

Rapporti tecnici

INGV

**BootMon: utility di controllo e
diagnosi per il modulo TN2 della
stazione sismica GAlA2**

230



Direttore

Enzo Boschi

Editorial Board

Raffaele Azzaro (CT)

Sara Barsotti (PI)

Mario Castellano (NA)

Viviana Castelli (BO)

Rosa Anna Corsaro (CT)

Luigi Cucci (RM1)

Mauro Di Vito (NA)

Marcello Liotta (PA)

Simona Masina (BO)

Mario Mattia (CT)

Nicola Pagliuca (RM1)

Umberto Sciacca (RM1)

Salvatore Stramondo (CNT)

Andrea Tertulliani - Editor in Chief (RM1)

Aldo Winkler (RM2)

Gaetano Zonno (MI)

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



Rapporti tecnici INGV

BOOTMON: UTILITY DI CONTROLLO E DIAGNOSI PER IL MODULO TN2 DELLA STAZIONE SISMICA GAIA2

Sandro Rao

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

230

Indice

1. Introduzione.....	5
2. L'interfaccia grafica di <i>BootMon</i>	6
3. Il SeedLink server.....	10
4. Sviluppi futuri e conclusioni.....	14
Appendice A – File di configurazione <i>seedlink.ini</i> del server <i>seedlink</i>	16
Appendice B – File di configurazione <i>seedlink.xml</i> del server <i>seedlink</i>	19
Appendice C – Comunicazione modulo TN2 e acquirente.....	24
Bibliografia.....	26
Ringraziamenti	26

Introduzione

La stazione sismica GAIA2, è sostanzialmente composta, almeno nella sua configurazione di base, da un acquisitore sismico a 24 bit a 4 canali, da un modulo GPS per la sincronizzazione dei segnali acquisiti e di una scheda TN2 (Transmission Node) che svolge il ruolo di unità di elaborazione centrale del tutto simile ad un PC, con architettura di tipo ARM. Il sistema operativo installato sul modulo in questione è una distribuzione linux debian, ricompilata appositamente per essere supportata dalla piattaforma ARM7. L'intero processo di boot e l'accesso alla modalità console, rappresentata da una finestra testuale con la possibilità di impartire dei comandi alla TN2, sono reindirizzati su una porta seriale (*service com*) alla quale l'utente può collegarsi con un qualsiasi programma di terminale (*hyperteminal* o *teraterm* in ambiente windows oppure *minicom* in ambiente linux) per poter interagire con il modulo TN2 oppure per osservarne la fase iniziale di boot.

I vari messaggi del boot risultano di fondamentale importanza qualora si volesse indagare sul corretto funzionamento della scheda, dal momento che offrono una panoramica completa sull'esito delle molteplici operazioni di inizializzazione dell'hardware o dei processi software in esecuzione.

Tali messaggi possono però risultare di difficile comprensione all'utente meno esperto per cui è nata l'esigenza di realizzare un software di interfacciamento *user friendly*, oggetto di questo lavoro, in grado di interpretare la fase di boot traducendone alcuni passi fondamentali in espliciti messaggi per la diagnosi del corretto funzionamento della stazione. Il software qui analizzato, offre quindi un *boot* "assistito" (da cui il nome *BootMon* ossia *Boot Monitoring*) permettendo tra l'altro, di evidenziare immediatamente eventuali messaggi di errore.

Inoltre terminata la fase di boot, viene offerta all'utente la possibilità di eseguire dei complessi comandi linux, senza conoscerne la sintassi, dal momento che questi sono stati associati a dei tasti grafici.

La stazione sismica GAIA2 al momento è largamente impiegata nella Rete Nazionale, rappresentandone il 40% ed in molte reti gestite da personale non INGV, come ad esempio quella gestita dall'Enel Green Power nei pressi della centrale geotermica di Larderello; in tale contesto *BootMon* è stato ideato per facilitare e velocizzare al massimo le operazioni di installazione e manutenzione della stazione da parte di personale anche non altamente specializzato e laddove fosse richiesto, fornire gli strumenti per raccogliere una serie di informazioni in formato testo, che successivamente possono essere inviate via email ai laboratori tecnici INGV per una analisi più approfondita.

BootMon è stato sviluppato in ambiente windows tramite il linguaggio Visual Basic, in modo da poter essere immediatamente integrato nel pacchetto software *EarthLab*, scritto nello stesso linguaggio, usato per tutte le operazioni di configurazione di una stazione sismica di tipo GAIA2.

1. L'interfaccia grafica di *BootMon*

Il programma *BootMon*, scritto in linguaggio *visual basic*, nasce per facilitare la comprensione di alcuni messaggi prodotti in fase di *boot* dal modulo TN2. Il programma appena viene avviato effettua un controllo sulla disponibilità delle porte seriali presenti sul proprio PC, per instaurare un collegamento RS232 alla velocità di 57600 bps con la porta seriale della scheda remota. L'utente dopo aver premuto il tasto *Boot*, può accendere la stazione ed attendere che il boot completi tutte le operazioni da eseguire. Tali operazioni possono essere monitorate utilizzando la finestra console evidenziata dal riquadro verde nella figura 1. Dei led virtuali indicano a seconda del loro colore (rosso indica un problema, verde il normale funzionamento) l'esito dei vari passaggi. Nel riquadro rosso ad esempio l'ultimo step ha dato un esito negativo. I primi due punti (Start booting ed Uncompressing Linux) sono fondamentali e indicano l'avvio regolare del sistema operativo. I successivi punti CompactFlash OK, Partition hda1/hda2 found ed Eth0 Network Connection indicano rispettivamente la presenza fisica di un dispositivo compact flash, il riconoscimento di due partizioni hda1 e hda2 nel medesimo dispositivo e lo status di un collegamento ethernet. Nel riquadro rosso, quindi, vengono monitorate alcune parti del boot *step-by-step* con indicazione immediata sull'esito dell'operazione. Il riquadro verde indica l'area di console sulla quale vengono presentati tutti i messaggi in formato di testo. L'area blu indica la zona in cui l'utente può invocare vari comandi verso il modulo TN2; infine il riquadro giallo è dedicato all'impostazione della porta seriale in uso.

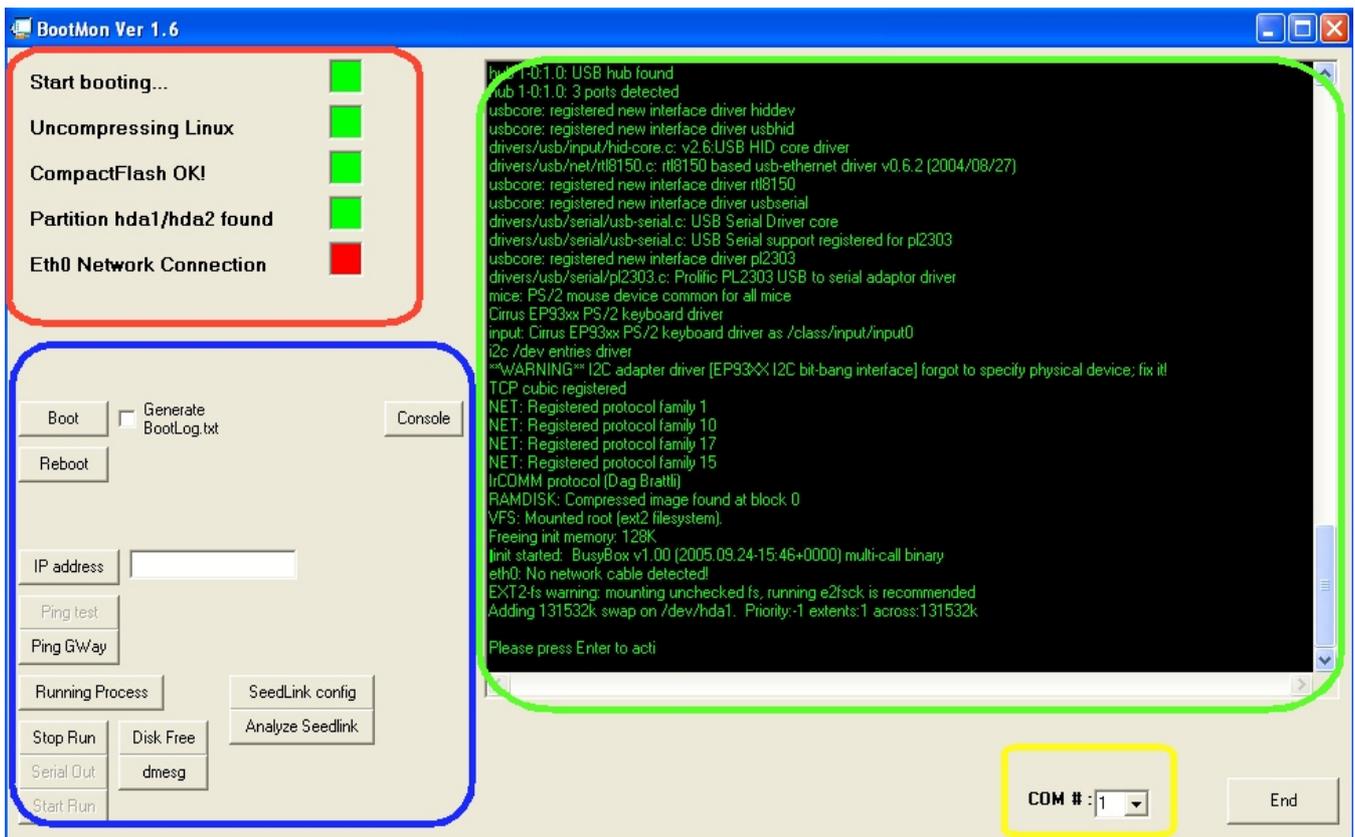


Figura 1. Interfaccia grafica del programma bootmon.

