UNIVERSITÀ DEGLI STUDI DI FERRARA

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI

Tesi di Laurea in Informatica

Progettazione e sviluppo del software di controllo qualità per il rivelatore di tracciamento dell'esperimento BABAR

Relatore: *Ch.mo Prof. Diego Bettoni*

Correlatore: *Prof.ssa Eleonora Luppi*

Laureando: *Luca Boni*

ANNO ACCADEMICO 2004/2005

Mentre invecchiamo

Si fa più strano il mondo, più complicato lo schema

Del morto e vivo. Non il momento intenso

Isolato, senza prima né dopo,

Ma in ogni momento una vita intera che brucia.

T. S. Eliot, *East Coker*

Indice

Introduzione

In queste pagine verrà mostrato il lavoro da me svolto presso il Dipartimento di Fisica dell'Università di Ferrara, sede di una sezione dell'INFN (Istituto Nazionale di Fisica Nucleare) nell'arco, approssimativamente, di sette mesi (da agosto 2003 a febbraio 2004) durante i quali ho progettato e sviluppato il software per il sistema di controllo di qualità dei rivelatori di tracciamento dell'esperimento BaBar.
Sebbene gran parte del tempo lo abbia passato nel piccolo laboratorio di via delle Scienze, non sono mancate occasioni per interessanti "trasferte" insieme a simpatici dottorandi che mi hanno accompagnato per l'intero periodo di tesi. In particolare, ho trascorso una settimana presso l'Università La Sapienza di Roma dove sono stato iniziato alle "meraviglie" dell'ambiente di sviluppo LabVIEW e alla prima definizione del lavoro che si sarebbe portato avanti nei mesi a seguire. Con l'arrivo dell'autunno si sono susseguiti diversi viaggi a Carsoli[1], in provincia dell'Aquila, sede della ditta Pol.Hi.Tech, incaricata della produzione dei rivelatori per l'esperimento BaBar. Qui è stata installata l'intera infrastruttura informatica (e non) sperimentata precedentemente a Ferrara insieme ad alcune apparecchiature messe a disposizione da Roma e dall'Università di Padova, anch'esse coinvolte nel progetto. Proprio Padova mi ha fornito un software sperimentale scritto in linguaggio C che ho ampiamente "rimaneggiato" per renderlo adatto alle nostre necessità (si veda §3.6).
I contributi al mio lavoro sono comunque pervenuti in tanti altri modi, ad esempio attraverso alcune discussioni tenute sulla mailing list creata per l'occasione, pratico mezzo per tenere in contatto un ampio numero di persone coinvolte in un medesimo progetto, e molto più spesso confrontandomi di persona con i simpatici dottorandi di cui sopra.

[1] Abitanti 5085, altitudine 616 metri. Significativo patrimonio naturale rappresentato dalle Grotte di Pietrasecca e del Cervo, di straordinaria bellezza e dal 1992 riserva naturale.

Capitolo 1

BaBar e LST

1.1 Introduzione

Nelle prossime pagine verrà presentato l'esperimento dal quale è nato il *progetto LST*, a cui ho preso parte sviluppando l'infrastruttura software per il sistema di controllo qualità. Poiché i concetti di fisica delle particelle elementari sono piuttosto complessi e una loro esaustiva trattazione esulerebbe dallo scopo del presente testo, l'esposizione sarà rigorosamente "per non addetti ai lavori".

1.2 L'esperimento BaBar

Figura 1.1. L'elefantino Babar è nato nel 1930 come personaggio di un libro illustrato per bambini, oggi logo dell'esperimento.

Scopo dell'esperimento è la *misura diretta degli effetti di violazione di CP attraverso lo studio dei decadimenti dei mesoni B*, processo misterioso che i fisici pensano possa essere la chiave per spiegare perché l'universo è composto quasi esclusivamente di materia.
Uno dei principali aspetti tuttora non compresi della fisica delle particelle è la misteriosa scomparsa dell'antimateria dall'universo primordiale. Lo spazio e il tempo sono nati circa 15 miliardi di anni fa dal *Big Bang*, un'enorme esplosione che ha creato materia e

antimateria in quantità esattamente uguali[2]. Con il tempo, tuttavia, la maggior parte dell'antimateria è scomparsa lasciando un universo fatto interamente di materia.
I fisici credono oggi di aver elaborato delle teorie in grado di spiegare, benché solo parzialmente, la scomparsa dell'antimateria e sono impegnati in svariati esperimenti che si spera aiuteranno a comprendere cosa è successo all'antimateria mancante. Tra i più importanti vi è proprio BaBar, un grande esperimento internazionale che usa un acceleratore di particelle chiamato *fabbrica di B*, situato allo *Stanford Linear Accelerator Center* (SLAC).
Esperimenti collegati a questo saranno il CESR, realizzato a Cornell, U.S.A. e il KEK in Giappone.

