ISSN 2039-7941
Anno 2012_Numero 228

rapporti tecnici INGV

Progetto HeliDAC

228

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 2012_Numero 228

rapporti tecnici

INGV

PROGETTO HELIDAC

Catello Acerra, Sandro Rao, Carlo Salvaterra, Leonardo Salvaterra, Stefano Silvestri, William Thorossian

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

228

Indice

Introduzione

Il progetto HeliDAC nasce dall'esigenza di visualizzare le forme d'onda, provenienti da un acquisitore sismico digitale, su di un supporto cartaceo termosensibile. Da diversi anni, infatti, grazie al progresso tecnologico, abbiamo assistito a una graduale trasformazione della Rete Sismica Nazionale con conseguente abbandono delle vecchie apparecchiature analogiche a favore dei sistemi sismici digitali.

Qualche anno fa, per il suddetto motivo, fu costruito un sistema prototipo di conversione digitale-analogico [Acerra et al, 2010] in grado di produrre un segnale elettrodinamico compatibile con i sismografi analogici (Helicorder) a carta termosensibile della Teledyne [1]. Le sue caratteristiche erano molto simili al sistema odierno con alcune sostanziali differenze di natura tecnologica, come ad esempio la possibilità di ricevere un flusso digitale solo da una porta seriale RS-232. Il prototipo ha affiancato per alcuni anni gli Helicorder analogici consentendo lo sviluppo della rete sismica digitale costruita intorno ai sistemi GAIA1. Il sistema elettronico di conversione permetteva un facile e rapido controllo dei segnali digitali provenienti da Collegamenti Diretti Numerici (CDN) di tipo seriale.

Sulla base di questa esperienza nasce il progetto HeliDAC con l'intento di realizzare un aggiornamento tecnologico dell'elettronica consentendo di ricevere le informazioni da un flusso digitale con protocollo TCP-IP. Infatti, l'attuale Rete Simica Nazionale utilizza esclusivamente collegamenti IP avendo abbandonato ormai da tempo i collegamenti seriali su linea CDN. Il nuovo progetto HeliDAC ha permesso la conservazione di una parte dei sistemi Helicorder Teledyne grazie alle stazioni con tecnologia GAIA, evitando così la loro definitiva scomparsa, con i vantaggi che questo comporta.

Sono stati progettati e realizzati 8 sistemi elettronici a loro volta ingegnerizzati all'interno di due unità rack 19'' insieme ad altrettanti Helicorder su carta termosensibile. I nuovi sistemi HeliDAC consentono la riproduzione corretta di un segnale sismico analogico in tutte le sue caratteristiche. Uno degli aspetti fondamentali è la corretta rispondenza temporale di un segnale digitalizzato su flusso TCP-IP, quindi non in tempo reale. L'HeliDAC è, infatti, in grado di estrarre la temporizzazione dal flusso digitale numerico e di generare localmente un riferimento temporale assoluto, perfettamente sincronizzato con il tempo origine prodotto durante la fase di digitalizzazione. Inoltre il sistema, a completamento della fruibilità del segnale sismico, riproduce un marcatempo analogico con tacca di riferimento, come da standard internazionali e, rende fruibili le informazioni in modo diretto e immediato con l'ausilio di un Display LCD multi righe. Per rendere più agevole e omogeneo il livello in tensione dei segnali si è predisposto un sistema di controllo del rapporto di amplificazione in forma digitale che agisce direttamente sulla scala di ri-conversione, il livello di tale intervento è chiaramente indicato sul display. A completamento del progetto elettronico, si è dovuto sviluppare un sistema software in grado di poter gestire i flussi digitali delle singole stazioni, programmando la destinazione verso gli heliDAC preposti. Nasce quindi un'interfaccia uomo-macchina che consente la scelta delle stazioni digitali da inviare verso la conversione in analogico, una sorta di associazione Helicorder-GAIA facilmente accessibile e programmabile essendo stata realizzata su piattaforma WEB e quindi accessibile da qualsiasi postazione.

1. HeliDAC

I dati di progetto da cui si è partiti per lo sviluppo del progetto HeliDAC, sono quelli inerenti all'adattamento tra l'ingresso analogico degli helicorder e i segnali digitali delle stazioni GAIA della rete nazionale:

- La dinamica del segnale analogico in ingresso agli amplificatori è di ±5 V;
- La generazione della tacca marcatempo per i minuti, le ore e la mezzanotte con tensione di 12 V;
- La frequenza massima rappresentabile è di 2 Hz alla velocità di trascinamento di 1 mm al secondo;
- Campionamento di 100 sps, invio di pacchetti dati di un secondo di 399 byte e trasmissione dati in protocollo TCP/IP.

1.1 Hardware

L'esperienza pluriennale nella programmazione di microprocessori compatibili con il core 8051 per lo sviluppo del progetto GAIA, ha reso conveniente basare lo sviluppo della parte hardware del progetto sul microcontrollore della Cygnal Silabs C8051F124 [2], il cui schema a blocchi è mostrato in fig. 1.

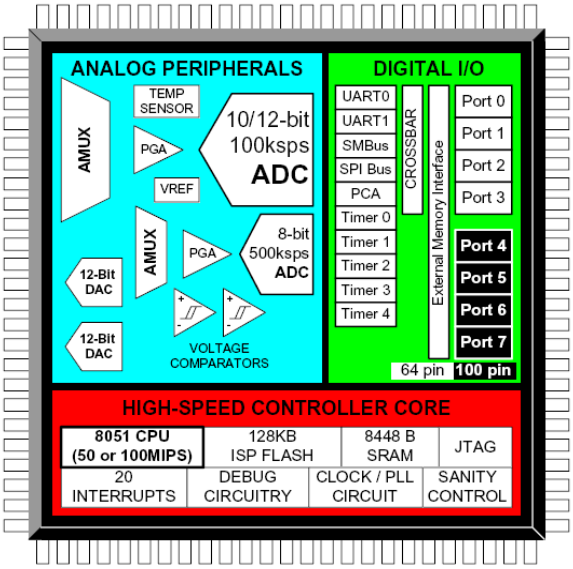


Figura 1. Schema a blocchi del processore CIGNAL della serie C8051Fxxx.

Il chip C8051F124 fa parte dei dispositivi chiamati "mixed-signal System-on-a-Chip MCUs" ed è caratterizzato da: 64 pin digitali di I/O, una pipeline ad alta velocità (50 MIPS) compatibile, come detto, col core dei microcontrollori della serie 8051, un'interfaccia di debug di tipo in-system, un ADC da 12 bit a 100 ksps con PGA e multiplexer analogico ad 8 canali, un ADC da 10 bit a 500 ksps anch'esso con PGA e multiplexer analogico ad 8 canali, due DAC programmabili a 12 bit, una tensione di riferimento, 128 kB di memoria FLASH interna, 8448 (8k + 256) bytes di RAM, interfaccia per memoria esterna con spazio di indirizzamento di 64 kB, bus SPI, SMBus/I2C, 2 UART, 5 contatori a 16 bit, Watchdog Timer, VDD Monitor e sensore di temperatura. Il microprocessore lavora con tensioni tra 2,7 V e 3,6 V con intervallo di temperature industriali (da -45° C a +85° C). Il C8051F124 utilizzato nel progetto è quello con il package TQFP a 100-pin [datasheet, 2002].

L'elenco delle caratteristiche appena descritto fa intuire come l'utilizzo di questo processore, quarzato a 22.118400 MHz, ha consentito di semplificare notevolmente la parte hardware del progetto. Le risorse utilizzate sono: una porta seriale UART [3] impostata a 57600 b/s per la ricezione dei dati digitali in TTL (la seconda UART impostata a 115200 b/s per il debug del firmware scritto in linguaggio assembly), il DAC a 12 bit per la generazione del segnale analogico, le porte di I/O per la generazione dell'impulso della

marcatempo e la lettura dei pulsanti della tastiera relativi all'amplificazione digitale, ed infine un display a 80 caratteri mappato in memoria.

Sono state inoltre realizzate le seguenti sezioni hardware:

- la sezione di alimentazione che produce 3V-200mA, 5V-500mA e ±12V-200mA;
- la sezione analogica di condizionamento del segnale sismico prodotto dal DAC sia in tensione sia in frequenza, necessaria per adattarsi alle caratteristiche di ingresso dell'amplificatore dell'helicorder;
- la sezione per il controllo di un relè a 12V necessario alla generazione della marca del tempo;
- la sezione relativa al controllo di un display a 80 caratteri;
- la sezione relativa all'interfaccia ethernet/seriale costituita dal convertitore EM203EV della Tibbo [4] alimentato a 5 V.

In fig. 2 si può vedere lo schema a blocchi del circuito dell'heliDAC.

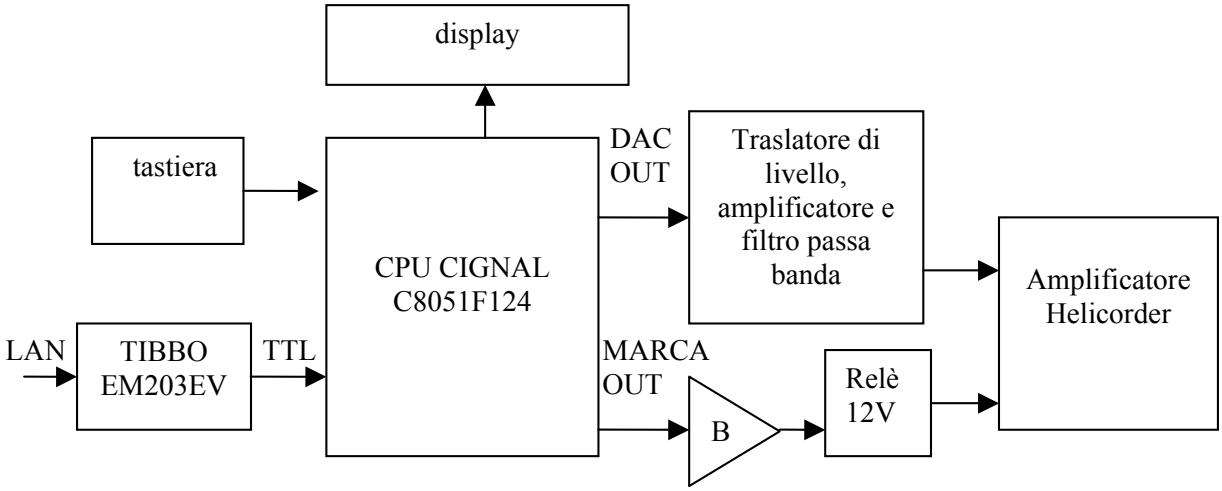


Figura 2. Schema a blocchi del circuito HelyDAC.

Per la sezione hardware di condizionamento del segnale sismico dinanzi elencata, è possibile evidenziarne tre parti costitutive: la prima svolge la funzione di traslatore di livello, portando la dinamica in tensione a lavorare a ±1,2V; la seconda è un preamplificatore X4, il quale porta la dinamica in tensione a lavorare a ±4,8V, quindi abbastanza vicina a quella dell'helicorder; infine la terza parte è un filtro attivo passa banda, di tipo VCVS (Voltage Controlled Voltage Source) di secondo ordine, con frequenze di taglio inferiore e superiore rispettivamente 0,5Hz e 5Hz, il tutto realizzato con amplificatori operazionali e altri componenti discreti. L'introduzione del filtro si è resa necessaria in quanto, come è noto, un segnale analogico prodotto da un DAC è un segnale a gradini di tensione, tanto più ampi, quanto più ampia è la variazione numerica del segnale digitale che lo produce e quanto più bassa è la velocità di campionamento (fig. 3).

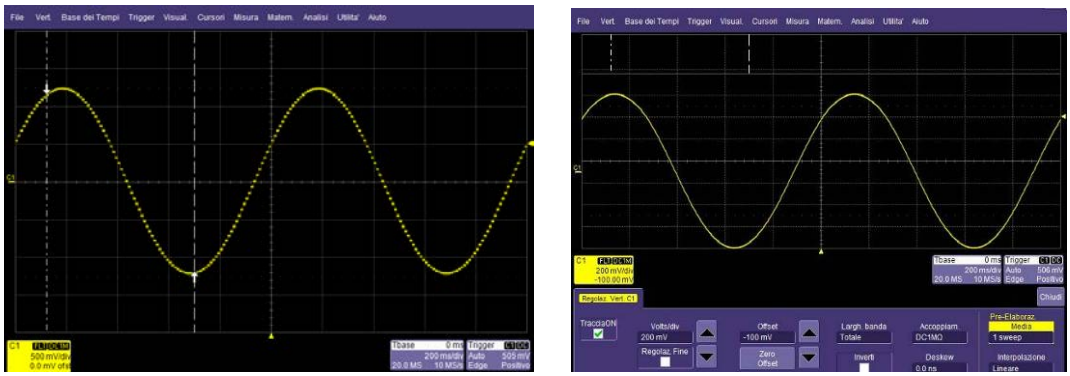


Figura 3. Andamento della forma d'onda di una sinusoide di prova in uscita al DAC prima e dopo il filtro.

I relativi transitori di commutazione rendono il segnale analogico risultante “sporco” di armoniche ad alta frequenza, per cui è necessaria la presenza del taglio di frequenza superiore, mentre è necessaria la presenza del taglio di frequenza inferiore per attenuare le componenti a bassa frequenza presenti nel segnale sismico proveniente dalle stazioni digitali , la maggior parte delle quali equipaggiate con sensore a larga banda, come ad esempio i sensori TRILLIUM ™ della Nanometrics [5], perché tali componenti generalmente molto più ampie del rumore di fondo, costringerebbero all'utilizzo del rullo con un'attenuazione globale troppo alta.

Al pilotaggio del display e alla lettura del tastierino, sono state predisposte due porte I/O (una da otto pin, e una da quattro pin) del microprocessore, per le quali non è stato necessario un adattamento tra segnali TTL a 3V della CPU e quelli TTL a 5V del display, in quanto le tolleranze dei segnali TTL del display sono sufficientemente ampie al riconoscimento dei segnali TTL della CPU. Un trimmer provvede alla regolazione del livello del contrasto. Infine, la retroilluminazione del display a barra LED viene alimentata dalla stessa sezione di alimentazione a 5V attraverso una resistenza di limitazione in corrente.

Il prototipo del circuito è visibile in fig. 4. Con il cerchio rosso è evidenziato il modulo di conversione ethernet/seriale TIBBO EM203EV [4] (fig. 5) collegato direttamente al microprocessore in TTL.

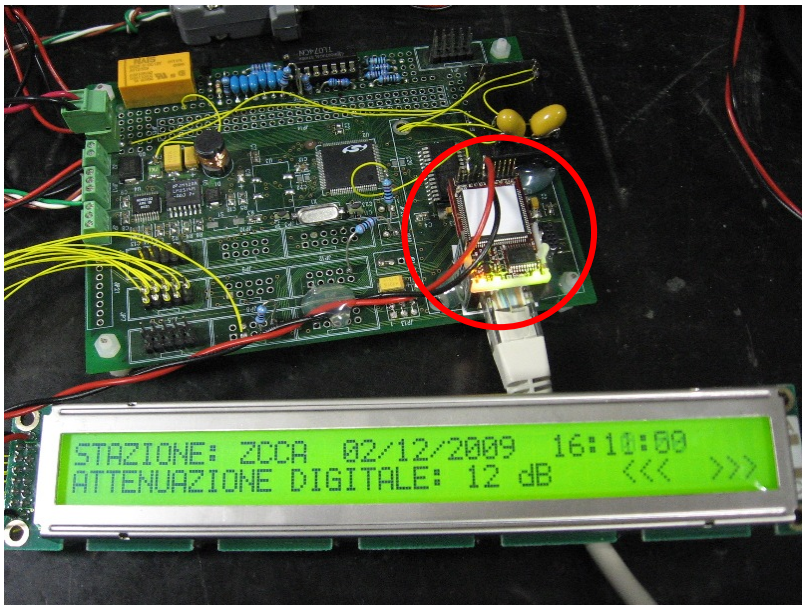


Figura 4. Prototipo hardware completo in funzione.

Questo dispositivo fa parte della famiglia dei moduli TIBBO che consentono di collegare un dispositivo munito di porta seriale a una rete locale Ethernet. Ciò permette l’accesso ai dispositivi seriali da tutti i PC della rete locale o da WAN. Il dispositivo ha un proprio indirizzo IP e può comunicare attraverso gli standard UDP e TCP supportando i protocolli ARP (utilizzato per risolvere gli indirizzi IP in indirizzi MAC) e ICMP (per rispondere alle richieste di “ping”).

Figura 5. Modulo TIBBO EM203EV.

In generale questo dispositivo server seriale (d'ora in avanti Serial Device Server - SDS) è munito di porta Ethernet, tipicamente 10baseT per la connessione alla rete locale con velocità 10/100Mbit/s e di una porta RS232 per il collegamento a una qualsiasi periferica di tipo seriale.

La differente programmabilità del dispositivo gli permette di lavorare sia come dispositivo server (slave) sia come client (master). In questo modo avremo:

- se viene programmato come client, l'SDS non attende richieste ed invia i dati ricevuti dalla seriale alla Ethernet non appena questi sono disponibili. In questa modalità è necessario che venga specificato nel SDS il proprio indirizzo IP e l'IP e la porta di destinazione. La Netmask e l'indirizzo IP del router saranno necessari nel caso il dispositivo di controllo e l'SDS appartengano a due sottoreti differenti;

- nel caso di programmazione server, l'SDS invia o riceve i dati solo dopo una richiesta sull'ethernet da un dispositivo remoto. La tipica applicazione è quella della gestione cosiddetta a "polling". In questa modalità, l'impostazione di rete, anche nel caso sia interposto un router tra il dispositivo remoto e l'SDS, prevede solo l'indirizzo IP e la pota di quest'ultimo.

Il dispositivo EM203EV utilizzato in questo progetto, è composto dall'EM203 (con controller DM9000B) e dal connettore Ethernet 10/100BaseT. Dispone di quattro led di segnalazione interni, un uscita seriale TTL full-duplex e half-duplex con velocità di trasmissione sino a 115 Kbps. La configurazione è di tipo server.

In fig. 6 sono riportate le finestre di configurazione del software DS Manager di gestione del modulo TIBBO. Per maggiori informazioni sui moduli TIBBO si rimanda all'Appendice C.