La presenza di un errore nelle partizioni, se non rilevata, comprometterebbe sicuramente il funzionamento dell'intera stazione, dal momento che su una partizione vengono creati dei file successivamente utilizzati dal

sistema operativo. Mentre un errore sulla connessione di rete (indicato dal led rosso su *Eth0 Network Connection*) indica l'assenza di un cavo ethernet o di un suo malfunzionamento.

Accanto al pulsante di *boot*, c'è una *check-box* (*Generate BootLog.txt*) che da la possibilità di catturare in un file di testo tutta la messaggistica del boot, per poi ad esempio essere analizzata attentamente o inviata via email a personale competente in grado di effettuare degli approfonditi controlli in caso di malfunzionamenti manifesti e non rilevabili da *BootMon*.

Il tasto *reboot*, ripete il boot della stazione. Il resto dei pulsanti evidenziato dal riquadro blu permette di sintetizzare l'esecuzione di comandi linux evitando di dover ricordare complicate sintassi. Alcuni di questi forniscono solo informazioni sullo stato della scheda TN2 (ad esempio il suo *IP Address*) altri invece permettono di modificarne i parametri (ad esempio quelli relativi al *seedlink*).

Uno dei problemi più diffusi per gli utilizzatori del modulo TN2 è quello di disconoscere il suo *ip-address* per esempio perché precedentemente cambiato e successivamente dimenticato. Tale informazione, chiaramente, non può essere ricavata via *ethernet*, ma solo con l'ausilio della *service com* del modulo TN2. Il pulsante *IP address* se premuto invia il comando linux **ifconfig** e visualizza il risultato nell'area di testo ubicata a destra del pulsante. A questo punto è significativo confrontare l'*output* del comando generato direttamente in ambiente linux con l'extrapolazione eseguita invece da *BootMon*. Nella figura 2 viene messo in evidenza come possa cambiare l'immediatezza dell'informazione nei due casi testé nominati. A sinistra è riportato l'*output* del comando **ifconfig** eseguito in una *shell* linux ed in neretto si può osservare l'ip della scheda; sulla destra lo stesso risultato ottenuto con il tasto *IP address* di *BootMon*. Chiaramente i due valori coincidono, ma è evidente la semplicità di lettura del secondo caso e la velocità con cui la si consegue.

```
/ # ifconfig
eth0  Link encap:Ethernet  HWaddr AC:DE:48:02:78:11
       inet addr:10.132.4.36  Bcast:10.132.4.255  Mask:255.252.0.0
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:0 errors:0 dropped:0 overruns:0 frame:0
       TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:1000
       RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
       Interrupt:39
```

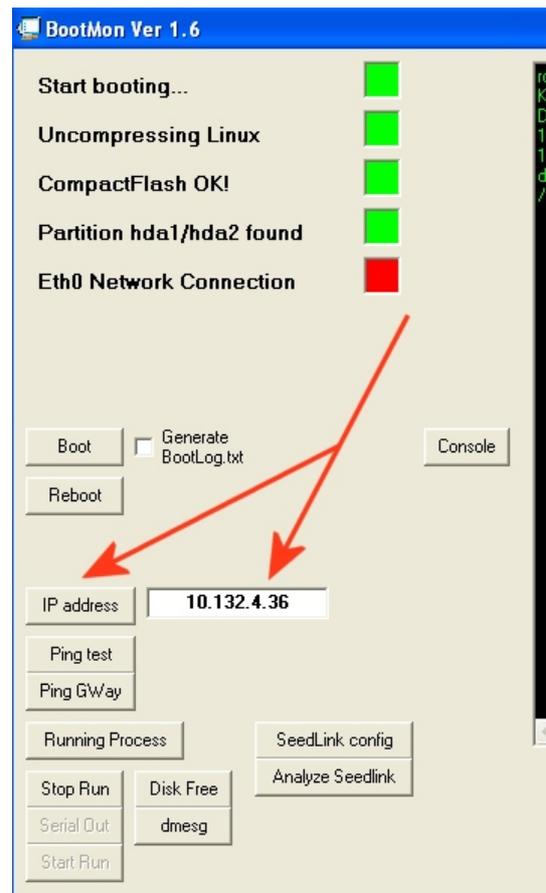
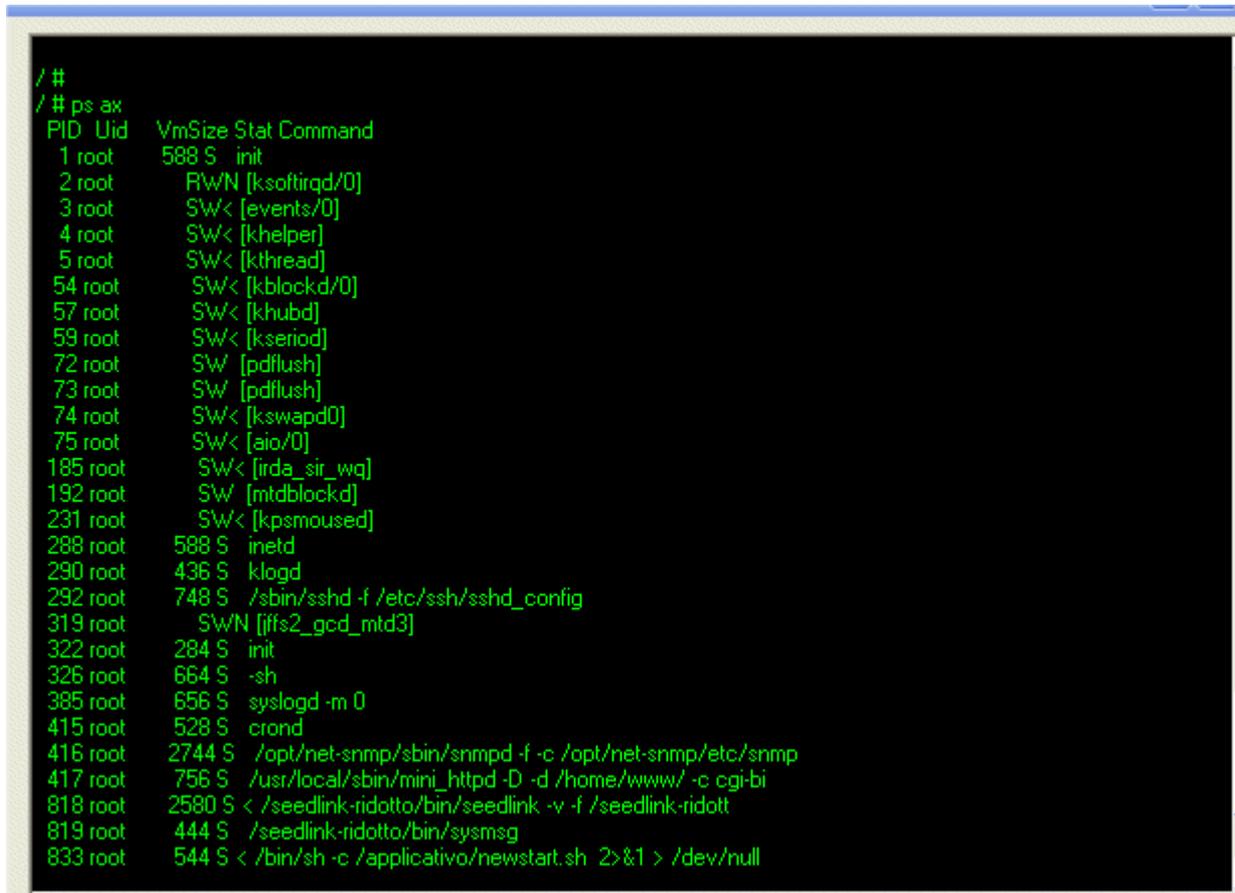


Figura 2. Output del comando **ifconfig** (a sinistra) in ambiente linux e suo equivalente tramite bootmon (a destra).

Il tasto *ping Gway* ha la funzione di eseguire un ping sul *gateway* di *default* della scheda, operazione molto importante nella fase di una installazione di una stazione per verificarne la trasmissione via rete *ethernet*. Senza

l'ausilio del software bisognerebbe o eseguire il ping direttamente sull'ip del *gateway*, che chiaramente deve essere noto, oppure scoprirlo lanciando un comando *pipe* del tipo **route | grep default** procedendo quindi con il ping su tale indirizzo. Questa operazione sfrutta contemporaneamente il comando *route* e ridirige la sua uscita in ingresso al comando *grep* che separerà la riga contenente la parola *default*. Anche in questo caso l'utente può evitare di conoscere la sintassi linux o il *networking* pure se a livello molto elementare.