Il nome dell'esperimento deriva dal fatto che in esso sono prodotte particelle di materia e antimateria chiamate rispettivamente mesoni B e Bbar.[2]
Per ogni particella (*materia*) c'è la corrispondente antiparticella (*antimateria*), identica alla sua particella sotto ogni aspetto tranne che per la carica, che è opposta. Per esempio il protone ha carica elettrica positiva, e l'antiprotone ha carica elettrica negativa; ma hanno la stessa identica massa, perciò sono soggetti alla gravità nella stessa identica maniera.
Gli acceleratori hanno una duplice funzione. Primo, dato che tutte le particelle si comportano come onde, gli acceleratori permettono di aumentare la quantità di moto di una particella, facendo diminuire così la sua lunghezza d'onda al punto da poterla costringere fin dentro un atomo. Secondo, l'energia delle particelle accelerate può venire usata per creare le particelle massive che si vogliono osservare. Di base, un acceleratore prende una particella, le fa guadagnare velocità grazie a forti campi elettromagnetici,

[2] Il modello del Big Bang, denominazione introdotta con senso dispregiativo nel 1950 da Fred Hoyle, uno degli autori della rivale *teoria dello stato stazionario*, teorizza un universo creato dal nulla in un istante a distanza finita nel passato, che si è espanso diventando via via sempre più freddo e rarefatto, fino al suo attuale stato di quiete. Nel 1948 due giovani ricercatori americani previdero che, se l'immagine del big bang di un passato caldo e denso era corretta, allora doveva essere rimasta qualche prova fossile di quell'ardente passato, una sorta di "eco" sotto forma di radiazione di fondo che si è raffreddata con l'espansione dell'universo. Nel 1965, Arno Penzias e Robert Wilson si imbatterono per caso in questo campo di radiazione mentre mettevano a punto un radioricevitore progettato per seguire satelliti artificiali; la radiazione aveva una temperatura di tre gradi al di sopra dello zero assoluto (cioè circa -270°C), quasi esattamente quella prevista dalla teoria diciassette anni prima. Non c'è alcun modo di produrre localmente, nell'universo, una radiazione avente queste caratteristiche. Ulteriori conferme alla teoria del big bang sono venute dalla corretta previsione dell'abbondanza cosmica di elio, deuterio e litio, tutti elementi prodotti dalle reazioni nucleari durante i primi tre minuti di espansione successivi al big bang (in proposito Steven Weinberg, premio Nobel della Fisica nel 1979, ha scritto il bel libro *I primi tre minuti*, Mondadori, Milano, 1986).[1]

Figura 1.2. L'acceleratore lineare di Stanford si trova subito a sud di San Francisco, in California, U.S.A. ed è l'acceleratore lineare più lungo al mondo: misura 2 miglia.

e la scaglia contro un bersaglio attorno al quale ci sono rivelatori che registrano le fasi fondamentali dell'evento. Nell'urto, a partire da fotoni o gluoni virtuali, nasceranno coppie di particelle e antiparticelle. Per separare le une dalle altre si adoperano campi magnetici.[3]

L'acceleratore PEP II di SLAC fa collidere elettroni con la loro controparte di anti-materia, i positroni, in modo da produrre coppie di particelle e anti-particelle pesanti esotiche, chiamate mesoni B e Bbar. Queste rare forme di materia e antimateria hanno una vita breve (circa un picosecondo, ovvero 10^{-12}secondi) e decadono a loro volta in altre particelle subatomiche più leggere. Osservando i decadimenti di più di 200 milioni di coppie di B e Bbar, i ricercatori hanno scoperto una forte asimmetria fra materia e antimateria. Anche se BaBar e altri esperimenti avevano già osservato in precedenza asimmetrie fra materia e antimateria, questo è il primo caso di una differenza ottenuta semplicemente contando il numero di decadimenti dei mesoni B, un fenomeno chiamato violazione di CP (simmetria di carica e parità) diretta.[4]