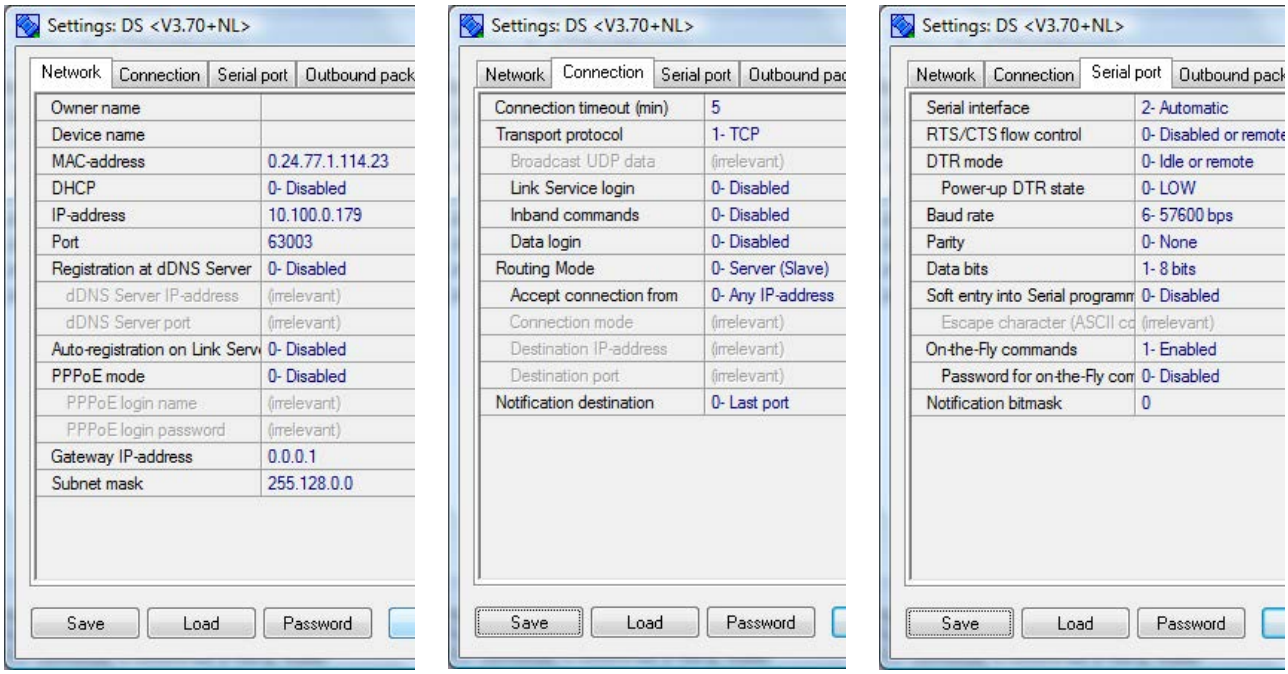


Figura 6. Finestre di configurazione del modulo TIBBO.

Si notano nella prima schermata la configurazione di rete con l'IP, il MacAddress, la subnet mask ecc, nella seconda la modalità Server in cui si accettano connessioni da chiunque e nella terza la configurazione della seriale (57600, 8 bit, nessuna parità).

In figura 7, infine, è riportata l'installazione completa in prova in cui sono visibili il rullo Teledyne, il modulo amplificatore Teledyne e il modulo prototipale.

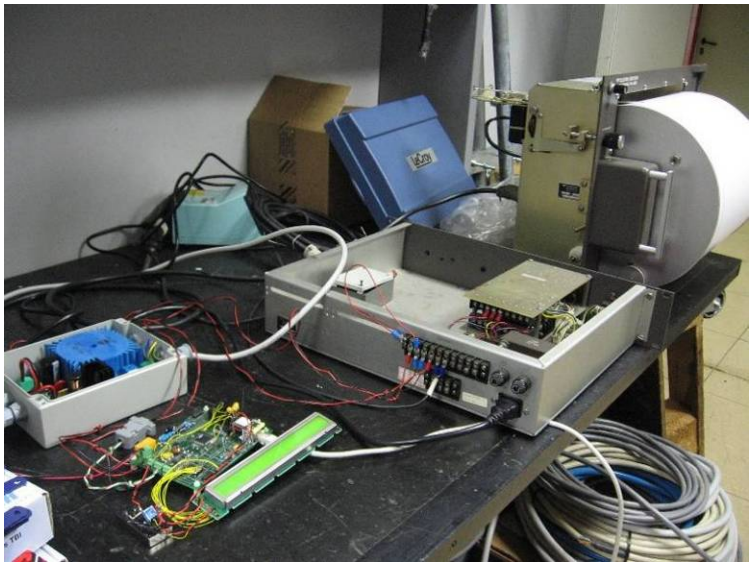


Figura 7. Installazione completa in prova.

Terminato lo sviluppo del prototipo, si è dovuto provvedere a ingegnerizzare il circuito elettronico, i cui schemi circuitali sono riportati in appendice B, per rendere facilmente assemblabile e ripetibile la costruzione del modulo HeliDAC. Questa operazione ha comportato una complessa fase di indagine industriale per poter selezionare i componenti elettronici da utilizzare. La selezione doveva rispondere ad alcuni requisiti fondamentali tra cui la compatibilità degli stessi con i componenti utilizzati nel prototipo, la loro reperibilità nei tempi richiesti dal progetto e il costo contenuto entro un certo range. Inoltre partendo dallo schema elettronico si è dovuto ingegnerizzare il circuito stampato idoneo ad ospitare componenti SMD (a montaggio superficiale) di ultima generazione. Per la realizzazione del circuito stampato e per l'assemblaggio ci si è avvalsi di aziende specializzate presenti sul territorio locale. L'assemblaggio dei componenti SMD deve essere fatta con macchine automatizzate a controllo numerico non disponibili presso i nostri laboratori. Al termine del processo d'industrializzazione si è provveduto a gestire la fase di acquisto coordinando le spedizioni tra i diversi fornitori nazionali. Infine l'ultima operazione è stata quella di coordinare le fasi di assemblaggio presso le aziende specializzate. Il risultato è mostrato fig. 8.

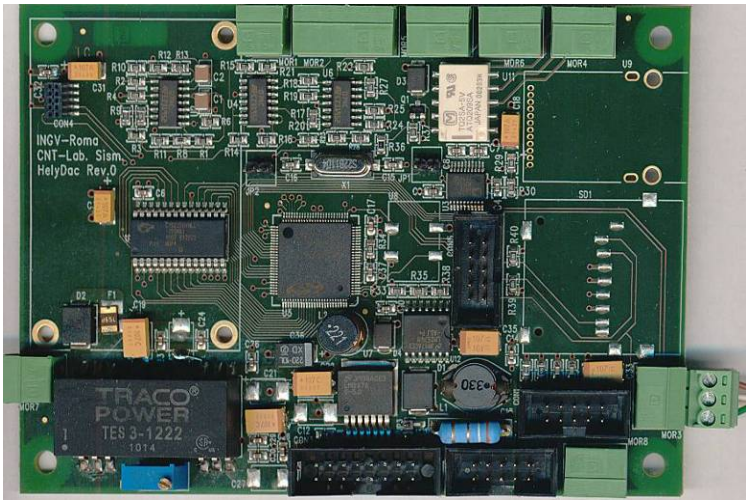


Figura 8. Scheda HeliDAC definitiva con componenti montati.

I prodotti completi sono poi stati sottoposti ad ispezione visiva e al test elettrico. Al termine del test si è passati alla fase di programmazione e collaudo in laboratorio e successivamente si è avviata la fase di assemblaggio nei moduli rack 19" (fig. 9).

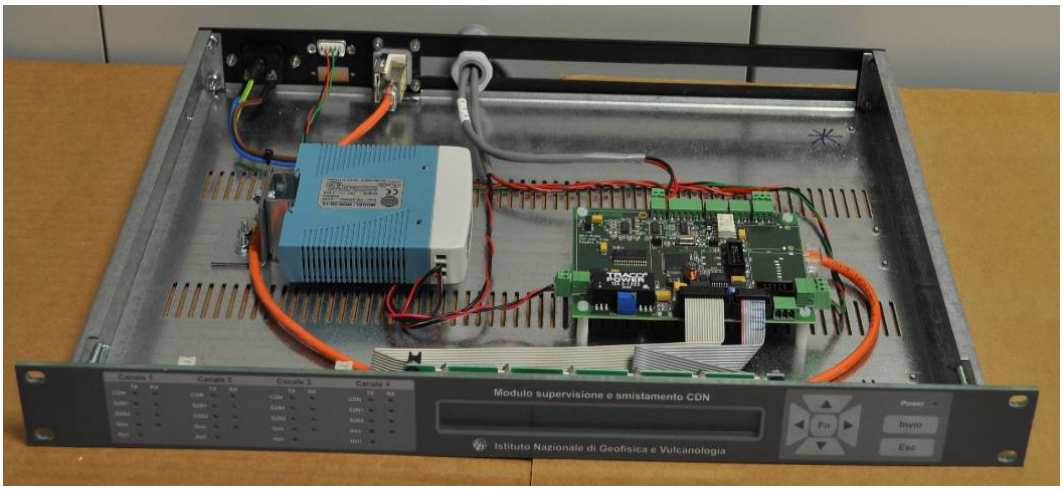


Figura 9. HeliDAC assemblato in modulo da rack 19".

Il costo finale è rimasto abbastanza contenuto perché si è potuto riutilizzare il rack da 19", con il relativo display, del progetto dei smistatori dei segnali numerici su CDN sviluppato nel 2000. L'alimentatore è un modulo switching commerciale da 20 W il cui costo è inferiore ai 30 €, il costo del circuito stampato è inferiore ai 10 €, infine per tutti i componenti la spesa complessiva è inferiore ai 30 €. In definitiva il totale del modulo HeliDAC in funzione è inferiore ai 100 €.

1.2 Firmware

Come detto, l'utilizzo del microprocessore della Cygnal ha permesso di ridurre al minimo lo sviluppo hardware concentrando lo sforzo sul firmware. Nello specifico, le parti più importanti del firmware applicativo sviluppato sono:

1. ricezione attraverso la seriale del flusso dati proveniente da una stazione sismica GAIA;
2. estrazione dall'header dei pacchetti in arrivo, delle informazioni relative al nome stazione, data, ora ed estrazione dei dati sismici da graficare (300 byte = 100 sps da 24bit);
3. gestione di un buffer di 5 secondi dei dati ricevuti al fine di compensare il ritardo che potrebbe introdurre il canale di comunicazione usato;
4. conversione del formato dei dati sismici ricevuti da complemento a 2 (a 24 bit) in formato normalizzato (a 12 bit) adatto al DAC;
5. la scrittura a 100 word rate sul DAC dei dati sismici normalizzati con fattore di attenuazione impostabile con due pulsanti di input (12 valori di attenuazione da 6dB a 72dB);
6. l'aggancio in fase ai pacchetti dati ricevuti al fine di garantire l'allineamento e la sincronizzazione di quelli scritti sul DAC;
7. aggiornamento delle informazioni di stazione sul display.

Per svolgere queste operazioni il firmware dell'heliDAC è stato diviso in due parti:

a) un processo principale di supervisione per il controllo delle operazioni e interazione con le interfacce esterne;

b) due routine, attivate con interrupt, relative alle due funzioni principali e cioè alla ricezione ed estrazione dati dalla telemetria, supportata da un buffer dati di 5 secondi, e alla scrittura sul DAC per la visualizzazione del segnale sul rullo.

La scelta di far svolgere le due operazioni principali attraverso vettori interrupt è stata imposta dal rispetto delle temporizzazioni in gioco, nel senso che sia per la telemetria dati sia per la scrittura sul DAC è necessaria la disponibilità immediata delle risorse del microprocessore. Nel primo caso perché eventuali ritardi sulla ricezione dati comporterebbero la perdita degli stessi e la conseguente presenza di discontinuità sul segnale sismico prodotto dal DAC; nel secondo caso in quanto la velocità di scrittura dei dati sul DAC (nel nostro caso specifico 100 campioni al secondo) deve essere garantita al fine di riprodurre fedelmente sul rullo la forma d'onda del segnale originale. Infatti, un elemento importantissimo è la velocità di avanzamento rotativo del rullo, il quale, come è noto, procede con una velocità fissa imposta dalla frequenza della rete 220V (50Hz ultra stabile) con la quale sono alimentati i suoi motori, producendo una rotazione continua che è completamente svincolata dalla sincronizzazione dei dati delle stazioni sismiche. Per questo motivo la priorità degli interrupt è a favore della scrittura del DAC.

I progressivi eventi che avvengono all'interno delle due routine d'interrupt sono segnalati al processo principale di supervisione attraverso opportuni flags. Pertanto, grazie ai processi serviti dai due vettori interrupt, il microprocessore riesce a scrivere 100 volte al secondo sul DAC e a ricevere contemporaneamente circa 400 byte al secondo dalla telemetria ethernet. Il listato completo e commentato del firmware in assembly è riportato nell'appendice A.

Il funzionamento del processo principale è chiarito dal diagramma di flusso di fig. 10. All'accensione il microprocessore esegue le operazioni d'inizializzazione del sistema specie quelle concernenti i segnali critici in fase di start-up del sistema come, ad esempio, il segnale che pilota il relè della marca del tempo il quale deve necessariamente rimanere interdetto, l'inizializzazione del segnale prodotto dal DAC che deve immediatamente configurarsi in posizione centrale per evitare dannosi movimenti del pennino del rullo all'accensione, l'inizializzazione del display con l'indicazione di attesa dati, ecc.

La successiva operazione è di attivare la seriale di ricezione dei dati sismici, attraverso l'abilitazione del relativo interrupt, e attendere che vengano accumulati 5 secondi di dati nel buffer. Il buffer è di tipo FIFO (first in first out) da otto posizioni con relativi otto flags di stato (1/0 = pieno/vuoto). La verifica dei 5 secondi, quindi, viene fatta testando il flag 5, quando da zero passa a uno il buffer è considerato pieno e si procede.

Con la successiva verifica (test del flag 6) si entra in una particolare routine per garantire l'allineamento, la sincronizzazione e la rimessa in passo tra la velocità dei dati convertiti dal DAC e quella di produzione dei dati dalle stazioni sismiche temporizzate GPS. La tecnica adottata per tale routine è stata quella di anello ad aggancio di fase (Phase Lock Loop) sullo stato di riempimento del buffer di ricezione, nel senso che, presupposta la presenza continua e completa di tutti i dati remoti e quindi in assenza di interruzioni sulle linee di comunicazione, la quantità di dati ricevuti è mediamente la stessa ossia ad un eventuale ritardo di arrivo dati dalla stazione remota dovrà necessariamente seguire un anticipo dei successivi (del resto, se cosi non fosse, ci sarebbero continue perdite di dati). È stata pertanto realizzata una routine di regolazione della frequenza word/rate di scrittura dei dati sul DAC, nell'intorno di questo stato medio di riempimento del buffer dati ricevuti, riproducendo mediamente la stessa velocità di conversione dati che avviene nell'acquisitore della stazione sismica remota ricevuta, garantendo quindi la velocità media di 100 sps. Il test del flag 6, quindi, fornisce l'informazione sull'anticipo o sul ritardo dei dati ricevuti e in base a tale informazione verrà leggermente aumentata o diminuita la velocità di scrittura sul DAC. Dopo questa impostazione si aggiorna il puntatore di lettura del buffer da parte della routine del DAC e si da il via alla scrittura abilitando il relativo interrupt.

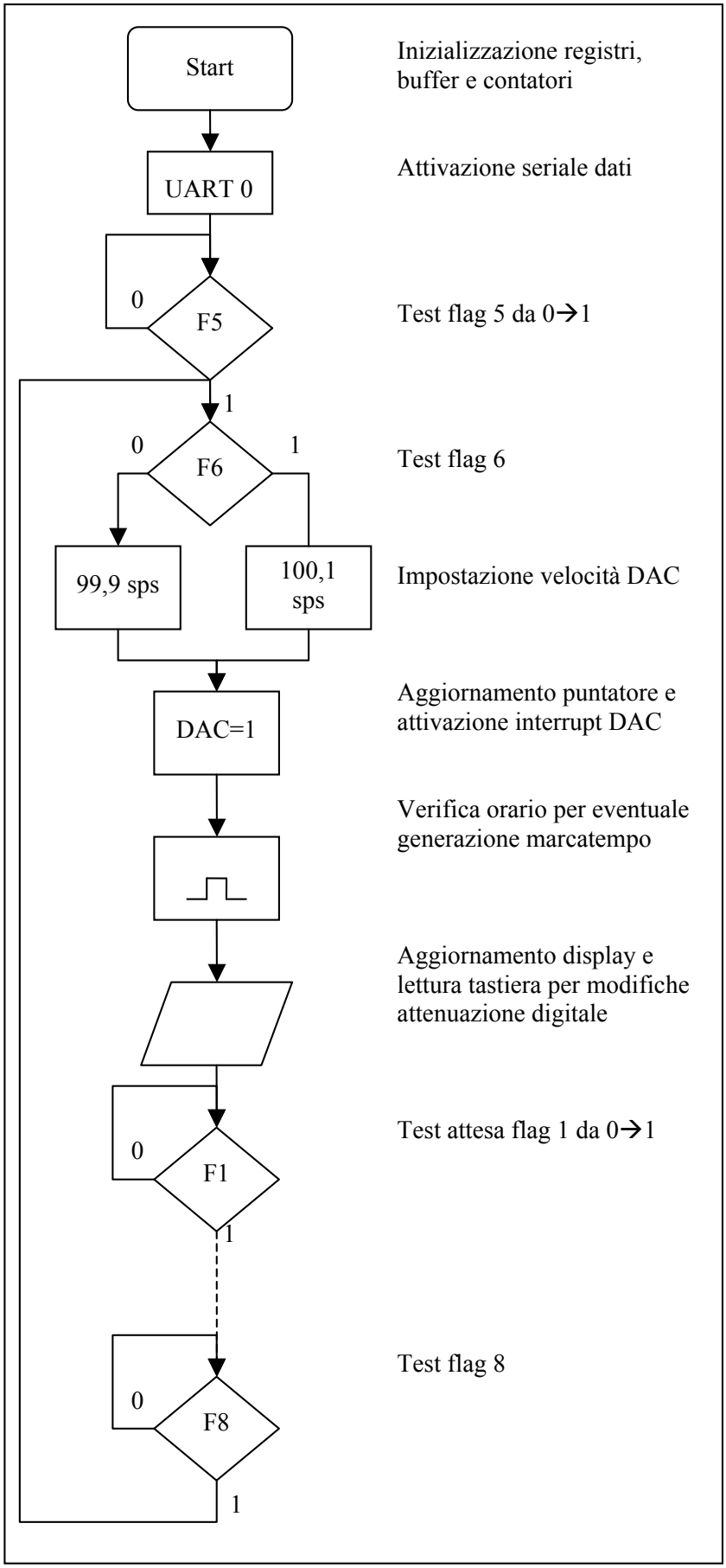


Figura 10. Diagramma di flusso del processo principale del firmware.

