ISSN 2039-7941
Anno 2011_Numero 212

rapporti tecnici INGV

WebMon: piccola interfaccia web della stazione sismica digitale GAIA2

212

Istituto Nazionale di Geofisica e Vulcanologia

Segreteria di Redazione
Francesca Di Stefano - coordinatore
Tel. +39 06 51860068
Fax +39 06 36915617
Rossella Celi
Tel. +39 06 51860055
Fax +39 06 36915617

redazionecen@ingv.it

ISSN 2039-7941
Anno 2011_Numero 212

rapporti tecnici

INGV

WEBMON: PICCOLA INTERFACCIA WEB DELLA STAZIONE SISMICA DIGITALE GAIA2

Sandro Rao e Leonardo Salvaterra

INGV (Istituto Nazionale di Geofisica e Vulcanologia, Centro Nazionale Terremoti)

212

Indice

Introduzione

Fin dall'inizio dello sviluppo dell'acquisitore sismico GAIA2, si è presentato il problema di come creare una rapida interfaccia utente per la gestione e il controllo della stazione sismica. Essendo quest'ultima un apparato hardware abbastanza complesso dotato di sistema operativo Linux (nel modulo TN-2), nel contempo offriva diverse opportunità di collegamento basate sostanzialmente su protocolli di comunicazione tramite porta seriale o di rete. Nel tempo è stato sviluppato un pacchetto software (EarthLab) in grado di sfruttare i protocolli testé citati e così l'utente poteva finalmente configurare i molteplici parametri della stazione secondo le sue esigenze di installazione. Il software principale del pacchetto è GAIASetup [Rapporti Tecnici n° 130 – Rao, Salvaterra, Acerra 2010] che consente, sia con la seriale sia con il protocollo SNMP, un controllo completo del setup della stazione sismica, ma è un po' limitato per quanto riguarda il monitoraggio dello status. Per un'analisi più approfondita l'unico modo era dato dall'accesso a Linux, tramite telnet o SSH, impartendo dei comandi da shell chiaramente ad appannaggio solo dei più esperti a conoscenza dell'architettura software della GAIA2. Per tali motivi si è deciso di sfruttare le potenzialità offerte da un opportuno web server, residente sulla stazione e accessibile con qualunque browser, per offrire all'utente un rapido strumento di controllo sul funzionamento dell'apparato.

Nel presente lavoro viene descritta l'interfaccia WEB sviluppata a tale scopo e le scelte che si sono compiute per svolgere delle operazioni apparentemente semplici, ma rese difficili dalle ridotte risorse di sistema a disposizione.

1. Mini_httpd e script CGI

La prima interfaccia web sviluppata per la GAIA2 prevedeva una pagina HTML statica con un collegamento che utilizza SeisGram2k di A. Lomax come applet java (programma scritto in linguaggio Java eseguito dal browser WEB) per la visualizzazione dei segnali sismici in tempo reale (fig. 1).

Figura 1. prima interfaccia WEB di GAIA2 e visualizzazione segnali sismici con SeisGram2K.

Chiaramente l'utilità di questa interfaccia era molto ridotta, ma obbligata dal server HTTP che la GAIA2 aveva a bordo all'inizio. Per aggiungere ulteriori funzionalità (conoscere lo status della stazione e impartire dei comandi), era necessario installare un server HTTP che fosse in grado di generare pagine dinamiche ed eseguire dei comandi attraverso script programmati da noi. La scelta è caduta sul server *mini_httpd* che consente, pur essendo di piccole dimensioni, l'esecuzione di programmi CGI (Common Gateway Interface). Inoltre *mini_httpd*, consente l'utilizzo dei metodi GET, HEAD e POST, un'autenticazione di base e sicurezza SSL/HTTPS e IPv6.

Nello specifico *mini_httpd* ha le seguenti opzioni [Poskanzer, J., 1999,2000]:

-C: specifica il file di configurazione;

-p: specifica una porta da cui accettare le connessioni (il default è 80);

-d: specifica la directory di avvio;

-c: specifica la directory per gli script CGI;

-u: specifica quale utente deve eseguire gli eventuali comandi;

-r: esegue il commando chroot all'avvio restringendo l'accesso ai soli file della directory corrente;

-V: mostra la versione di mini_httpd.