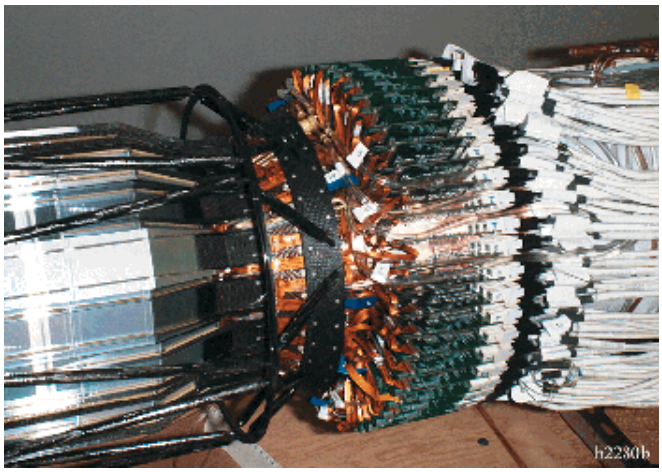

Figura 1.3. A sinistra, il rivelatore per lo studio delle collisioni tra elettroni e positroni, a destra i cavi di segnale.

Secondo il *Modello Standard*[3], la violazione di simmetria di CP è possibile, oltre che per i decadimenti dei mesoni[4] K, anche per i mesoni B. La scoperta di un simile effetto porterebbe all'evidenza di una seconda differenza tra materia e antimateria, ma ancor più interessante sarebbe l'eventualità che la misura di violazione di CP nei decadimenti di B non potesse essere spiegata completamente all'interno del modello ma dovesse richiedere un'estensione oggi non prevista, segnale di "nuova fisica".

I recenti risultati dell'esperimento BaBar sono riportati a pagina xxx.

[3] Teoria fisica elaborata negli anni '60 e presentata per la prima volta in un'elaborazione unitaria, nel 1974, da John Iliopoulos. Si propone di descrivere sia la materia che tutte le forze dell'universo, esclusa la gravità. La sua bellezza sta nella capacità di spiegare centinaia di particelle e interazioni complesse con poche particelle e interazioni fondamentali. In particolare, secondo tale modello, ogni tipo di interazione fondamentale agisce "mediante" una particella mediatrice di forza (ad esempio il fotone) e considera tutte le particelle materiali finora conosciute composte da particelle più fondamentali, i quark e i leptoni. Ci sono sei tipi di leptoni, di cui tre con carica negativa (l'elettrone, il muone e il tau) e tre privi di carica elettrica (i neutrini) più i corrispettivi di antimateria, gli *antileptoni*, con massa uguale e carica opposta. Per quanto riguarda i quark, ce ne sono sei, ma normalmente i fisici li raggruppano in tre coppie: Up/Down, Charm/Strange e Top/Bottom e i corrispondenti quark di antimateria (*antiquark*).

[4] I quark vivono solo insieme ad altri quark, in gruppi che formano particelle composte dette *adroni*, di cui esistono due classi: i *barioni*, composti da 3 quark, e i *mesoni* composti da un quark e un antiquark. Una nota curiosa: la massa di un adrone non è interamente dovuta alla somma delle masse dei quark che lo compongono! Infatti gran parte della massa dell'adrone deriva dall'energia cinetica e potenziale del sistema. La legge fisica più conosciuta della storia, $E = mc^2$ di Albert Einstein, ci ricorda l'equivalenza tra massa ed energia; nel caso degli adroni, l'energia si manifesta nel sistema come massa.

1.3 L'Italia e BaBar

I gruppi italiani hanno contribuito in modo notevole e visibile all'esperimento BaBar durante la fase di costruzione e continuano a farlo in quella attuale di presa dati e analisi. Si tratta della componente nazionale più importante, dopo quella statunitense: partecipano dodici gruppi che operano nelle sezioni INFN e nei Dipartimenti di Fisica di Bari, Ferrara, Genova, Milano, Napoli, Padova, Pavia, Pisa, Roma, Torino, Trieste e nei laboratori nazionali di Frascati. Le altre componenti provengono da Canada, Cina, Francia, Germania, Gran Bretagna, Norvegia e Russia. Un contributo specifico molto significativo per l'INFN e l'industria italiana è costituito dal solenoide superconduttore del rivelatore e dalla relativa criogenia. Il magnete è stato costruito dall'ANSALDO di Genova su progetto del gruppo superconduttività della sezione INFN di Genova e sotto il controllo di esperti dell'esperimento BaBar.