Il processo principale prosegue con la verifica dell'orario del pacchetto per l'eventuale generazione della tacca marca tempo (un impulso di durata 1 secondo in corrispondenza dello scadere di ogni minuto, 4 secondi allo scadere dell'ora e 8 secondi allo scadere della mezzanotte), con l'aggiornamento del display scrivendo il nome della stazione dalla quale riceve il segnale digitale, la data, l'ora ed il valore della attenuazione digitale espressa in decibel (dB), ed infine con la lettura dello stato dei pulsanti di incremento o decremento dell'attenuazione digitale da utilizzare nela routine di scrittura del successivo pacchetto.

Da notare che la temporizzazione presa dal pacchetto ricevuto rende ogni heliDAC indipendente ed in un certo senso più preciso temporalmente rispetto alla tecnologia analogica precedente, dove la temporizzazione era unica e uguale per tutti i segnali in arrivo e non considerava le differenze nei ritardi dovuti ai collegamenti.

Infine, prima di processare il flag 2, il microprocessore attende che la routine di scrittura abbia terminato, testando il flag 1 (da 1 deve passare a 0). Le operazioni viste sono ripetute per ogni flag fino all'ottavo per poi riprendere dal primo ciclicamente.

Passando all'analisi delle due routine ad interrupt, quella di ricezione dati viene eseguita quando arriva l'interrupt della seriale: il processore ferma l'esecuzione del processo principale, va a prelevare il dato dal buffer della porta seriale e lo mette nel buffer principale in memoria cambiando da 0 a 1 il flag relativo quando riceve tutti i 100 campioni (da 24 bit) di un pacchetto.

La routine di scrittura del DAC è leggermente più complessa.

L'abilitazione del relativo interrupt, da parte del processo principale, fa partire un timer che dà la frequenza di lettura dal buffer e successiva scrittura nel DAC (variabile come detto in precedenza da 99,9 a 100,1 sps). A ogni fine ciclo del timer e con conseguente generazione dell'interrupt, il processore opera l'attenuazione digitale mediante shift binario e la conversione del dato dal complemento a due a 24 bit al formato normalizzato a 12 bit.

Come è noto nel complemento a due il bit più significativo (nel nostro caso di valore 2^{23}) ha segno negativo per cui si va (in caso di notazione binaria 8 bit) dal valore negativo più piccolo 10000000 al valore positivo più grande 01111111, mentre il formato normalizzato è sempre positivo e va da 00000000 (minimo) a 11111111 (massimo). Ricordiamo che i dati in ingresso nel nostro caso sono a 24 bit mentre quelli per il DAC a 12.

Svuotato il buffer, la routine termina con il reset (da 1 a 0) del relativo flag.

2. Software di smistamento e WEB Interface

Associare direttamente una stazione GAIA ad un HeliDAC senza nessun software di gestione ad alto livello, chiaramente non era praticabile per la complessità nell'operare le modifiche in caso di problemi. Si è deciso pertanto di utilizzare un computer che facesse da interfaccia fra la rete di stazioni GAIA e tutti gli HeliDAC. Per far ciò è stato necessario operare a due livelli di progettazione: un programma che si occupasse della ricezione, elaborazione e successivo smistamento dei pacchetti sismici e un'interfaccia che rendesse facile e intuitiva l'interazione col programma stesso e la scelta delle stazioni. Il programma è stato sviluppato in linguaggio C [Schildt, 1998], mentre per l'interfaccia si è optato per lo sviluppo in linguaggio PHP [6] per l'utilizzo di un qualsiasi browser web, tutto su sistema operativo Linux [Masini, (2006)].

2.1 Programma C: Receive, Computing & Send (RCS)

Le stazioni GAIA sono state progettate con la possibilità di spedire i dati, via TCP/IP, verso un server alla porta di default 63003, nel formato INGV-TWF, formato storico dei dati sismici in cui un pacchetto di dati sismici campionati a 100 sps, sia composto da un header di 64 byte, 300 byte di dati e 35 di dati GPS per un totale di 399 byte [Pintore e Salvaterra, 2007].

La spedizione dei dati prevede un pacchetto per canale, mentre agli heliDAC serve solo il canale verticale inoltre l'eventuale offset presente sul segnale analogico dei sensori, si ripercuote sulla riconversione D/A dell'elettronica sviluppata per ogni helicorder diminuendone la dinamica.

Per tutti questi motivi il programma RCS è stato sviluppato secondo la programmazione concorrente (processi) e comunicazione con i socket (schema a blocchi in fig. 11). Il processo padre si preoccupa solamente di gestire le richieste di connessioni in ingresso alla porta 63003 dalla rete delle stazioni GAIA mentre, per ogni connessione stabilita, crea un nuovo processo. Ogni figlio, quindi, si occupa della ricezione dei dati, l'elaborazione

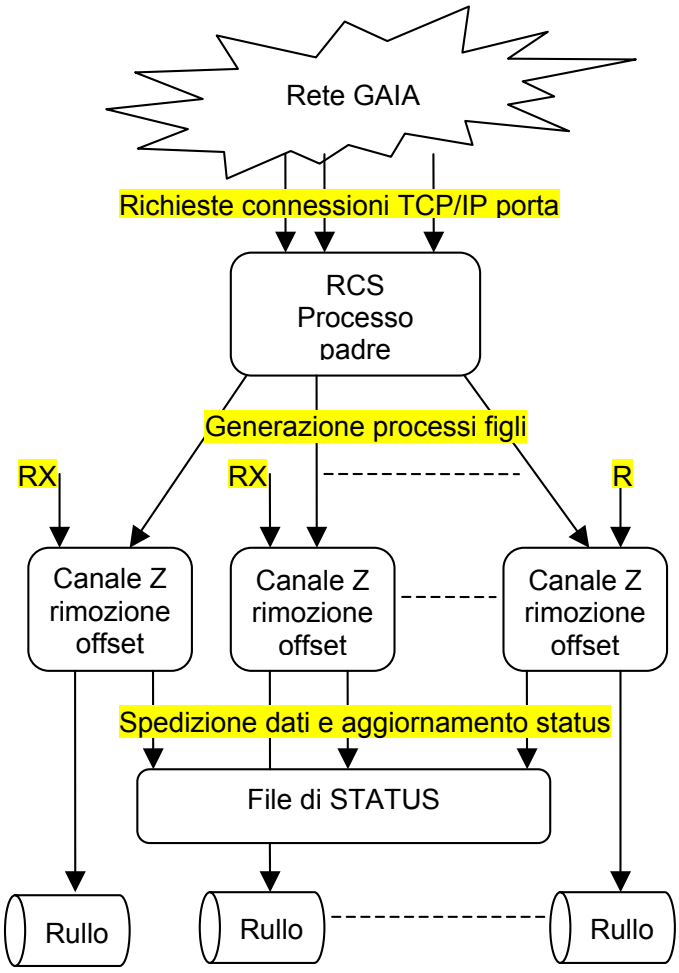


Figura 11. Schema a blocchi di RCS.

(selezione del solo canale verticale e rimozione dell'offset) e successiva spedizione verso l'interfaccia ethernet dell'elettronica di uno degli heliDAC secondo quanto scritto in un file di configurazione. Tale file è composto di coppie di nomi di indirizzi IP, stazione/heliDAC, per cui le stazioni presenti seguono il flusso descritto mentre tutte le altre non vengono elaborate (attualmente, infatti, le stazioni configurate per la spedizione dei dati INGV-TWF per gli 8 heliDAC sono 19). Il valore dell'offset da rimuovere è calcolato e aggiornato ogni minuto di dati. Inoltre RCS si occupa di aggiornare due file necessari all'interfaccia WEB: un file contente i pid (process-id) dei processi figli, per poter resettare il singolo processo di una stazione, e un file di status di tutti gli otto collegamenti. L'accesso al file di status è regolato da quello che, in linguaggio di programmazione, viene chiamato semaforo: se il file è occupato da un processo (per la scrittura) tutti gli altri processi attendono la fine dell'operazione. Il listato C della funzione principale è visibile nell'appendice D.

2.2 Script PHP: index.php

La scelta del WEB per l'interfaccia della gestione delle stazioni sismiche nasce dalla portabilità di funzionamento: basta, infatti, un qualunque browser WEB per accedervi, senza nessuna installazione e configurazione. Il linguaggio scelto per la generazione della pagina WEB dinamica è il PHP (Hypertext Preprocessor) con script Bash [Mendel Cooper, 2006] e il metodo di passaggio variabili è POST.

Lo script PHP (appendice E), eseguito sul server dove risiede, genera la pagina HTML che viene spedita al browser dell'utente a cui viene data la possibilità di eseguire le operazioni di restart del software, restart del processo di una singola stazione e modifica dell'associazione stazione-heliDAC. Ogni singola operazione è memorizzata in un file di log.

L'interfaccia è raggiungibile all'indirizzo http://nara.int.ingv.it/rulli/index.php, e come, ovvio, gira sul server NARA di sala sismica, lo stesso su cui gira il software di monitoraggio delle stazioni MON. Analizzandone la struttura a tabella della pagina principale riportata in fig.12, le righe fanno riferimento al flusso dati stazione GAIA – heliDAC.

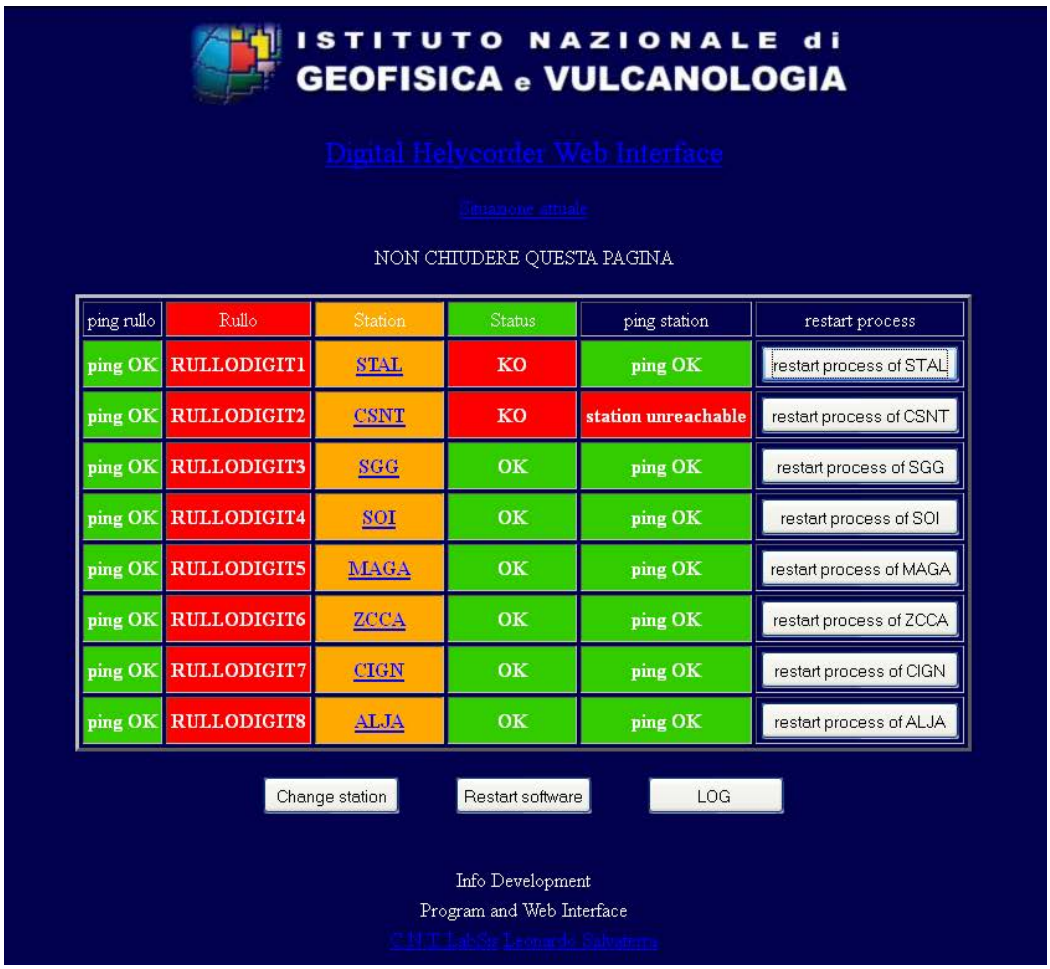


Figura 12. Interfaccia web rulli digitali.

Per ogni riga, la prima casella fornisce l'informazione sull'esito (**ping OK** o **rullo unreachable**) del comando ping verso l'heliDAC indicato nella seconda casella (compare il nome logico dell'indirizzo IP) cui è connessa la stazione indicata nella terza casella (sigla internazionale). Nella quarta, è indicato lo status del collegamento, tratto dal file di status gestito dal programma RCS, e può assumere due valori: OK con sfondo verde, KO con sfondo rosso. La quinta casella dà un'informazione simile alla prima e cioè, il risultato del comando ping verso la stazione (**ping OK** o **station unreachable**).

Nell'ultima casella è presente il pulsante che dà la possibilità, al turnista, di riavviare il processo che gestisce la relativa catena stazione – heliDAC (fig. 13). Anche in questo caso è utilizzato il file contenente i pid dei processi gestito da RCS.

Figura 13. Risultato riavvio processo.

La maggior parte delle volte in cui nello status c'è il KO è dovuta alla disconnessione e riconnessione del software della stazione che avviene in tempi rapidi e in tali casi i risultati dei ping sono OK. Questi ultimi sono un ausilio al turnista per escludere problemi dovuti o alla stazione e al relativo collegamento, oppure al modulo di conversione ethernet – seriale di cui è dotato l'hardware elettronico del "rullo digitale". Nel primo caso, dopo aver appurato, con gli altri strumenti in dotazione al turnista, dell'esistenza di un problema duraturo alla stazione si deve apportare la modifica dell'associazione stazione – heliDAC con il pulsante "Change station", nel secondo si deve avvisare il laboratorio.

I comandi ping vengono eseguiti ad ogni ricarica della pagina, che avviene in automatico ogni 60 secondi oppure cliccando sul link "Situazione attuale" che esegue un refresh della pagina. Vi sono, inoltre, il pulsante "Restart software" con l'ovvia funzione e il pulsante "LOG" che visualizza il file log del software (fig. 14).

Figura 14. log del software.

La pagina per la modifica dell'associazione è visibile in fig.15: è composta da una tabella in cui nella prima colonna vi sono gli heliDAC, nella seconda le attuali stazioni connesse e nella terza vi sono i menù a tendina per la selezione della nuova stazione. Dopo aver finito le variazioni, cliccando sul pulsante "Apply" vengono modificati i file di configurazione e effettuato il riavvio del software. In caso di ripetizione di una stazione viene avvisato l'utente e non eseguito il riavvio.

Figura 15. Interfaccia web per la modifica.

Attualmente le stazioni disponibili sono 19 (STAL, CTL8, MAGA, FVI, PIEI, CRE, CSNT, ZCCA, VVLD, GUAR, FAGN, VCEL, CIGN, SGG, CRAC, ORI, SOI, ALJA, MILZ), scelte per disporre di almeno due preferenze per heliDAC e per zona geografica, ma nulla vieta di abilitarle in maniera differente.

Infine in fig. 16 ci sono due screenshots relativi al restart del software: il primo riguarda la finestra d'inserimento del motivo del riavvio, il secondo il risultato del riavvio stesso. C'è da aggiungere che lo script PHP si preoccupa anche di controllare lo status generale: in caso di colonna *status* tutta rossa avvia un restart straordinario.

Figura 16. Inserimento motivo e risultato del restart.

3. Conclusioni

Il valore storico della visualizzazione su carta termosensibile e degli allarmi dati dai sistemi analogici Teledyne è ormai patrimonio di tutti, proprio questo valore non quantificabile ha reso necessario il progetto HeliDAC che quindi, oltre ad essere un progetto tecnologico, può essere definito come progetto di "recupero storico". Grazie ad esso si è prolungato l'utilizzo degli Helicorder e ora il limite finale è dato dalla vita degli stessi (fuori produzione da metà anni '90). Nel mese di agosto 2010, sono state installate e rese operative in sala sismica quattro unità rack e successivamente, altre quattro nel mese di ottobre 2010 (fig. 17). Ad esclusione di brevi interruzioni dovute a guasti relativi alle stazioni remote assegnate, oppure a piccole modifiche del programma di instradamento segnali, il loro utilizzo, sino alla data odierna è stato pressoché continuativo, pertanto, può essere considerato come collaudo finale del sistema, con esito positivo.

Figura 17. I due rack da quattro helicorder montati in sala sismica.