L'eseguibile di *mini_httpd* è stato preso dalla distribuzione Debian per microprocessore ARM (processore della scheda TN-2) e installato nella GAIA2. La riga di comando usata è:

```
/usr/local/sbin/mini_httpd -d /home/www/ -c cgi-bin/* -u root
```

dove, come si può notare facilmente, la directory di avvio è /home/www/, gli script risiedono in cgi-bin e l'utente è root perché alcuni comandi degli script possono essere eseguiti solo se dotati di privilegi di root.

Per proseguire nella descrizione bisogna dare qualche informazione sul significato delle pagine web dinamiche e la loro creazione. Com'è noto quando un browser WEB (client) si collega ad un sito WEB (server) può richiedere documenti HTML, immagini, suoni, video o script. In quest'ultimo caso però lo script non viene ricevuto, ma viene eseguito restituendo pagine HTML dinamiche cioè costruite sul momento. Uno script è quindi a tutti gli effetti un programma che funge da interfaccia tra un client, un server web, il sistema operativo, le periferiche hardware o anche altri server. La Common Gateway Interface è uno standard per interfacciare applicazioni esterne con un server WEB. Le applicazioni che usano tale standard per comunicare con il server prendono il nome di programmi o script CGI. Tali programmi permettono di rendere dinamico lo scambio dei dati tra client e server. Per esempio, per fare ricerche all'interno di un database utilizzando una interfaccia WEB (una form), il server che offre tale servizio ha bisogno sicuramente di un programma CGI perché deve ricevere la richiesta (che di solito è scritta usando il linguaggio umano) e tradurla secondo le regole del linguaggio del database. Il programma CGI riceve l'interrogazione, la traduce per il database ricercando il documento richiesto ed infine crea la pagina HTML per mostrare il risultato.

Esistono diversi linguaggi di programmazione per costruire script CGI, a seconda del sistema operativo utilizzato: ad es. su un sistema Unix si può usare il linguaggio della shell o il linguaggio C, se il server sta girando su un Macintosh, si usa il linguaggio Applescript, e così via, anche se il linguaggio di programmazione più usato per scrivere degli script CGI è sicuramente il PERL (Practical Extraction and Reporting Language).

Per gli script CGI della GAIA è stato scelto il linguaggio BASH (Bourne again shell) intrinseco nel sistema operativo Linux (con qualche limitazione) non avendo risorse a sufficienza per l'installazione del PERL.

La shell [Masini, D., 2006]è un programma interfaccia (a caratteri) che gestisce la comunicazione fra l'utente ed il sistema operativo, interpretando ed eseguendo i comandi impartiti (la shell viene chiamata anche interprete dei comandi). Ogni shell è dotata di un linguaggio di programmazione (scripting language) più o meno evoluto, che consente di scrivere semplici programmi (script) interpretabili dalla stessa shell. Lo scripting language si compone di comandi interni, file eseguibili ed operatori aritmetico-logici. È possibile pertanto realizzare degli script, cioè file di testo eseguibili che contengono istruzioni comprensibili dalla shell e da questa interpretate ed eseguite. Scrivere uno script di shell è come scrivere una sequenza di comandi da poter far eseguire in blocco alla shell indicando semplicemente il nome del file che li contiene.

BASH mette a disposizione un potente scripting language [Mendel Cooper, 2006], con la valutazione di espressioni condizionali, cicli, inclusioni di file, che permette di automatizzare in maniera piuttosto semplice qualunque operazione di sistema.

Generalmente la prima riga di uno script Bash è la seguente:

#!/bin/bash o #!/bin/sh

in modo tale che, se tale file viene specificato come comando, la shell lanci in esecuzione il file /bin/bash (o /bin/sh), cioè l'interprete BASH, al quale fare interpretare il file stesso.

Va notato il fatto che le linee che iniziano con il carattere '#' sono considerate dei commenti dallo scripting language di Bash, pertanto tale riga viene ignorata dall'interprete stesso quando esegue il file.

Uno script CGI, come detto, è un listato che, oltre ad interagire col sistema, genera le pagine html inviate dal server WEB, semplicemente mandando in output i risultati formattandoli con i tag HTML [Cecchin, W.,], con funzioni javascript ecc, tramite il comando *echo*, ad esempio la riga

```
echo "<BODY bgcolor=#FFFACD><H1><center>Benvenuto in WebMon</center>"
```

è usata per far comparire nella pagina HTML la scritta Benvenuto in WebMon centrata, con sfondo giallo.

Per approfondire i comandi di scripting shell si veda il riferimento nella bibliografia.

2. WEBMON

Lo script principale è *webmon.cgi* (v. appendice A) che viene richiamato cliccando sull'omonimo link della nuova home page della stazione (fig. 2).