1.4 Il progetto LST - Limited Streamer Tube

Un componente essenziale dell'apparato rivelatore dell'esperimento BaBar, che prende il nome di *IFR* (Instrumented Flux Return), si compone di tre sezioni: una centrale (*barrel*) e due tappi laterali (*endcaps forward - backward*) e rappresenta la parte più esterna del rivelatore, là dove giungono le particelle maggiormente energetiche e penetranti. Questa parte del rivelatore è ottimizzata per la rivelazione di muoni e particelle adroniche neutre e sfrutta rivelatori di traccia del tipo *RPC* (Resistivity Plate Chamber). Alcuni problemi legati ad un notevole e precoce effetto di *invecchiamento* hanno portato, nel corso dei primi tre *run*, ad un costante calo di efficienza sino a valori di 0.4 nel barrel e 0.7 nel forward, con una perdita mensile pari al 2.54% nel primo caso e 1.21% nel secondo. Questi motivi hanno indotto ad un primo intervento di ricostruzione della parte forward in seguito al quale l'efficienza si è mantenuta intorno ad un valore di circa 0.9. La parte centrale dell'IFR, il barrel, sarà invece ricostruita in due fasi utilizzando rivelatori di traccia di tipo *LST* (Limited Streamer Tube): fra agosto e settembre 2004 sono stati sostituiti due sestanti, quello superiore e quello inferiore, mentre l'intervento sui restanti quattro avverrà nel corso dello spegnimento (*shutdown*) del rivelatore nell'anno 2005 al termine del quinto run.[5]

In seguito all'approvazione del progetto di aggiornamento del barrel, avvenuto nel maggio 2003, la produzione delle prime camere LST è iniziata nel novembre dello stesso

anno, affidata alla ditta Pol.Hi.Tech. di Carsoli (AQ) e si è conclusa nel giugno 2004 per un numero totale di camere prodotte pari a 1418 unità.

1.4.1 Struttura degli LST prodotti

Il tubo a streamer è un rivelatore economico che offre, nella sua versione tradizionale, una efficienza del 90%. L'efficienza è limitata da fattori geometrici e costruttivi: spessore delle pareti, spaziatori e altri parametri costruttivi che possono essere ottimizzati in base alle esigenze di utilizzo.
Questo tipo di rivelatore è stato impiegato con successo in una dozzina di esperimenti per la fisica delle alte energie. Uno dei vantaggi fondamentali riguarda la resistenza alla carica assorbita. Test di laboratorio hanno dimostrato che per valori di carica assorbita di quasi 1 C/cm^2 non si osservano significativi effetti di degrado nelle prestazioni. Due sono i problemi riscontrati con l'utilizzo degli LST: l'alta mortalità infantile e l'innesco di scariche autosostenute, in corrispondenza di eventuali difetti nella lavorazione della superficie.

Il profilo e la camicia esterna degli LST prodotti per BaBar sono realizzati in PVC e stampati per estrusione. Il profilo è prodotto in due differenti tipi: otto celle di lunghezza 358 e 318 cm e sette celle di lunghezza 358 cm, mentre la sezione delle celle vale 15x17 mm^2. Nel seguito con "lungo" e "corto" ci si riferirà a queste due lunghezze. Utilizzando questi tre formati, vengono assemblati moduli in quattro diverse combinazioni tali da obbedire ai requisiti geometrici imposti dalle dimensioni dei layers dell'IFR. Il numero di tubi prodotti per le differenti dimensioni sono riassunti in tabella.

LUNGHEZZA	**NUMERO CELLE**	**QUANTITÀ**
358 cm	8	972
318 cm	8	150
358 cm	7	257

Tabella 1.1. Tipi di camere LST prodotte.

La superficie interna dei profili è stata grafitata con l'utilizzo di una sospensione colloidale di carbone micronizzato in una miscela di acqua e polimero a basso peso molecolare. I test di resistività effettuati su larga scala rivelano una resistività superficiale media variabile da 100 $K\Omega$ a 800 $K\Omega$.

