ISSN 2039-7941
Anno 2018_Numero 403

rapporti tecnici INGV

Evoluzione ventennale (1998-2018) del sistema di acquisizione (Mag-Net) dei segnali dalla rete magnetica dell'Etna e dell'isola di Stromboli

403

Istituto Nazionale di Geofisica e Vulcanologia

REGISTRAZIONE AL TRIBUNALE DI ROMA N.173 | 2014, 23 LUGLIO

Rappresentante legale: Carlo DOGLIONI
Sede: Via di Vigna Murata, 605 | Roma

ISSN 2039-7941
Anno 2018_Numero 403

rapporti tecnici

INGV

EVOLUZIONE VENTENNALE (1998-2018) DEL SISTEMA DI ACQUISIZIONE (MAG-NET) DEI SEGNALI DALLA RETE MAGNETICA DELL'ETNA E DELL'ISOLA DI STROMBOLI

Antonino Sicali, Alfio Amantia, Pasqualino Cappuccio

INGV (Istituto Nazionale di Geofisica e Vulcanologia, Sezione di Catania - Osservatorio Etneo)

403

Come citare: Sicali A., Amantia A., Cappuccio P., (2018). Evoluzione ventennale (1998-2018) del sistema di acquisizione (Mag-Net) dei segnali dalla rete magnetica dell'Etna e dell'isola di Stromboli. Rapp. Tec. INGV, 403: 1-38.

Indice

Introduzione

Nel 1997 esisteva già la rete magnetica del M.te Etna. Con il sistema *Mag-Net* [Del Negro et al, 1998, 1999, 2002] si è cercato di rinnovarla risolvendo i problemi fino ad allora aperti. I problemi maggiori riguardavano soprattutto la gestione dell'alimentazione e la continuità del dato. In commercio esistevano già sistemi pronti all'uso che richiedevano solo qualche adattamento però ritrovandosi con un sistema chiuso (*black box*) per cui l'evoluzione è pressoché impossibile [Sicali et al, 2016]. Si è scelto in alternativa di realizzare su un sistema completamente aperto, *Mag-Net*, assecondando una visione futuristica e di lungo periodo. Dalla progettazione primordiale si è passati alla sua realizzazione e successivamente a uno studio sistematico durato 20 anni, durante il quale si è cercato di valutare il sistema quando più oggettivamente è possibile. Negli anni si è arrivati a definire i tre concetti di *stabilità, efficienza e automatismo* [Sicali et al, 2016] che hanno permesso di far evolvere negli anni *Mag-Net* (figura 1) finché si è potuto ed arrivare a definire un nuovo sistema di acquisizione *IDRA* (Interactive Device for Remote Acquision) (figura 2) che risolvesse totalmente i problemi.

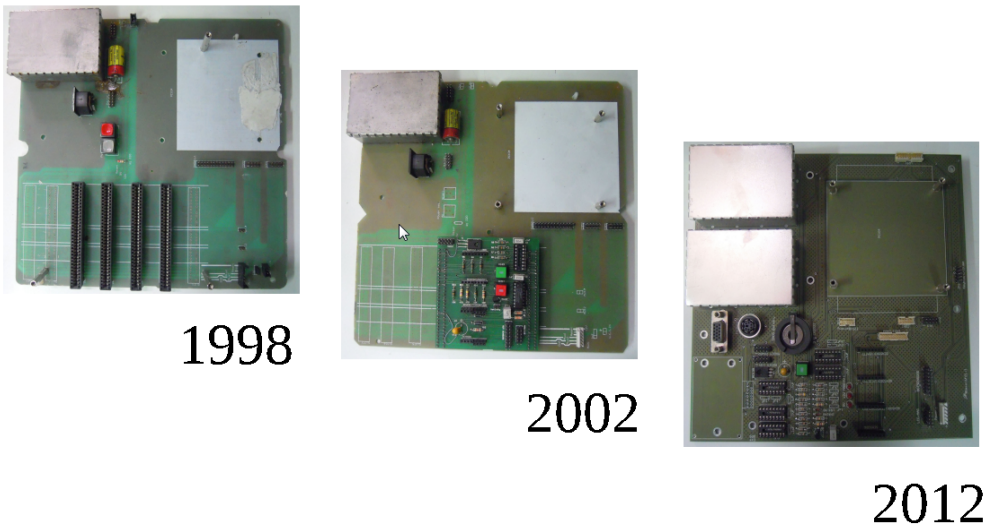

Figura 1. Evoluzione della *mainboard* del sistema *Mag-Net.*

1. I sistemi di acquisizione precedenti a *Mag-Net*

Prima del 1998 le stazioni magnetiche erano costruite per trasmettere in continuo, al centro di raccolta dati situato nella sede di Catania, il dato acquisito (sistema *Ulisse*). Una tale organizzazione aveva sicuramente una grande valenza per la sorveglianza ma conteneva nel suo modello alcuni limiti [Sicali et al, 2016]. Le stazioni erano dotate di un unico canale di trasmissione unidirezionale che non permetteva di interagire con la strumentazione in caso di malfunzionamenti: ogni problema doveva essere risolto recandosi sul sito poiché non si conosceva nulla sullo stato della strumentazione e non si poteva interagire con la strumentazione. In presenza di un guasto al sistema di trasmissione o del semplice deterioramento dello stesso, si creavano buchi enormi nell'acquisizione nonostante lo strumento fosse perfettamente funzionante. Tutto ciò perché semplicemente i dati non erano archiviati in locale. Si capì che era necessario rendere indipendenti l'acquisizione e la trasmissione, abbandonando la trasmissione in continuo e impiegando l'energia recuperata a vantaggio di una logica di comando più flessibile e sofisticata. Il canale di trasmissione divenne bidirezionale. Una trasmissione bidirezionale ha permesso di aggiungere qualcosa fino ad allora trascurata: la diagnostica [Sicali et al, 2017]. La diagnostica, essenziale per qualsiasi sistema, avrebbe permesso di conoscere in anticipo i problemi che affliggevano un determinato sito di misura, di attivare degli interventi mirati e ottimizzando le risorse disponibili. Allo stesso tempo l'abbassamento dei consumi energetici ha permesso di aumentare molto la continuità nelle misure. Grazie al nuovo tipo di trasmissione bidirezionale (*GSM*) veniva assicurata comunque una tempestiva disponibilità del dato, non solo al centro di

raccolta dati ma a tutti coloro che ne avevano necessita, rendendo disponibili le misure a luoghi diversi. Si potevano recuperare dati, conoscere lo stato della strumentazione ed eventualmente intervenire da qualsiasi luogo. Con il passare degli anni si è comunque compreso che per mantenere un certo equilibrio energetico del sistema remoto, si dovevano comunque limitare i collegamenti a quelli strettamente necessari, soprattutto di notte ed durante l'inverno, evitando di recuperare più volte le medesime misure.

2. Origini del sistema *Mag-Net*

Con il sistema *Mag-Net* sono stati introdotti nella rete magnetica del M.te Etna anche i magnetometri ad effetto *Overhauser* [Overhauser, 1953] prodotti dalla *GEM Systems*, una ditta canadese all'avanguardia nella produzione di magnetometri. I magnetometri *Overhauser*, rispetto ai semplici magnetometri a precessione di protoni, hanno consumi più ridotti poiché la polarizzazione in continuo (*DC current*) è stata sostituita da una polarizzazione in radio frequenza (circa 55 Mhz), che permette di ridurre i consumi di quasi un ordine di grandezza (circa 8 W contro 1.5 W). I magnetometri scelti (dei *GSM90)*, sono molto robusti e con un'architettura (sia *hardware* che *software*) che ricordava molto quella impiegata dai primi satelliti artificiali, li rende ideali per essere utilizzati nelle condizioni estreme del M.te Etna. Le misure erano molto accurate: con una sensibilità di 20 pT contro i 100 pT erano pochi i casi in cui fallivano la misurazione. Avevano un'elettronica più moderna e potevano ricevere direttamente comandi da un qualsiasi terminale *RS232C*. In generale non necessitavano di modifiche o interventi e permettevano facilmente di riprodurre una stazione di acquisizione. Strumenti così flessibili e facilmente gestibili sono molto rari, ancora oggigiorno. Per affiancare uno strumento dalle potenzialità così spiccate si è pensato a un computer industriale in standard *PC/104* che riprende l'architettura *ISA* della *IBM*. Inizialmente si era pensato a un sistema ibrido e molto interattivo basato su Microsoft Windows (appendice A1), soprattutto per ridurre i costi in termini di ore di programmazione. Purtroppo il sistema operativo risentiva degli improvvisi blackout energetici e fu ben presto soppiantato poichè non adatto a sostenere i sistemi di acquisizione [Sicali et al, 2016]. A distanza di decenni si è riscontrato lo stesso problema sul sistema di gestione dei dilatometri da pozzo [Sicali et al, 2013]. Per quel sistema, più chiuso e rigido rispetto a *Mag-Net*, si è fortunatamente risolto facendo semplicemente comunicare il sistema di acquisizione con il sistema energetico. Una soluzione simile funziona bene per i distacchi energetici programmati ma malissimo per quelli imprevisti e tutt'oggi non ha una soluzione se non quella di sostiture il sistema operativo centrale. Tutto perché in un sistema di alto livello le operazioni di *startup* e *shutdown* non sono semplici [Sicali et al, 2013] mentre le installazioni sull'Etna e sull'isola di Stromboli richiedono semplicità [Sicali et al, 2016]. La soluzione definitiva per *Mag-Net* è stata quella di usare un sistema operativo come l'*MS-DOS* privo di una procedura di *shutdown*.