Figura 2. nuova home page della GAIA2 con il link a WEBMON.

Dopo aver cliccato, il server chiede l'autenticazione (fig. 3) ed infine genera la pagina di WebMon (fig. 4).

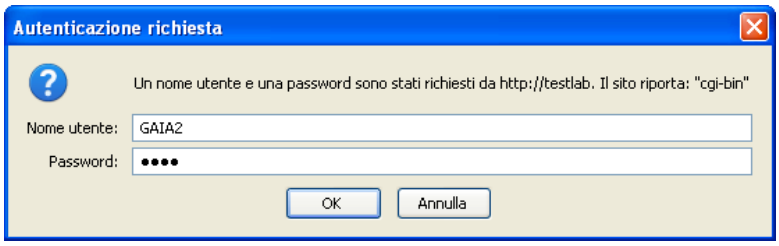

Figura 3. Richiesta autenticazione.

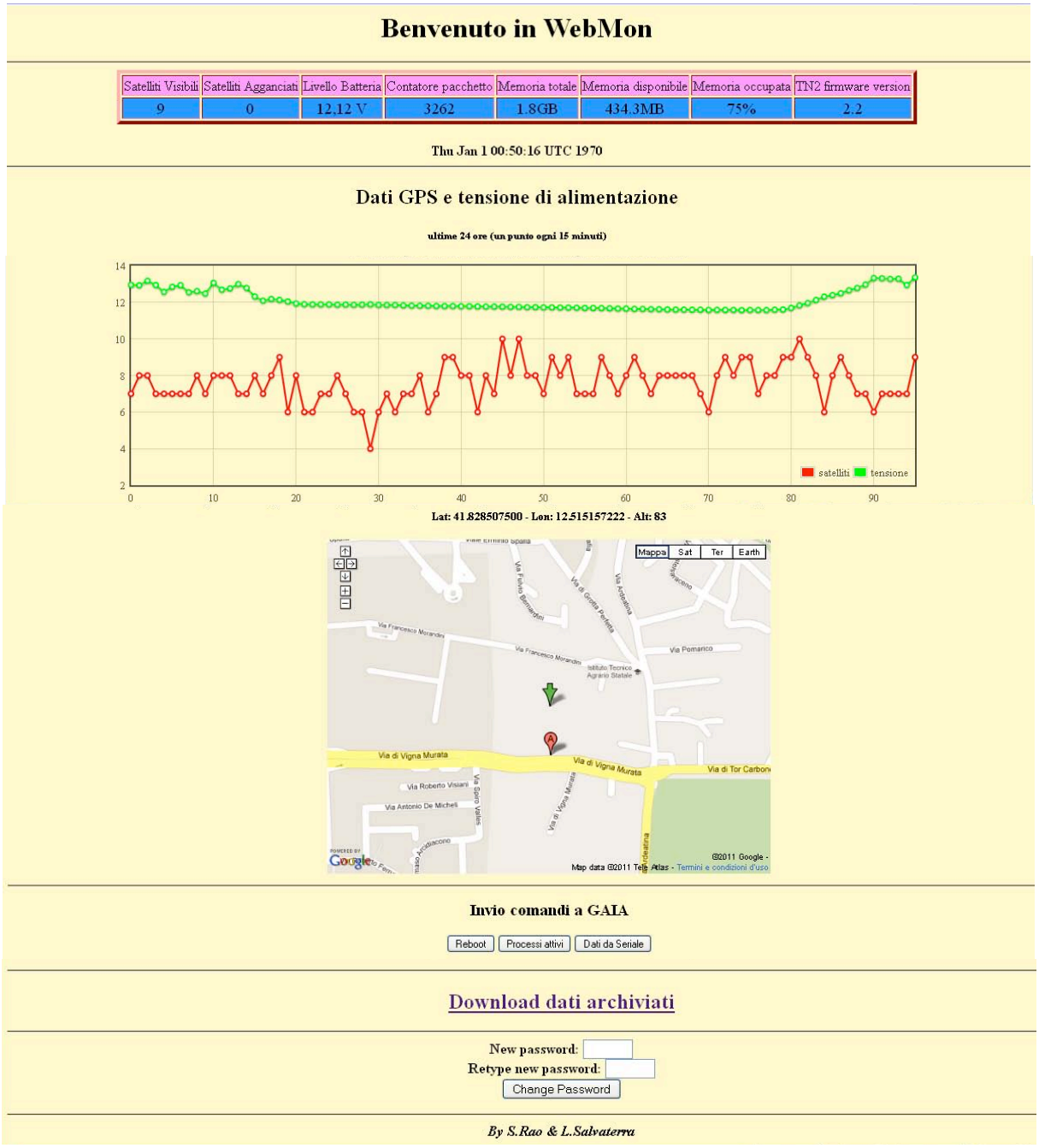

Figura 4. Pagina di WebMon.