Ciascuna cella è delimitata da una parete in PVC avente spessore di 1 mm, mentre alle estremità i profili sono smussati e sagomati in modo da poter supportare il cavaliere reggifilo (*PCB holder*), sul quale sono montate le schede per la saldatura dei fili ed il collegamento con le boccole per l'High Voltage (HV) e la massa. Fra *PCB holder* (stampato in polietilene rigido) e profilo, viene posta una gomma conduttiva al fine di ottimizzare il contatto elettrico con il substrato in grafite sul fondo. Il fissaggio della gomma è assicurato da un'astina che si incastra sul *PCB holder*. Alle estremità il profilo è chiuso da due tappi, (*endcap A* e *B*). Sull'*endcap A* sono presenti quattro boccole per la connessione all'HV, un jack per la massa e due connettori per consentire il flusso della miscela gassosa, mentre l'*endcap B* presenta unicamente le sedi per i connettori del sistema del gas.

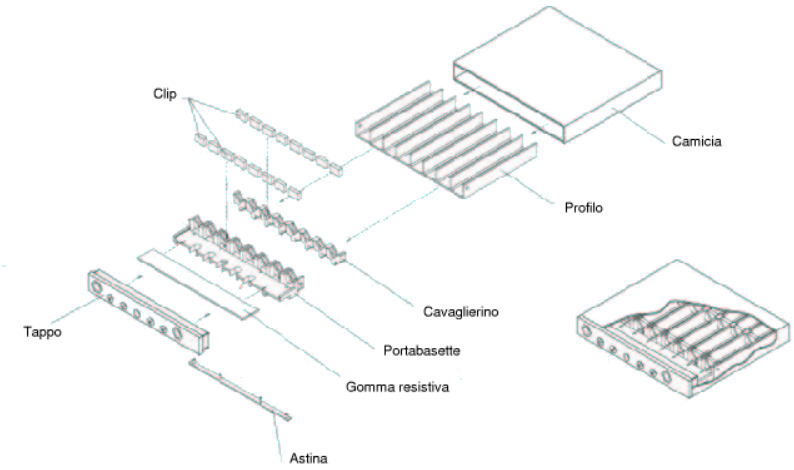

Figura 1.4. Struttura di una camera LST in dettaglio.

Lungo il profilo, separati da una distanza di 40 cm, sono disposti dei cavalieri reggifilo in polietilene, i *wireholder*. Su ciascuno di essi, in corrispondenza della singola cella, è ricavato un alloggiamento dentro il quale va innestata una *clip*. Il filo è vincolato a rimanere fra *wire holder* e *clip* attraverso l'intercapedine prodotta dall'unione dei due elementi. La filatura viene eseguita da una macchina che dispone di otto bobine, una per cella, e utilizza un filo in lega berillio rame (2:1) con un flash di nichel (trattamento con plasma di Ni) e copertura superficiale di 1 μm di oro e cobalto (1:0.25). Il diametro del filo è pari a 100 ± 1 μm ed al momento della filatura viene applicata una tensione di 250 grammi.

In seguito alla filatura il profilo viene pulito mediante l'aspirazione dei possibili residui

polverulenti, inserito all'interno d'una camicia in PVC chiusa alle estremità dai due *endcap A* e *B*, quindi sigillato con mastice. Quest'utlima operazione è necessaria poiché la camera verrà riempita con una miscela di gas ternario composto da Argon (3%), Isobutano (8%) e CO_2 (89%) consentendone l'accensione e la messa in funzione.

1.4.2 Il sistema produttivo e di controllo qualità (QC)

Studi eseguiti su prototipi, finalizzati all'analisi del design delle camere LST, hanno messo alla luce che la qualità con la quale sarebbero state costruite avrebbe inciso considerevolmente sulle prestazioni dell'intero rivelatore di BaBar. Il sistema di *QC* (quality control) alla Pol.Hi.Tech è quindi principalmente rivolto a monitorare ogni passo del processo di produzione, includendo controlli sulle proprietà meccaniche ed elettriche dei tubi, sulla loro tenuta stagna e il loro funzionamento. A tutto questo è aggiunto un *test di lungo termine* (si veda §3.7) per assicurare l'affidabilità delle camere nel tempo.

Gli obiettivi del *QC* possono essere riassunti come segue:

1. Effettuare una verifica sui materiali durante la fase di assemblamento per rivelare eventuali discrepanze con gli standard qualitativi previsti.
2. Applicare un controllo durante le fasi di produzione così da poter mettere in luce eventuali problemi ed applicare interventi retroattivi di miglioramento.
3. Controllare il singolo rivelatore alla fine del processo produttivo, prima della sua definitiva accettazione e spedizione negli U.S.A.
4. Popolare un database con la storia dettagliata di ogni singolo rivelatore servendosi dei risultati dei test effettuati.