L'utilizzo di *MS-DOS* unita alla versatilità del linguaggio *C++* permise di formare un sistema *software* molto versatile. Grazie alla compatibilità dei *PC/104* con i *Personal Computer* è stato possibile scrivere e correggere il *software* molto velocemente. Ciononostante, l'eccessiva somiglianza con i sistemi desktop per ufficio, che inizialmente permise di sviluppare velocemente il sistema ne limitò in seguito molto l'evoluzione *hardware*. Alla fine del 1998 il sistema fu terminato e installato sul M.te Etna. Il *software* negli anni si è evoluto adattandosi alle diverse richieste molto facilmente. I problemi maggiori si sono avuti e si hanno ancora sull'*hardware* che diventa velocemente obsoleto ed è troppo rigido per alcune applicazioni (per esempio l'acquisizione analogica). Ciò nonostante il sistema mostrò la sua duttilità e venne impiegato per molte altre applicazioni e strumentazioni diverse come per esempio i gravimetri [Greco et al, 2008]. L'esperienza maturata dallo studio di *Mag-Net* ha permesso l'installazione dei dilatometri da pozzo nel 2011 [Sicali et al, 2013] e nel 2014 [Bonaccorso et al, 2015]. A *Mag-Net* è stato successivamente aggiunto anche un sistema di sincronizzazione delle misure utilizzante i *GPS* [Sicali et al, 2016].

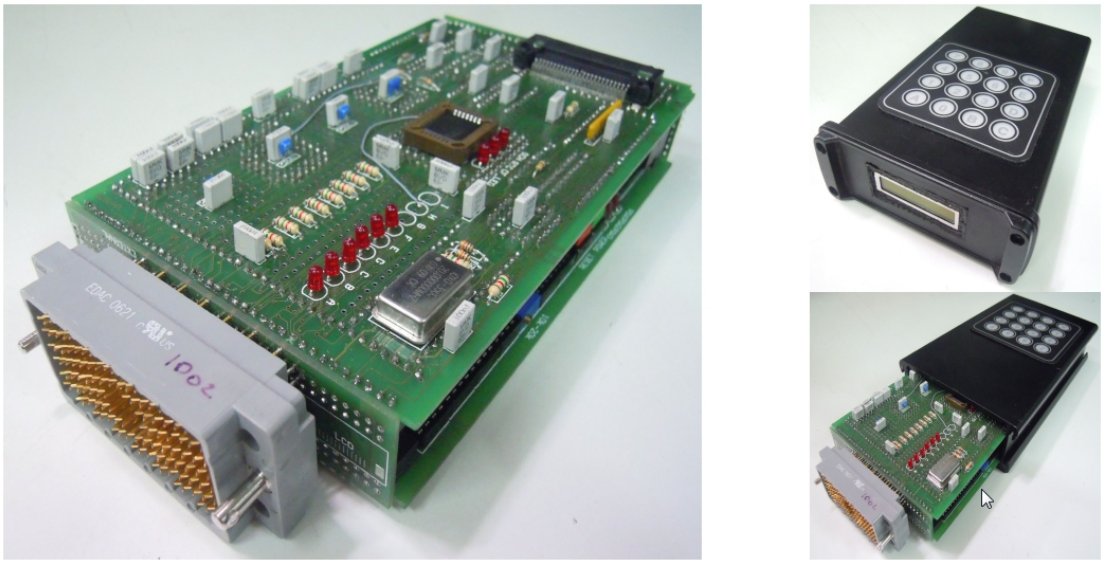

Figura 2. Sistema di acquisizione *IDRA*.

3. Evoluzione di *Mag-Net* e il nuovo sistema *IDRA* (*Interactive Device for Remote Acquisition*)

Dallo studio decennale di *Mag-Net* è nato il progetto *IDRA* (figura 2) che mira a dare una soluzione a tutti quei problemi di un sistema oramai divenuto obsoleto, soprattutto dal punto di vista *hardware*. L'idea principale è quella di rimpiazzare *l'MS-DOS* del sistema *Mag-Net* con un ambiente multiprogrammato (*OSGM*) [Sicali et al, 2018] in cui coesistono diversi processi arbitrati da un *kernel* monolitico stabile e robusto. Ogni task del sistema attuale sarà un processo del nuovo sistema, isolato perfettamente e capace solamente di elaborare le informazioni ricevute in ingresso. Degli oggetti semplici che possono essere sostituiti al *runtime* dal *kernel* in caso di malfunzionamento, non permettendo alcuna perdita di informazione. Nel nuovo sistema verranno presi molto seriamente gli errori, da quelli più semplici a quelli critici, ponendo al primo posto la salvaguardia delle informazioni elaborate e il continuo ed esaustivo funzionamento nel tempo del sistema, riducendo o addirittura annullando il numero di buchi nel dato acquisito. Il progetto si è sviluppato in due fasi successive. La prima fase di progettazione *software* ha visto la realizzazione di un emulatore dell'*hardware* per eseguire il *software* durante la fase di sviluppo (figura 3). Si veda in proposito l'esempio di emulazione del circuito integrato 8059 riportato nell'appendice B1 e B2. Per ogni dispositivo hardware è stato realizzato un software che ne riproducesse pienamente il comportamento. Ogni volta che si rendeva necessario un nuovo componente si creava un nuovo equivalente software. Tutti i componenti sono collegati tra loro attraverso un bus di sistema virtuale come accade nell'architettura di Von Neumann.

Figura 3. Emulatore dell'*hardware* di *IDRA*.