La pagina si presenta divisa in cinque parti: la prima, composta da una tabella, fornisce un'indicazione in tempo reale sullo status della GAIA2 [Rapporti tecnici n.130, Rao, Salvaterra e Acerra 2010] (satelliti GPS, tensione di alimentazione, memoria, versione del firmware) e l'orario UTC di visualizzazione delle informazioni; la seconda, composta da un grafico e una mappa, fornisce l'andamento della tensione di alimentazione e del numero di satelliti GPS agganciati nelle ultime ventiquattro ore e, con le coordinate prese dal ricevitore GPS, la localizzazione con Google Maps ™; la terza,che ospita tre pulsanti, consente di inviare i comandi di riavvio del sistema operativo Linux della TN-2, di interrogazione dei processi attivi (fig. 5) e di lettura dei dati in arrivo alla seriale di comunicazione con l'acquisitore a 24 bit chiamato AGDF2 della GAIA2 (fig. 6); la quarta, composta da un collegamento, rimanda ad una pagina (fig. 7) in cui viene presentato l'elenco dei file giornalieri dei dati sismici a pieno campionamento memorizzati sulla stazione, con la possibilità di farne il download; infine l'ultima parte consente di modificare la password di accesso. I comandi, la modifica della password e la creazione della pagina per il download, vengono effettuati tramite altrettanti script CGI (*command.cgi, download.cgi e passwd.cgi*, v. Appendice A).

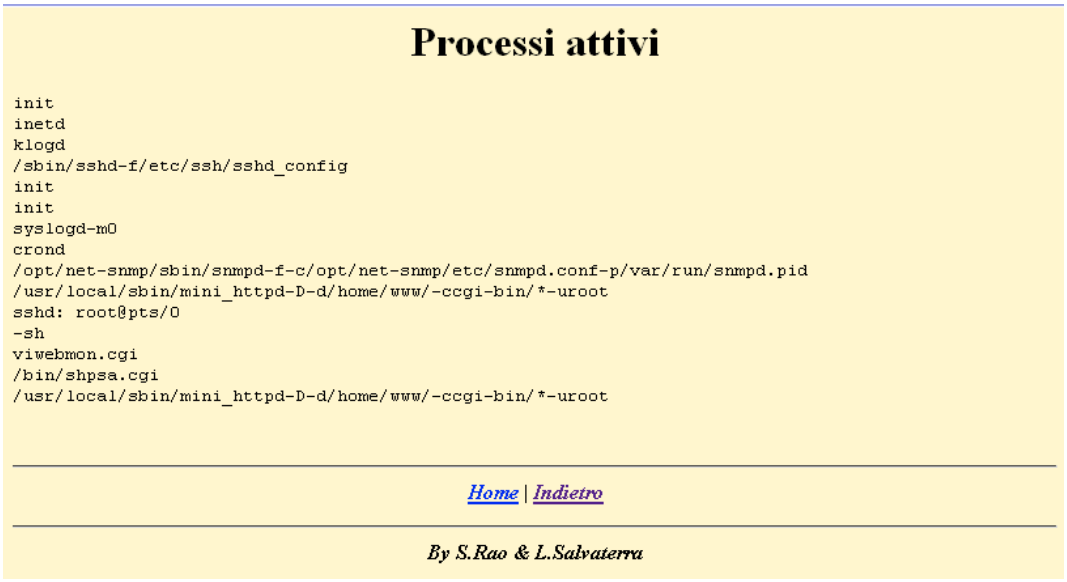

Figura 5. Risultato del comando "Processi attivi".

Figura 6. Risultato del comando “Dati da Seriale”.