Il tasto *running process* ci mostra i processi attivi ed il corrispondente numero di PID sfruttando il comando linux **ps ax**. Gli applicativi in esecuzione compariranno nell'area di console come appare in figura 3.



```
/ #
/ # ps ax
PID Uid VmSize Stat Command
1 root 588 S init
2 root RW N [ksoftirqd/0]
3 root SW < [events/0]
4 root SW < [khelper]
5 root SW < [kthread]
54 root SW < [kblockd/0]
57 root SW < [khubd]
59 root SW < [kseriod]
72 root SW [pdflush]
73 root SW [pdflush]
74 root SW < [kswapd0]
75 root SW < [aio/0]
185 root SW < [irda_sir_wq]
192 root SW [mtblockd]
231 root SW < [kpsmoused]
288 root 588 S inetd
290 root 436 S klogd
292 root 748 S /sbin/sshd -f /etc/ssh/sshd_config
319 root SW N [nfsd/0]
322 root 284 S init
326 root 664 S -sh
385 root 656 S syslogd -m 0
415 root 528 S crond
416 root 2744 S /opt/net-snmp/sbin/snmpd -f -c /opt/net-snmp/etc/snmp
417 root 756 S /usr/local/sbin/mini_httpd -D -d /home/www/ -c cgi-bin
818 root 2580 S < /seedlink-ridotto/bin/seedlink -v -f /seedlink-ridott
819 root 444 S /seedlink-ridotto/bin/sysmsg
833 root 544 S < /bin/sh -c /applicativo/newstart.sh 2>&1 > /dev/null
```

Figura 3. Il tasto *running process* permette di visualizzare tutti i processi attivi nella CPU del modulo TN2 ed il rispettivo numero di PID (Process Identifier).

Il tasto *disk free* ci offre un altro esempio di sintetizzazione di un comando linux in un click del mouse con l'apertura di una ulteriore form in cui sono riassunte le informazioni riguardanti il supporto di memoria di tipo *compact flash*.

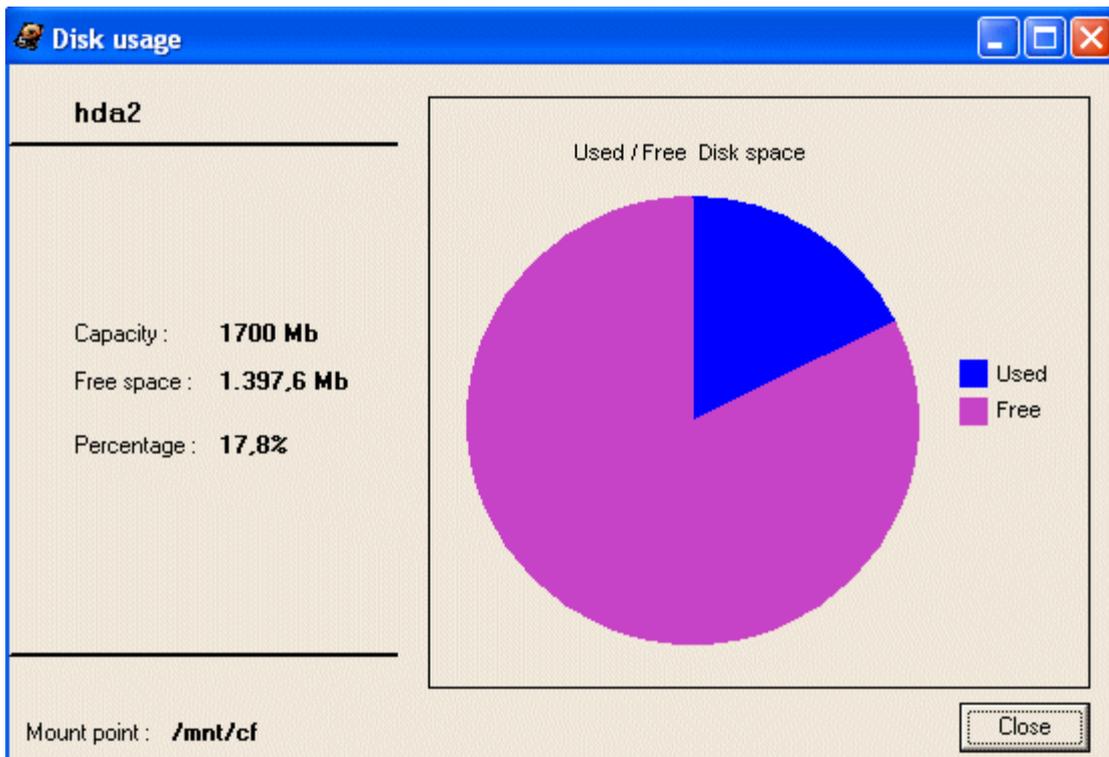


Figura 4. Form associata al pulsante Disk Free. Attraverso il grafico a torta si ottiene una visione immediata della quantità di spazio disco disponibile o usato.

L'alternativa in questo caso sarebbe stata quella dataci dal comando linux shell `df -h | grep hda2` che avrebbe fornito l'informazione, visibile nella figura 5, meno immediata e leggibile del precedente grafico.

```
df -h | grep hda2
/dev/hda2      1.7G  302.7M  1.3G  18% /mnt/cf
/#
```

Figura 5. Output del comando linux `df -f | grep hda2`.

Il tasto *dmesg* ci ripropone tutti i messaggi generati dal boot, sulla console di visualizzazione.

La trasmissione dei dati sismici dall'acquisitore al modulo TN2 avviene attraverso una porta seriale interna al connettore sul quale è montato quest'ultimo modulo. Il gruppo di tasti in basso a sinistra, ha la funzione di investigare la presenza del flusso dati testé citato;

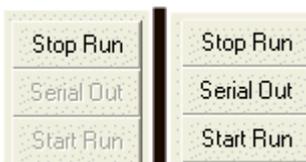


Figura 6. Tasti correlati con la visualizzazione dati sulla porte seriale interna tra acquisitore e modulo TN2.

come si può osservare dalla figura 6 dopo aver premuto il tasto *Stop Run* gli altri due tasti vengono abilitati. Ciò è dovuto al fatto che il processo di verifica ha bisogno di assumere il controllo della porta seriale che in normali condizioni di funzionamento appartiene all'applicativo di acquisizione. Il tasto *Stop Run* infatti lancia il comando linux `/mnt/mtdb/applicativo/tn2_stop -s`. A questo punto premendo *Serial Out* l'utente deve far eseguire al

HHE, HHU) con campionamento (*rate*) di 100 campioni al secondo. Il quarto canale (U) è definito dal protocollo *miniseed* come canale ausiliario e può essere usato per una variabile scalare come ad esempio la temperatura. Se si volesse impostare un altro tipo di *stream* bisognerebbe aggiungere un altro blocco di istruzioni per il parametro **proc**.

```
<?xml version="1.0" standalone="yes"?>
<streams>
  <proc name="INGV_4x100+10s">
    <tree>
      <input name="HHZ" channel="Z" location="" rate="100"/>
      <input name="HHN" channel="N" location="" rate="100"/>
      <input name="HHE" channel="E" location="" rate="100"/>
      <input name="HHU" channel="U" location="" rate="100"/>
    </tree>
  </proc>
</streams>
```

Figura 8. Procedura “INGV_4x100+10s” contenuta all’interno del file xml che imposta il flusso di dati sismici (streams).

L’esigenza di uno *stream* diverso da quello della figura 8 può presentarsi nel momento in cui si deve installare un nuovo sensore con un campionamento diverso o più in generale ogni volta che si vogliono generare dei canali non previsti tra quelli standard. In tal caso senza modifiche di configurazione la GAIA2 non genererebbe nessuno *streaming* in quanto manca l’impostazione del parametro **proc** relativo ai canali di tipo e campionamento diverso allo standard. Come si può osservare le operazioni da svolgere sono varie e di una certa complessità, ma grazie all’ausilio del software *BootMon* basterà premere *SeedLink config* per scaricare sul proprio PC il file .xml come riportato in figura 9. Il tasto sintetizza il comando linux **cat/mnt/seedlink-ridotto/config/streams.xml**.