La seconda fase ha visto la realizzazione dell'*hardware* necessario per eseguire il *software* durante il normale utilizzo. La suddivisione del progetto in due parti non era strettamente necessaria ma aiutatò nello sviluppo. La realizzazione dell'emulatore, anche se impiega ulteriore tempo di programmazione, permette sul lungo periodo di ridurre i tempi di sviluppo, così come successe per il sistema *Mag-Net* e il *PC/104*. Con un emulatore completamente aperto e flessibile è possibile progettare gli strumenti di sviluppo secondo le proprie esigenze. In un sistema di sviluppo commerciale basato su *ICD* (In-Circuit Debugger) gli strumenti sono in numero finito e addirittura potrebbero anche essere insufficienti a risolvere determinati problemi. Supponiamo di ricercare un *bug* che danneggia una zona di memoria (*buffer overflow bug*). Di *bug* del genere per fortuna se ne creano pochi però sono tra i più difficili da togliere poiché gli effetti sono visibili solo successivamente alla posizione del *bug* e molte volte vengono influenzati anche dal *software* utilizzato per il *debug*. Un emulatore dell'*hardware* può, nel momento esatto in cui si verifica la corruzione della memoria, generare un'eccezione e interrompere l'esecuzione del programma grazie al pieno controllo che ha del sistema, a qualsiasi livello e dettaglio. Per fare lo stesso con un *ICD* si dovrebbe ricercare un processore (o microcontrollore) con tale funzionalità limitando molto le scelte possibili. Utilizzando un emulatore *hardware* si può, per esempio, prevedere uno strumento che in automatico verifichi la stabilità del sistema introducendo errori casuali sul disco o nella memoria. Di strumenti così c'è ne potrebbero essere infiniti, tanti quanti sono i problemi che affliggono i sistemi di acquisizione e la flessibilità dell'ambiente di sviluppo aiuta alla loro realizzazione. Alla stabilizzazione del *software* e dopo aver chiarito tutte le necessità, si potrà passare allo sviluppo dell*'hardware* e risolvere eventuali problemi esclusivamente con l'ausilio di un *ICD*. L'emulatore realizzato copriva il 99% dell'*hardware* e ha permesso di provare ogni singola parte del *software* permettendo allo stesso tempo di definire quale struttura *hardware* fosse necessaria ancor prima di costruirla e senza necessariamente spendere denaro e tempo per la prototipazione. Una parte del sistema *Mag-Net* era stata scritta in *ASSEMBLY* usando il set di istruzioni *x86* della Intel. Quella scelta era stata presa in considerazione per aumentare la velocità di esecuzione di quelle routine che venivano usate per la compressione dei dati prima di archiviarli e spedirli. Nella prima versione dell'*hardware*, per utilizzare la maggior parte del *software* di *Mag-Net* e concentrarsi solo sulla parte innovativa, si è deciso di utilizzare ancora un processore Intel, senza comunque escludere in un futuro di passare ad un processore diverso, ad esempio. *ARM*. Nonostante il processore (*80C188*) non sia molto recente i consumi non sono elevati, e tutto il sistema consuma meno di 1 W (il consumo del *PC/104* attualmente è circa 6 W). Grazie alla multiprogrammazione e a un sistema operativo studiato ad hoc, i processi saranno molto snelli e minimali. La memoria di soli 1 MB sarà sufficiente per tutto il sistema. La zona di memoria alta di sola lettura (FLASH EP*ROM*), ospiterà il *kernel* in esecuzione, quindi non potrà essere corrotta da alcun *bug* salvaguardando la consistenza dello stesso. La zona di memoria bassa (*RAM*) ospiterà i processi in esecuzione che verranno richiamati dal disco (*Compact Flash*). Sia il *kernel* che i processi possono essere aggiornati in remoto attraverso delle procedure di aggiornamento sicure. Anche l'aggiornamento dei processi non bloccherà mai l'acquisizione. Il disco sarà organizzata in modo molto semplice per mezzo di una *FAT* (*File Allocation Table*) e affiancata da un tipo molto semplice di *journaling* che assicura sempre la consistenza del *filesystem* in presenza di blackout. Le misure verranno memorizzati in file duplicati (tipo *RAID1*) i cui cluster saranno frammentati volontariamente sul disco per evitare che un eventuale errore, che di solito influenza una porzione contigua del disco, possa creare perdita di dati. I *file* saranno compressi con un sistema studiato ad hoc [Sicali, 2000]. Le periferiche di *I/O sono* connesse al bus di sistema e vi si può accedere attraverso delle semplici istruzioni di lettura. Si potranno collegare un numero illimitato di periferiche ed tutte saranno capaci di funzionare indipendentemente dal microprocessore (funzionamento asincrono) che sarà avvertito solo in presenza di informazione. Nel sistema *Mag-Net* era molto complicato inserire una nuova periferica, in *IDRA* diviene solo una formalità. Il sistema *IDRA* avrà una gestione asincrona delle risorse per non permettere che un processo ostacoli gli altri. Il *kernel* farà da arbitro attuando un modello con pre-rilascio (*preemptive*). Nel sistema *Mag-Net* è stata introdotta la trasmissione bidirezionale, in *IDRA* la trasmissione acquista una veste tutta nuova: diventa un trasmissione multicanale permettendo a ogni processo di scambiare dati e di accedere all'unico mezzo fisico disponibile. La trasmissione assicura un canale attivo e stabile tra la stazione remota e il centro di controllo per permettere anche in caso di avaria di avviare test diagnostici e richiedere informazioni sullo stato e agire di conseguenza. Un canale affidabile permetterà di sostituire parti *software* difettose riavviare processi bloccati tutto nella massima trasparenza e senza perdita d' informazioni.

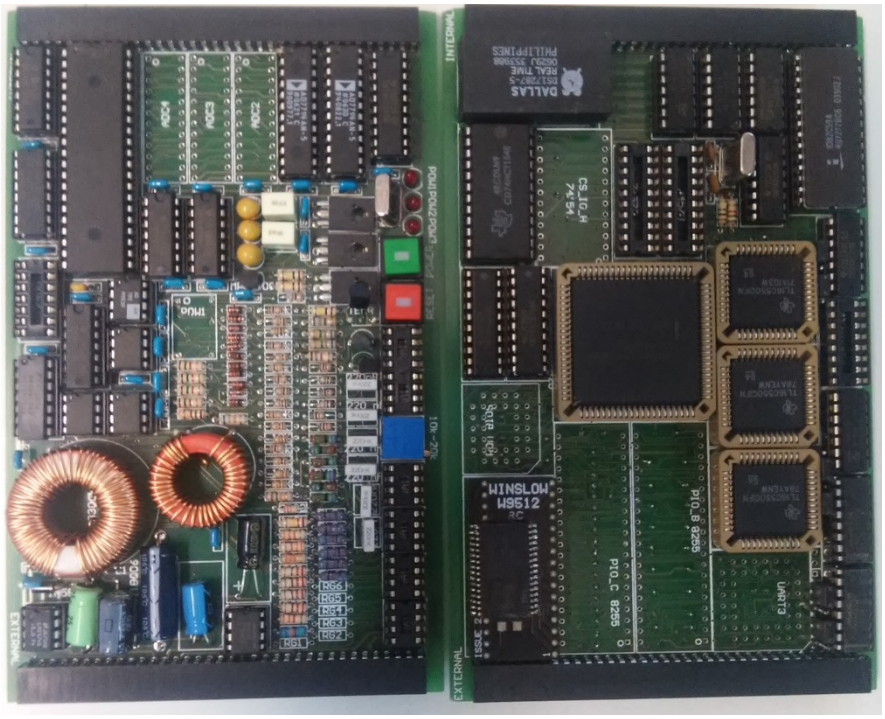

Figura 4. Dettaglio dei componenti utilizzati su *IDRA*.

Anche dal punto di vista *hardware* sono state approntati diversi miglioramenti: la componentistica che di solito usciva di produzione è stata sostituita da integrati comuni reperibili in qualsiasi negozio di elettronica allungando la vita di ciascuna versione *hardware* di *IDRA* prima della successiva. Si è cercato di usare dei componenti grandi (figura 4) per facilitare interventi in urgenza in sito dove la manualità non è semplice. Si sono eliminate le connessioni attraverso fili per diminuire il tempo di assemblaggio e il rischio di falsi contatti. *IDRA* è stato progettato per essere multi-parametrico e può interfacciarsi con diversi tipi di strumentazione, anche contemporaneamente. Il prototipo di *IDRA* possiede 6 seriali *RS232/TTL*, 25 canali *ADC* a 24 bit (che divengono 5 se usati in continuo) circa 70 uscite/ingressi digitali. I valori sono solo indicativi e si possono estendere, i *PC/104* finora usati hanno 4 seriali *RS232/TTL*, circa 16 porte uscite/ingressi e nessun canale di acquisizione analogica (tabella 1).

Caratteristica	***Mag-Net***	***IDRA***
Consumo	6 W	1 W
Dimensioni	25x25x10 cm	12x20x5 cm
Sistema Operativo	MS-DOS	OSGM [Sicali et al, 2018]
Seriali RS-232C	4	6
Canali analogici	0	5 in acquisizione continua o 25 in acquisizioni singole. Tutti a 24 bit bit tipo **ΣΔ**.
GPIO (General Purpose Input Output)	Porta Parallela, con alcuni solo uscite e altri solo ingressi (8-10)	70 completamente configurabili

Tabella 1. Dati riassunti delle caratteristiche di *Mag-Net* e *IDRA*.

Conclusioni

A distanza di 20 anni il sistema *Mag-Net* è ancora in uso nella rete magnetica dell'Etna e dell'isola di Stromboli e assicura l'acquisizione nelle 24h per ogni giorno dell'anno. Nonostante *IDRA* sia ancora allo stadio prototipale, ci auspichiamo che possa ben presto, grazie alle sue caratteristiche sostituire non solo il sistema *Mag-Net* ma anche altri sistemi per l'acquisizione di segnali analogici e digitali.

Bibliografia

Overhauser A.W., (1953). *Polarization of Nuclei in Metals*. Phys. Rev. 92, 411.

Del Negro C., Di Bella A., Ferrucci F., Napoli R., Sicali A., (1998). *Automated System for Magnetic Surveillance of Active Volcanoes*. In Final Report Tekvolc: Technique and Method Innovation in Geophysical Research, Monitoring and Early Warning at Active Volcanoes. Commission of European Communities Environment Programme, Contract ENV4 CT95 0251.

Del Negro C., Di Bella A., Napoli R., Sicali A., (1999). *Development of an automated console prototype for control of remote magnetic sensors.* In Interim Report Tomave: Electromagnetic and potential field integrated tomographies applied to volcanic environments. Commission of European Communities Environment Programme, Contract ENV4 CT98 0697.

Sicali A., (2000). *Entropia dei segnali continui nel tempo.* Computer Programming n. 89, marzo 2000, pp. 68-72, ISSN 1123-8526.

Del Negro C., Napoli R. & Sicali A., (2002). *Automated system for magnetic monitoring of active volcanoes*. Bull. Volcanol. 64, 94-99.

Greco F., Carmisciano C., Del Negro C., Loretti I., Sicali A., Stefanelli P. (2008). *Seismic-induced accelerations detected by two coupled gravity meters in continuous recording with a high sampling rate at Etna volcano*. Annals of Geophysics.