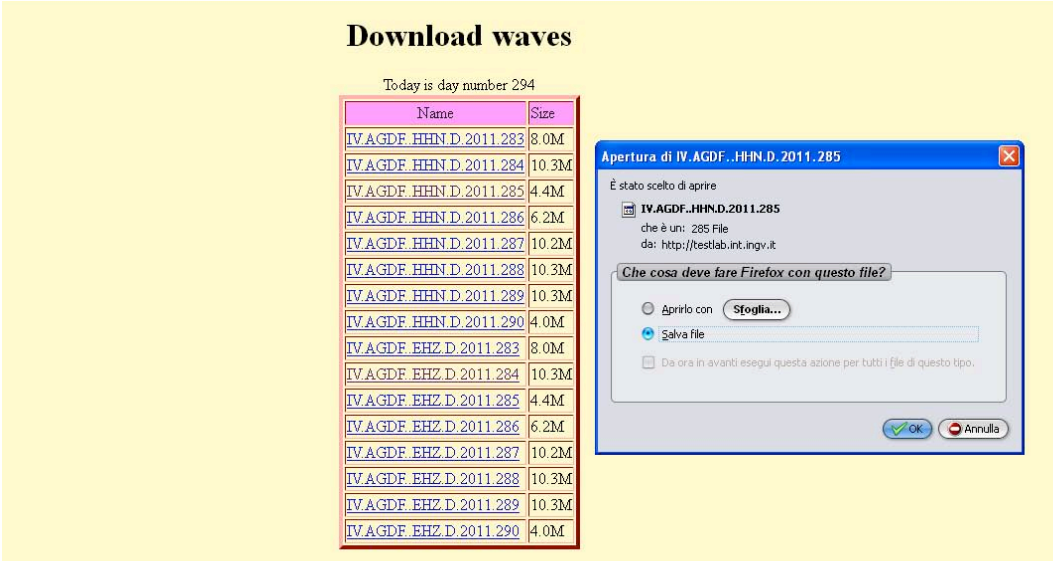

Figura 7. Pagina per il download dei dati archiviati con finestra di dialogo per il salvataggio di un file.

Per la generazione del grafico indicante il livello della tensione di alimentazione e il numero di satelliti GPS agganciati, vengono utilizzati due javascript. Al momento del caricamento della pagina vengono scaricati dalla stazione nel computer client ed eseguiti dal browser locale, senza appesantire la GAIA2 che deve solo generare la pagina html con i valori da graficare.

Il contenuto della directory cgi-bin con gli script è il seguente:

js: directory con javascript per la generazione del grafico;

command.cgi: script CGI per l'esecuzione dei comandi dei tre pulsanti;

download.cgi: script CGI di gestione del download dei file archiviati;

passwd.cgi: script CGI per la modifica della password;

sat.sh: script per la memorizzazione, ogni 15 minuti, del numero di satelliti (nel file di testo sat) e dei valori della tensione di alimentazione (nel file power) per il grafico;

webmon.cgi: script CGI principale.

3. Conclusioni e ulteriori sviluppi

Un proficuo utilizzo di WebMon c'è stato nell'installazione della Stazione sismica di Monte Pizzetto (BO), in collegamento UMTS e alimentazione a pannelli solari, dopo la quale è stato possibile monitorare lo status della GAIA2 tramite uno smartphone dotato di accesso ad internet UMTS: si è verificato, dall'albergo senza connessione internet, che la mattina seguente la batteria non stava caricando correttamente rendendo obbligatorio un intervento in sito sul regolatore di tensione. Ma l'utilità maggiore si è riscontrata nei "feedback" positivi di utenti non tecnici.

Allo stato attuale si sta testando la possibilità di scegliere un intervallo di tempo, nella pagina di download dei file, per non dover scaricare per forza un file giornaliero completo di grosse dimensioni che, con collegamenti con banda limitata, potrebbe essere moto lento. Ulteriori sviluppi potrebbero riguardare l'inserimento di altre informazioni sullo status della stazione, la grafica con la quale presentare i vari dati e altri suggerimenti di utenti, ma soprattutto la possibilità di effettuare il setup della stazione tramite web.

4. Appendice A: listato degli script webmon.cgi, command.cgi, passwd.cgi e download.cgi

Di seguito trovano posto i listati degli script webmon.cgi e command.cgi

Webmon.cgi

```
#!/bin/sh
echo Content-type: text/html
echo "<HTML><HEAD><TITLE>WebMon</TITLE></HEAD>"
echo "<BODY bgcolor=#FFFACD><H1><center>Benvenuto in WebMon</center>"
echo " <!--[if IE]><script language='javascript' type='text/javascript'
src='js/excanvas.min.js'></script><![endif]-->"
echo " <script language=\"javascript\" type=\"text/javascript\"
src=\"js/jquery_002.js\"></script>"
echo "<script language=\"javascript\" type=\"text/javascript\"
src=\"js/jquery.js\"></script>"
. /var/tn-2/1
echo "</H1><p><hr>"
```