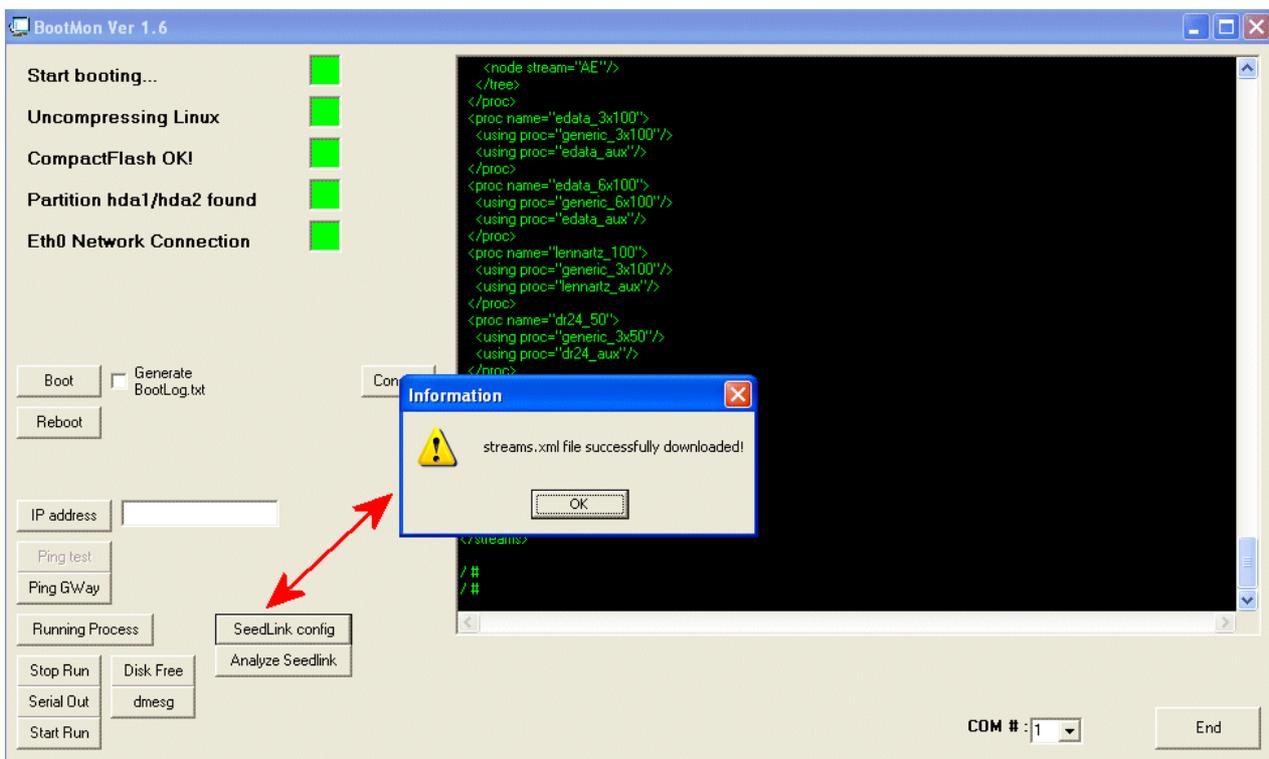


Figura 9. Il tasto SeedLink config permette di scaricare il file xml contenuto nel modulo TN2, tramite porta seriale (service com) sul proprio PC. Le informazioni contenute nel file vengono anche visualizzate nell’area console.

A questo punto grazie al pulsante *Analyze Seedlink* si può agevolmente procedere con la modifica del file appena scaricato sul proprio PC. Con l'ausilio delle successive figure osserveremo il processo di creazione di un nuovo campo **proc** nel file .xml.

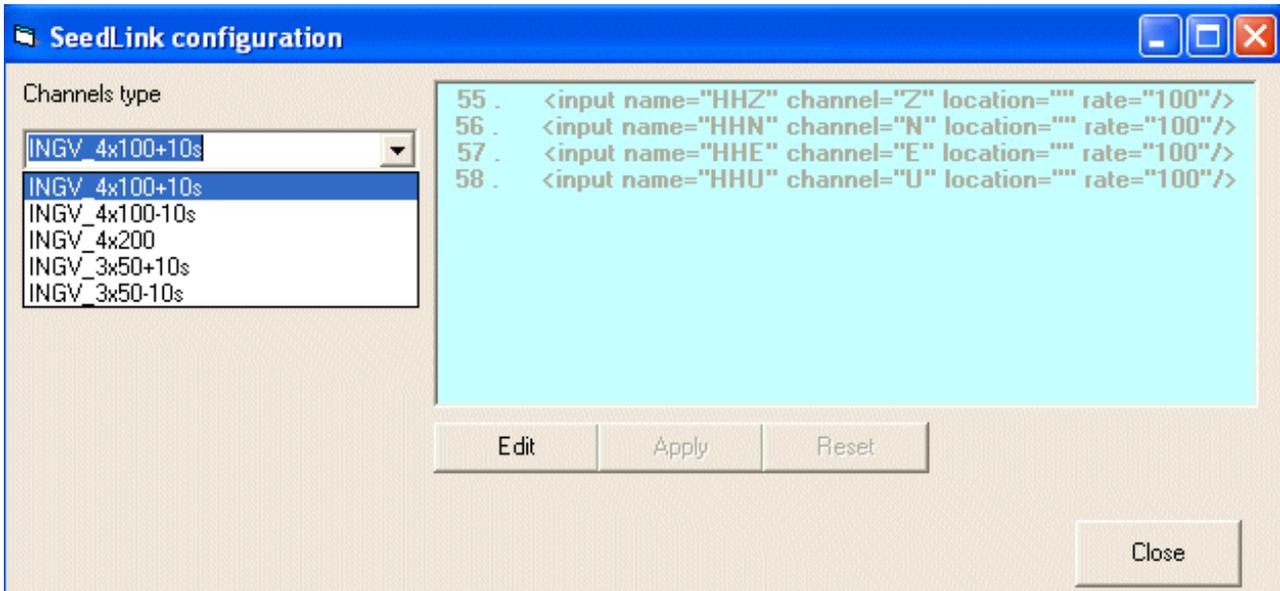


Figura 10. Finestra di editing del file seedlink.xml.

Dopo aver premuto *Analyze Seedlink* si apre la finestra della figura 10: sulla sinistra ci sono i vari nomi dei campi **proc** e al momento della loro selezione sulla destra compaiono le relative impostazioni. Se si desidera apportare delle modifiche bisognerà cliccare sul pulsante *Edit* dopodiché, il testo nella parte destra della finestra potrà essere editato come mostrato nella seguente figura dove, per esempio, si è introdotto un nuovo canale "E" con campionamento a 125Hz per un sensore S13.

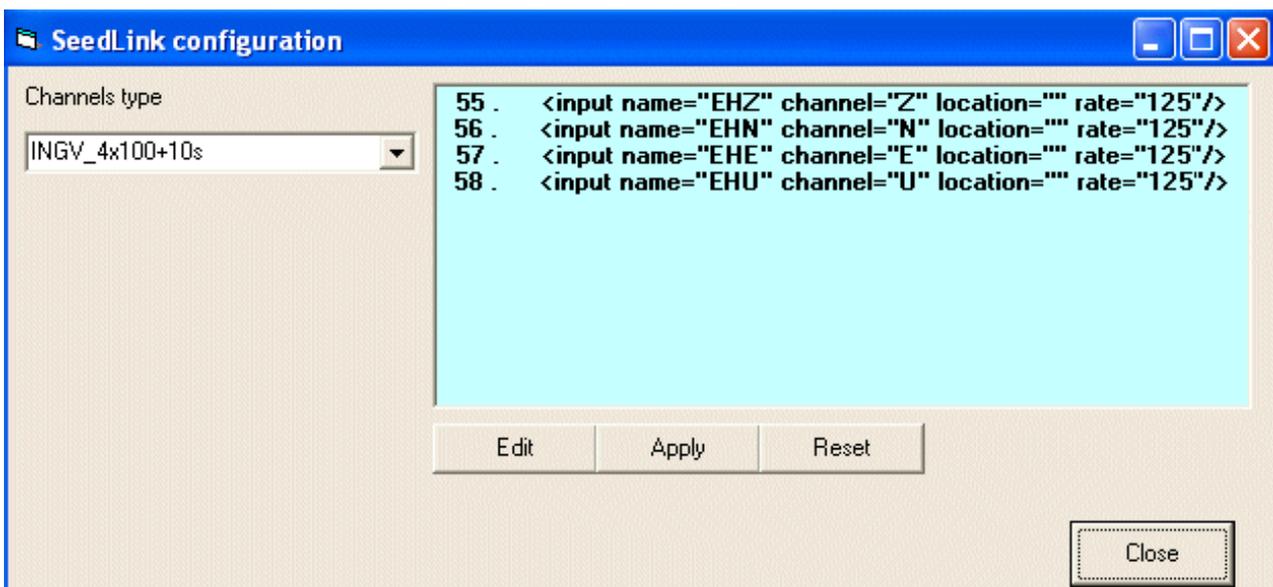


Figura 11. Attivazione della console (parte destra) per la modifica del file seedlink.xml.

Finiti i cambiamenti si potrà procedere al loro salvataggio e invio sulla scheda TN2 utilizzando il tasto *Apply* divenuto attivo nel frattempo, che una volta premuto genererà una finestra di conferma in cui verrà visualizzato il messaggio visibile nella figura successiva.