Sicali A., Bonaccorso A., (2013). *Gestione dei dilatometri installati in pozzi profondi all'Etna*. Rapporti Tecnici INGV, n° 258, ISSN 2039-7941.

Bonaccorso A., Currenti G., Sicali A., (2015). *La rete dei dilatometri in pozzi profondi dell'Etna*. Quaderni di Geofisica, n° 126, ISSN 1590-2595.

Sicali A., Amantia A., Cappuccio P., (2016). *Linee guida e criticità nella progettazione di sistemi per l'acquisizione di dati geofisici in prossimità di vulcani attivi.* Rapporti Tecnici INGV. n° 347, ISSN 2039-7941.

Sicali A., Cappuccio P., Amantia A., (2016). *Software per la sincronizzazione dei sistemi di acquisizione attraverso l'uso del GPS: implementazione hardware per l'architettura PC/104.* Rapporti Tecnici INGV, n° 356, ISSN 2039-7941.

Sicali A., Cappuccio P., Amantia A., (2017). *Strumento di Diagnostica e Amministrazione Remota di reti per il monitoraggio di parametri geofisici.* Rapporti Tecnici INGV n° 375, ISSN 2039-7941.

Sicali A., Amantia A., Cappuccio P., (2018). *Realizzazione di un sistema operativo Client-Server per la gestione di stazioni geomagnetiche remote e relativo simulatore in ambiente Unix.* Rapporti Tecnici INGV, n° 395, ISSN 2039-7941.