Generazione della tabella di status:

```
echo "<CENTER> <table border='4' BORDERCOLOR=RED BORDERCOLORLIGHT=ORANGE
BORDERCOLORDARK=RED >"
echo "<tr bgcolor=#FF99FF>"
echo "<td>Satelliti Visibili</td>"
echo "<td>Satelliti Agganciati</td>"
echo "<td>Livello Batteria</td>"
echo "<td>Contatore pacchetto</td>"
echo "<td>Memoria totale</td>"
echo "<td>Memoria disponibile</td>"
echo "<td>Memoria occupata</td>"
echo "<td>TN2 firmware version</td></tr>"
echo "<tr  bgcolor=#3399FF>"
echo "<td ><center><font size='4' face='Times'>"
echo $visibleSatellites
echo "</td>"
echo "<td><center><font size='4' face='Times'>"
echo $trackedSatellites
echo "</td>"
echo "<td><center><font size='4' face='Times'>"
echo $(($powerSupply/1000)),$(($powerSupply%1000))
echo " V</td>"
echo "<td><center><font size='4' face='Times'>"
echo $packetsCounter
```

```
echo "</td>"
echo "<td><center><font size='4' face='Times'>"
echo $(df -h | grep hda2 | awk 'BEGIN{FS=" "}{print $2}')B
echo "</td>"
echo "<td><center><font size='4' face='Times'>"
echo $(df -h | grep hda2 | awk 'BEGIN{FS=" "}{print $4}')B
echo "</td>"
echo "<td><center><font size='4' face='Times'>"
echo $(df -h | grep hda2 | awk 'BEGIN{FS=" "}{print $5}')
echo "</td>"
echo "<td><center><font size='4' face='Times'>"
echo $(cat /etc/VERSION)
echo "</td></tr>"
echo "</table></CENTER><br>"
echo "<CENTER><b>"
date
echo "</center>"
echo "<HR><center>Dati GPS e tensione di alimentazione </H2>
</center><H6> <center >ultime 24 ore (un punto ogni 15
minuti)</H6></center>"
```

Generazione del grafico

```
echo "<CENTER><table width= '80%'><tr><td colspan='3'>"
echo "<div id=\"placeholder\" style=\"width: 100%; height: 300px;
position: relative; \"><div class=\"tickLabels\" style=\"font-size:
smaller; color: rgb(84, 84, 84);\"></div></div>"
echo "<script id=\"source\" language=\"javascript\"
type=\"text/javascript\">"
echo "\$(function () {"
echo "var sat = [];"
for riga in $(cat ./sat)
do
   echo "sat.push([ $i, $riga]) "
   i=$(( $i + 1 ))
done
i=0
echo "var power = [];"
for riga in $(cat ./power)
do
   echo "power.push([ $i, $riga/1000 ]) "
   i=$(( $i + 1 ))
```

```
done
echo "var options = {"
echo "lines: { show: true },"
echo "legend: {position: \"se\" , noColumns:2, backgroundOpacity: 0},"
echo "points: { show: true }"
echo "};"
echo "$.plot(\$(\"#placeholder\"), [{ data: sat , label: \"satelliti
\"},{ data: power , label: \"tensione \"}], options );"
echo "});"
echo "</script> </td></tr>"
```

Calcolo delle coordinate per la mappa di Google ™

```
altitude=$(($altitude/100))
latitude=$(echo "scale=9; $latitude / 3600000" | bc)
longitude=$(echo "scale=9; $longitude / 3600000" | bc)
echo "<tr><td><b><center> Lat: $latitude - Lon: $longitude - Alt:
$altitude</td></center>"
echo "</td></tr></table>
echo "<br><CENTER> <iframe width=\"640\" height=\"480\"
frameborder=\"0\" scrolling=\"no\" marginheight=\"0\" marginwidth=\"0\"
src=\"http://maps.google.it/maps?f=q&amp;source=s_q&amp;hl=it&amp;geocod
e=&amp;q=$latitude+$longitude&amp;sll=$latitude,$longitude&amp;sspn=0.00
9729,0.021951&amp;ie=UTF8&amp;ll=$latitude,$longitude&amp;spn=0.009729,0
.021951&amp;z=16&amp;output=embed\"></iframe><br/>"
echo "</CENTER><p><p><hr>"
```