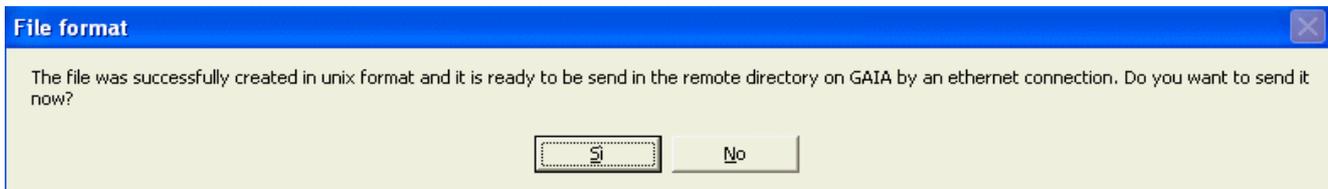


Figura 12. Finestra di conferma.

Esso ci informa che il file è stato modificato e convertito con successo nel formato unix ed è pronto per essere inviato al modulo TN2. Infatti il file .xml viene modificato in ambiente *windows* causando l'inserimento di caratteri che non sono compatibili con una piattaforma di tipo unix (linux) come i *carriage return* (codice ascii esadecimale 13H). A tale scopo nel software è stata integrata una *routine* già implementata e scaricabile dal *web* dal nome DOS2UNIX che viene richiamata per la conversione finale del file. L'*upload* sulla scheda TN2 non avviene su porta seriale (come il *download*) perché in questo caso sarebbe stato necessario l'utilizzo di un protocollo tipo *kermit* che girando sotto linux avrebbe operato come ponte per il trasferimento dei byte; quindi si è optato per il trasferimento via TCP/IP su porta 22 tramite protocollo SSH (*secure shell*) dal momento che sul modulo TN2 è attivo un demone *sshd* in grado di stabilire una connessione *client* con qualsiasi *host* che si trovi nella stessa classe di indirizzi. Per cui collegando il cavo di rete del proprio PC e configurando la scheda di rete in modo opportuno si può procedere con l'*upload* del file .xml. A tale proposito si noti come la spia *Eth0 Network Connection*, in figura 9, sia divenuta verde, ad indicare una connessione di rete valida. Agendo sul tasto *Si* nel messaggio di figura 12, verrà attivata una finestra di dialogo come quella della figura 13 che ci invita ad immettere l'indirizzo IP della scheda remota o a scoprirlo con l'apposito tasto *IP address* descritto in precedenza.

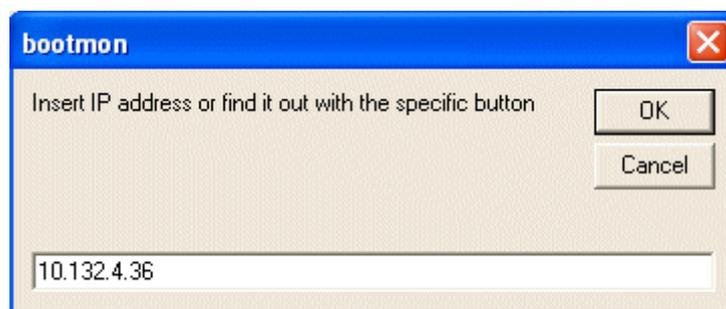


Figura 13. Richiesta dell'IP address del modulo TN2.

Dopo aver immesso l'IP e premuto *OK*, *BootMon* attiva un'altra *routine*, anche questa scaricabile dal *web*, che stabilisce un collegamento con protocollo SSH con la stazione GAIA; il software utilizzato si chiama *pscp.exe* ed il comando che viene inviato da *BootMon* è:

```
pscp -pw password streams/streams.xml root@10.132.4.36:/mnt/mtdb/seedlink-  
ridotto/config/streams.xml.
```

Quindi nel comando sono già inclusi password (password) username (root) ip (10.132.4.36) del collegamento e *pattern* finale di archiviazione (/mnt/mtdb/seedlink-ridotto/config). Cliccando sul pulsante *OK* verrà attivata una shell DOS come quella della figura successiva (fig. 14) dove verrà richiesta l'archiviazione della chiave RSA a cui è consigliabile rispondere no, per evitare che in una fase successiva

collegandosi con lo stesso PC ad un diverso modulo TN2 avente però uno stesso IP, ci si trovi nella condizione di *man in the middle attack* (MITM) ⁽¹⁾ ed il programma ci impedisca l'accesso per motivi di sicurezza; tale situazione è molto facile che si verifichi, dal momento che tutti i moduli TN2 escono dal laboratorio INGV configurati con il medesimo IP (10.132.4.36).

L'ultima riga nella figura successiva (fig.14) ci informa dell'avvenuto trasferimento del file *streams.xml* avente una dimensione di 10 kB, avvenuto alla velocità di 10.3 kB/s. Alla fine il software visualizzerà una finestra messaggio indicante la buona riuscita dell'*upload* (fig.15).

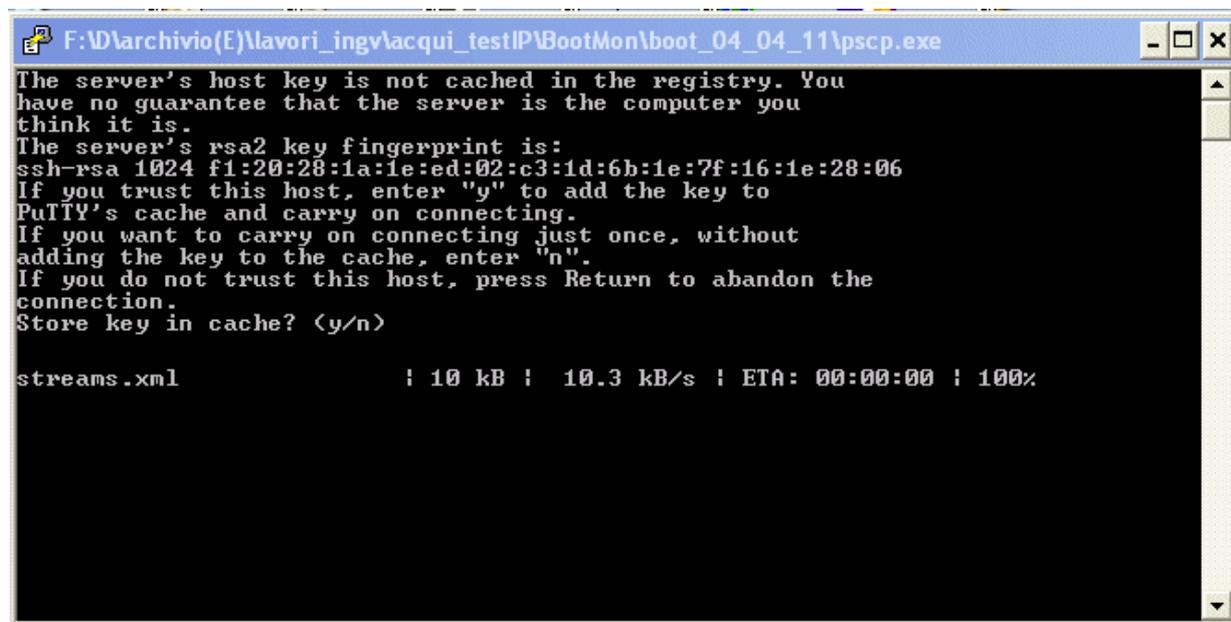


Figura 14. La shell di DOS con la richiesta di archiviazione RSA e nell'ultima riga il messaggio indicante alcune informazioni sul trasferimento file.



Figura 15. Finestra finale con il messaggio di riuscita del trasferimento del file.

Nota (1): Il MITM è un attacco informatico in cui viene, ad esempio, sostituita una macchina di rete con un'altra macchina, apparentemente identica alla precedente, ma sotto il controllo di un hacker. Quindi, nel nostro caso, il programma *pscp.exe* si accorge dalla chiave RSA, se accettata, dell'avvenuta sostituzione di un modulo TN2 anche se dotato di IP uguale.

3. Sviluppi futuri e conclusioni

Il software oggetto di questo lavoro nasce per semplificare e velocizzare alcune operazioni che l'utente di una stazione sismica di tipo GAIA2 si trova ad eseguire in fase di installazione o in quella di una verifica funzionale. Come è stato illustrato, *BootMon*, attraverso un'interfaccia grafica semplice ed intuitiva

si interpone tra utente finale e stazione GAIA2 convertendo i comandi grafici in comandi linux altrimenti difficili da ricordare, consentendo una drastica riduzione dei tempi di lavoro.

Al momento il programma è disponibile nella versione compatibile con il sistema operativo Windows XP essendo scritto in Visual Basic 6.0. Tale versione si basa sull'utilizzo di file dll (*dynamic link library*) che costituiscono una parte integrante dell'applicazione. Com'è noto le dll non sono del tutto compatibili con il sistema operativo Windows 7 che al contrario sfrutta i *frameworks* come librerie di sistema. Quindi è già in cantiere la sua riscrittura in ambiente Visual Basic 2010 che supporta l'utilizzo dei *frameworks*. In questa fase è previsto un miglioramento dell'interfaccia grafica e l'implementazione di nuovi comandi. Sarà ad esempio possibile cambiare svariati parametri di configurazione, sia dell'acquisitore che del modulo TN2, che al momento sono modificabili solo tramite il protocollo *snmp*.

