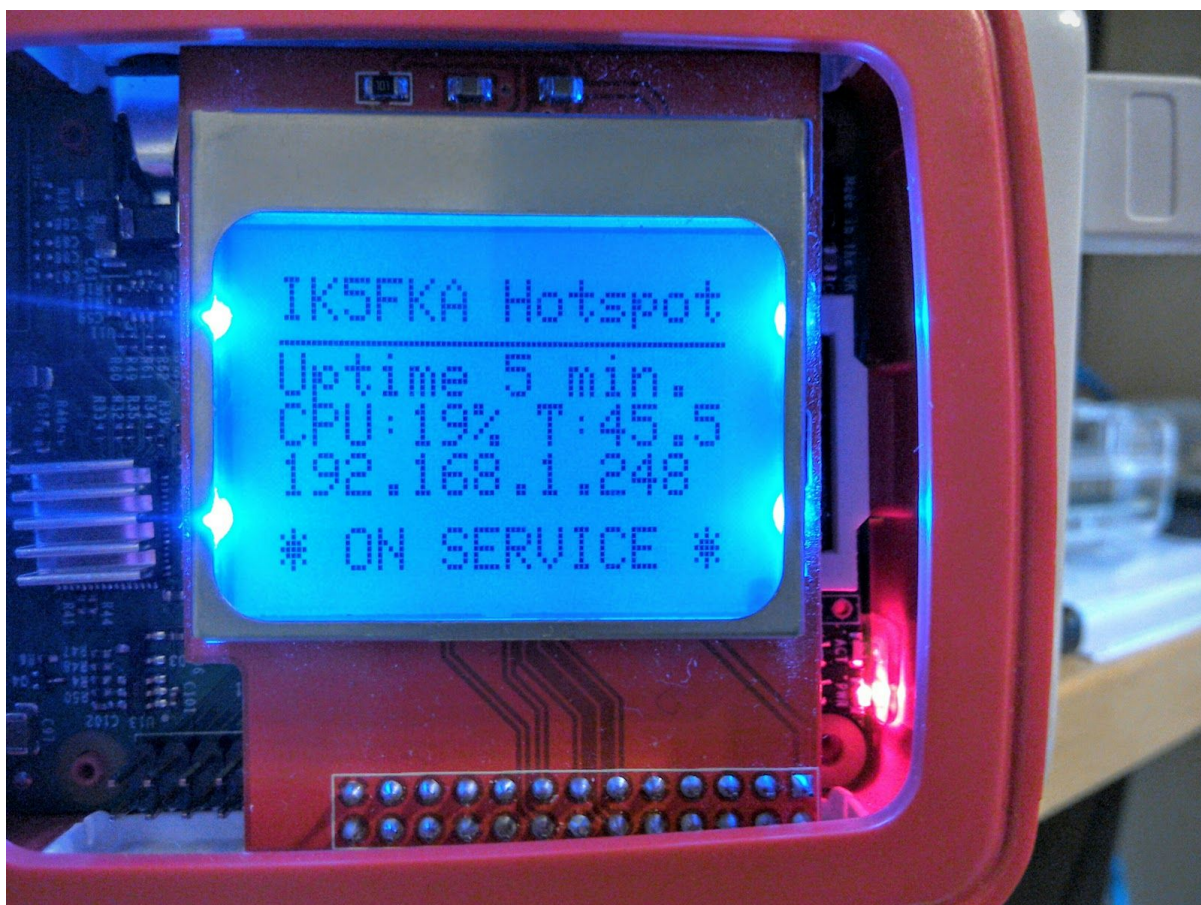


Un piccolo accessorio per tenere a controllo il Raspberry Pi

... con o senza DMR.

Ecco qua un minuscolo display LCD per il noto "Raspino", ormai disponibile per pochi euro da Amazon o simili e attualmente ospite nel mio hotspot DMR. E che una volta installato e modificato il software fornito dal produttore, mostra lo stato della CPU (carico di sistema e temperatura) nonché l'indirizzo IP della connessione e altre cose...



Come si vede, il modulo si inserisce facilmente nel connettore I/O del Raspberry. La risoluzione non è elevata (84 x 48 pixel in b/n) ma la gestione è semplice partendo dai codici sorgente scaricabili dal sito <http://www.sunfounder.com> con la procedura lì descritta.

Il software è fornito sia in Python che in C e comprende già la gestione di scansione della matrice di pixel del display oltre che procedure per l'acquisizione di alcuni dati di sistema (che sono di default il tempo trascorso dall'accensione, il carico della CPU e la quantità di RAM disponibile) e la scrittura dei messaggi, ed è un buon esempio per modifiche o aggiunte. Tra i sorgenti, disponibili a scelta nei linguaggi Python e C, ho preferito scaricare quest'ultimo set per evitare di impararne uno nuovo... ma penso che non sia difficile la modifica anche nell'altro caso.