4. Appendice A: listato assembly del firmware HeliDAC

```
$include (c8051f120.inc)

rit1        equ  30h
rit2        equ  31h
input_m     equ  32h   ;riservato ai dati di input
input_c     equ  33h   ;riservato ai dati di input
input_l     equ  34h   ;riservato ai dati di input
input_ok    equ  35h   ;riservato al flag di segnalazione
ram_b1      equ  0000h ;puntatore dati buffer1
ram_b2      equ  0200h ;puntatore dati buffer2
ram_b3      equ  0400h ;puntatore dati buffer3
ram_b4      equ  0600h ;puntatore dati buffer4
ram_b5      equ  0800h ;puntatore dati buffer5
ram_b6      equ  0a00h ;puntatore dati buffer6
ram_b7      equ  0c00h ;puntatore dati buffer7
ram_b8      equ  0e00h ;puntatore dati buffer8
info_b1     equ  012ch ;puntatore dati info stazione
info_b2     equ  032ch ;puntatore dati info stazione
info_b3     equ  052ch ;puntatore dati info stazione
info_b4     equ  072ch ;puntatore dati info stazione
info_b5     equ  092ch ;puntatore dati info stazione
info_b6     equ  0b2ch ;puntatore dati info stazione
info_b7     equ  0d2ch ;puntatore dati info stazione
info_b8     equ  0f2ch ;puntatore dati info stazione
qta_camp    equ  100d  ;quantita' campioni
ram_in_h    equ  36h ;riservato puntatori ram dati input
ram_in_l    equ  37h ;riservato puntatori ram dati input
output_ok   equ  38h   ;riservato al flag di segnalazione
ram_out_h   equ  39h   ;riservato puntatori ram dati output
ram_out_l   equ  3ah   ;riservato puntatori ram dati output
cont_out    equ  3bh   ;riservato al contatore dati ram output
anno_h      equ  3ch   ;riservato
anno_l      equ  3dh   ;riservato
mese        equ  3eh   ;riservato
giorno      equ  3fh   ;riservato
ore         equ  40h   ;riservato
minuti      equ  41h   ;riservato
secondi     equ  42h   ;riservato all'informazione sui secondi ricevuti
staz_1      equ  43h   ;riservato
staz_2      equ  44h   ;riservato
staz_3      equ  45h   ;riservato
staz_4      equ  46h   ;riservato
staz_5      equ  47h   ;riservato
user1       equ  48h
user2       equ  49h
user3       equ  4ah
passo       equ  4bh   ;riservato ai passi di ricezione byte header
contatore   equ  4ch ;riservato contatore byte header
contatore1  equ  4dh   ;riservato alla quantita' di byte per ogni campione
attenuazione equ 4eh   ;riservato indicatore di attenuazione
buffer_s    equ  4fh   ;riservato buffer status
appo_ramh   equ  50h   ;riservato appoggio ram
appo_raml   equ  51h   ;riservato appoggio ram
info_qta    equ  52h   ;riservato
orem        equ  53h   ;riservato
minutim     equ  54h   ;riservato
secondim    equ  55h   ;riservato
att_val     equ  56h   ;riservato valore attenuazione
att_val_ini equ  2d    ;valore dell'attenuazione NON DEVE MAI ESSERE ZERO!!
 1d=att./2  2d=att./4  3d=att./8 ecc. ecc.
bta1        equ  57h   ;riservato
bta2        equ  58h   ;riservato
bta3        equ  59h   ;riservato
cont_car    equ  5ah   ;riservato
appo_dph    equ  5bh   ;riservato
appo_dpl    equ  5ch   ;riservato

;mappa memoria ram usata
;0000H - 012CH 300 byte buffer1 + 12 byte in coda di info stazione
;0200H - 032CH 300 byte buffer2 + 12 byte in coda di info stazione
;0400H - 052CH 300 byte buffer3 + 12 byte in coda di info stazione
;0600H - 072CH 300 byte buffer4 + 12 byte in coda di info stazione
;0800H - 092CH 300 byte buffer5 + 12 byte in coda di info stazione
;0a00H - 0b2CH 300 byte buffer6 + 12 byte in coda di info stazione
;0c00H - 0d2CH 300 byte buffer7 + 12 byte in coda di info stazione
;0e00H - 0f2CH 300 byte buffer8 + 12 byte in coda di info stazione
;le porte p2 e p3 sono inizializzate nel bios helidac.asm
;tutta la porta p2 e' usta per i dati del display
;p3.0 e' il segnale per il pulsante incremento attenuazione(freccia destra)
;p3.1 e' il segnale per il pulsante decremento attenuazione(freccia sinistra)
;p3.2 e' il segnale per il pulsante Fn
;p3.3 e' il segnale della marca del tempo (ex pulsante Esc)
;p3.4 e' il segnale per il pulsante Invio
;p3.5 e' il segnale RS del display
;p3.6 e' il segnale E del display
;p3.7 e' il segnale PPS

        org 2000h
        ljmp applicativo
        org 2073h
        ljmp timer3_int
```

```
        org 20a3h
        ljmp ser1_int
        org 2100h
db 'CODE-heli2010-21/05/2010-10:00'
db 0ah,0dh
db '&'

applicativo:
        mov a,#38h      ;set 8 bit,caratteri 5x7
        lcall wir_disp  ;scrive istruction register wir
        mov a,#0ch      ;display on e cursor off
        lcall wir_disp
        mov a,#06h      ;cursor shift to the right
        lcall wir_disp
        lcall azz_ram
        mov a,dac0l
        cjne a,#00h,pa          ;controlla prima accensione
        mov att_val,#att_val_ini    ;inizializza valore di attenuazione
        mov dac0l,#01h          ;memorizza prima accensione
pa:     setb p3.0   ;inizializza porta pulsante incremento attenuaz.
        setb p3.1   ;inizializza porta pulsante decremento attenuaz.
        clr p3.3    ;inizializza porta marca del tempo
        orl ckcon,#00000001b    ;imposta prescaler clock timer1 sysclk/4
        mov th1,#0d0h       ;reload timer1 per seriale 1 a 57600b/s

;OSCILLATORE ESTERNO 22.1184Mhz
;nota calcolo del b/s della seriale 1
;si parte dal timer 1 in modo 2 reload 8 bit temporizzato da sysclk/4
;22118400/4=5529600
;poi 5529600/2=2764800
;poi 2764800/38400=72       divide per la velocita' seriale scelta
;poi 256-72=184  cioe' in hex il reload th1 e' : B8H
;mentre per 57600 b/s e' D0H

        mov ref0cn,#00000011b
        mov sfrpage,#01h;seleziona pagina 1
        orl dac1cn,#10000000b ;attivazione del dac1
        mov dac1l,#00000000b
        mov dac1h,#00001000b    ;posizione centrale dac1
        mov rcap3h,#0b8h ;reload timer3 alla frequenza 100 campioni
        mov rcap3l,#00h  ;frequenza centrale
        clr tf3         ;cancella flag interrupt timer 3
        setb tr3        ;avvia timer 3
        mov sfrpage,#00h;seleziona pagina 0

;OSCILLATORE ESTERNO 22.1184Mhz
;nota calcolo timer 3 - si parte dalla 22.118.400/12=1843200
;poi 1843200/100campioni=18432
;poi 65536-18432=47104 , cioe il reload (rcap3) in hex e' : B800H
;nel caso di 100,1 campioni cioe' aumento frequenza il reload e' : B812H
;nel caso di 99,9 campioni cioe' diminuzione frequenza il reload e' : B7EDH

        mov passo,#01h      ;inizializza passo ricezione byte
        mov contatore,#04h          ;inizializza contatore byte
        orl eip2,#01000000b ;priorita' interrupt seriale1
        setb ea                     ;abilita tutte le interruzioni
        mov sfrpage,#01h;seleziona pagina 1
        clr ri1         ;azzera flag seriale 1
        clr ti1         ;azzera flag seriale 1
        mov sfrpage,#00h;seleziona pagina 0

        mov cont_car,#80d           ;scrive messaggio  connessione.....
        mov dptr,#connessione
connessione1:  mov a,#00h
        movc a,@a+dptr
        lcall wdr_disp
        inc dptr
        djnz cont_car,connessione1

attendi:    jnb ri0,vai
uscita:     clr ri0
        mov a,sbuf0
        cjne a,#'F',vai
        anl eie2,#10111110b ;disabilita interrupt timer3 e seriale1
        clr ea                      ;disabilita tutte le interruzioni
        anl eip2,#10111111b ;toglie priorita' interrupt seriale1
        mov sfrpage,#01h;seleziona pagina 1
        clr tr3     ;spegne timer 3
        clr tf3     ;cancella flag interrupt timer 3
        anl dac1cn,#00000000b ;disattivazione del dac1
        mov sfrpage,#00h;seleziona pagina 0
        jmp 0000h

vai:        mov input_ok,#10000000b ;inizializza flag di segnalazione
        mov output_ok,#00h
        mov buffer_s,#00000000b ;inizializza buffer status
        mov cont_out,#qta_camp  ;inizializza contatore campioni
        orl eie2,#01000000b ;abilita interrupt seriale 1

attendis:   jnb ri0,attendiss
        jmp uscita
```

```
attendiss:    mov a,buffer_s
              cjne  a,#00011111b,attendis    ;attende
fino al buffer 5
              mov dptr,#ram_b1   ;inizializza puntatore
buffer 1
              mov ram_out_h,dph
              mov ram_out_l,dpl
              orl  eie2,#00000001b  ;abilita  interrupt
timer 3
              mov a,#01        ;display clear
              lcall wir_disp

attendi1:     jnb ri0,attendi11
uscita1:      jmp uscita
attendi11:    mov a,output_ok
              cjne a,#0ffh,attendi1
              mov output_ok,#00h
              anl    buffer_s,#11111110b    ;aggiorna
buffer 1 usato
              mov a,buffer_s
              jnb  acc.5,dim1    ;se non c'e' buffer 6
diminuisce
              lcall aumenta    ;se c'e buffer 6 aumenta
              jmp aum1
dim1:         jnb acc.1,uscita1 ;controlla presenza
buffer 2
              lcall diminuisci
aum1:              mov  dptr,#ram_b2    ;inizializza
puntatore buffer 2
              mov ram_out_h,dph
              mov ram_out_l,dpl
              orl  eie2,#00000001b  ;abilita  interrupt
timer 3
              mov dptr,#info_b2
              lcall out_display   ;scrive su seriale 0 info
stazione
              lcall marcatempo

attendi2:     jnb ri0,attendi22
uscita2:      jmp uscita
attendi22:    mov a,output_ok
              cjne a,#0ffh,attendi2
              mov output_ok,#00h
              anl    buffer_s,#11111101b    ;aggiorna
buffer 2 usato
              mov a,buffer_s
              jnb  acc.6,dim2    ;se non c'e' buffer 7
diminuisce
              lcall aumenta    ;se c'e buffer 7 aumenta
              jmp aum2
dim2:         jnb acc.2,uscita2 ;controlla presenza
buffer 3
              lcall diminuisci
aum2:   mov dptr,#ram_b3   ;inizializza puntatore
buffer 3
              mov ram_out_h,dph
              mov ram_out_l,dpl
              orl  eie2,#00000001b  ;abilita  interrupt
timer 3
              mov dptr,#info_b3
              lcall out_display   ;scrive su display info
stazione
              lcall marcatempo

attendi3:     jnb ri0,attendi33
uscita3:      jmp uscita
attendi33:    mov a,output_ok
              cjne a,#0ffh,attendi3
              mov output_ok,#00h
              anl    buffer_s,#11111011b    ;aggiorna
buffer 3 usato
              mov a,buffer_s
              jnb  acc.7,dim3    ;se non c'e' buffer 8
diminuisce
              lcall aumenta    ;se c'e buffer 8 aumenta
              jmp aum3
dim3:         jnb acc.3,uscita3 ;controlla presenza
buffer 4
              lcall diminuisci
aum3:              mov  dptr,#ram_b4    ;inizializza
puntatore buffer 4
              mov ram_out_h,dph
              mov ram_out_l,dpl
              orl  eie2,#00000001b  ;abilita  interrupt
timer 3
              mov dptr,#info_b4
              lcall out_display   ;scrive su seriale 0 info
stazione
              lcall marcatempo

attendi4:     jnb ri0,attendi44
uscita4:      jmp uscita
attendi44:    mov a,output_ok
              cjne a,#0ffh,attendi4
              mov output_ok,#00h
              anl    buffer_s,#11110111b    ;aggiorna
buffer 4 usato
              mov a,buffer_s
              jnb  acc.0,dim4    ;se non c'e' buffer 1
diminuisce
              lcall aumenta    ;se c'e buffer 1 aumenta
              jmp aum4
dim4:         jnb acc.4,uscita4 ;controlla presenza
buffer 5
              lcall diminuisci
aum4:              mov  dptr,#ram_b5    ;inizializza
puntatore buffer 5
              mov ram_out_h,dph
              mov ram_out_l,dpl
              orl  eie2,#00000001b  ;abilita  interrupt
timer 3
              mov dptr,#info_b5
              lcall out_display   ;scrive su seriale 0 info
stazione
              lcall marcatempo

attendi5:     jnb ri0,attendi55
uscita5:      jmp uscita
attendi55:    mov a,output_ok
              cjne a,#0ffh,attendi5
              mov output_ok,#00h
              anl    buffer_s,#11101111b    ;aggiorna
buffer 5 usato
              mov a,buffer_s
```

```
        jnb acc.1,dim5     ;se non c'e' buffer 2 diminuisce
        lcall aumenta    ;se c'e buffer 2 aumenta
        jmp aum5
dim5:       jnb acc.5,uscita5 ;controlla presenza buffer 6
        lcall diminuisci
aum5:           mov dptr,#ram_b6     ;inizializza puntatore buffer 6
        mov ram_out_h,dph
        mov ram_out_l,dpl
        orl eie2,#00000001b ;abilita interrupt timer 3
        mov dptr,#info_b6
        lcall out_display   ;scrive su seriale 0 info stazione
        lcall marcatempo

attendi6:     jnb ri0,attendi66
uscita6:      jmp uscita
attendi66:    mov a,output_ok
        cjne a,#0ffh,attendi6
        mov output_ok,#00h
        anl     buffer_s,#11011111b     ;aggiorna buffer 6 usato
        mov a,buffer_s
        jnb acc.2,dim6     ;se non c'e' buffer 3 diminuisce
        lcall aumenta    ;se c'e buffer 3 aumenta
        jmp aum6
dim6:       jnb acc.6,uscita6 ;controlla presenza buffer 7
        lcall diminuisci
aum6:           mov dptr,#ram_b7     ;inizializza puntatore buffer 7
        mov ram_out_h,dph
        mov ram_out_l,dpl
        orl eie2,#00000001b ;abilita interrupt timer 3
        mov dptr,#info_b7
        lcall out_display   ;scrive su seriale 0 info stazione
        lcall marcatempo

attendi7:     jnb ri0,attendi77
uscita7:      jmp uscita
attendi77:    mov a,output_ok
        cjne a,#0ffh,attendi7
        mov output_ok,#00h
        anl     buffer_s,#10111111b     ;aggiorna buffer 7 usato
        mov a,buffer_s
        jnb acc.3,dim7     ;se non c'e' buffer 4 diminuisce
        lcall aumenta    ;se c'e buffer 4 aumenta
        jmp aum7
dim7:       jnb acc.7,uscita7 ;controlla presenza buffer 8
        lcall diminuisci
aum7:           mov dptr,#ram_b8     ;inizializza puntatore buffer 8
        mov ram_out_h,dph
        mov ram_out_l,dpl
        orl eie2,#00000001b ;abilita interrupt timer 3
        mov dptr,#info_b8
        lcall out_display   ;scrive su seriale 0 info stazione
        lcall marcatempo

attendi8:     jnb ri0,attendi88
uscita8:      jmp uscita
attendi88:    mov a,output_ok
        cjne a,#0ffh,attendi8
        mov output_ok,#00h
        anl     buffer_s,#01111111b     ;aggiorna buffer 8 usato
        mov a,buffer_s
        jnb acc.4,dim8     ;se non c'e' buffer 5 diminuisce
        lcall aumenta    ;se c'e buffer 5 aumenta
        jmp aum8
dim8:       jnb acc.0,uscita8 ;controlla presenza buffer 1
        lcall diminuisci
aum8:           mov dptr,#ram_b1     ;inizializza puntatore buffer 1
        mov ram_out_h,dph
        mov ram_out_l,dpl
        orl eie2,#00000001b ;abilita interrupt timer 3
        mov dptr,#info_b1
        lcall out_display   ;scrive su seriale 0 info stazione
        lcall marcatempo
        jmp attendi1

timer3_int:   push acc
        push dph
        push dpl
        push psw
        mov sfrpage,#01h;seleziona pagina 1
        clr tf3
        mov sfrpage,#00h;seleziona pagina 0
        mov dph,ram_out_h
        mov dpl,ram_out_l
        movx a,@dptr
        mov input_l,a     ;lsb
        inc dptr
        movx a,@dptr
        mov input_c,a     ;csb
        inc dptr
        movx a,@dptr
        mov input_m,a     ;msb
        inc dptr
        mov ram_out_h,dph
        mov ram_out_l,dpl
        jb acc.7,negativo
```