Appendice A – File di configurazione seedlink.ini del server *seedlink*

Il file *seedlink.ini* contiene i settaggi per l’inizializzazione del server *seedlink* e nel suo codice, evidenziati in neretto, ci sono i richiami ad altri files di configurazione, come ad esempio il file *xml* per la generazione degli *strams* sismici o al parametro **proc** usato all’interno del suddetto file *xml*.

```
[seedlink]

* Organization and default network code
organization = "INGV"
network = IV

* GAIAdockfile path
#lockfile = /seedlink-ridotto/status/seedlink.pid
lockfile = /var/run/seedlink.pid

* Directory where Seedlink will store its disk buffers. Each station
* requires its private "<station_name>" and "<station_name>/segments"
* subdirectories, eg., create "/home/sysop/seedlink/EIGAIA/" and
* "/home/sysop/seedlink/EIGAIA/segments/" for station EIL
*filebase = /mnt/flash/seedlink-ridotto/seedlink
*filebase = /seedlink-ridotto/seedlink
filebase = /var/seedlink

* Paths to config files of StreamProcessor
filters = /seedlink-ridotto/config/filters.fir
streams = /seedlink-ridotto/config/streams.xml

* GAIAdist of trusted addresses
trusted = "127.0.0.0/8"

* Check start and end times of streams
stream_check = enabled

* If stream_check = enabled, also check for gaps in all channels that
* match given pattern. Register all gaps that are larger than +-0.5 seconds.
gap_check_pattern =
[BGAIAVU]H[ZNE] |[VU]M[ZNE] |LN[ZNE] |L(CQ|EP) |U(CD|EP|F[CP12]|K2|[TV]1) |V[EK]1|A(C
D|F[CP]|E([0-9]|P))
gap_treshold = 500000

* Disable window extraction from arbitrary Internet hosts
window_extraction = disabled

* Enable window extraction from localhost
window_extraction_trusted = enabled

* INFO provided to arbitrary Internet hosts: ID, CAPABIGAIAITIES, STATIONS
info = all

* INFO provided to trusted hosts: ID, CAPABIGAIAITIES, STATIONS, STREAMS,
* GAPS, CONNECTIONS, AGAIAL
info_trusted = all

* Show requests in log file
request_log = disabled

* Give warning if an input channel has time gap larger than 2 us
proc_gap_warn = 2

* Flush streams if an input channel has time gap larger than 2 s
```

```

proc_gap_flush = 2000000

* Maximum allowed deviation from the sequence number of oldest packet if
* packet with requested sequence number is not found. If seq_gap_limit is
* exceeded, data flow starts from the next packet coming in, otherwise
* from the oldest packet in buffer.
seq_gap_limit = 1000

* Server's TCP port
port = 18000

* Number of recent Mini-SEED packets to keep in memory
buffers = 10

* Number of temporary files to keep on disk for backup access
* (couldn't think of a better term than 'segment' :-()
segments = 2

* Size of one segment in 512-byte blocks
segsizes = 200

* Total number of TCP/IP connections allowed
connections = 400

* Maximum speed per connection (0: throttle disabled)
bytespersec = 0

* Defaults for all plugins. All of these parameters take value in seconds;
* zero disables the corresponding feature.
*
* timeout -- shut down the plugin if no data arrives in this time period
* [0 = wait for data forever];
* start_retry -- restart terminated plugins after this time period (plugins
* can terminate themselves because of some internal error, or they can be
* shut down by the Seedlink if timeout occurs or invalid data received)
* [0 = don't restart terminated plugins];
* shutdown_wait -- wait this time period for a plugin to terminate after
* sending the TERM signal. If a plugin will not terminate, it will be
* terminated the hard way by sending the KILL signal [0 = wait forever].
*plugin_timeout = 0
*plugin_start_retry = 60
*plugin_shutdown_wait = 10

* The first station is always available in uni-station mode (regardless
* of the "access" parameter). Add a dummy station, so uni-station mode
* cannot be used.

station .dummy access = 0.0.0.0 description = "INGV"

* GAIAist of plugins which supply data to the Seedlink server. Several
* instances of a plugin under different names can be used, eg.
* plugin edata1 cmd="/home/sviluppo/seedlink/bin/serial_plugin -v"
timeout=600
* plugin edata2 cmd="/home/sviluppo/seedlink/bin/serial_plugin -v"
timeout=600
* In this case, also corresponding "edata1" and "edata2" sections in the
* plugins.ini file will have to be defined. The plugin parameters above can
* be set individually for each plugin here (without the "plugin_" suffix).

*plugin TN1 cmd="/applicativo/master2bn-seed-com1 /applicativo/config.staz 99999
> /dev/null 2>&1 &"

plugin TN2 cmd="/applicativo/newstart.sh 2>&1 > /dev/null"

```

```
        timeout = 60
        start_retry = 6
        shutdown_wait = 20
*        dynamic_station = enabled
*        station_description = "INGV"

station GAIA description = "INGV1"
            name = GAIA
            network = IV
            proc = INGV
station BBB description = "INGV2"
            name = BBB
            network = IV
            proc = INGV
```

Appendice B – File di configurazione seedlink.xml del server *seedlink*

```
<?xml version="1.0" standalone="yes"?>
<streams>
  <proc name="generic_3x100">
    <tree>
      <input name="Z" channel="Z" location="" rate="100"/>
      <input name="N" channel="N" location="" rate="100"/>
      <input name="E" channel="E" location="" rate="100"/>

<!-- Uncomment this to enable 100Hz HH? streams -->
<!-- -->
<!-- <node stream="HH"/> -->

<!-- Uncomment this to enable 50Hz SH? streams -->
<!-- -->
<!-- <node filter="F96C" stream="SH"/> -->

      <node filter="FS2D5" stream="BH">
        <node filter="F96C">
          <node filter="ULP" stream="LH">
            <node filter="VLP" stream="VH"/>
          </node>
        </node>
      </node>
    </tree>
  </proc>
  <proc name="generic_6x100">
    <tree>
      <input name="Z1" channel="Z" location="00" rate="100"/>
      <input name="N1" channel="N" location="00" rate="100"/>
      <input name="E1" channel="E" location="00" rate="100"/>
      <input name="Z2" channel="Z" location="10" rate="100"/>
      <input name="N2" channel="N" location="10" rate="100"/>
      <input name="E2" channel="E" location="10" rate="100"/>

<!-- Uncomment this to enable 100Hz HH? streams -->
<!-- -->
<!-- <node stream="HH"/> -->

<!-- Uncomment this to enable 50Hz SH? streams -->
<!-- -->
<!-- <node filter="F96C" stream="SH"/> -->

      <node filter="FS2D5" stream="BH">
        <node filter="F96C">
          <node filter="ULP" stream="LH">
            <node filter="VLP" stream="VH"/>
          </node>
        </node>
      </node>
    </tree>
  </proc>

  <proc name="INGV_4x100+10s">
    <tree>
      <input name="HHZ" channel="Z" location="" rate="100"/>
      <input name="HHN" channel="N" location="" rate="100"/>
      <input name="HHE" channel="E" location="" rate="100"/>
      <input name="HHU" channel="U" location="" rate="100"/>
    </tree>
  </proc>
</streams>
```

```

        <node stream="HH"/>
        <node filter="FS2D5" stream="BH">
            <node filter="F96C">
                <node filter="ULP" stream="LH">
                    <node filter="VLP" stream="VH"/>
                </node>
            </node>
        </node>
    </tree>
</proc>

<proc name="INGV_4x100-10s">
    <tree>
        <input name="EHZ" channel="Z" location="" rate="100"/>
        <input name="EHN" channel="N" location="" rate="100"/>
        <input name="EHE" channel="E" location="" rate="100"/>
        <input name="EHU" channel="U" location="" rate="100"/>
        <node stream="EH"/>
    </tree>
</proc>

<proc name="INGV_4x200">
    <tree>
        <input name="HNZ" channel="Z" location="" rate="200"/>
        <input name="HNN" channel="N" location="" rate="200"/>
        <input name="HNE" channel="E" location="" rate="200"/>
        <input name="HNU" channel="U" location="" rate="200"/>
        <node stream="HN"/>
    </tree>
</proc>

<proc name="INGV_3x50+10s">
    <tree>
        <input name="BHZ" channel="Z" location="" rate="50"/>
        <input name="BHN" channel="N" location="" rate="50"/>
        <input name="BHE" channel="E" location="" rate="50"/>
        <node stream="BH"/>
    </tree>
</proc>

<proc name="INGV_3x50-10s">
    <tree>
        <input name="SHZ" channel="Z" location="" rate="50"/>
        <input name="SHN" channel="N" location="" rate="50"/>
        <input name="SHE" channel="E" location="" rate="50"/>
        <node stream="SH"/>
    </tree>
</proc>
<proc name="generic_3x50">
    <tree>
        <input name="Z" channel="Z" location="" rate="50"/>
        <input name="N" channel="N" location="" rate="50"/>
        <input name="E" channel="E" location="" rate="50"/>
        <node stream="SH"/>
        <node filter="F96C" stream="BH">
            <node filter="FS2D5">
                <node filter="FS2D5" stream="LH">
                    <node filter="VLP" stream="VH"/>
                </node>
            </node>
        </node>
    </node>
</tree>

