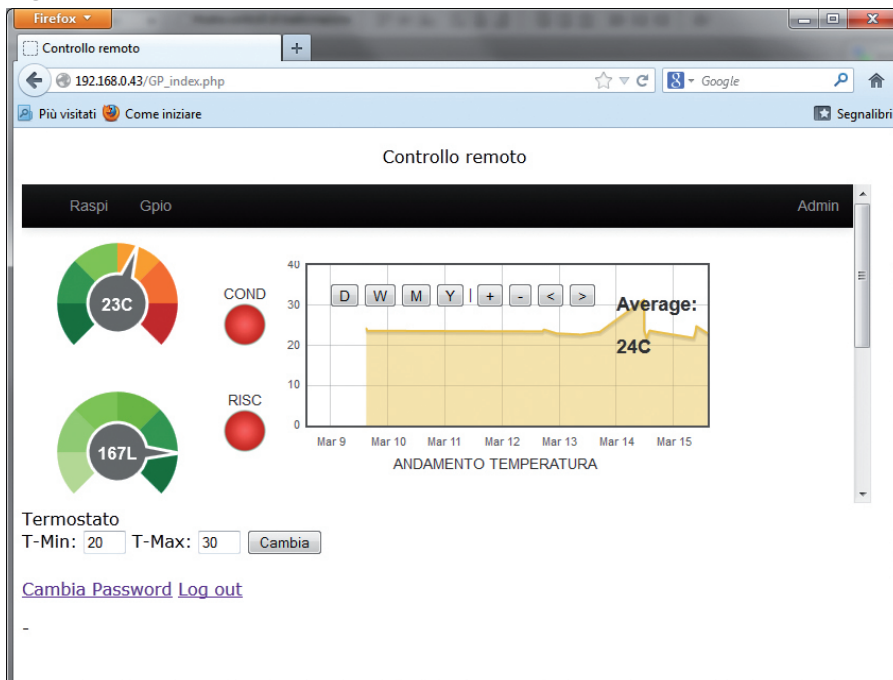
**Fig. 7**

```

192.168.0.43 - KITTYY
root@raspberrypi:~# nohup python /home/GPIOServer3.py > /dev/null &
[1] 4221
root@raspberrypi:~# nohup: ignoring input and redirecting stderr to stdout

root@raspberrypi:~# ps -e | grep python
4221 pts/0    00:00:04 python
root@raspberrypi:~# ps 4221
  PID TTY          STAT       TIME COMMAND
 4221 pts/0    S          0:04 python /home/GPIOServer3.py
root@raspberrypi:~# kill 4221
root@raspberrypi:~# ps 4221
  PID TTY          STAT       TIME COMMAND
[1]+  Terminated                  nohup python /home/GPIOServer3.py > /dev/null
root@raspberrypi:~#

```

**Fig. 8****Fig. 9**

i messaggi successivamente, a scopo di debug (Fig. 6 e Fig. 7). Nel primo caso usiamo il comando

```
nohup python /home/GPIOServer3.py > /dev/null &
```

nel secondo caso

```
nohup python /home/GPIOServer3.py > /home/GPIOServer.log &
```

Per provare l'applicazione usciamo da emoncms con "Logout", richiamiamo la pagina principale con l'URL

```
http://<IP-RaspberryPi/GP_index.php
```

alla pagina di login rispondiamo con "admin" e "gpio" (Fig. 8). Cambiando i valori di soglia della temperatura e scaldando la resistenza NTC tenendola tra le dita possiamo vedere attivarsi i LED veri e simulati a seconda delle condizioni di temperatura. Nel caso in cui l'aspetto grafico della pagina non vi soddisfi, riaprite emoncms in amministrazione, modificate il layout e riprovate (Fig. 9). Eccoci alla fine del percorso che ci permette di progettare e realizzare applicazioni embedded in ambiente GNU/Linux in generale, non soltanto utilizzando RaspberryPi. Chiaramente si è trattato solo di un assaggio, ma speriamo di essere riusciti a trasferire le informazioni necessarie e capire dove andare a guardare quando avremo necessità particolari. Ai tempi dei libri, quello che abbiamo cercato di descrivere stava nascosto all'interno di una "mezza libreria" di manuali. Oggi con internet a disposizione è molto più semplice approfondire architetture e tecniche, a patto di sapere dove cercare. Buona sperimentazione. ■