Sezione dei pulsanti: chiamata dello stesso script (command.cgi) con parametro diverso

```
echo "<CENTER><H2>Invio comandi a GAIA</H2>"
echo "<input type=button value=Reboot
onclick=location.href='command.cgi?reboot'>"
echo "<input type=button value='Processi attivi'
onclick=location.href='command.cgi?ps'>"
echo "<input type=button value='Dati da Seriale'
onclick=location.href='command.cgi?serial'>"
echo "<p><p><hr>"
```

Sezione per il download e della modifica della password

```
echo "<CENTER><H2><a href='download.cgi'>Download dati
archiviati</a></H2>"
echo "<p><p><hr>"
echo "<form action='passwd.cgi'>"
echo "<div><label>New password: <input name='password'
size='5'></label></div>"
echo "<div><label>Retype new password: <input name='new_password'
size='5'></label></div>"
echo "<input type=submit Value='Change Password'>"
```

```
echo "</form>"
echo "<hr><B><I>By S.Rao & L.Salvaterra"
echo "</CENTER></BODY></HTML>"
```

Command.cgi

```
#!/bin/sh
echo Content-type: text/html
echo
echo "<HTML>"
echo "<HEAD><TITLE>WebMon - dati da seriale</TITLE></HEAD>"
echo "<BODY bgcolor=#FFFACD><H1><center>Dati da seriale</H1></center>"
```

A seconda del parametro passato da webmon.cgi lo script esegue comandi diversi

- reboot

```
if [ $1 = reboot ]
then
        reboot
```

- lettura dati dalla seriale

```
elif [ $1 = serial ]
then
      killall -2 master2bn-seed-com1 2> /dev/null
      rm serial.txt
      sleep 1
      cat /dev/ttyS1 > serial.txt &
      pid=$(ps | grep "cat /dev/ttyS1" | awk 'BEGIN{FS=" "}{print $1}')
      sleep 5
      kill -9 $pid 2> /dev/null
      echo "<CENTER>"
      echo " <table border='2' width='80%'>"
      echo "<tr><td>"
      echo  $(cat serial.txt)
      echo "</td></tr></table>"
      echo "<B><blink>Attenzione applicativo fermo: si riavvierà entro un
      minuto</blink>"
      echo "</CENTER><HR>"
```

- elenco dei processi attivi

```
elif [ $1 = ps ]
then
      echo "<PRE>"
      ps | grep "S " | awk 'BEGIN{FS=" "}{print $1}' > ps.txt
```

```
      for line in $(cat ps.txt)
      do
               echo $(cat /proc/$line/cmdline 2> /dev/null)
         done
      echo "</PRE><HR>"
fi
echo "<CENTER><B><I><a href='../index.html'>Home</a> | <a
href='webmon.cgi'>Indietro</a>"
echo "<hr><a href=\"mailto:sandro.rao@ingv.it,
leonardo.salvaterra@ingv.it\"><B><I>By S.Rao & L.Salvaterra </a>"
echo "</CENTER></BODY></HTML>"
```

Passwd.cgi

```
#!/bin/sh
echo Content-type: text/html
echo "<HTML>"
echo "<HEAD><TITLE>WebMon - New Password</TITLE></HEAD>"
echo "<BODY bgcolor=#FFFACD><H1><center>New Password</H1></center>"
password=$(echo "$QUERY_STRING" | awk 'BEGIN{FS="&"}{print $1}' | awk
'BEGIN{FS="="}{print $2}')
new_password=$(echo "$QUERY_STRING" | awk 'BEGIN{FS="&"}{print $2}' |
awk 'BEGIN{FS="="}{print $2}')
if [ $password = $new_password ];
then
  /usr/local/sbin/htpasswd -cb /home/www/cgi-bin/.htpasswd GAIA2
$password 2>&1 > /dev/null &
  echo "<br><br><br><H2><CENTER><blink>PASSWORD
CHANGED</blink></CENTER></H2><br><br>"
else echo "no passwd"
fi
echo "<hr><CENTER><B><I><a href='../index.html'>Home</a> | <a
href='webmon.cgi'>Indietro</a>"
echo "<hr><a href=\"mailto:sandro.rao@ingv.it,
leonardo.salvaterra@ingv.it\"><B><I>By S.Rao & L.Salvaterra </a>"
echo "</CENTER></BODY></HTML>"
```