```

```

    </tree>
</proc>
<proc name="generic_3x40">
  <tree>
    <input name="Z" channel="Z" location="" rate="40"/>
    <input name="N" channel="N" location="" rate="40"/>
    <input name="E" channel="E" location="" rate="40"/>

<!-- Uncomment this to enable 40Hz SH? streams -->
<!-- -->
<!-- <node stream="SH"/> -->

    <node filter="F96C" stream="BH">
      <node filter="F96C">
        <node filter="ULP" stream="LH">
          <node filter="VLP" stream="VH"/>
        </node>
      </node>
    </node>
  </tree>
</proc>
<proc name="generic_3x20">
  <tree>
    <input name="Z" channel="Z" location="" rate="20"/>
    <input name="N" channel="N" location="" rate="20"/>
    <input name="E" channel="E" location="" rate="20"/>
    <node stream="BH"/>
    <node filter="F96C">
      <node filter="ULP" stream="LH">
        <node filter="VLP" stream="VH"/>
      </node>
    </node>
  </tree>
</proc>
<proc name="stream_40">
  <tree>
    <input name="BHZ" channel="Z" location="" rate="40"/>
    <input name="BHN" channel="N" location="" rate="40"/>
    <input name="BHE" channel="E" location="" rate="40"/>
    <node stream="BH"/>
  </tree>
</proc>
<proc name="stream_20">
  <tree>
    <input name="BHZ" channel="Z" location="" rate="20"/>
    <input name="BHN" channel="N" location="" rate="20"/>
    <input name="BHE" channel="E" location="" rate="20"/>
    <node stream="BH"/>
  </tree>
</proc>
<proc name="irae">
  <!-- STS-2 -->
  <tree>
    <input name="Z0" channel="Z" location="" rate="125"/>
    <input name="N0" channel="N" location="" rate="125"/>
    <input name="E0" channel="E" location="" rate="125"/>
    <!-- BHZ, BHN, BHE generated here -->
    <node filter="FS2D5" stream="BH"/>
  </tree>
  <!-- CMG-5 -->
  <tree>
    <input name="Z1" channel="Z" location="" rate="125"/>
    <input name="N1" channel="N" location="" rate="125"/>

```

```

    <input name="E1" channel="E" location="" rate="125"/>
    <!-- SNZ, SNN, SNE generated here -->
    <node filter="FS2D5" stream="SN"/>
</tree>
<!-- STS-2 LP -->
<tree>
    <input name="Z2" channel="Z" location="" rate="125/128"/>
    <input name="N2" channel="N" location="" rate="125/128"/>
    <input name="E2" channel="E" location="" rate="125/128"/>
    <!-- LHZ, LHN, LHE generated here -->
    <node stream="LH"/>
</tree>
</proc>
<proc name="tiltmeter">
    <tree>
        <input name="N" channel="N" location="" rate="1/10"/>
        <input name="E" channel="E" location="" rate="1/10"/>
        <node stream="VA"/>
    </tree>
    <tree>
        <input name="TN" channel="N" location="" rate="1/10"/>
        <input name="TE" channel="E" location="" rate="1/10"/>
        <node stream="XE"/>
    </tree>
</proc>
<proc name="stream_processor_test">
    <tree>
        <input name="HHZ" channel="Z" location="01" rate="100"/>
        <input name="HHN" channel="N" location="01" rate="100"/>
        <input name="HHE" channel="E" location="01" rate="100"/>
        <node filter="F96C">
            <node filter="F96C" stream="BH">
                <node filter="F96C">
                    <node filter="ULP" stream="LH">
                        <node filter="VLP" stream="VH"/>
                    </node>
                </node>
            </node>
        </node>
    </tree>
    <tree>
        <input name="BHZ0" channel="Z" location="02" rate="20"/>
        <input name="BHN0" channel="N" location="02" rate="20"/>
        <input name="BHE0" channel="E" location="02" rate="20"/>
        <input name="BHZ1" channel="Z" location="12" rate="40"/>
        <input name="BHN1" channel="N" location="12" rate="40"/>
        <input name="BHE1" channel="E" location="12" rate="40"/>
        <node stream="BH"/>
        <node filter="F96C">
            <node filter="ULP" stream="LH">
                <node filter="VLP" stream="VH"/>
            </node>
        </node>
    </tree>
</proc>
<proc name="edata_aux">
    <tree>
        <input name="S1" channel="1" location="" rate="1"/>
        <input name="S2" channel="2" location="" rate="1"/>
        <input name="S3" channel="3" location="" rate="1"/>
        <input name="S4" channel="4" location="" rate="1"/>
        <input name="S5" channel="5" location="" rate="1"/>
        <input name="S6" channel="6" location="" rate="1"/>

```

```

    <input name="S7" channel="7" location="" rate="1"/>
    <input name="S8" channel="8" location="" rate="1"/>
    <input name="PLL" channel="P" location="" rate="1"/>
    <node stream="AE"/>
  </tree>
</proc>
<proc name="lennartz_aux">
  <tree>
    <input name="T" channel="T" location="" rate="1"/>
    <input name="B" channel="B" location="" rate="1"/>
    <input name="X" channel="X" location="" rate="1"/>
    <input name="Y" channel="Y" location="" rate="1"/>
    <node stream="AE"/>
  </tree>
</proc>
<proc name="dr24_aux">
  <tree>
    <input name="S0" channel="0" location="" rate="1/10"/>
    <input name="S1" channel="1" location="" rate="1/10"/>
    <input name="S2" channel="2" location="" rate="1/10"/>
    <input name="S3" channel="3" location="" rate="1/10"/>
    <input name="S4" channel="4" location="" rate="1/10"/>
    <input name="S5" channel="5" location="" rate="1/10"/>
    <input name="S6" channel="6" location="" rate="1/10"/>
    <input name="S7" channel="7" location="" rate="1/10"/>
    <input name="S8" channel="8" location="" rate="1/10"/>
    <input name="S9" channel="9" location="" rate="1/10"/>
    <node stream="AE"/>
  </tree>
</proc>
<proc name="edata_3x100">
  <using proc="generic_3x100"/>
  <using proc="edata_aux"/>
</proc>
<proc name="edata_6x100">
  <using proc="generic_6x100"/>
  <using proc="edata_aux"/>
</proc>
<proc name="lennartz_100">
  <using proc="generic_3x100"/>
  <using proc="lennartz_aux"/>
</proc>
<proc name="dr24_50">
  <using proc="generic_3x50"/>
  <using proc="dr24_aux"/>
</proc>

<proc name="INGV">
  <using proc="INGV_4x100+10s"/>
  <using proc="INGV_4x100-10s"/>
  <using proc="INGV_4x200"/>
  <using proc="INGV_3x50+10s"/>
  <using proc="INGV_3x50-10s"/>
</proc>
</streams>

```

Appendice C – Comunicazione modulo TN2 e acquisitore

La figura 16 ritrae una stazione sismica GAIA2 nella sua configurazione di base. Il modulo TN2 è collegato al digitalizzatore a 24 bit attraverso un connettore DIP da 40 pin, tramite il quale la scheda viene alimentata e dove è stata inserita la seriale TTL concepita per il passaggio dei dati che può essere controllato con il tasto *Stop Run* come descritto a pag.9 e nelle figure 6 e 7. Nelle figure 17, 18 e 19 si può osservare tale connettore sia sull'acquisitore che sul modulo TN2 con evidenziati i pin TX ed RX della seriale TTL. Accanto al modulo TN2 è possibile notare alloggiato anche il ricevitore GPS di bordo per la sincronizzazione dei segnali acquisiti. Infine viene riportato il *pin-out* del connettore DIP40.