Appendice A1 – Sorgenti dell'applicazione Windows del software di *Mag-Net* del 1998

```
#include <windows.h>
#include <conio.h>
#include <string.h>
#include <stdio.h>
#include <gsm90.h>
#include <dos.h>

void ReadCFG(void);
void CreaCFG(void);
void DialogStatus(HANDLE hInstance,HWND hwnd);
BOOL FAR PASCAL dialogstatus(HWND hwnd,WORD message,WORD wParam,LONG lParam);
int GetTime(HANDLE hInstance,HWND hwnd);
BOOL FAR PASCAL dialogtime(HWND hwnd,WORD message,WORD wParam,LONG lParam);
void Lettura(HWND parent,HINSTANCE hInstance);
BOOL FAR PASCAL Gestore_Letture(HWND hwnd,WORD message,WORD wParam,LONG lParam);
HWND InitGSM90(HANDLE hInstance,int nCmdShow);
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int
nCmdShow);
BOOL FAR PASCAL Gestore(HWND parent,WORD message,WORD wParam,LONG lParam);
void MenuItems(HWND parent,WORD wParam,HINSTANCE hInstance);
BOOL InitSerial(HWND hwnd,HANDLE hInstance);
BOOL FAR PASCAL dialogerr(HWND hwnd,WORD message,WORD wParam,LONG lParam);
void DialogErr(HANDLE hInstance,HWND hwnd,LONG lParam);
BOOL Salva_Set_Misure(char **disco,char *memoria,unsigned int *count);
void FormatFile(char *name,unsigned int count);
void Compact(void);
void SeparaDecine(unsigned char a,unsigned char *d,unsigned char *u);
void SalvaMisura(char *buffer,char **pmisure,TIMESTRUCT tempo_misura);
void GetCmos(TIMESTRUCT *tempo_misura);
void ERRORE_OVERFLOW(void);
void ERRORE_DISCO(void);
void ERRORE_TIMEOUT(TIMESTRUCT tempo_misura);
void EffettuaMisura(HWND hwnd);
int Compress(char *source,char *drain,int nmisure);
BOOL AggiornaCFG(void);

// variabili globali
static int COM_FILE=0;
static HINSTANCE hInst;
static char ERRORE[BYTE_MISURA+1];
static char *perrore=ERRORE;
static char MISURE[NUMERO_FILE_MEMORIA*NUMERO_MISURE_FILE*BYTE_MISURA];
static char *pdisco=MISURE;
static char *pmisure=MISURE;
static int COMPACT=NO;
static int Q_timeout=NO;
static int BLOCK=NO;
static HWND HMAIN;
static FARPROC letture;
static HWND hletture=0;
static int TIMER_CHANGE=NO;
static GSM90_PARAMETER STANDARD_PARAMETERS;
static int SALVA_SET = YES;
static HMENU MENU_BAR;

#pragma argsused

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR lpCmdLine, int
nCmdShow)
```

```
{
        int esci=NO;
        HWND hwnd;
        MSG msg;

        if (hPrevInstance == 0) {
                                        hInst=hInstance;
                                        hwnd=InitGSM90(hInstance,nCmdShow);
                                        HMAIN=hwnd;
                                        if (hwnd == 0) { esci=YES; }
                                        while (esci == NO) {
                                                                if
(PeekMessage(&msg,NULL,0,0,PM_REMOVE) != 0) {



        if (msg.message == WM_QUIT) { esci=YES; }

                                                                if
(Salva_Set_Misure(&pdisco,pmisure,&(STANDARD_PARAMETERS.THIS_FILE))     ==     NO)
{ SALVA_SET = NO; COMPACT=NO; ERRORE_DISCO(); }
                                                        }

                                    }
         return(msg.wParam);

}


void EffettuaMisura(HWND hwnd)
{
        char buffer[20];
        char s[128];
        TIMESTRUCT tempo_misura;
        int i;
        int c=0;
        MSG msg;
        DWORD t1;
        DWORD t2;
         DWORD t;

        c=0;
        GetCmos(&tempo_misura);
        if (TIMER_CHANGE == YES) {
                                        TIMER_CHANGE = NO;
                                        KillTimer(hwnd,ID_TIMER);

        SetTimer(hwnd,ID_TIMER,STANDARD_PARAMETERS.DELAY*1000,NULL);
                                    }
        Q_timeout=NO;
         SALVA_SET = YES;
        Compact();
        TransmitCommChar(COM_FILE,'F');
        BLOCK = YES;
        t1=GetTickCount();

        while (c<10 && Q_timeout == NO) {
                                        t2=GetTickCount();
                                        if (t1 <= t2) { t=t2-t1; }
                                                    else
                                                    {
                                                       t=0xFFFFFFFF-t1;
```

```
                                                    t += t2;
                                                }
                                        if                  (t                  >=
(STANDARD_PARAMETERS.TIMEOUT*1000)) {

        Q_timeout = YES;

             }
                                        if   (ReadComm(COM_FILE,buffer,1)   ==   1)
{ s[c]=buffer[0]; c++; }
                                         }
        s[c]=0;
        if   (Q_timeout   ==   YES)   {   ERRORE_TIMEOUT(tempo_misura);   }     else
{ SalvaMisura(s,&pmisure,tempo_misura); }
        BLOCK= NO;
}

HWND InitGSM90(HANDLE hInstance,int nCmdShow)
{
        WNDCLASS hmain;
         HWND hwnd;
        HMENU hmenu;

        SetErrorMode(SEM_NOOPENFILEERRORBOX | SEM_FAILCRITICALERRORS);

        STANDARD_PARAMETERS.a=0;
        STANDARD_PARAMETERS.DELAY=TEMPO_TRA_LE_MISURE;
        STANDARD_PARAMETERS.b=2;
        STANDARD_PARAMETERS.TIMEOUT=TEMPO_DI_TIMEOUT;
        STANDARD_PARAMETERS.c=4;
        STANDARD_PARAMETERS.COMPRESS=YES;
        STANDARD_PARAMETERS.d=6;
        STANDARD_PARAMETERS.THIS_FILE=0;
        STANDARD_PARAMETERS.e=8;
        STANDARD_PARAMETERS.LAST_FILE=(MASSIMO_NUMERO_FILE-1);
        STANDARD_PARAMETERS.f=10;
        STANDARD_PARAMETERS.LAST_ERROR=IDE_NOONE;

        ReadCFG();
        strcpy(ERRORE,"NOERROR");

        hmain.style=CS_NOCLOSE;
        hmain.lpfnWndProc= Gestore;
        hmain.cbClsExtra=0;
        hmain.cbWndExtra=0;
        hmain.hInstance=hInstance;
        hmain.hIcon=LoadIcon(hInstance,MAKEINTRESOURCE(IDI_MAIN));
        hmain.hCursor=LoadCursor(NULL,IDC_ARROW);
        hmain.hbrBackground=CreateSolidBrush(BIANCO);
        hmain.lpszMenuName=NULL;
        hmain.lpszClassName="GSM90";

        RegisterClass(&hmain);
        hwnd=CreateWindow("GSM90","GSM90",WS_OVERLAPPEDWINDOW,0,0,640,480,NULL,N
ULL,hInstance,NULL);
        hmenu=LoadMenu(hInstance,MAKEINTRESOURCE(ID_BARMENU));
        MENU_BAR=hmenu;
        SetMenu(hwnd,hmenu);
        if (InitSerial(hwnd,hInstance) == NO) { return(0); }
        ShowWindow(hwnd,nCmdShow);
        //SetMenuItemBitmaps(hmenu,IDM_COMPRESS,MF_BYCOMMAND,NULL,NULL);
        if (STANDARD_PARAMETERS.COMPRESS == YES) {
```

```
        CheckMenuItem(MENU_BAR,IDM_COMPRESS,MF_BYCOMMAND | MF_CHECKED);
                                                }
                                                else
                                                {

        CheckMenuItem(MENU_BAR,IDM_COMPRESS,MF_BYCOMMAND | MF_UNCHECKED);
                                                  }

        SetTimer(hwnd,ID_TIMER,STANDARD_PARAMETERS.DELAY*1000,NULL);
        return(hwnd);
}

BOOL FAR PASCAL Gestore(HWND parent,WORD message,WORD wParam,LONG lParam)
{
        if                      (BLOCK                      ==                      YES)
{ return(DefWindowProc(parent,message,wParam,lParam)); }
        switch (message) {
                                case WM_COMMAND:
                                        MenuItems(parent,wParam,hInst);
                                        break;
                                case WM_TIMER:
                                        if (wParam == ID_TIMER) {

        EffettuaMisura(parent);
                                                                        return(0);
                                                                }
                        }
        return(DefWindowProc(parent,message,wParam,lParam));
}

#pragma argsused

void MenuItems(HWND parent,WORD wParam,HINSTANCE hInstance)
{
        static int r;
        switch (wParam) {
                                case IDM_COMPRESS:
                                        if (STANDARD_PARAMETERS.COMPRESS == YES) {

        CheckMenuItem(MENU_BAR,IDM_COMPRESS,MF_BYCOMMAND | MF_UNCHECKED);

        STANDARD_PARAMETERS.COMPRESS = NO;
                                                                }
                                                                else
                                                                {

        CheckMenuItem(MENU_BAR,IDM_COMPRESS,MF_BYCOMMAND | MF_CHECKED);

        STANDARD_PARAMETERS.COMPRESS = YES;
                                                                }
                                        CreaCFG();
                                        break;
                                case IDM_ESCI:
                                        PostQuitMessage(0);
                                        break;
                                case IDM_TEMPOMISURA:
                                        r=GetTime(hInstance,parent);
                                        if (r != -1) {
                                                        if (r > 65) {

        DialogErr(hInstance,parent,(LONG) "Periodo troppo lungo");
```

```
                                                                    break;
                                                                }
                                                        if          (r          <=
STANDARD_PARAMETERS.TIMEOUT+3 ) {

        DialogErr(hInstance,parent,(LONG) "Periodo troppo breve");
                                                                    break;
                                                                }
                                                        TIMER_CHANGE = YES;

        STANDARD_PARAMETERS.DELAY=r;
                                                        CreaCFG();
                                                    }
                                    break;
                                case IDM_TEMPOCONCESSO:
                                    r=GetTime(hInstance,parent);
                                    if (r != -1) {
                                                        if          (r+3          >=
STANDARD_PARAMETERS.DELAY) {

        DialogErr(hInstance,parent,(LONG) "Periodo troppo lungo");
                                                                    break;
                                                                }
                                                        if (r < 4 ) {

        DialogErr(hInstance,parent,(LONG) "Periodo troppo breve");
                                                                    break;
                                                                }
                                                        while (BLOCK == YES) {
                                                                            }

        STANDARD_PARAMETERS.TIMEOUT=r;
                                                        CreaCFG();
                                                    }
                                    break;
                                case IDM_STATO:
                                    DialogStatus(hInstance,parent);
                                    break;
                                case IDM_LETTURA:
                                    Lettura(parent,hInstance);
                                    break;
                        }

}

void DialogStatus(HANDLE hInstance,HWND hwnd)
{
        FARPROC de;

        de=MakeProcInstance(dialogstatus,hInstance);
        DialogBoxParam(hInstance,MAKEINTRESOURCE(IDD_GSM90),hwnd,de,(LONG)
ERRORE);
        FreeProcInstance(de);
        strcpy(ERRORE,"NOERROR");
}

BOOL FAR PASCAL dialogstatus(HWND hwnd,WORD message,WORD wParam,LONG lParam)
{
        static char s[128];
        if (message == WM_INITDIALOG) {
                                    SetDlgItemText(hwnd,ID_TEXT_ERROR,(LPSTR)
lParam);
```

```
                                        sprintf(s,"Tempo   tra   le   misure:   %d
sec",STANDARD_PARAMETERS.DELAY);
                                        SetDlgItemText(hwnd,ID_DELAY,(LPSTR) s);
                                        sprintf(s,"Tempo necessario al GSM90 per
rispondere: %d sec",STANDARD_PARAMETERS.TIMEOUT);
                                        SetDlgItemText(hwnd,ID_TIMEOUT,(LPSTR) s);
                                        return(TRUE);
                                      }
        if (message == WM_COMMAND && wParam == IDOK) {
                                                    EndDialog(hwnd,TRUE);
                                                    return(TRUE);
                                                  }
        return(FALSE);
}


int GetTime(HANDLE hInstance,HWND hwnd)
{
        FARPROC de;
        int r;

        de=MakeProcInstance(dialogtime,hInstance);
        r=DialogBox(hInstance,MAKEINTRESOURCE(IDD_TIME),hwnd,de);
        FreeProcInstance(de);
        return(r);
}

#pragma argsused

BOOL FAR PASCAL dialogtime(HWND hwnd,WORD message,WORD wParam,LONG lParam)
{
        static int newtime;
        static int error;

        if (message == WM_COMMAND && wParam == IDOK) {

        newtime=GetDlgItemInt(hwnd,ID_EDIT_TIME,&error, 0);
                                                    if (error != 0) {

        EndDialog(hwnd,newtime);

        return(TRUE);
                                                                    }
                                                  }
        if (message == WM_COMMAND && wParam == IDCANCEL) {
                                                          EndDialog(hwnd,-1);
                                                          return(TRUE);
                                                        }
        return(FALSE);
}

void Lettura(HWND parent,HINSTANCE hInstance)
{
         if (hletture != 0) { return; }
         letture=MakeProcInstance(Gestore_Letture,hInstance);

hletture=CreateDialog(hInstance,MAKEINTRESOURCE(IDD_LETTURE),parent,letture);
}

#pragma argsused

BOOL FAR PASCAL Gestore_Letture(HWND hwnd,WORD message,WORD wParam,LONG lParam)
```

```
{
        if (message == WM_COMMAND && wParam == IDOK) {
                                                DestroyWindow(hwnd);
                                                FreeProcInstance(letture);
                                                hletture=0;
                                                   return(TRUE);
                                              }
        return(FALSE);
}


BOOL InitSerial(HWND hwnd,HANDLE hInstance)
{
        DCB dcb;
         int err;
        char s[128];

        COM_FILE=OpenComm("COM1",10,1);
        if (COM_FILE < 0) {
                                DialogErr(hInstance,hwnd,(LONG) "OpenComm");
                                return(NO);
                           }

        err=BuildCommDCB("COM1:9600,n,8,1", &dcb);
        if ( err < 0) {
                          DialogErr(hInstance,hwnd,(LONG) "BuildCommDCB");
                          return(NO);
                        }

        err=SetCommState(&dcb);
        if ( err < 0) {
                                        sprintf(s,"SetCommState %d",dcb.Id);
                                        DialogErr(hInstance,hwnd,(LONG) s);
                                        return(NO);
                                      }


        return(YES);
}

void DialogErr(HANDLE hInstance,HWND hwnd,LONG lParam)
{
        FARPROC de;

        de=MakeProcInstance(dialogerr,hInstance);
        DialogBoxParam(hInstance,MAKEINTRESOURCE(IDD_ERROR),hwnd,de,lParam);
         FreeProcInstance(de);
}

BOOL FAR PASCAL dialogerr(HWND hwnd,WORD message,WORD wParam,LONG lParam)
{
        if (message == WM_INITDIALOG) {
                                        SetDlgItemText(hwnd,ID_TEXT_ERROR,(LPSTR)
lParam);
                                     }
        if (message == WM_COMMAND && wParam == IDOK) {
                                                EndDialog(hwnd,TRUE);
                                                return(TRUE);
                                              }
        return(FALSE);
}
```

```
BOOL Salva_Set_Misure(char **disco,char *memoria,unsigned int *count)
{
        char name[128];
        OFSTRUCT file;
        HFILE hfile;
        int nbyte;
        char dati_compressi[NUMERO_MISURE_FILE*BYTE_MISURA];
         int *pdati;
        TIMESTRUCT tempo_errore;

        if (COMPACT == YES | SALVA_SET == NO) { return(YES); }
        if ((memoria-*disco) < NUMERO_MISURE_FILE*BYTE_MISURA) { return(YES); }
        FormatFile(name,*count);
        hfile=OpenFile(name,&file,OF_CREATE | OF_READWRITE);
        if (hfile == HFILE_ERROR) { return(NO); }

        //compressione dati
        pdati=dati_compressi;
        dati_compressi[9]=0x0d;
        dati_compressi[8]=0x0a;
        if (STANDARD_PARAMETERS.COMPRESS == YES) {
                                *pdati=FILE_COMPRESSO;

        nbyte=Compress(*disco,dati_compressi+10,NUMERO_MISURE_FILE);
                                pdati++;
                                *pdati=nbyte;
                                pdati++;
                                  *pdati=NUMERO_MISURE_FILE;
                                if    (_lwrite(hfile,dati_compressi,nbyte+10)    ==
HFILE_ERROR) { return(NO); }
                               }
                               else
                               {
                                  *pdati=FILE_NONCOMPRESSO;
                                  pdati++;
                                  pdati++;
                                  *pdati=NUMERO_MISURE_FILE;
                                  if       (_lwrite(hfile,dati_compressi,10)       ==
HFILE_ERROR) { return(NO); }
                                  if
(_lwrite(hfile,*disco,NUMERO_MISURE_FILE*BYTE_MISURA)       ==       HFILE_ERROR)
{ return(NO); }
                               }
        if (_lclose(hfile) == HFILE_ERROR) { return(NO); }
        (*count)++;
        if (*count >= MASSIMO_NUMERO_FILE) *count=0;
        COMPACT=YES;
        if (STANDARD_PARAMETERS.THIS_FILE == STANDARD_PARAMETERS.LAST_FILE) {

        STANDARD_PARAMETERS.LAST_FILE++;
                                                                          if
(STANDARD_PARAMETERS.LAST_FILE              >=              MASSIMO_NUMERO_FILE)
{ STANDARD_PARAMETERS.LAST_FILE=0; }
                                                                        }
        if (AggiornaCFG() == NO) {
                                        GetCmos(&tempo_errore);

STANDARD_PARAMETERS.LAST_ERROR=IDE_UPDATE;
                                        SalvaMisura("NO
UPDATE",&perrore,tempo_errore);
                                 }
```

```
        CreaCFG();
        return(YES);
}

void FormatFile(char *name,unsigned int count)
{
        char number[9];

        strcpy(name,PATH);
        sprintf(number,"%5.5hd",count);
        strcat(name,number);
        strcat(name,".");
        strcat(name,EXT);
}

void Compact(void)
{
        int c;
        int i;
        if (COMPACT == YES) {
                                pdisco += NUMERO_MISURE_FILE*BYTE_MISURA;
                                c=pmisure-pdisco;
                                pmisure=MISURE;
                                for (i=0;i<c;i++) { pmisure[i]=pdisco[i]; }
                                pdisco=MISURE;
                                pmisure += c;
                                COMPACT = NO;
                           }
}

void SalvaMisura(char *buffer,char **pmisure,TIMESTRUCT tempo_misura)
{
        char misura[BYTE_MISURA+1];
        int i=0,j=0;

        misura[BYTE_MISURA]=0;

        SeparaDecine(tempo_misura.giorno,misura+i,misura+i+1);
        i += 2;
        misura[i]='/';
        i++;
        SeparaDecine(tempo_misura.mese,misura+i,misura+i+1);
        i += 2;
        misura[i]='/';
         i++;
        SeparaDecine(tempo_misura.anno,misura+i,misura+i+1);
        i += 2;
        misura[i]=0x20;

         i++;
        SeparaDecine(tempo_misura.ore,misura+i,misura+i+1);
        i += 2;
        misura[i]=':';
         i++;
        SeparaDecine(tempo_misura.minuti,misura+i,misura+i+1);
        i += 2;
        misura[i]=':';
         i++;
        SeparaDecine(tempo_misura.secondi,misura+i,misura+i+1);
        i += 2;
        misura[i]=0x20;
```

```
        i++;
        for (j=0;j<10;j++) { misura[j+i]=buffer[j]; }

        for (j=0;j<BYTE_MISURA;j++) { *(*pmisure+j)=misura[j]; }

        if (hletture != 0) {
                                if
(SendDlgItemMessage(hletture,ID_LB_LETTURE,LB_GETCOUNT,0,0) >= 7) {



        SendDlgItemMessage(hletture,ID_LB_LETTURE,LB_ADDSTRING,0,(LONG) misura);
                            }
        if (*pmisure == ERRORE) { return; }
        *pmisure += BYTE_MISURA;

        if (*pmisure >= (MISURE+sizeof(MISURE))) {
                                                        ERRORE_OVERFLOW(); }
}

void SeparaDecine(unsigned char a,unsigned char *d,unsigned char *u)
{
        *d=a/16;
        *u=a-(*d)*16;
        *d += 0x30;
        *u += 0x30;
}

void ERRORE_DISCO(void)
{
        // setta le veriabili
        // il set di misure verr\E0 perso
        TIMESTRUCT tempo_errore;

        GetCmos(&tempo_errore);
        STANDARD_PARAMETERS.LAST_ERROR=IDE_DISK;
        SalvaMisura("DISCO ERR",&perrore,tempo_errore);
}


void ERRORE_OVERFLOW(void)
{
      // richiedi accesso al disco
      // al verificarsi di un errore i dati verranno persi
      int i;
      TIMESTRUCT tempo_errore;

      KillTimer(HMAIN,ID_TIMER);
      for (i=0;i<NUMERO_FILE_MEMORIA;i++) {
      if  (Salva_Set_Misure(&pdisco,pmisure,&(STANDARD_PARAMETERS.THIS_FILE))  ==
NO) {

        GetCmos(&tempo_errore);

        STANDARD_PARAMETERS.LAST_ERROR=IDE_OVERFLOW;

        SalvaMisura("OVERFLOW",&perrore,tempo_errore);
                                                                    break;
                                                            }
                                        Compact();
                                        }
      SetTimer(HMAIN,ID_TIMER,STANDARD_PARAMETERS.DELAY*1000,NULL);
```

```
       pmisure=MISURE;
       pdisco=MISURE;
       COMPACT=NO;
}

void ERRORE_TIMEOUT(TIMESTRUCT tempo_misura)
{
     char err[10];
     strcpy(err,"TIME OUT");
     err[8]=0x0A;
     err[9]=0x0D;
     SalvaMisura(err,&pmisure,tempo_misura);
     GetCmos(&tempo_misura);
     STANDARD_PARAMETERS.LAST_ERROR=IDE_TIMEOUT;
     SalvaMisura(err,&perrore,tempo_misura);     //questo

}

void GetCmos(TIMESTRUCT *tempo_misura)
{
        struct time t;
        struct date d;

        gettime(&t);

        tempo_misura->secondi=(t.ti_sec/10)*16+t.ti_sec-((t.ti_sec/10)*10);
        tempo_misura->minuti =(t.ti_min/10)*16+t.ti_min-((t.ti_min/10)*10);
        tempo_misura->ore    =(t.ti_hour/10)*16+t.ti_hour-((t.ti_hour/10)*10);

        getdate(&d);

        tempo_misura->giorno=(d.da_day/10)*16+d.da_day-((d.da_day/10)*10);
        tempo_misura->mese  =(d.da_mon/10)*16+d.da_mon-((d.da_mon/10)*10);
        d.da_year = d.da_year-((d.da_year/100)*100);
        tempo_misura->anno  =(d.da_year/10)*16+d.da_year-((d.da_year/10)*10);
        /*
        outp(0x70,0);
        tempo_misura->secondi=inp(0x71);
        outp(0x70,2);
        tempo_misura->minuti=inp(0x71);
        outp(0x70,4);
        tempo_misura->ore=inp(0x71);
        outp(0x70,7);
        tempo_misura->giorno=inp(0x71);
        outp(0x70,8);
        tempo_misura->mese=inp(0x71);
        outp(0x70,9);
        tempo_misura->anno=inp(0x71);
        */
}

void ReadCFG(void)
{
        OFSTRUCT file;
        HFILE hfile;
        char file_cfg[128];
        int *p;
        int i;
        GSM90_PARAMETER new_parameters;

        new_parameters.a=1;
```

```
        sprintf(file_cfg,"%s%s",PATH,FILE_CONFIGURAZIONE);

        hfile=OpenFile(file_cfg,&file,OF_READ);
        if (hfile == HFILE_ERROR) {
                                        CreaCFG();
                                        return;
                                }
        _lread(hfile,&new_parameters,sizeof(new_parameters));
        _lclose(hfile);
        p=&new_parameters;
        for (i=0;i<sizeof(new_parameters)/2;i += 2) {
                                                        if (*(p+i) != i) {

        CreaCFG();

        return;
                                                                        }
                                                }
        STANDARD_PARAMETERS=new_parameters;
}

void CreaCFG(void)
{
        char file_cfg[128];
        OFSTRUCT file;
        HFILE hfile;

        sprintf(file_cfg,"%s%s",PATH,FILE_CONFIGURAZIONE);

        hfile=OpenFile(file_cfg,&file,OF_CREATE);
        _lwrite(hfile,&STANDARD_PARAMETERS,sizeof(STANDARD_PARAMETERS));
        _lclose(hfile);
}

// la funzione ritorna il numero di byte da scrivere sul disco
int Compress(char *source,char *drain,int nmisure)
{
        int i;
        TIMESTRUCT tempo_misura;
        DWORD tempo;
        DWORD misura;
        DWORD *pdrain;
        char s[128];

        pdrain=drain;

        for (i=0;i<nmisure;i++)  {
                                        tempo_misura.giorno
=(*(source+(i*BYTE_MISURA)   )-0x30)*10;
                                        tempo_misura.giorno
+=(*(source+(i*BYTE_MISURA)+ 1)-0x30);

                                        tempo_misura.mese
=(*(source+(i*BYTE_MISURA)+ 3)-0x30)*10;
                                        tempo_misura.mese
+=(*(source+(i*BYTE_MISURA)+ 4)-0x30);

                                        tempo_misura.anno
=(*(source+(i*BYTE_MISURA)+ 6)-0x30)*10;
                                        tempo_misura.anno
+=(*(source+(i*BYTE_MISURA)+ 7)-0x30);
```

```
                                        tempo_misura.ore
=(*(source+(i*BYTE_MISURA)+ 9)-0x30)*10;
                                        tempo_misura.ore
+=(*(source+(i*BYTE_MISURA)+10)-0x30);

                                        tempo_misura.minuti
=(*(source+(i*BYTE_MISURA)+12)-0x30)*10;
                                        tempo_misura.minuti
+=(*(source+(i*BYTE_MISURA)+13)-0x30);

                                        tempo_misura.secondi
=(*(source+(i*BYTE_MISURA)+15)-0x30)*10;
                                        tempo_misura.secondi
+=(*(source+(i*BYTE_MISURA)+16)-0x30);

                                        tempo= tempo_misura.secondi;
                                        tempo += (tempo_misura.minuti*60L);
                                        tempo += (tempo_misura.ore   *60L*60L);
                                        tempo_misura.giorno--;
                                        tempo                                   +=
(tempo_misura.giorno*60L*60L*24L);
                                        tempo_misura.mese--;
                                        tempo         +=         (tempo_misura.mese
*60L*60L*24L*31L);
                                        tempo         +=         (tempo_misura.anno
*60L*60L*24L*31L*12L);

                                        if  (*(source+(i*BYTE_MISURA)+18L)  !=  'T')
{

                                        misura  =  ((*(source+(i*BYTE_MISURA)+18L)-
'a')*10000000L )    ; // 10^7
                                        misura                                  +=
((*(source+(i*BYTE_MISURA)+19L)-0x30)*1000000L)    ; // 10^6
                                        misura                                  +=
((*(source+(i*BYTE_MISURA)+20L)-0x30)*100000L )    ; // 10^5
                                        misura                                  +=
((*(source+(i*BYTE_MISURA)+21L)-0x30)*10000L   )    ; // 10^4
                                        misura                                  +=
((*(source+(i*BYTE_MISURA)+22L)-0x30)*1000L    )    ; // 10^3
                                        misura                                  +=
((*(source+(i*BYTE_MISURA)+23L)-0x30)*100L     )    ; // 10^2
                                        misura                                  +=
((*(source+(i*BYTE_MISURA)+24L)-0x30)*10L      )    ; // 10^1
                                        misura                                  +=
((*(source+(i*BYTE_MISURA)+25L)-0x30)*1L       )    ; // 10^0

 }

else
                                                                                {

misura=41704977;
                                                                                }
                                        *pdrain= tempo;
                                        pdrain++;
                                          *pdrain=misura;
                                        pdrain++;
                                    }

        return((int) pdrain-(int) drain);
```

```
}

BOOL AggiornaCFG(void)
{
        OFSTRUCT file;
        HFILE hfile;
        char file_cfg[128];
        int *p;
         int i;
        GSM90_PARAMETER new_parameters;

        new_parameters.a=1;

        sprintf(file_cfg,"%s%s",PATH,FILE_CONFIGURAZIONE_UPDATE);

        hfile=OpenFile(file_cfg,&file,OF_READ);
        if (hfile == HFILE_ERROR) {
                                        return(NO);
                                }
        _lread(hfile,&new_parameters,sizeof(new_parameters));
        _lclose(hfile);
        p=&new_parameters;
        for (i=0;i<sizeof(new_parameters)/2;i += 2) {
                                                        if (*(p+i) != i) {

        return(NO);
                                                                        }
                                                }
        STANDARD_PARAMETERS.LAST_FILE=new_parameters.LAST_FILE;
        return(YES);
}
```