;controlla se e' segnale negativo o positivo per la trasformazione da rappresentazione in complemento a due in rappresentazione normalizzata ed essere utilizzato dal dac.

```
        mov attenuazione,att_val
posi_att:   mov a,input_m       ;segnale positivo
        clr c
        rrc a           ;shifta
        mov input_m,a       ;salva
        mov a,input_c
        rrc a           ;shifta
        mov input_c,a       ;salva
        mov a,input_l
        rrc a           ;shifta
        mov input_l,a       ;salva
        djnz attenuazione,posi_att
        mov a,input_m
        jnz fs_posi         ;fondo scala positivo
        mov a,input_c
        anl a,#11111000b        ;maschera bit
segnale
        jnz fs_posi         ;fondo scala positivo
        mov a,input_l       ;scrive valore nel dac
        mov sfrpage,#01h;seleziona pagina 1
        mov dac1l,a
        mov sfrpage,#00h;seleziona pagina 0
        mov a,input_c       ;scrive valore nel dac
        setb acc.3          ;impone valori positivi
        mov sfrpage,#01h;seleziona pagina 1
        mov dac1h,a
        mov sfrpage,#00h;seleziona pagina 0
        jmp esci_t33
fs_posi:    mov sfrpage,#01h;seleziona pagina 1
        mov dac1l,#0ffh         ;scrive valori di
fondo scala
        mov dac1h,#0ffh
        mov sfrpage,#00h;seleziona pagina 0
        jmp esci_t33

negativo:   mov attenuazione,att_val
nega_att:       mov  a,input_m          ;segnale
negativo
        setb c
        rrc a           ;shifta
        mov input_m,a       ;salva
        mov a,input_c
        rrc a           ;shifta
        mov input_c,a       ;salva
        mov a,input_l
        rrc a           ;shifta
        mov input_l,a       ;salva
        djnz attenuazione,nega_att
        mov a,input_m
        cjne    a,#0ffh,fs_nega     ;fondo scala
negativo
        mov a,input_c
        orl  a,#00000111b       ;maschera bit
segnale
        cjne    a,#0ffh,fs_nega     ;fondo scala
negativo
        mov a,input_l       ;scrive valore nel dac
        mov sfrpage,#01h;seleziona pagina 1
        mov dac1l,a
        mov sfrpage,#00h;seleziona pagina 0
        mov a,input_c       ;scrive valore nel dac
        clr acc.3           ;impone valori negativi
        mov sfrpage,#01h;seleziona pagina 1
        mov dac1h,a
        mov sfrpage,#00h;seleziona pagina 0
        jmp esci_t33
fs_nega:        mov sfrpage,#01h;seleziona pagina
1
        mov dac1l,#00h          ;scrive valori di
fondo scala
        mov dac1h,#00h
        mov sfrpage,#00h;seleziona pagina 0
esci_t33:   djnz cont_out,esci_t3
        mov  cont_out,#qta_camp     ;ricarica
contatore campioni
        mov output_ok,#0ffh
        anl eie2,#11111110b ;disabilita interrupt
timer3
esci_t3:    pop psw
        pop dpl
        pop dph
        pop acc
        reti

ser1_int:   push acc
        push dph
        push dpl
        push psw
        mov  appo_ramh,ram_in_h     ;salva
puntatore ram
        mov appo_raml,ram_in_l
        mov sfrpage,#01h;seleziona pagina 1
        clr ri1     ;cancella flag seriale
        clr ti1
        mov b,sbuf1
        mov sfrpage,#00h;seleziona pagina 0
        mov a,passo
        cjne a,#01h,passo2      ;controlla primo ff
        mov a,b
        cjne a,#0ffh,esci_ser11
        mov passo,#02h
        dec contatore
esci_ser11:    jmp esci_ser1
esci_sernoo:   jmp esci_serno
passo2:     cjne a,#02h,passo3
        mov a,b
        cjne a,#0ffh,esci_sernoo  ;impone altre 3
volte ff
        djnz contatore,esci_ser11
        mov passo,#03h
        mov contatore,#04h
        jmp esci_ser1

passo3:     cjne a,#03h,passo4
        mov a,b
        cjne  a,#00h,esci_sernoo    ;impone  4
volte 00
        djnz contatore,esci_ser11
        mov passo,#04h
        mov contatore,#2d
        jmp esci_ser1

passo4:     cjne a,#04h,passo5
```

```
        djnz    contatore,esci_ser11    ;salta    2
caratteri
        mov passo,#05h
        jmp esci_ser1

passo5:     cjne a,#05h,passo6
        mov anno_l,b
        mov passo,#06h
        jmp esci_ser1
passo6:     cjne a,#06h,passo7
        mov anno_h,b
        mov passo,#07h
        jmp esci_ser1
passo7:     cjne a,#07h,passo8
        mov mese,b
        mov passo,#08h
        jmp esci_ser1
passo8:     cjne a,#08h,passo9
        mov giorno,b
        mov passo,#09h
        jmp esci_ser1

passo9:     cjne a,#09h,passoa
        mov ore,b
        mov passo,#0ah
        jmp esci_ser1

passoa:     cjne a,#0ah,passob
        mov minuti,b
        mov passo,#0bh
        jmp esci_ser1

passob:     cjne a,#0bh,passoc
        mov secondi,b
        mov passo,#0ch
        mov contatore,#7d
        jmp esci_ser1
passoc:     cjne a,#0ch,passod
        djnz    contatore,esci_ser12    ;salta    7
caratteri
        mov passo,#0dh
        jmp esci_ser1

passod:     cjne a,#0dh,passoe
        mov staz_1,b
        mov passo,#0eh
        jmp esci_ser1
passoe:     cjne a,#0eh,passof
        mov staz_2,b
        mov passo,#0fh
        jmp esci_ser1
passof:     cjne a,#0fh,passo10
        mov staz_3,b
        mov passo,#10h
        jmp esci_ser1
passo10:    cjne a,#10h,passo11
        mov staz_4,b
        mov passo,#11h
        jmp esci_ser1
passo11:    cjne a,#11h,passo12
        mov staz_5,b
        mov passo,#12h
        mov contatore,#2d
        jmp esci_ser1

passo12:    cjne a,#12h,passo13
        djnz    contatore,esci_ser12    ;salta    2
caratteri
        mov passo,#13h
        jmp esci_ser1

esci_ser12:  jmp esci_ser1
esci_serno1:  jmp esci_serno

passo13:    cjne a,#13h,passo14
        mov a,b
        cjne    a,#'Z',esci_serno1      ;impone    il
carattere Z
        mov passo,#14h
        mov contatore,#32d
        jmp esci_ser1

passo14:    cjne a,#14h,passo15
        djnz    contatore,esci_ser12    ;salta    32
caratteri
        mov passo,#15h
        mov contatore,#qta_camp
        mov contatore1,#03h
        jmp esci_ser1

passo15:    cjne a,#15h,passo16
        lcall memo_in
        djnz  contatore,esci_ser12  ;memorizza
la quantita' campioni
        mov  contatore,#qta_camp    ;per 3 byte
di ogni campione
        djnz contatore1,esci_ser12
        mov passo,#16h
        mov contatore,#32d
        jmp esci_ser1

passo16:    cjne a,#16h,passo17
        djnz    contatore,esci_ser12    ;salta    32
caratteri
        mov passo,#17h
        jmp esci_ser1

passo17:    cjne a,#17h,passo18
        mov a,b
        cjne    a,#'E',esci_serno1      ;impone    il
carattere E
        mov passo,#18h
        jmp esci_ser1

passo18:    cjne a,#18h,passo19
        mov a,b
        cjne    a,#'O',esci_serno1      ;impone    il
carattere O
        mov passo,#19h
        jmp esci_ser1

passo19:    mov a,b
        cjne    a,#'B',esci_serno1      ;impone    il
carattere B
```

```
            mov b,staz_1
            lcall memo_in
            mov b,staz_2
            lcall memo_in
            mov b,staz_3
            lcall memo_in
            mov b,staz_4
            lcall memo_in
            mov b,staz_5
            lcall memo_in
            mov b,giorno
            lcall memo_in
            mov b,mese
            lcall memo_in
            mov b,anno_h
            lcall memo_in
            mov b,anno_l
            lcall memo_in
            mov b,ore
            lcall memo_in
            mov b,minuti
            lcall memo_in
            mov b,secondi
            lcall memo_in
            mov passo,#01h      ;inizializza nuova
ricezione
            mov contatore,#04h   ;inizializza nuova
ricezione
            mov a,input_ok
            rl a
            mov input_ok,a
            cjne a,#00000001b,buff2
            orl buffer_s,a     ;aggiorna buffer status
            mov dptr,#ram_b2   ;inizializza puntatore
buffer 2
            mov ram_in_h,dph
            mov ram_in_l,dpl
            jmp esci_ser1

buff2:      cjne a,#00000010b,buff3
            orl buffer_s,a     ;aggiorna buffer status
            mov dptr,#ram_b3   ;inizializza puntatore
buffer 3
            mov ram_in_h,dph
            mov ram_in_l,dpl
            jmp esci_ser1

buff3:      cjne a,#00000100b,buff4
            orl buffer_s,a     ;aggiorna buffer status
            mov dptr,#ram_b4   ;inizializza puntatore
buffer 4
            mov ram_in_h,dph
            mov ram_in_l,dpl
            jmp esci_ser1

buff4:      cjne a,#00001000b,buff5
            orl buffer_s,a     ;aggiorna buffer status
            mov dptr,#ram_b5   ;inizializza puntatore
buffer 5
            mov ram_in_h,dph
            mov ram_in_l,dpl
            jmp esci_ser1

buff5:      cjne a,#00010000b,buff6
            orl buffer_s,a     ;aggiorna buffer status
            mov dptr,#ram_b6   ;inizializza puntatore
buffer 6
            mov ram_in_h,dph
            mov ram_in_l,dpl
            jmp esci_ser1

buff6:      cjne a,#00100000b,buff7
            orl buffer_s,a     ;aggiorna buffer status
            mov dptr,#ram_b7   ;inizializza puntatore
buffer 7
            mov ram_in_h,dph
            mov ram_in_l,dpl
            jmp esci_ser1

buff7:      cjne a,#01000000b,buff8
            orl buffer_s,a     ;aggiorna buffer status
            mov dptr,#ram_b8   ;inizializza puntatore
buffer 8
            mov ram_in_h,dph
            mov ram_in_l,dpl
            jmp esci_ser1

buff8:      orl buffer_s,a     ;aggiorna buffer status
            mov dptr,#ram_b1   ;inizializza puntatore
buffer 1
            mov ram_in_h,dph
            mov ram_in_l,dpl
            jmp esci_ser1

esci_serno:   mov ram_in_h,appo_ramh
            mov ram_in_l,appo_raml
            mov passo,#01h             ;esce se non
corrispondono
            mov contatore,#04h      ;le condizioni
                ;e inizializza sempre nuova ricezione
esci_ser1:    pop psw
            pop dpl
            pop dph
            pop acc
            reti

memo_in:      mov dph,ram_in_h     ;scrittura in
ram di input
            mov dpl,ram_in_l
            mov a,b
            movx @dptr,a
            inc dptr
            mov ram_in_h,dph
            mov ram_in_l,dpl
            ret

aumenta:      mov sfrpage,#01h;seleziona pagina
1
            mov rcap3h,#0b8h      ;reload timer3
aumento della frequenza
            mov rcap3l,#12h
            mov sfrpage,#00h;seleziona pagina 0
            ret
diminuisci:  mov sfrpage,#01h;seleziona pagina 1
```

```
        mov rcap3h,#0b7h      ;reload timer3
diminuzione della frequenza
        mov rcap3l,#0edh
        mov sfrpage,#00h;seleziona pagina 0
        ret

;OSCILLATORE ESTERNO 22.1184Mhz
;nota calcolo timer 3 - si parte dalla
22.118.400/12=1843200
;poi 1843200/100campioni=18432
;poi 65536-18432=47104 , cioe il reload (rcap3) in
hex e' : B800H
;nel caso di 100,1 campioni cioe' aumento
frequenza il reload e' : B812H
;nel caso di 99,9 campioni cioe' diminuzione
frequenza il reload e' : B7EDH

marcatempo:   mov a,orem
        cjne a,#00d,c_ora      ;controlla se
mezzanotte
        mov a,minutim
        cjne a,#00d,c_minuti ;controlla ore
        mov a,secondim
        cjne a,#00d,c_41s   ;controlla i minuti
        setb p3.3
        ret
c_41s:      cjne a,#01d,c_42s
        setb p3.3
        ret
c_42s:      cjne a,#02d,c_43s
        setb p3.3
        ret
c_43s:      cjne a,#03d,c_44s
        setb p3.3
        ret
c_44s:      cjne a,#04d,c_45s
        setb p3.3
        ret
c_45s:      cjne a,#05d,c_46s
        setb p3.3
        ret
c_46s:      cjne a,#06d,c_47s
        setb p3.3
        ret
c_47s:      cjne a,#07d,c_48s
        setb p3.3
        ret
c_48s:      clr p3.3
        ret
c_ora:     mov a,minutim
        cjne a,#00d,c_minuti
        mov a,secondim
        cjne a,#00d,c_21s
        setb p3.3
        ret
c_21s:      cjne a,#01d,c_22s
        setb p3.3
        ret
c_22s:      cjne a,#02d,c_23s
        setb p3.3
        ret
c_23s:      cjne a,#03d,c_24s
        setb p3.3
        ret
c_24s:      clr p3.3
        ret
c_minuti:     mov a,secondim
        cjne a,#00d,c_11s
        setb p3.3
        ret
c_11s:      clr p3.3
        ret

send_a:     jnb ti0,$
        clr ti0
        mov sbuf0,a
        ret

azz_ram:     mov dptr,#0000h
azz_ram1:    mov a,#00h
        movx @dptr,a
        inc dptr
        mov a,dph
        cjne a,#20h,azz_ram1
        ret

out_display:  jb p3.0,contp31
        mov a,att_val
        cjne a,#12d,inc_att
        jmp no_att
inc_att:    inc att_val
        jmp no_att
contp31:     jb p3.1,no_att
        mov a,att_val
        cjne a,#1d,dec_att
        jmp no_att
dec_att:     dec att_val
no_att:
        mov appo_dph,dph   ;salva dptr per dati
info
        mov appo_dpl,dpl
        mov cont_car,#10d  ;scrive STAZIONE:
        mov dptr,#stazione
stazione1:    mov a,#00h
        movc a,@a+dptr
        lcall wdr_disp
        inc dptr
        djnz cont_car,stazione1
        mov dph,appo_dph      ;ripristina dptr per
dati info
        mov dpl,appo_dpl
        mov info_qta,#5d          ;scrive nome
stazione 5 caratteri
out_dis:     movx a,@dptr
        inc dptr
        lcall wdr_disp
        djnz info_qta,out_dis
        mov a,#' '          ;scrive spazio
        lcall wdr_disp
        movx a,@dptr
        inc dptr
        lcall s_bin2ascii    ;scrive giorno
        mov a,#'/'          ;scrive /
        lcall wdr_disp
```

```
        movx a,@dptr
        inc dptr
        lcall s_bin2ascii     ;scrive mese
        mov a,#'/'            ;scrive /
        lcall wdr_disp
        mov a,#'2'        ;scrive 20 dell'anno 2000
        lcall wdr_disp
        mov a,#'0'
        lcall wdr_disp
        inc dptr              ;salta msb dell'anno
        movx a,@dptr          ;lsb dell'anno
        inc dptr
        subb a,#0d0h          ;sottrae anno 2000 e
fino a 2042 esatto
        lcall  s_bin2ascii          ;scrive  differenza
anno attuale - 2000
        mov a,#' '            ;scrive spazio
        lcall wdr_disp
        mov a,#' '            ;scrive spazio
        lcall wdr_disp
        movx a,@dptr
        inc dptr
        mov orem,a
        lcall s_bin2ascii     ;scrive ore
        mov a,#':'            ;scrive :
        lcall wdr_disp
        movx a,@dptr
        inc dptr
        mov minutim,a
        lcall s_bin2ascii     ;scrive minuti
        mov a,#':'            ;scrive :
        lcall wdr_disp
        movx a,@dptr
        inc dptr
        mov secondim,a
        lcall s_bin2ascii     ;scrive secondi
        lcall  s_attenuaz             ;scrive  valore
attenuazione
        ret

wir_disp:     clr p3.5     ;abilita registro istruzioni
        mov p2,a
        setb p3.6
        mov user3,#80h
        djnz user3,$
        clr p3.6
        mov user3,#00h
        djnz user3,$
        ret

wdr_disp:     setb p3.5     ;abilita registro dati
        mov p2,a
        setb p3.6
        mov user3,#80h
        djnz user3,$
        clr p3.6
        mov user3,#00h
        djnz user3,$
        ret

s_bin2ascii:   lcall bin2ascii
        mov a,bta2
        lcall wdr_disp
        mov a,bta3
        lcall wdr_disp
        ret

bin2ascii:      mov b,#100d           ;divisione del
numero per 100 decimale
        div ab            ;l'intero e' in (A) ed il resto
in (B)
        add  a,#30h             ;trasformazione  in
numero ASCII
        mov  bta1,a             ;trasferimento  nel
registro di uscita
        mov a,b           ;resto da dividere in (A)
        mov b,#10d        ;divisione del resto per
10 decimale
        div ab  ;l'intero e' in (A) ed il resto in (B)
        add  a,#30h             ;trasformazione  in
numero ASCII
        mov  bta2,a             ;trasferimento  nel
registro di uscita
        mov a,b           ;resto in (A)
        add  a,#30h             ;trasformazione  in
numero ASCII
        mov  bta3,a             ;trasferimento  nel
registro di uscita
        ret             ;fine routine

s_attenuaz:     mov cont_car,#4d          ;scrive 4
spazi per terminare riga
        mov a,#' '
attenuaz1:     lcall wdr_disp
        djnz cont_car,attenuaz1
        mov  cont_car,#23d                ;scrive
attenuazione digitale:
        mov dptr,#att_mess
attenuaz2:     mov a,#00h
        movc a,@a+dptr
        lcall wdr_disp
        inc dptr
        djnz cont_car,attenuaz2
        mov cont_car,#3d        ;scrive <<<
        mov a,#'<'
attenuaz3:     lcall wdr_disp
        djnz cont_car,attenuaz3
        mov a,att_val
        cjne a,#01d,ca_02
        mov a,#' '
        lcall wdr_disp
        mov a,#'6'
        lcall wdr_disp
        jmp fine_att
ca_02:        cjne a,#02d,ca_03
        mov a,#'1'
        lcall wdr_disp
        mov a,#'2'
        lcall wdr_disp
        jmp fine_att
ca_03:        cjne a,#03d,ca_04
        mov a,#'1'
        lcall wdr_disp
        mov a,#'8'
```

```
            lcall wdr_disp
            jmp fine_att
ca_04:      cjne a,#04d,ca_05
            mov a,#'2'
            lcall wdr_disp
            mov a,#'4'
            lcall wdr_disp
            jmp fine_att
ca_05:      cjne a,#05d,ca_06
            mov a,#'3'
            lcall wdr_disp
            mov a,#'0'
            lcall wdr_disp
            jmp fine_att
ca_06:      cjne a,#06d,ca_07
            mov a,#'3'
            lcall wdr_disp
            mov a,#'6'
            lcall wdr_disp
            jmp fine_att
ca_07:      cjne a,#07d,ca_08
            mov a,#'4'
            lcall wdr_disp
            mov a,#'2'
            lcall wdr_disp
            jmp fine_att
ca_08:      cjne a,#08d,ca_09
            mov a,#'4'
            lcall wdr_disp
            mov a,#'8'
            lcall wdr_disp
            jmp fine_att
ca_09:      cjne a,#09d,ca_10
            mov a,#'5'
            lcall wdr_disp
            mov a,#'4'
            lcall wdr_disp
            jmp fine_att
ca_10:      cjne a,#10d,ca_11
            mov a,#'6'
            lcall wdr_disp
            mov a,#'0'
            lcall wdr_disp
            jmp fine_att
ca_11:      cjne a,#11d,ca_12
            mov a,#'6'
            lcall wdr_disp
            mov a,#'6'
            lcall wdr_disp
            jmp fine_att
ca_12:      mov a,#'7'
            lcall wdr_disp
            mov a,#'2'
            lcall wdr_disp
fine_att:   mov a,#'d'
            lcall wdr_disp
            mov a,#'B'
            lcall wdr_disp
            mov cont_car,#3d        ;scrive >>>
            mov a,#'>'
attenuaz4:  lcall wdr_disp
            djnz cont_car,attenuaz4
            mov cont_car,#7d        ;scrive 7 spazi
            mov a,#' '
attenuaz5:  lcall wdr_disp
            djnz cont_car,attenuaz5
            ret
stazione:
db 'STAZIONE: '
att_mess:
db  'ATTENUAZIONE  DIGITALE:  '             ;23
CARATTERI
connessione:
db 'connessione.....                    ' ;40 caratteri
db '                                    ' ;40 caratteri

        end
```