Le modifiche

Il display, di default, mostra rispettivamente il tempo dall'accensione (Uptime), il carico di sistema e la quantità di RAM disponibile. Quest'ultima informazione, nella gestione dell'hotspot, non la ritengo necessaria, a favore invece di altri parametri: in particolare ho trovato utile far vedere la **temperatura della CPU** e l'**indirizzo IP** della connessione, quest'ultimo utile ad esempio per un login da remoto via LAN se si usa il DHCP e non un indirizzo statico.

Ho infine previsto un'ultima riga per visualizzare brevi messaggi di stato, ad esempio per conoscere se il programma di gestione della DV4mini è attivo oppure no.

Temperatura, ip address e altre cose

... non è tutto farina del mio sacco hi! per la gestione del primo parametro ho infatti utilizzato e adattato la stessa procedura presente nel file "functions.php" del "Dashboard" creato da **DG9VH** e utilmente riportato nella sezione D-Star del nostro sito.

Per quanto riguarda l'indirizzo IP ho utilizzato la funzione del C "get_ip()" per ricercare il valore utilizzato dall'interfaccia attiva (eth0, wlan0 o usb0). Infine la riga finale viene visualizzata leggendo un file di testo contenente un breve messaggio predisposto da una qualsiasi procedura esterna e che dev'essere presente in una directory scrivibile dalla stessa.

Le caratteristiche del dispositivo e le istruzioni per lo scarico e l'installazione del software originale si leggono a questa pagina:

<http://www.sunfounder.com/index.php?c=downloadscs&a=Manualdetails&id=70&name=>

e prima di utilizzare il codice sorgente modificato occorre seguire quanto mostrato nella sezione "For C users". Occorre inoltre scaricare nel Raspberry Pi, se non già presente, anche il pacchetto "libc6-dev" che contiene i riferimenti aggiuntivi del C necessari per la compilazione del nuovo sorgente:

```
sudo apt-get install libc6-dev
```

Per la compilazione dei sorgenti in C è necessario eseguire lo script **compile.sh** già presente nella cartella dei file scaricati. Il programma originale, **pcd8544_rpi.c** va spostato in un'altra locazione prima di essere sostituito dall'omonimo modificato (e sperabilmente corretto) scaricabile da questo sito.

L'unica probabile modifica al suo interno è, se necessario, nella riga che definisce directory e nome del file di testo contenente il breve messaggio da mostrare in fondo al display (nel mio caso la riga è *Statfile = fopen("/root/status.txt","r")*) coerentemente con la locazione voluta per il file.

Eseguire infine lo script **compile.sh**.

Se non vi sono errori sarà alla fine creato l'eseguibile **cpushow**. Eseguirlo per prova dalla linea di comando ma poi, come spiegato alla fine dell'articolo, va mantenuto in esecuzione in background. Il contenuto del display viene aggiornato di default ogni 2 secondi, il tempo può essere modificato cambiando nel file sorgente il valore presente (in millisecondi) nell'istruzione *sleep()* del codice sorgente.

Esempio di script di check status

per l'hotspot DMR con il software dedicato alla chiavetta DV4mini

hotchk.sh: Verifica il corretto funzionamento del programma di gestione della chiavetta DV4mini. Ogni 5 secondi controlla l'esistenza in esecuzione di **dv_serial** (la routine che interfaccia la DV4mini) e aggiorna il messaggio da visualizzare.

```
#!/bin/bash
while true
do
if $(ps ax | grep dv_serial | grep -v grep > /dev/null ) ;
then
    echo "* ON SERVICE *" > /root/status.txt
else
    echo "* OFF SERVICE!" > /root/status.txt
fi
sleep 5
done
```

NOTE

- Il tempo di aggiornamento di quest'ultimo script è regolato dal valore di *sleep* (in secondi).
- Ad ogni modifica al sorgente **pcd8544_rpi.c** va seguita la compilazione e, se già avviato **cpushow** in background, fermarla con il comando

```
sudo killall cpushow
```

prima dell'avvio del programma modificato.

- I programmi **cpushow** e **hotchk.sh** vanno eseguiti in background! quindi - nella linea di comando - con la "&" finale. Una tipica collocazione per l'esecuzione all'avvio è nel file */etc/rc.local*.

Aprire *rc.local* con

```
sudo nano /etc/rc.local
```

e aggiungere, alla fine del file e prima del comando `exit 0` , le due linee

```
/home/cpu_show/cpushow > /dev/null &
/usr/local/bin/hotchk.sh > /dev/null &
```

*(nella mia realizzazione ho memorizzato lo script **hotchk.sh** in */usr/local/bin* mentre ho tenuto per comodità **cpushow** nella sua cartella originale)*