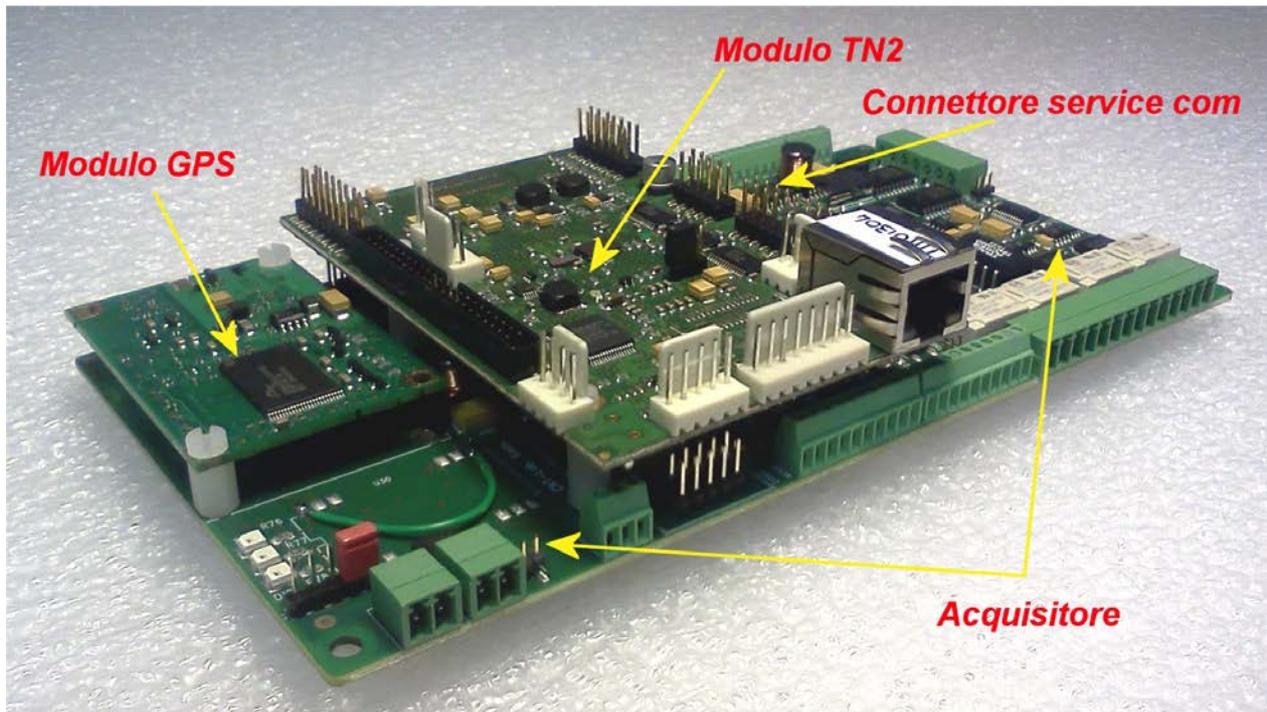


Figura 16. Stazione sismica GAIA2 nella sua configurazione hardware di base.

Il connettore *service com* è la seriale RS-232 utilizzata da *BootMon* per colloquiare con la scheda TN2.

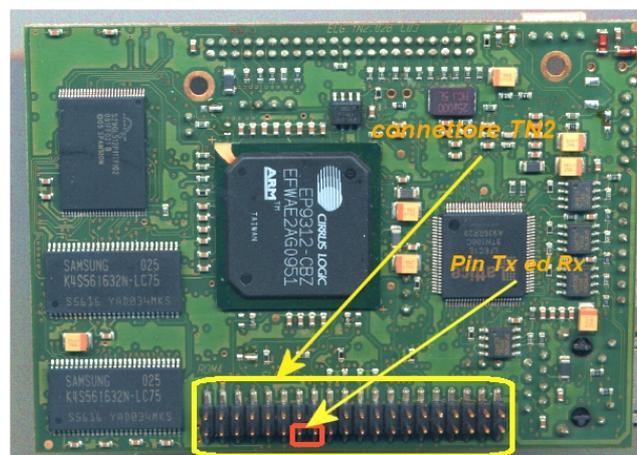


Figura 17. Connettore DIP da 40 pin maschio installato sul modulo TN2. In rosso sono evidenziati i pin TX ed RX della seriale che collega la scheda all'acquisitore.

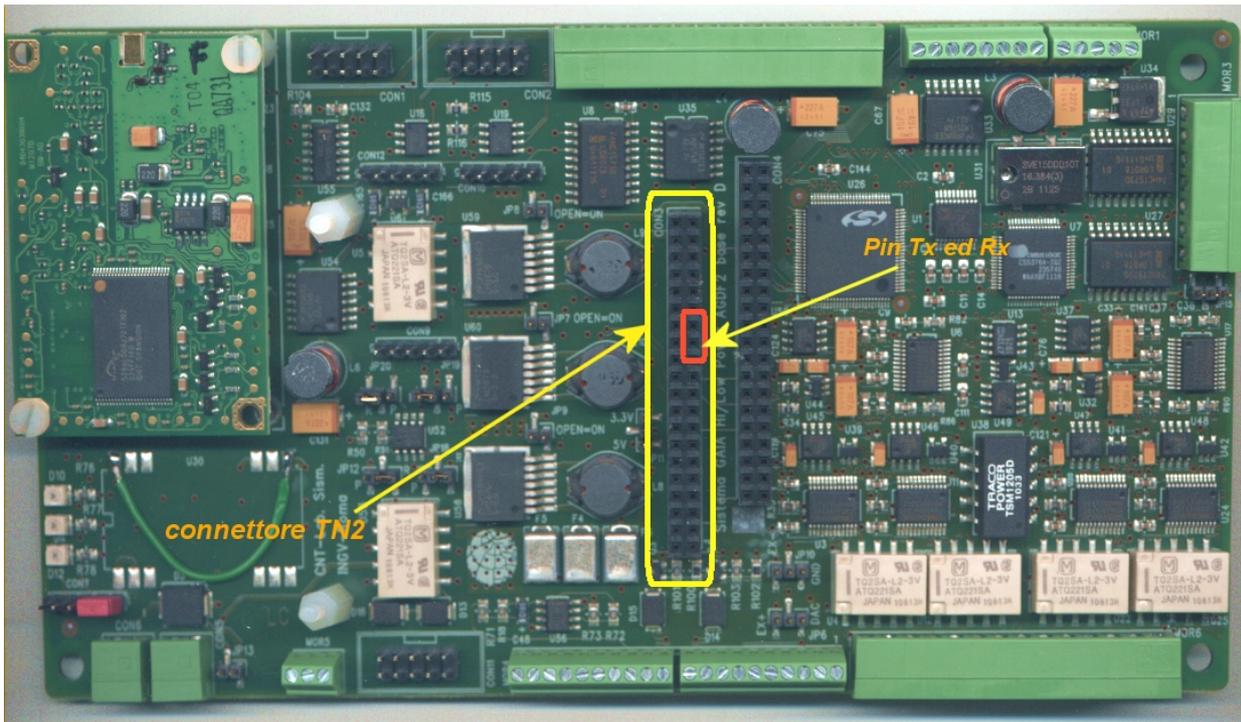


Figura 18. Connettore DIP da 40 pin femmina installato sul digitalizzatore. In rosso sono evidenziati i pin TX ed RX della seriale che collega la scheda al modulo TN2.

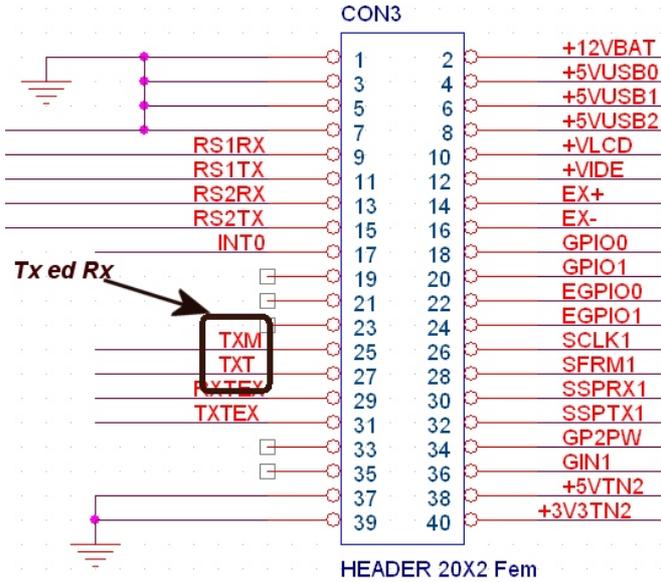


Figura 19. Pin-out del connettore DIP40. Nel riquadro sono evidenziati i pin TX ed RX della seriale interna di collegamento tra modulo TN2 ed acquisitore.

Bibliografia

- [1]. *Software per l'installazione e la configurazione della stazione sismica Gaia2* - Sandro Rao, Leonardo Salvaterra, Catello Acerra, rapporto tecnico INGV n° 130 del 2010.
- [2]. *Visual Basic 6* – Rob Thayer, Ediz. Apogeo 1999.
- [3]. *Learning the bash shell* – Cameron Newham, Bill Rosenblatt, Ed. O'Reilly 1998.
- [4]. *SEED Reference Manual, Seed Format version 2.4 May, 2010, IRIS – Appendix A: channel Naming.*
- [5]. <http://www.seiscomp3.org/wiki/doc/applications/seedlink>
- [6]. <http://www.iris.edu/manuals/>
- [7]. *Upgrading Microsoft Visual Basic 6.0 to Microsoft Visual Basic .Net*, Ed. Robinson, R.I. Oliver, M. Bond (Jan 5, 2002).

Ringraziamenti

Si ringrazia il Dott. Ing. Leonardo Salvaterra per aver fornito la documentazione tecnica riguardante il sistema operativo del modulo TN2 e di alcuni applicativi personalmente sviluppati.

Coordinamento editoriale e impaginazione

Centro Editoriale Nazionale | INGV

Progetto grafico e redazionale

Daniela Riposati | Laboratorio Grafica e Immagini | INGV

© 2012 INGV Istituto Nazionale di Geofisica e Vulcanologia

Via di Vigna Murata, 605

00143 Roma

Tel. +39 06518601 Fax +39 065041181

<http://www.ingv.it>



Istituto Nazionale di Geofisica e Vulcanologia