5. Appendice B: schemi circuitali dell'HeliDAC

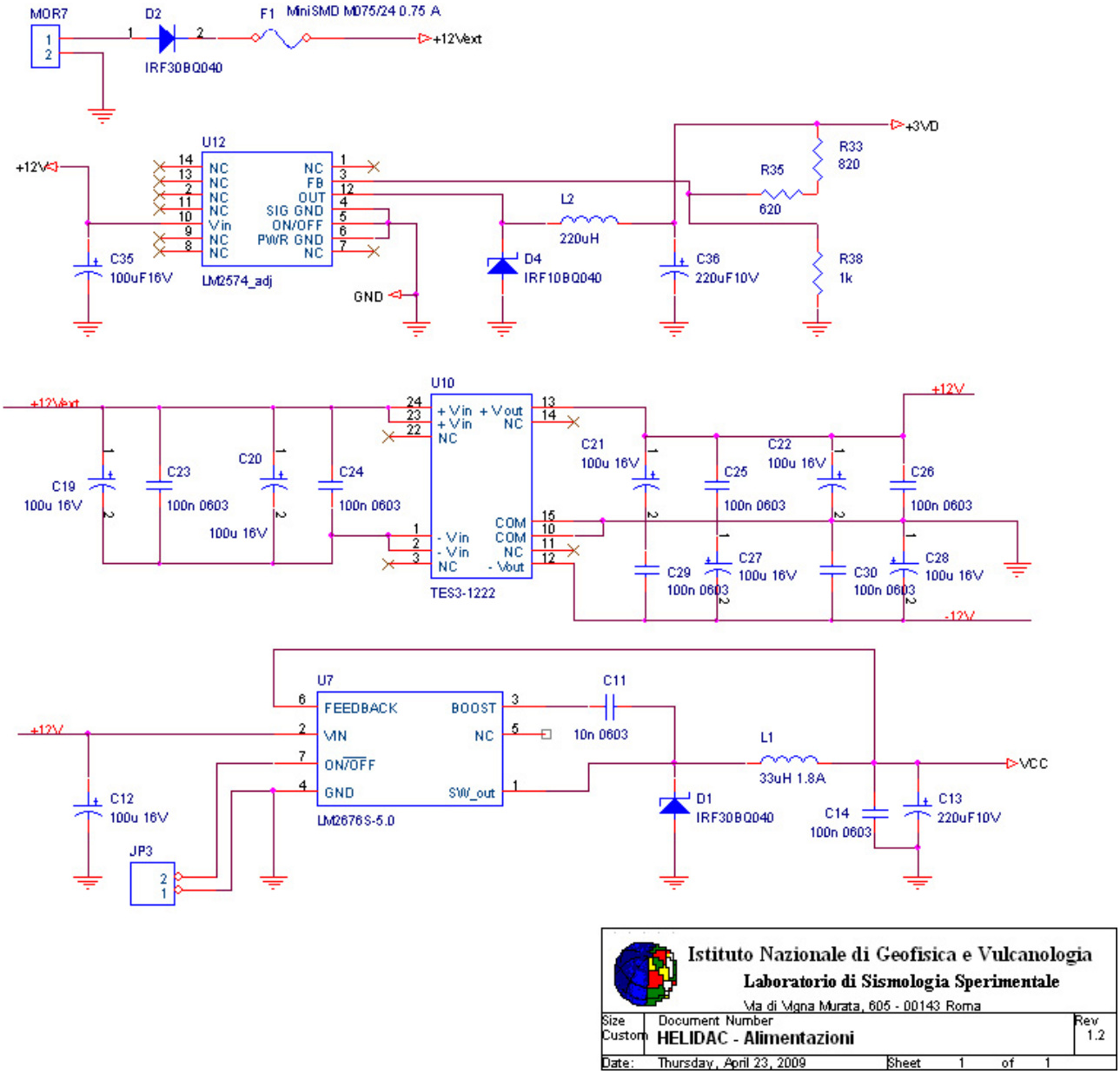


Figura 18. Sezione di alimentazione.

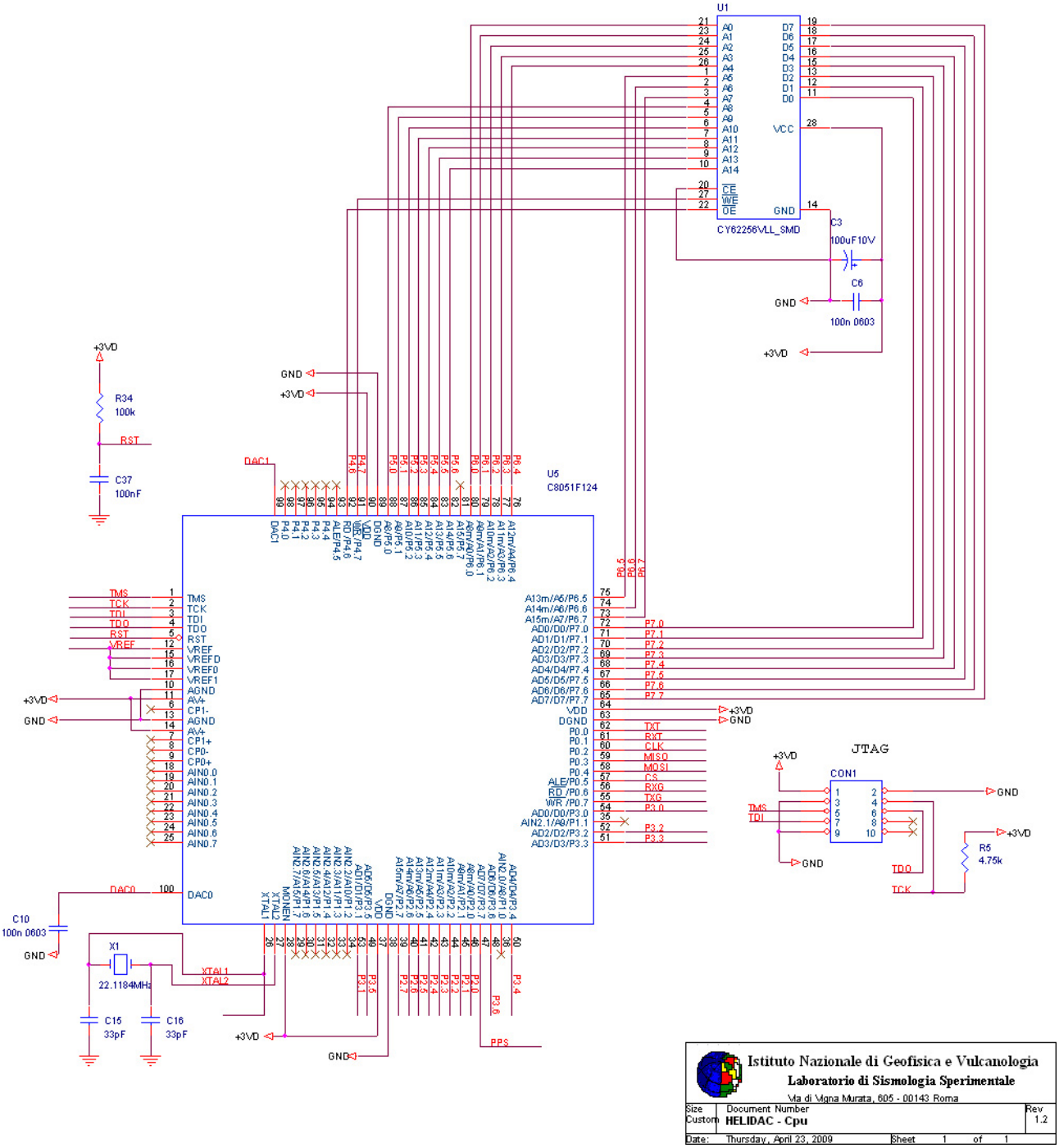
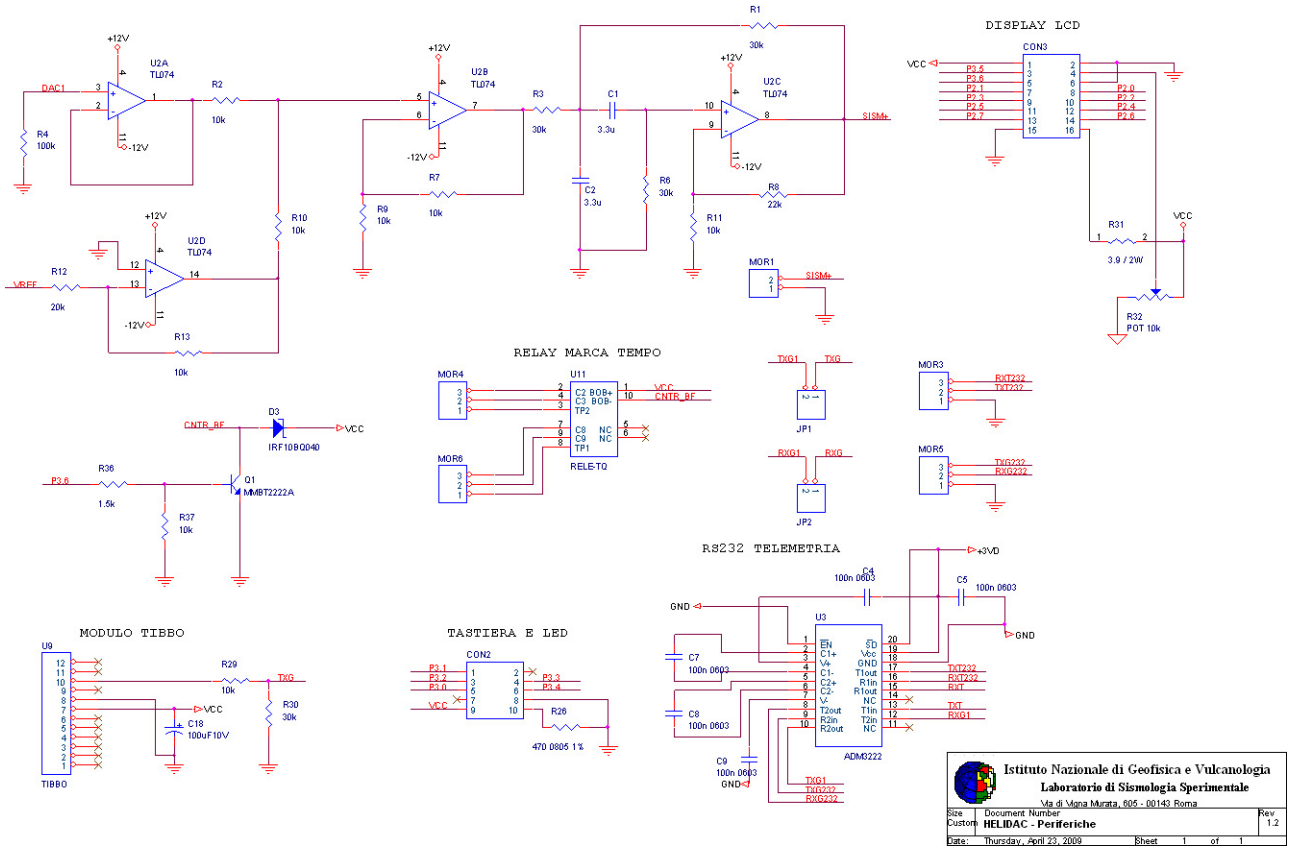


Figura 19. Microprocessore con interfacce.


DISPLAY LCD
CON3
RELAY MARCA TEMPO
MOR4
U11
RELE-TQ
MOR6
JP1
JP2
MOR3
MOR5
RS232 TELEMETRIA
U3
ADM3222
MODULO TIBBO
TIBBO
TASTIERA E LED
CON2
Istituto Nazionale di Geofisica e Vulcanologia
Laboratorio di Sismologia Sperimentale
Via di Vigna Murata, 605 - 00143 Roma
HELIDAC - Periferiche
Thursday, April 23, 2009

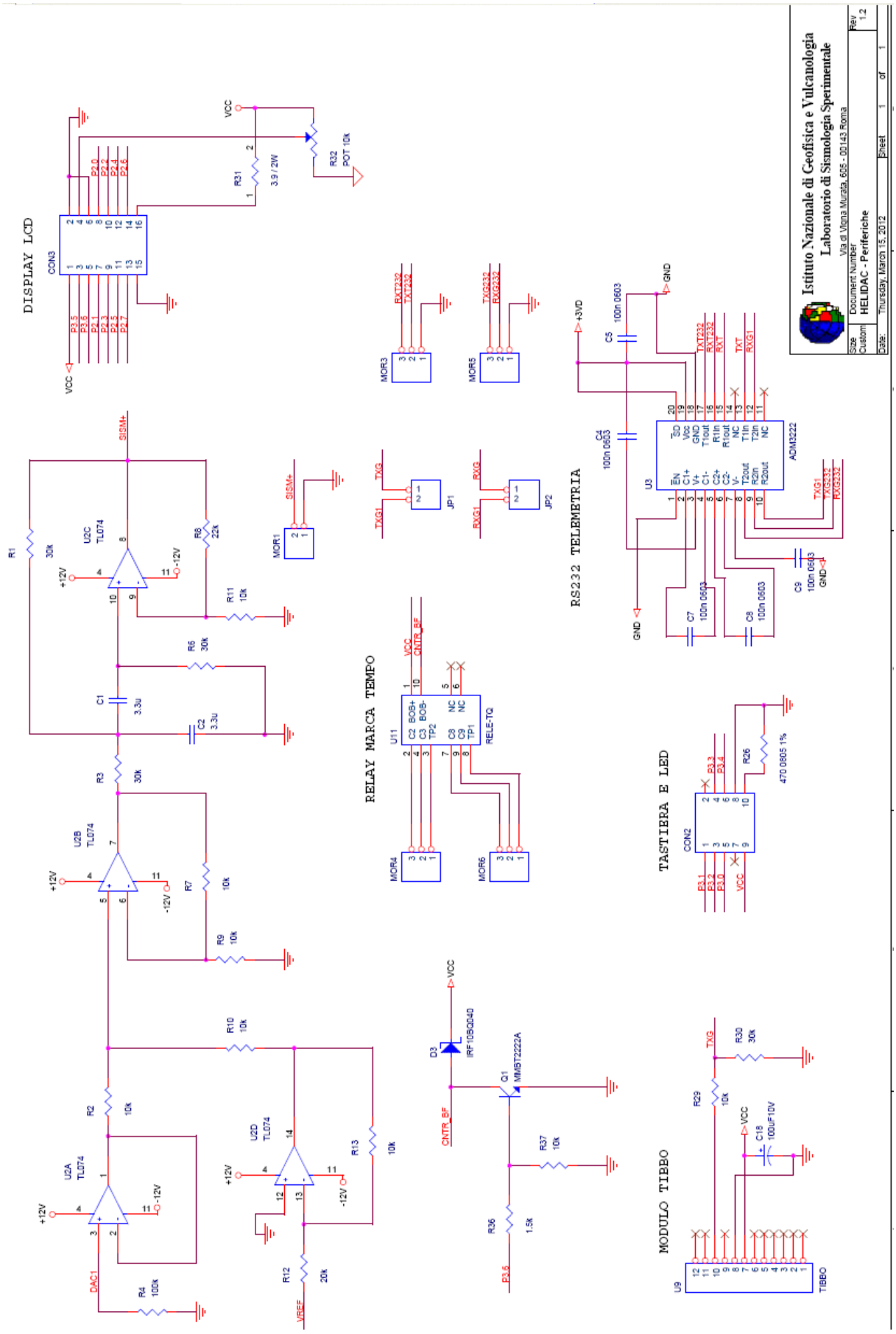


Figura 20. Periferiche.

6. Appendice C: Dispositivo Server Seriale TIBBO

Questi SDS sono caratterizzati da tre diverse modalità operative:

1. Conversione Seriale Ethernet e viceversa *(Normal Mode);*
2. *P*rogrammazione via porta seriale *(Serial Programming Mode);*
3. Upgrade del firmware *(Firmware Download Mode).*

La funzione principale è la prima e permette di convertire e instradare i dati tra la propria porta Ethernet e quella seriale, sfruttando i protocolli UDP/IP o TCP/IP.

Le porte utilizzate per la trasmissione dei comandi sono la 65535 del TCP e la 1001 (utilizzata di default e configurabile dall'utente) per i dati.

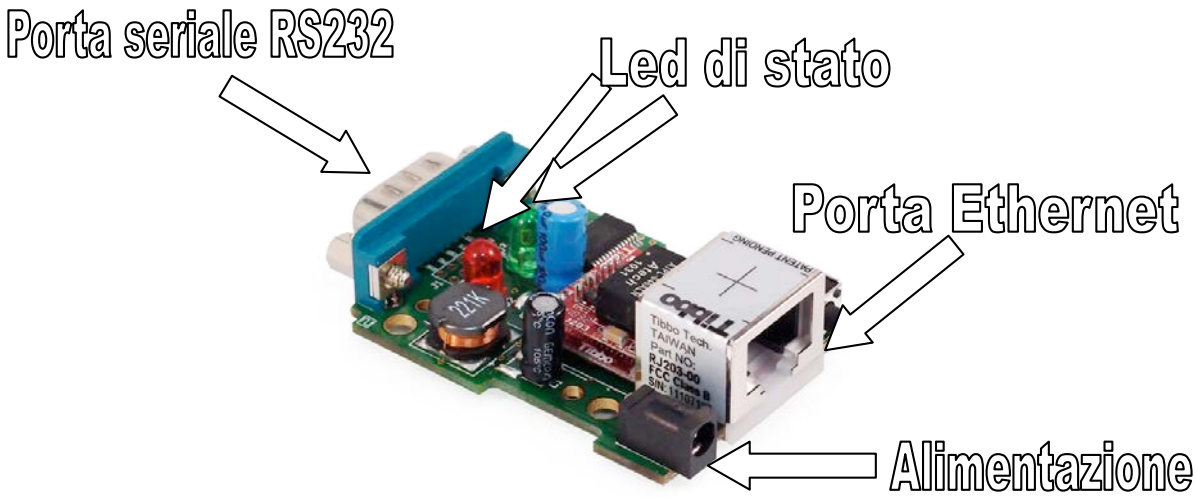


Figura 21. Modulo TIBBO.

Le specifiche tecniche sono riassumibili nella tabella seguente:

- Dimensioni: 30,1 x 20 x 15,5 mm.
- Consumo: 220 mA / 5 V.
- Porta Ethernet 10/100BaseT Auto-MDIX con connettore RJ45 ed Ethernet magnetico.
- Una porta seriale che può funzionare sia in modo full-duplex sia half-duplex (il full-duplex è indicato per funzionare con porte RS232 e RS422, mentre, in half-duplex, la trasmissione dei dati mediante i protocolli RS232/422/485 non è supportata).
- Baud rate: oltre 115200 bps.
- 5 Modi di parità supportati: nessuno, pari, dispari, mark, space.
- Trasmissione dati: 7 e 8 bit.
- La porta seriale include le linee: TX, RX, RTS, CTS, DTR, DSR, ciascuna linea può essere utilizzata per il controllo remoto di I/O.
- 4 LED che indicano lo stato di funzionamento del dispositivo.
- Buffer dati da 8 KB (uno in ogni direzione).
- Controllo diretto del modem ADSL.
- EEprom: memorizza i dati di configurazione utente.
- Protocolli di trasmissione supportati: UDP, TCP, ARP, ICMP (PING), DHCP, PPPoE e LCP.