Appendice B1 – Esempio di circuito integrato simulato: PIC 8059 (header file).

```
#ifndef _8259_H
#define _8259_H

#include "config.h"
#include "itypes.h"

#define NPORT_8059 0x02

#define ICW1 0
#define ICW2 1
#define ICW3 2
#define ICW4 3
#define OCW  4

#define FIRST  0
#define SECOND 1
#define THIRD  2

#define MODE_8086 1
#define MODE_8085 0

#define IR0 0
#define IR1 1
#define IR2 2
#define IR3 3
#define IR4 4
#define IR5 5
#define IR6 6
#define IR7 7

#define NON_SPECIFIC_EOI                0x01
#define SPECIFIC_EOI                    0x03
#define ROTATE_ON_NON_SPECIFIC_EOI      0x05
#define ROTATE_IN_AUTOMATIC_EOI_SET     0x04
#define ROTATE_IN_AUTOMATIC_EOI_CLEAR   0x00
#define ROTATE_ON_SPECIFIC_EOI          0x07
#define SET_PRIORITY_COMMAND            0x06
#define NO_OPERATION                    0x02

void InterruptRequest(int n);
byte inta(void);
void reset8259(void);
void write8259(bit a0,byte data);
byte read8259(bit a0);
ibool processInterrupt(byte *n,ibool test);

#endif
```