Download.cgi

```
#!/bin/sh
echo "<HTML>"
echo "<HEAD><TITLE>Download data</TITLE></HEAD>"
echo "<BODY bgcolor=#FFFACD><center><H1> Download waves </center> </H1>"
echo "<center>Today is day number"
```

```
date  +%j
echo "</center>"
stat=$(cat /seedlink-ridotto/config/seedlink.ini | grep "description =
\"INGV1" | awk 'BEGIN{FS=" "}{print $2}')
anno=$(date | awk 'BEGIN{FS=" "}{print $6}')
echo "<center><table border='4' BORDERCOLOR=RED BORDERCOLORLIGHT=ORANGE
BORDERCOLORDARK=RED ></center>"
echo "<tr bgcolor=#FF99FF>"
echo "<td><center>Name</td></center>"
echo "<td>Size</td></center>"
echo "<td>Cut</td></tr></center>"
du -ah data/archive/$anno/IV/$stat/H* | grep "IV\." >  H.txt
if [ -s H.txt ]
  then
  while read line
  do
      file_name=$(echo $line | awk 'BEGIN{FS=" "}{print $2}')
      echo "<td> <a href=$(echo $line | awk 'BEGIN{FS=" "}{print $2}')
color=\"#FF0000\" title='click to download full wave' >$(echo $line |
awk 'BEGIN{FS="/"}{print $7}') </a></font></center></td>"
      echo "<td> $(echo $line | awk 'BEGIN{FS=" "}{print $1}')
</font></center></td></tr>"
  done < H.txt
fi
du -ah data/archive/$anno/IV/$stat/E* | grep "IV\." >  E.txt
if [ -s E.txt ]
  then
  while read line
  do
      file_name=$(echo $line | awk 'BEGIN{FS=" "}{print $2}')
      echo "<td> <a href=$(echo $line | awk 'BEGIN{FS=" "}{print $2}')
color=\"#FF0000\" title='click to download wave' >$(echo $line | awk
'BEGIN{FS="/"}{print $7}') </a></font></center></td></tr>"
  done < E.txt
fi
echo "</table></CENTER><br>"
echo "<CENTER><B><I><a href='../index.html'>Home</a> | <a
href='webmon.cgi'>Indietro</a>"
echo "</CENTER></BODY></HTML>"
```

5. Appendice B: elenco e sommaria descrizione dei comandi Linux usati negli script

- **Awk:** es. awk 'BEGIN{FS="/"}{print $2}'

è utilizzato per dividere una stringa in tanti parti quante volte compare un determinato carattere di separazione (es. /) e, con l'opzione *print $n*, visualizzare i caratteri appartenenti all'n-esima parte;

- **Bc:** es. scale=9; $latitude / 3600000" | bc

è utilizzato per effettuare divisioni con la virgola, l'opzione scale indica il numero di cifre decimali del quoziente;

- **cat:** es. cat serial.txt

è utilizzato per leggere file di testo;

- **date:**

è utilizzato per visualizzare l'ora di sistema, con l'opzione +%j fornisce il solo giorno dell'anno (1-365);

- **df:** es. df -h

è utilizzato per visualizzare la quantità di spazio disponibile e utilizzata nei file system;

- **du:** es. du -ah data/archive/$anno/IV/$stat/E*

è utilizzato per fare l'elenco dei file archiviati con le relative grandezze;

- **grep:** es df -h | grep hda2

è utilizzato per filtrare l'output di altri comandi in base ad una determinata stringa: nel nostro esempio restituisce le informazioni sulle dimensioni, date dal comando df, riguardanti il solo dispositivo hda2;

- **kill e killall:** es. kill -9 $pid e killall -2 master2bn-seed-com1

sono utilizzati per la terminazione di processi;

- **ps**

è utilizzato per fare l'elenco dei processi attivi;

- **reboot**

è utilizzato per riavviare il sistema operativo;

- **sleep:** es. sleep n

è utilizzato per sospendere l'esecuzione dello script per n secondi.

Per ulteriori informazioni si rimanda ai relativi manuali disponibili in internet.

Bibliografia

Poskanzer, J., (1999,2000) manuale on line di mini_httpd;

Mendel Cooper, (2006) Guida avanzata di scripting Bash;

Masini, D., (2006) Informatica e GNU/Linux;

Cecchin, W., Guida HTML su www.html.it;

Rao, S., Salvaterra, L., Acerra, C., (2010) Software per l'installazione e la configurazione della stazione sismica GAIA2.

Coordinamento editoriale e impaginazione
Centro Editoriale Nazionale | INGV

Progetto grafico e redazionale
Daniela Riposati | Laboratorio Grafica e Immagini | INGV

Via di Vigna Murata, 605
00143 Roma
Tel. +39 06518601 Fax +39 065041181

http://www.ingv.it

Istituto Nazionale di Geofisica e Vulcanologia