I driver forniti a corredo permettono il corretto funzionamento in ambiente Windows e alcuni software di gestione consentono la totale programmazione del dispositivo. Svariate sono le applicazioni in cui può essere utilizzato, ma essenzialmente si dividono in tre tipi principali:

1. Collegamento di una periferica seriale senza modifica del software di controllo già esistente. I driver del SDS sono in grado di creare delle porte COM virtuali sul PC (chiamate VSP Virtual Serial Ports). Esse, dal punto di vista del programma e dell'utente, si comportano come porte standard COM hardware, ma in realtà trasformano i dati in pacchetti TCP inviati sulla LAN che vengono poi convertiti dall' SDS in formato seriale.
2. E' possibile creare un nuovo software di gestione, per mezzo del quale è possibile la comunicazione direttamente con la periferica seriale, senza utilizzare le VSP. Il SDS utilizza i protocolli di trasmissione UDP/IP e TCP/IP che sono disponibili come plug-in in diversi linguaggi di programmazione. Come esempio è possibile citare quello di Visual Basic 6 della Microsoft, per il quale la Tibbo mette a disposizione un manuale (scaricabile liberamente da Internet dal sito http://www.tibbo.com) in cui è possibile studiare la comunicazione con il DS100 (altro famoso SDS di casa TIBBO).
3. Utilizzando due SDS, è possibile creare un "Collegamento Seriale Virtuale", collegando due periferiche seriali utilizzanti la connessione Ethernet. Si viene a creare cioè una connessione che, agli occhi delle periferiche, è di tipo seriale, ma che in realtà si basa sulla tecnologia Ethernet. Per spiegare meglio, il collegamento avviene tra un SDS alla porta COM di un PC e il secondo SDS alla porta seriale di una periferica, le due interfacce Ethernet invece si connettono tramite un cavo RJ45 diretto. Il software "vede" così un normale collegamento seriale senza neppure utilizzare le VSP (si accederà direttamente in hardware alle porte COM), mentre in realtà si sfrutta la tecnologia e i protocolli di comunicazione Ethernet. Avendo a disposizione un collegamento WAN (ad esempio ADSL) sarà possibile connettere direttamente la porta Ethernet del SDS, rendendo accessibile la periferica da qualunque PC collegato alla WAN.

Il pacchetto ***Tibbo Device Server Toolkit***, fornito dal produttore, contiene diversi software di gestione e programmazione. Uno di questi è il ***Connection Wizard*** con il quale è possibile rendere immediatamente operativo il dispositivo, impostando: un collegamento virtuale, utilizzare una VSP, scegliere il protocollo di comunicazione (TCP e/o UDP), impostare la porta seriale, impostare il dispositivo come Master o come Slave, specificare l'indirizzo IP, specificare la porta logica sia del dispositivo SDS che di quello remoto.

Con il ***DS Manager*** (fig.22) è possibile modificare manualmente tutte le impostazioni che sono invece semiautomatizzate con il modulo software precedente. Si ha così la possibilità di intervenire in maniera più mirata alla configurazione del dispositivo.

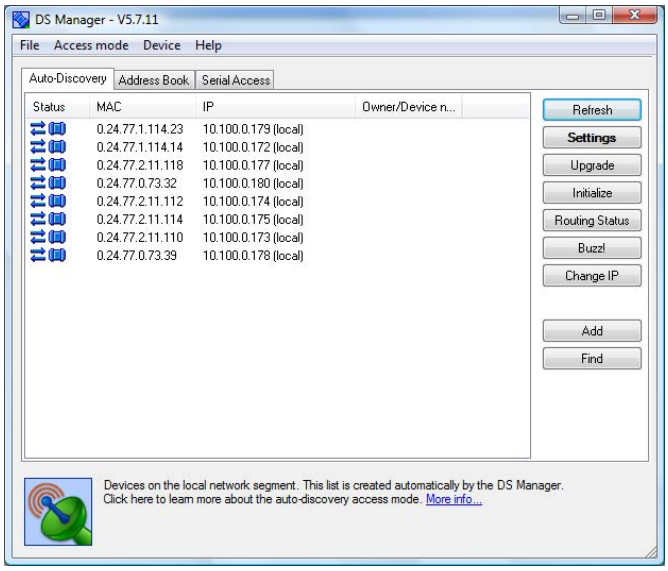


Figura 22. Schermata principale di DS Manager.

In ultimo c'è il modulo software ***VSP Manager*** (fig. 23) con il quale si possono programmare le COM virtuali.

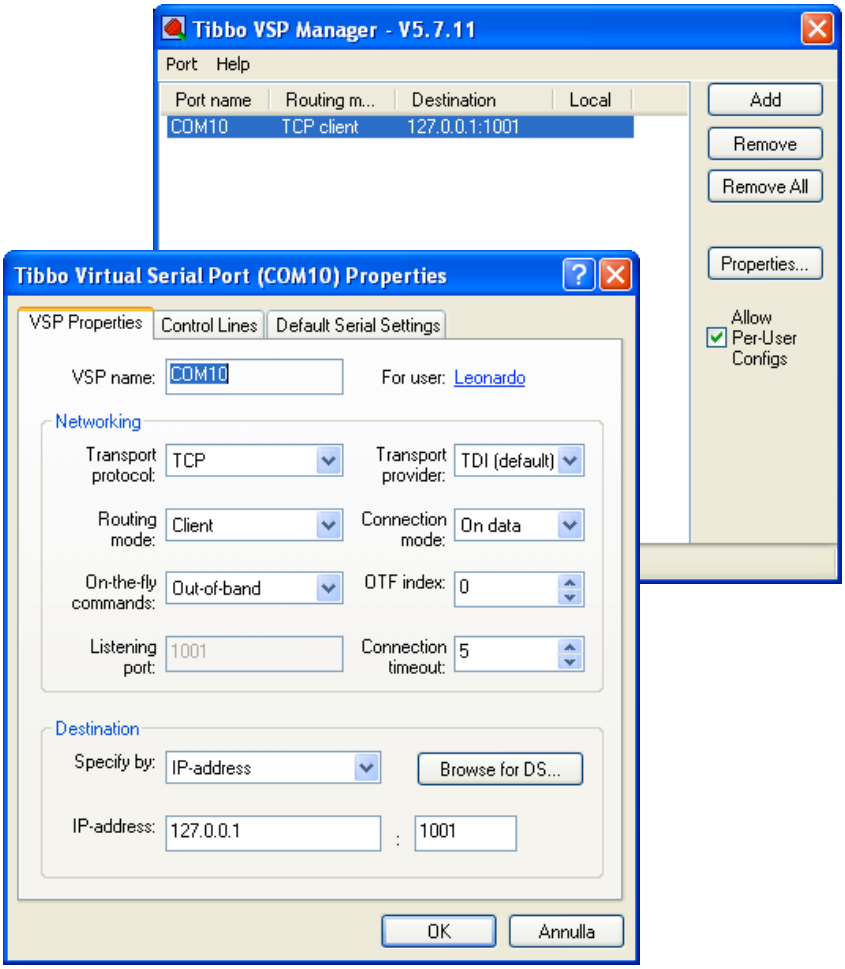


Figura 23. Schermata principale di VSP Manager.

Un'ultima considerazione è relativa al collegamento di un SDS a più nodi, ossia accedere a una singola periferica seriale da più dispositivi remoti (ad esempio PC), oppure accedere e/o ricevere da più dispositivi seriali i dati da una singolo computer. Nel primo caso il SDS deve essere messo in modalità Slave affinché possa rispondere a tutte le richieste dei diversi PC remoti. Al fine di evitare "conflitti di connessione", occorre scegliere il protocollo TCP orientato alla "Connessione".

Nel secondo caso i vari SDS devono essere impostati come Master e avere lo stesso indirizzo IP di destinazione. Anche in questo caso, al fine di evitare problemi se ogni blocco di dati seriali, non viene inviato in un unico pacchetto, bisogna utilizzare il protocollo TCP e mantenere aperti sul PC remoto, diversi socket per ogni sorgente dati.

7. Appendice D: listato in C della funzione main del programma RCS

```
int main(int argc, char *argv[])
{
  const unsigned char hdr_mask[8] = {0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00};
  unsigned char buffer[MAX_DIM], rullo_addr[30];
  int rullo_socket, NuovoSocket, length, ris;
  FILE *config_file_id, *status_file_id, *file_pid;
  unsigned int first = 0;
  fpos_t position;
  int exitCond, socket_chiuso=1;
  int byte_ric, i, j = 0, OK = 0;
  int w,semval=0,semid=0; //variabili per il semaforo per connection
  struct sembuf sops[1];
  int sons_pid=-10, l_wait=40;
  long int *somma_punt;
  long chiave_sem=50;
  long int somma_temp = 0, media = 0, temp = 0;
  unsigned char buffer_int[400];
  char config_file[20]="./rullo.txt", station[30], rullo[30], status_file[20] = "./connection", pid[11] =
"./file_pid", connection[2], rullo_con[30];
  struct sockaddr_in rullo_sock;
  struct sockaddr_in client;
  struct hostent * station_addr;
  fd_set write_fds;
  struct sigaction sa;
 // Attivazione intercettore SIGINT
  signal(SIGINT, catcher_SIGINT);
  //Attivazione intercettore SIGTERM
  signal(SIGTERM, catcher_SIGTERM);

  if( argc >= 2 ) strcpy(config_file,argv[1]);
  printf("\n*********************************************\n");
  printf("\nStarting Server: ");
  fflush(stdout);
  system("date");
  if ((status_file_id = fopen (status_file , "r+" )) == NULL) printf("Errore nell'apertura file di
configurazione\n");
  if ((file_pid = fopen (pid , "a+" )) == NULL) printf("Errore nell'apertura filepid\n");

 //riempio la struttura per i rulli
  rullo_sock.sin_family=AF_INET;
  rullo_sock.sin_port=htons(PORT);
  length = sizeof( client );

  semget(IPC_PRIVATE,1,0666);
  if(semid == -1){
       printf("errore creazione set di semafori\n");
      exit(0);
  }
  else semctl(semid,0,SETVAL,1);
  station_socket = socket_in(PORT);
  if (station_socket == -1){
    printf("Server: Terminato per errore station_socket.\n");
    fflush(stdout);
    kill(getpid(), 15 );
  }

  sa.sa_handler = sigchld_handler; // elimina tutti i processi morti
  sigemptyset(&sa.sa_mask);
  sa.sa_flags = SA_RESTART;
  if (sigaction(SIGCHLD, &sa, NULL) == -1) {
```

```
            perror("sigaction");
            exit(1);
        }

        while (1){
            printf("Server: Attendo connessioni...\n");
            fflush(stdout);
          //Test sul socket: accept blocca l'esecuzione fino all'arrivo di una connessione.
          if ((NuovoSocket=accept(station_socket, (struct sockaddr *)&client, &length))!=-1){
                OK = 0;
                sops[0].sem_num = 0; /* operazione sul semaforo 0 */
                sops[0].sem_flg = 0; /* che fare: wait se semaforo = 0 */

                if ((station_addr = gethostbyaddr((const void *)&(client.sin_addr), sizeof(client.sin_addr),
AF_INET)) == NULL) printf("gethostbyaddr error\n");
                printf("%s accettata: ", station_addr->h_name);
                fflush(stdout);
                system("date");
                 if ((config_file_id = fopen (config_file , "r" )) == NULL) printf("Errore nell'apertura file di
configurazione\n");
                rewind(config_file_id);
                while(!feof(config_file_id)) /*scorro tutto il file per vedere se la stazione e' gia stata associata
ad un rullo */
                {
                     fscanf(config_file_id,"%s %s", station, rullo);
                      if(!strcmp((station_addr->h_name), station)){
                            strcpy(rullo_addr, rullo);
                             OK = 1;
                       }
                 } //printf("dopo while\n");
                  fclose(config_file_id);
        //Il socket deve essere non bloccante per usare la select
                  fcntl(NuovoSocket,F_SETFL,O_NONBLOCK);

                  if ((sons_pid = fork())== 0){
                       if(fprintf(file_pid, "%s %i\n",  station_addr->h_name, (int)getpid() ) ==-1)  printf("errore
scrittura pid\n"); //salvataggio pid
                        fflush(file_pid);
                        close(station_socket); //il figlio non ne ha bisogno
                        if (!OK){
                           printf("%s non configurata\n\n", station_addr->h_name);
                           fflush(stdout);
                           while(1){
                                // leggo comunque i dati dalla stazione per non far fare continue aperture di socket
                                  socket_chiuso = sync_finder(NuovoSocket, station_addr->h_name);
                                  if  (  (read_test_select(NuovoSocket,  40,  station_addr->h_name)  ==  1)  &&
(socket_chiuso != 0)){
                                    if  ((byte_ric=read(NuovoSocket,  (buffer  +  WORD_SYNC),  MAX_DIM  -
WORD_SYNC))<=0 ){
                                       ChiudiSocket(NuovoSocket);
                                         printf("%s connection down stazione non configurata read ", station_addr-
>h_name);
                                         fflush(stdout);
                                         system("date");
                                         exit(0);
                                      }
                                      else memset(buffer, 0, MAX_DIM);
                                  }
                                  else {
                                       ChiudiSocket(NuovoSocket);
                                       printf("%s  connection  down  stazione  non  configurata  test  select  ",
station_addr->h_name);
```

```
                                fflush(stdout);
                                system("date");
                                exit(0);
                        }
                    }
                }

                //apro e connetto socket con rullo non bloccante
                if ((rullo_socket = socket_out(rullo_addr, PORT)) == 0 ){
                    printf("rullo_socket errore\n");
                    fflush(stdout);
                    ChiudiSocket(rullo_socket);
                    socket_chiuso = 0;
                }
                else{
                    sops[0].sem_op = -1; /* decrementa il semaforo di 1 */
                    semop(semid,sops,1);
                    rewind(status_file_id);
                    while(!feof(status_file_id)) /*scorro tutto il file */
                    {
                        if ((fgetpos(status_file_id, &position)) == -1) printf("errore scrittura status\n");
                        fscanf(status_file_id, "%s %s\n", rullo_con, connection);
                        if(!strcmp(rullo_con, rullo_addr)){
                            printf("%s connessa a rullo %s \n\n", station_addr->h_name, rullo_addr);
                            fflush(stdout);
                            fsetpos(status_file_id, &position);
                            if(fprintf(status_file_id, "%s %s", rullo_con, "OK" ) ==-1)    printf("errore
scrittura status\n"); //connection up
                            fgetc(status_file_id);
                            if((fgetc(status_file_id)) == EOF) break;
                            fflush(status_file_id);
                            break;
                        }
                    }
                    sops[0].sem_op = 1; /* tipo di operazione: incrementa il semaforo libero il file*/
                    semop(semid,sops,1);
                }
                while(socket_chiuso){
                    memset(buffer_int, 0 , 400);
                    memset(buffer, 0, MAX_DIM);
                    socket_chiuso = sync_finder(NuovoSocket, station_addr->h_name);
            //Lettura dei dati dal socket (messaggio ricevuto)
                    if (( read_test_select(NuovoSocket, l_wait, station_addr->h_name ) == 1) &&
(socket_chiuso != 0)) //problemi con l'apertura del socket con la stazione
                    {
                      if ((byte_ric=read_new(NuovoSocket, (buffer + WORD_SYNC), MAX_DIM -
WORD_SYNC, station_addr->h_name))<=0) //problemi con la ricezione dati dalla stazione
                    {
                        printf("No dati\n");
                        fflush(stdout);
                        ChiudiSocket(NuovoSocket);
                        socket_chiuso = 0;
                        sops[0].sem_op = -1; /* decrementa il semaforo di 1, aspetto se il file non e'
disponibile */
                        semop(semid,sops,1);
                    }
                    else if ((memcmp(buffer + MAX_DIM - 3, "EOB", 3) == 0) && (memcmp(buffer + 30,
"HZ", 2) == 0)){
            //Elaborazione dati ricevuti: rimozione offset e send passo da tre a quattro byte per rimozione offset
                        j = 0;
                        for(i = 64; i < MAX_DIM - 35; i+=3){
                            memcpy(buffer_int + j, buffer + i, 3);
```

```
                                if ((buffer[i+2] & 128) == (unsigned char) 0x80 ) buffer_int[j+3] = (unsigned
char) 0xFF; //estendo il segno
                                else buffer_int[j+3] = (unsigned char) 0x00;
                                j+=4;
                            }
                            for(i = 0; i < 400; i+=4){  //rimozione offset
                                somma_temp += *((long int *) &buffer_int[i]);
                            }
                            if (memcmp(buffer + 16, "0", 1) == 0)   { //al cambio del minuto calcolo la media
sugli ultimi 6000 campioni
                              if(first == 1){
                                  media = somma_temp / 6000; //la prima volta la assegno comunque
                                  first = 0;
                              }
                              if((media<0) && ((somma_temp / 6000) < 0)){
                                   if ( abs(media * 100) > abs(somma_temp / 6000)) media =
somma_temp/6000;
                              }
                              else if((media>0) && ((somma_temp / 6000) < 0)){
                                if ((media * 100) > abs(somma_temp / 6000)) media = somma_temp / 6000;
                                }
                                else if((media>0) && ((somma_temp / 6000) > 0)){
                                    if ((media * 100) > (somma_temp / 6000)) media = somma_temp / 6000;
                                }
                                else if((media<0) && ((somma_temp / 6000) > 0)){
                                  if (abs(media * 100) > (somma_temp / 6000)) media = somma_temp / 6000;
                                  }
                                 else first = 1;
                                 somma_temp = 0;
                            }
                            for(i = 0; i < 400; i+=4){
                                somma_punt = (long int *) &(buffer_int[i]);
                                temp =  (*(long int *) &(buffer_int[i])) - media;
                                *somma_punt = temp;
                            }
                            j = 0; //ritorno ai tre byte
                            for(i = 64; i < 364; i+=3){
                                memcpy(buffer + i, buffer_int + j, 3);
                                j+=4;
                            }
                            memcpy(buffer, hdr_mask, 8);    //ricreo la parola di sincronismo
                             //rullo_socket e' non bloccante
                            if ((ris = write(rullo_socket, buffer, MAX_DIM))<=0)  //problemi con la connessione
col rullo digitale
                            {
                                printf("Impossibile mandare il messaggio verso %s. exit, ris %d ", rullo_addr,
ris);
                                fflush(stdout);
                                system("date");
                                socket_chiuso = 0;
                                sops[0].sem_op = -1; /* decrementa il semaforo di 1, aspetto se il file non e'
disponibile */
                                semop(semid,sops,1);
                                ChiudiSocket(rullo_socket);
                            }
                        }
                    }
                    else{
                      printf("source socket chiuso ");
                      fflush(stdout);
                      socket_chiuso = 0;
```

```
                    sops[0].sem_op = -1; /* decrementa il semaforo di 1, aspetto se il file non e' disponibile
*/
                    semop(semid,sops,1);
                  }
                }//while(socket_chiuso)

              //aggiornamento del file connection di status alla disconnessione
                fsetpos(status_file_id, &position);
                fprintf(status_file_id, "%s %s", rullo_con, "KO" ); //connection down
                fflush(status_file_id);
                printf("%s connection down %s, ", station_addr->h_name, rullo_con);
                fflush(stdout);
                system("date");
                printf("\n");
                fflush(stdout);
                sops[0].sem_op = 1; /* tipo di operazione: incrementa il semaforo e sblocco il file*/
                semop(semid,sops,1);
                ChiudiSocket(NuovoSocket);
                ChiudiSocket(rullo_socket);
                sleep(2);
                exit(0);
              }
               else   sleep(1);
            }//accept
          }
            ChiudiSocket(station_socket); //Chiusura del socket
          printf("Server: Terminato.\n");
          fflush(stdout);
          kill(getpid(), 15 );
        }
```