Appendice B2 – Esempio di circuito integrato simulato: PIC 8059 (main file).

```
#include "config.h"
#include "8088.h"
#include "8259.h"
#include <stdio.h>

ibool initialization=ICW1;
ibool rotate_on_automatic_eoi=false;
byte highest_interrupt=1;

union {
        byte v;
        struct {
                byte ir0 : 1;
                byte ir1 : 1;
                byte ir2 : 1;
                byte ir3 : 1;
                byte ir4 : 1;
                byte ir5 : 1;
                byte ir6 : 1;
                byte ir7 : 1;
         };
} irr,isr;

union {
        byte v;
        struct {
                byte ic4  : 1;
                byte sngl : 1;
                byte adi  : 1;
                byte ltim : 1;
                byte d4   : 1;
                byte a5   : 1;
                byte a6   : 1;
                byte a7   : 1;
         };
} icw1;

union {
        byte v;
        struct {
                byte a8    : 1;
                byte a9    : 1;
                byte a10   : 1;
                byte a11   : 1;
                byte a12   : 1;
                byte a13   : 1;
                byte a14   : 1;
                byte a15   : 1;
        };
        struct {
                byte d0    : 1;
                byte d1    : 1;
                byte d2    : 1;
                byte t3    : 1;
                byte t4    : 1;
                byte t5    : 1;
                byte t6    : 1;
                byte t7    : 1;
        };
```

```
} icw2;

union {
        byte v;
        struct {
                byte id0  : 1;
                byte id1  : 1;
                byte id2  : 1;
                byte d3   : 1;
                byte d4   : 1;
                byte d5   : 1;
                byte d6   : 1;
                byte d7   : 1;
         };
} icw3;

union {
        byte v;
        struct {
                byte upm  : 1;
                byte aeoi : 1;
                byte ms   : 1;
                byte buf  : 1;
                byte sfnm : 1;
                byte d5   : 1;
                byte d6   : 1;
                byte d7   : 1;
         };
} icw4;

union {
        byte v;
        struct {
                byte m0 : 1;
                byte m1 : 1;
                byte m2 : 1;
                byte m3 : 1;
                byte m4 : 1;
                byte m5 : 1;
                byte m6 : 1;
                byte m7 : 1;
         };
} ocw1;

union {
        byte v;
        struct {
                byte l0  : 1;
                byte l1  : 1;
                byte l2  : 1;
                byte d3  : 1;
                byte d4  : 1;
                byte eoi : 1;
                byte sl  : 1;
                byte r   : 1;
         };
} ocw2;

union {
        byte v;
        struct {
                byte ris  : 1;
```

```
                byte rr    : 1;
                byte p     : 1;
                byte d3    : 1;
                byte d4    : 1;
                byte smm   : 1;
                byte esmm  : 1;
                byte d7    : 1;
        };
} ocw3;

int getLog2(int v)
{
        int i;
        byte b=1;

        for (i=0;i<8;i++) {
                if (v & b) break;
                b <<= 1;
                        }
        return i;
}

ibool processInterrupt(byte *n,ibool test)
{
        byte ir=irr.v;
        byte b=highest_interrupt;
        int i;

        if (ocw3.smm && ocw3.esmm) ir &= ~ocw1.v;

        if (!ir) return false;

        for (i=0;i<8;i++) {
                if (isr.v & b) return false;
                if (ir & b) break;
                b <<= 1;
                if (!b) b=1;
                          }

        if (test) return true;

        *n=getLog2(b);

        if (!icw4.aeoi) isr.v |= b;
                        else
                        {
                        if   (rotate_on_automatic_eoi)   highest_interrupt=(b   ==
0x80?b=1:b << 1);
                         }
        irr.v &= ~b;
        return true;
}

void processOCW2(void)
{
        int cmd=(ocw2.v & 0xE0) >> 5;
        int l=ocw2.v & 0x07;
        int i;
        byte b;
         ibool rotate=false;

        switch(cmd) {
```

```
                case ROTATE_ON_NON_SPECIFIC_EOI:
                        rotate=true;
                case NON_SPECIFIC_EOI:
                        b=highest_interrupt;
                        for (i=0;i<8;i++) {
                                if (isr.v & b) break;
                                b <<= 1;
                                if (!b) b=1;
                                        }
                        if (i < 8) {
                                isr.v &= ~b;
                                if (rotate) highest_interrupt=(b == 0x80)?1:b <<
1;
                                   }
                        break;
                case ROTATE_ON_SPECIFIC_EOI:
                        rotate=true;
                case SPECIFIC_EOI:
                        b=1;
                        for (i=0;i<l;i++) b <<= 1;
                        isr.v &= ~b;
                        if (rotate) highest_interrupt=(b == 0x80)?1:b << 1;
                        //printf("[ISR: %d IRR: %d]",isr.v,irr.v);
                        break;
                case ROTATE_IN_AUTOMATIC_EOI_SET:
                        rotate_on_automatic_eoi=true;
                        break;
                case ROTATE_IN_AUTOMATIC_EOI_CLEAR:
                        rotate_on_automatic_eoi=false;
                        break;
                case SET_PRIORITY_COMMAND:
                        b=1;
                        for (i=0;i<l;i++) b <<= 1;
                        highest_interrupt=(b == 0x80?1:b << 1);
                        break;
                case NO_OPERATION:
                        break;
        }
}

void write8259(bit a0,byte data)
{
        if (!a0 && (data & 0x10)) {// && (initialization == ICW1 ||
initialization == OCW)) {
                        //printf("[ICW1]");
                        initialization=ICW2;
                        icw1.v=data;
                        return;
                           }
        if (a0 && initialization == ICW2) {
                        //printf("[ICW2]");
                        initialization=ICW3;
                        icw2.v=data;
                        return;
                           }
        if (a0 && initialization == ICW3) {
                        //printf("[ICW3]");
                        if       (icw1.ic4)       initialization=ICW4;       else
initialization=OCW;
                        icw3.v=data;
                        return;
                           }
```

```
        if (a0 && initialization == ICW4) {
                        //printf("[ICW4]");
                        initialization=OCW;
                        icw4.v=data;
                        return;
                           }
        if (a0 && initialization == OCW) {
                                        //printf("[OCW1]");
                                        ocw1.v=data;
                                        return;
                                           }
        if (!a0 && !(data & 0x10) && !(data & 0x08) && initialization == OCW) {
                                                                //printf("[OCW2]");
                                                                ocw2.v=data;
                                                                processOCW2();
                                                                return;
                                                                     }
        if (!a0 && !(data & 0x10) && (data & 0x08) && initialization == OCW) {

        //printf("[OCW3]");

        ocw3.v=data;
                                                                        return;
                                                                               }
}

byte read8259(bit a0)
{
        byte r;

        if (ocw3.p) {
                ocw3.rr=true;
                ocw3.ris=false;
                ocw3.p=false;
                if (processInterrupt(&r,false)) r |= 0x8000;
                return r;
                    }
        if (!a0 && ocw3.rr && !ocw3.ris) return irr.v;
        if (!a0 && ocw3.rr && ocw3.ris) return isr.v;
        if (a0) return ocw1.v;                          // imr
        return 0;
}

void reset8259(void)
{
        initialization=ICW1;
        icw1.v=0;
        icw2.v=0;
        icw3.v=0;
        icw4.v=0;
        ocw1.v=0;
        ocw2.v=0;
        ocw3.v=0;
      ocw3.rr=true;
        isr.v=0;
        irr.v=0;
}

void convertInterval4(byte *r)
{
        switch(*r) {
                case IR0:
```

```
                        *r=0x1c;
                         break;
                case IR1:
                        *r=0x18;
                        break;
                case IR2:
                        *r=0x14;
                        break;
                case IR3:
                        *r=0x10;
                        break;
                case IR4:
                        *r=0x0c;
                        break;
                case IR5:
                        *r=0x08;
                        break;
                case IR6:
                        *r=0x04;
                        break;
                case IR7:
                        *r=0x00;
                        break;
        }
}

void convertInterval8(byte *r)
{
        switch(*r) {
                case IR0:
                        *r=0x38;
                        break;
                case IR1:
                        *r=0x30;
                        break;
                case IR2:
                        *r=0x28;
                        break;
                case IR3:
                        *r=0x20;
                        break;
                case IR4:
                        *r=0x18;
                        break;
                case IR5:
                        *r=0x10;
                        break;
                case IR6:
                        *r=0x08;
                        break;
                case IR7:
                        *r=0x00;
                        break;
        }
}

byte inta(void)
{
        static int i=FIRST;
        byte r;

      if (initialization != OCW) return 0;
```

```
      if (icw1.ic4 && icw4.upm == MODE_8086) {
                                if (i == FIRST) {       i=SECOND; return 0; }
                                if (i == SECOND) {
                                i=FIRST;
                                        processInterrupt(&r,false);
                                        return (icw2.v & 0xf8) | r;
                                                }
                                        }
                                        else
                                        {
                                if (i == FIRST) {
                                         i=SECOND; return 0xcd; }
                                if (i == SECOND) {
                                        i=THIRD;
                                        processInterrupt(&r,false);
                                        if  (icw1.adi)  convertInterval4(&r);  else
convertInterval8(&r);
                                        return   icw1.adi?((icw1.v   &   0xe0)   |
r):((icw1.v & 0xc0) | r);
                                                }
                                if (i == THIRD) {
                                        i=FIRST;
                                        return icw2.v;
                                                }
                                        }
        return 0;
}

void InterruptRequest(int n)
{
        int b,ir;
        int i;

        //printf("[IR %d]",n);

        if (n < 0 || n > 7) return;

        ir=1;
        for (i=0;i<n;i++) ir <<= 1;

    irr.v |= ir;
}
```