8. Appendice E: listato script PHP

```
<?php
function check_admin(){
	$admin_servers=array(
		"10.132.4.14", //golden-eagle.int.ingv.it
		"10.152.0.5", //golden-eaglewifi
		"10.100.60.78", //ss2turnista.int.ingv.it
		"10.132.4.20", //fermi.int.ingv.it
		"10.132.4.24", //anakin3.int.ingv.it
		"10.132.4.27", //linus.int.ingv.it
		);
	$elems=count($admin_servers);
	for($i=0; $i<$elems; $i++)
		if($_SERVER["REMOTE_ADDR"]==$admin_servers[$i])
			return true;
	return false;
}
$use_utf = true;
if ($use_utf)
  header('Content-type: text/html; charset=UTF-8');
print "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 3.2//EN\"><html><head><title>Digital Helycorder
Web Interface</title><!-- <meta http-equiv=\"Content-Type\" content=\"text/html;charset=iso-8859-15\"> --
></head><body text=\"white\" bgcolor=\"#01004E\" link=\"blue\" alink=\"yellow\" vlink=\"blue\">
<center><img src=\"./titolo.jpg\" border=\"0\" alt=\"[Istituto Nazionale di Geofisica e Vulcanologia
logo]\"></center><br><font size=+2 ><center><a
href=./Interfaccia%20web%20per%20i%20rulli%20digitali.pdf color=\"#FF0000\" title='click to download
manual' > Digital Helycorder Web Interface </a></font></center><table  border=\"3\" cellspacing=\"3\"
cellpadding=\"3\" align=\"center\" >";
if ($use_utf)
  print "\n";
$charset = $_POST['charset'];
$new_station = $_POST['new_station'];
$command=$_POST['command'];
$submit=$_POST['submit'];
$new_stat=$_POST['new_stat']; //new_stat
$motivo=$_POST['motivo'];
$restart_stat= trim(strtolower(substr($submit, strpos($submit, ' '))));
$submit= substr($submit, 0, strpos($submit, ' '));
$restart_stat=substr($restart_stat, strpos($restart_stat, 'of '));
$restart_stat=trim(substr($restart_stat, strpos($restart_stat, ' ')));

if(($submit =="Aggiungi") && ($new_station!="Sigla Stazione")){
   if ($name == "rulli2010")   {
     $stat_file=fopen("./station", "a+");
     fwrite($stat_file, $new_station."\n");
     fclose($stat_file);
     print "<center>stazione $new_station aggiunta </center>";
   }
   if (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] == 'on')
    $result = 'https://';
  else
    $result = 'http://';
  $result .= $_SERVER['HTTP_HOST'] .$_SERVER['PHP_SELF'];
  print "<br><hr><a href='$result'>Back</a>\n";
}
elseif($submit == "restart"){
	$connect = file('./rullo.txt');
	$lines = file('./file_pid');
	$file_OK = file('./connection');
	$nuovo_conn = fopen('./connection', w);
```

```
        sleep(1);
        foreach ($lines as $line_num => $line){
                $line2=substr($line, 0, strpos($line, ' '));
                $line2 = substr($line2, 0, strpos($line, '.'));
                if ($restart_stat == $line2)
                {
                        $pid = substr($line, strpos($line, ' '));
                }
        }
        $i=0;
        foreach ($connect as $conn_num => $conn_line) //aggiornamento file connect
        {
                $conn_riga = substr($conn_line, 0, strpos($line, ' '));
                $conn_riga = substr($conn_riga, 0, strpos($conn_riga, '.'));
                $conn_riga2 = substr($conn_line, strpos($line, ' '));
                if ($conn_riga == $restart_stat) {
                        $file_OK[$i] = "rullodigit".($i + 1).".int.ingv.it KO\n";
                }
                fwrite($nuovo_conn, $file_OK[$i]);
                $i = $i + 1;
        }
        fclose($nuovo_conn);
        print " <br> <font> <center> restart process of station $restart_stat <br> <br> <br> <H2>attendere
qualche minuto per la riconnessione automatica </center> </font>";
        system('sudo -u root /bin/kill -9 '.$pid.' >/dev/null', $retval);
        system('echo "restart process of '.$restart_stat.' da '.gethostbyaddr($_SERVER['REMOTE_ADDR']).'"
>> /var/log/server_restart.log', $retval);
        system('date >> /var/log/server_restart.log', $retval);
        print "<tr> <td><a href='$result'>Back</a> </tr> </td>";
        print "<meta http-equiv='refresh' content=5 url='$result'>";
}
elseif(($submit =="Restart") && ($motivo !=" ")){
        print "  <form name='test3' id='test3' method='post'";
        if ($use_utf)
        print " accept-charset='UTF-8'>";
        $motivo="\"$motivo\"";
        system('sudo -u root ./script_restart '.gethostbyaddr($_SERVER['REMOTE_ADDR']).' '.$motivo.'>
/dev/null ', $retval);
        if ($retval == 0 ) print "<br><br><center> <H2>restart effettuato </center> </H2>";
        $result = 'http://';
        $result .= $_SERVER['HTTP_HOST'] .$_SERVER['PHP_SELF'];
        print "<tr> <td><a href='$result'>Back</a> </tr> </td>";
        print "<meta http-equiv='refresh' content=10 url='$result'>";
}
elseif($submit =="Change"){
   $lines = file('./rullo.txt');
   $stat = file('./station');
   print "  <form name='test2' id='test2' method='post'";
   if ($use_utf)
    print " accept-charset='UTF-8'>";
print " <br> <font> <center> Selezione associazione stazione - rullo </center> </font>
        <br> <font> <center> fare attenzione a non selezionare la stessa stazione </center></font>
        <br>";
  print "<tr>   <td bgcolor=\"#FF0000\"> Rullo</td> <td bgcolor=\"#33CC00\"> Stazione </td> <td
bgcolor=\"#01009E\"> Nuova Stazione </td>";
   for ($i=1; $i < 9; $i++)   {
        foreach ($lines as $line_num => $line)  {
          $riga_stat = explode (' ', strtoupper ($line));
          $riga_stat[1] = trim($riga_stat[1]);
          if ($riga_stat[1] == "RULLODIGIT".$i.".INT.INGV.IT")   {
            $rullo_ok  = $riga_stat[0];
            break;
```

```
         }
         else $rullo_ok = "";
        }
         print "<tr>   <td bgcolor=\"#FF0000\"><b> RULLODIGIT$i </b></td><td bgcolor=\"#33CC00\">
<b>$rullo_ok </b></td> <td bgcolor=\"#01009E\"> <select name='new_stat[$i]'><option
value='$rullo_ok'>$rullo_ok </option>";
         foreach($stat as $stat_line_num => $stat_line) {
              $stazione = trim(strtoupper($stat_line));
              if ($stazione != $rullo_ok) {
                    print " <option value='$stazione'> $stazione </option>";
              }
         }
         print "</select> </td></tr>";
   }
       print "<tr><td><input type='submit' name='submit' value='Apply '></td></tr>";
       if (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] == 'on')
        $result = 'https://';
       else
       $result = 'http://';
       $result .= $_SERVER['HTTP_HOST'] .$_SERVER['PHP_SELF'];
       print "<tr> <td><a href='$result'>Back</a>\n </tr> </td>";
}
elseif($submit =="Apply"){
    $lines = file('./file_pid');
    $kill_lines=file('./rullo.txt');
    print "  <form name='test3' id='test3' method='post'";
    if ($use_utf)
    print " accept-charset='UTF-8'>";
     for ($i = 1; $i < 9; $i++) {
          for($j = $i + 1; $j < 9 - 1; $j++)
          {
               if (($new_stat[$i] == $new_stat[$j]) && ($new_stat[$i] != ""))
               {
                 print "<br> <font> <center> errore nell'assegnazione della stazione $new_stat[$j]</center>
</font> <br>";
                 $i = 9;
                 break;
               }
          }
       }
       if ($i != 10) {
         for($mul = 1; $mul < 9; $mul++){
                    if($new_stat[$mul] !="NULL") {
                          foreach ($kill_lines as $kill_line_num => $kill_line) {
                                $kill_line2=trim(strtoupper(substr($kill_line, 0, strpos($kill_line, ' '))));
                                if ($new_stat[$mul] == $kill_line2) {
                                      $pid=0;
                                      break;
                                }
                                else {
                                      foreach ($lines as $line_num => $line) {
                                            $line2=trim(strtoupper(substr($line, 0, strpos($line, ' '))));
                                            if ($new_stat[$mul]== $line2) {
                                                  $pid = substr($line, strpos($line, ' '));
                                            }
                                      }
                                }
                          }
                          if($pid != 0) system('sudo -u root /bin/kill -9'.$pid.' >/dev/null', $retval);
                     }
         }
         $rullo_file=fopen("./rullo.txt", "w");
```

```
        for($mul = 1; $mul < 9; $mul++){
                if($new_stat[$mul] !="NULL") fwrite($rullo_file, strtolower($new_stat[$mul])." " . "rullodigit" .
$mul . ".int.ingv.it \n");
                else fwrite($rullo_file, " rullodigit".$mul.".int.ingv.it \n");
        }
        fclose($rullo_file);
      system('sudo -u root ./script_restart '.gethostbyaddr($_SERVER['REMOTE_ADDR']).' change_station >
/dev/null ', $retval);
      if ($retval == 0 ) print "<br><br><center> <H2>modifica e restart effettuati </center> </H2>";
      }
      else print "<br> <font> <center> riavvio non effettuato</center> </font> <br>";
      $result = 'http://';
      $result = $_SERVER['PHP_SELF'];
      print "<tr> <td><a href='$result'>Back</a>\n </tr> </td>";
      print "<meta http-equiv='refresh' content=10 url='$result'>";
}
elseif($submit =="LOG"){
   print "  <form name='test3' id='test3' method='post'";
   if ($use_utf)
   print " accept-charset='UTF-8'>";
   print "<pre>";
   system('tail -200 /var/log/server.log', $retval);
   print "</pre>";
   $result = 'http://';
   $result = $_SERVER['PHP_SELF'];
print "<meta http-equiv='refresh' content=60 >";
   print "<tr> <td><a href='$result'>Back</a> </tr> </td>";
}
else{
  $lines = file('./rullo.txt');
  $connect = file('./connection');
  print "  <form name='test' id='test' method='post'";
  if ($use_utf)
 print " accept-charset='UTF-8'>";
print "<br> <font> <center> <A HREF=index.php title='click to refresh'>Situazione attuale</A>  </font>";
print " <br><br>NON CHIUDERE QUESTA PAGINA</center><br><tr> <td><center>ping rullo</td>
      <td bgcolor=\"#FF0000\"><center> Rullo </td> <td bgcolor=\"#FFAA00\"><center> Station </td>
      <td bgcolor=\"#33CC00\" width='15%'> <center>Status </td> <td><center>ping station</td>
<td><center>restart process</td></tr>";
  $ko=0; //variabile per verifica numero ko per restart automatico
  foreach ($lines as $line_num => $line)
  {
      $riga = explode (' ', strtoupper ($line));
      $punto = strpos($riga[1], '.');
      $punto2 = strpos($riga[0], '.');
      $riga[1] = substr($riga[1], 0, $punto);
      $STA = substr($riga[0], 0, $punto2);
      foreach ($connect as $conn_num => $conn_line) {
            $conn_riga = explode (' ', strtoupper ($conn_line));
            $conn_punto = strpos($conn_riga[0], '.');
            $conn_riga[0] = substr($conn_riga[0], 0, $conn_punto);
            if ($conn_riga[0] == $riga[1]) $status = trim($conn_riga[1]);
      }
      $cmd = "ping -c 1 -i 3 -q $riga[1] | grep % | awk 'BEGIN{FS=\",\"}{print $3}' |awk 'BEGIN{FS=\" \"}{print
$1}'";
      $result = exec($cmd, $retval);
      if ($result == "0%") {
            $result = "ping OK";
            $color="#33CC00";
      }
      else {
            $result = "rullo unreachable";
```

```
            $color="#FF0000";
        }
        print "<tr><td bgcolor=\"$color\"><center> <b> $result </center></td ></b>";
     if ($status == "OK") $color="#33CC00";
     elseif($status == "KO") {
        $color="#FF0000";
        $ko++;
     }
     else  //caso in cui c'e' un errore nel file connection    {
        system('sudo -u root ./script_restart nara file_connection_error > /dev/null ', $retval);
        print "<center> <H2>Attenzione, restart automatico in esecuzione causa errore, attendere... </center>
</H2>";
     }
href=\"http://webapp.int.ingv.it/sit/index.php?page=find&find=$STA&ok=Cerca\" target= \"_blank\" title='link to
SIT'>$STA </center></a></b></td >
print "<td bgcolor=\"#FF0000\"> <b> $riga[1] </b></td ><td bgcolor=\"#FFAA00\" width='15%'> <b><center>
<a  href=\"http://hgp2.int.ingv.it/~ads/seisnet_interface/index.php?-table=station&-
action=browse&id_inter==$STA\" target= \"_blank\" title='link to SIT'>$STA </center></a></b></td >
        <td bgcolor=\"$color\"><center> <b> $status </b> </center></td >";
          $cmd = "ping -c 1 -q -i 3 $STA | grep % | awk 'BEGIN{FS=\",\"}{print $3}' |awk 'BEGIN{FS=\" \"}{print
$1}'";
        $result = exec($cmd, $retval);
        if ($result == "0%")     {
              $result = "ping OK";
              $color="#33CC00";
        }
        else  {
            $result = exec($cmd, $retval);
            if ($result != "0%"){
              $result = "station unreachable";
              $color="#FF0000";
            }
        }
        print "<td bgcolor=\"$color\"><center> <b> $result </center></td ></b>";
        if (check_admin()==true ) {
              print "<td><center> <input type='submit' name='submit' value='restart process of $STA'
style='height: 30px; width: 160px'> </center></td> </tr>";
        }
  }
  if($ko > 6) //errore nel restart dovuto a problemi di rete
  {
        system('sudo -u root ./script_restart nara restart_error > /dev/null ', $retval);
        print "<center> <h2>attenzione, restart automatico in esecuzione causa errore, attendere... </center>
</h2>";
  }
   if (check_admin()==true ) {
        print "</table> <table  border=\"0\" cellspacing=\"0\" cellpadding=\"0\" align=\"center\" ><tr><br>
          <td width='100px'><center><input type='submit' name='submit' value='Change station' style='height:
30px; width: 110px'> </center></td>
        <td width='200px'><center><input type='submit' name='submit' value='Restart software' style='height:
30px; width: 110px' onclick='messaggio()'> </center></td>
        <td width='30px'><center><input type='submit' name='submit' value='LOG '  style='height: 30px; width:
110px'> </center></td></tr><tr><td></td><td><center> <br><input id='motivo' type='hidden' name='motivo'
length='100' > </center></td>
        </tr> </table>";
   }
print "<meta http-equiv='refresh' content=60 url='$result'>";
}
print" <script>
function messaggio(){
    var messaggio =\"\";
    messaggio= prompt (\"Inserisci il motivo del restart:\");
```

```
    if (messaggio) {
    document.getElementById('motivo').value = messaggio;
  }
  else {
      document.getElementById('motivo').value = ' ';;
  }
}
</script>";
print " <br> <table align=\"center\">
       <tr><td colspan=\"2\" align=\"center\"> Info Development </td></tr>
        <tr><td colspan=\"2\" align=\"center\" valign=\"top\" > Program and Web Interface </td>
        </tr><tr><td  align=\"center\" valign=\"top\" > <a
href=\"http://webcnt.rm.ingv.it/index.php?area=24&pagina=3&modules=3&operazione=Laboratorio%20di%2
0Sismologia&li=it\" title='go to C.N.T. LabSis website'> C.N.T. LabSis </a> </td>
       <td  align=\"center\" valign=\"top\"> <a
href=\"mailto:leonardo.salvaterra@ingv.it?subject=Rulli%20Interface\" title='send a mail'> Leonardo
Salvaterra </a> </td></tr>";
print "</table></form> </body></html>";
?>
```

Bibliografia

Acerra, C., Rao, S., Salvaterra, C. , (2010). Helicorder digitale - Rapporti Tecnici INGV, n° 145

Datasheet microprocessore Cygnal C8051F124 (2002)

Macro Assembler and Utilities for 8051 and Variants User guide (2000)

Pintore, S., Salvaterra, L., (2007). Il Progetto TN-1 - Rapporti Tecnici INGV, n° 40

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

Cecchin, W., Guida HTML WEB: http://xhtml.html.it/guide/leggi/51/guida-html/

Schildt, H., (1998). La guida completa C++. MC Graw Hill

Cooper, M., (2006). Guida avanzata di scripting Bash

Sitografia

[1] http://www.alphageofisica.com.br/geotech/geo_instrument/ds-rv301b_br.pdf

[2] http://www.silabs.com/products/mcu/mixed-signalmcu/Pages/C8051F12x3x.aspx

[3] http://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/index.html

[4] http://docs.tibbo.com/soism/index.html?em203.htm (2008)

[5]http://www.nanometrics.ca/products

[6] http://www.php.net/

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