ISSN 1590-2595

Quaderni di Geofisica

http://istituto.ingv.it/it/le-collane-editoriali-ingv/quaderni-di-geofisica.html

I Quaderni di Geofisica coprono tutti i campi disciplinari sviluppati all'interno dell'INGV, dando particolare risalto alla pubblicazione di dati, misure, osservazioni e loro elaborazioni anche preliminari, che per tipologia e dettaglio necessitano di una rapida diffusione nella comunità scientifica nazionale ed internazionale. La pubblicazione on-line fornisce accesso immediato a tutti i possibili utenti. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-7941

rapporti tecnici INGV

http://istituto.ingv.it/it/le-collane-editoriali-ingv/rapporti-tecnici-ingv.html

I Rapporti Tecnici INGV pubblicano contributi, sia in italiano che in inglese, di tipo tecnologico e di rilevante interesse tecnico-scientifico per gli ambiti disciplinari propri dell'INGV. La collana Rapporti Tecnici INGV pubblica esclusivamente on-line per garantire agli autori rapidità di diffusione e agli utenti accesso immediato ai dati pubblicati. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-6651

miscellanea INGV

http://istituto.ingv.it/it/le-collane-editoriali-ingv/miscellanea-ingv.html

La collana Miscellanea INGV nasce con l'intento di favorire la pubblicazione di contributi scientifici riguardanti le attività svolte dall'INGV (sismologia, vulcanologia, geologia, geomagnetismo, geochimica, aeronomia e innovazione tecnologica). In particolare, la collana Miscellanea INGV raccoglie reports di progetti scientifici, proceedings di convegni, manuali, monografie di rilevante interesse, raccolte di articoli, ecc.

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