L'organizzazione dei test ed il numero di LST trattati è stato stabilito in modo da evitare accumuli ottimizzando la durata del test in funzione del rate di produzione giornaliero (15 camere al giorno). Le procedure di controllo sono state per quanto possibile automatizzate, creando interfacce di facile utilizzo così da poter essere usate dal personale non specializzato operante al sito di produzione. Per ottemperare ai compiti specialistici, da due a tre persone appartenenti alla collaborazione BaBar sono state regolarmente presenti. Il sito produttivo è stato munito di una rete locale (LAN) di tipo *wireless* per consentire la comunicazione tra le stazioni di test (DAQ) e il server centrale, a sua volta connesso alla rete esterna tramite una linea telefonica di tipo ADSL.

In corrispondenza d'ogni tappa del *QC*, le DAQ producevano file in formato testo nei quali venivano incluse le grandezze di rilievo relative al test condotto (chiamati d'ora in poi *file di log*). Al termine di ogni test, le DAQ trasmettevano i propri file ad un server centrale che memorizzava queste informazioni nel database locale di produzione. A fine giornata tutti i dati venivano trasferiti su un server esterno per essere importati nel database ufficiale di LST.

Nel seguito sono elencate in dettaglio le fasi che hanno scandito il processo produttivo. Le lettere maiuscole comprese tra parentesi quadre indicano le operazioni corrispondenti a misure necessarie al *QC*.

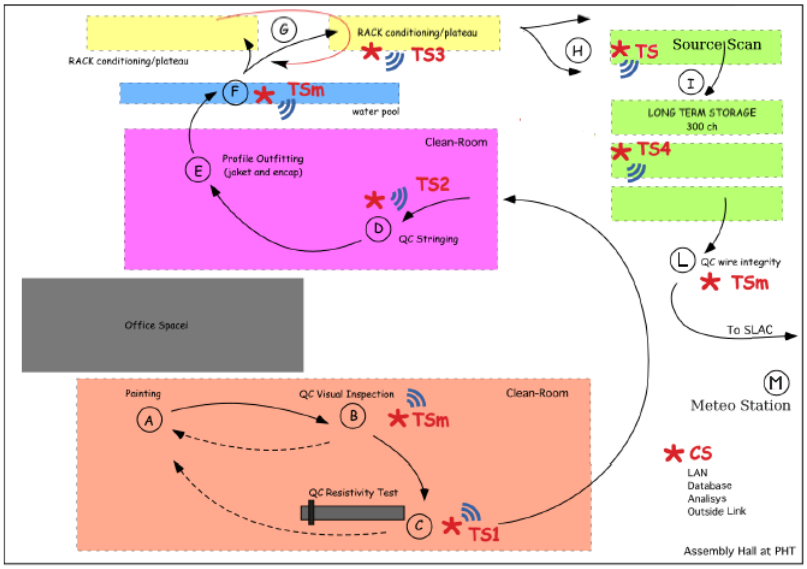

Figura 1.5. Catena produttiva e di QC presso la Pol.Hi.Tech.

1. Controllo dell'integrità meccanica dei profili e dei componenti plastici (*clips*, *wire holders*, *PCB holders* ed *endcap*) prima dell'assemblamento. Gli *endcap* sono stampati in ABS caricata con fibre di vetro, tutte le altre *small parts* in polietilene.
2. Il profilo viene grafitato nella camera pulita e successivamente sottoposto ad *ispezione visiva* [B] quindi al test di *resistività superficiale sul fondo e sulle alette* [C].
3. In questa fase sono assemblati i componenti e le schede sulle quali vengono saldati i fili con una tensione di 250 grammi. *I fili sono identificati da un numero relativo alla bobina dalla quale vengono svolti* [D]. Per ciascuna di queste, un campione è stato inviato al *Princeton Image and Analisys Center* per un test di controllo al microscopio

elettronico. La parte più interna della bobina è stata scartata.

4. Il tubo viene definitivamente assemblato: il profilo in seguito alla filatura è inserito nell'involucro in PVC, quindi, dopo aver applicato le connessioni di HV e massa, viene chiuso all'estremità dagli *endcap* e sigillato.
5. I tubi escono dalla camera pulita e vengono provati per verificare eventuali perdite di gas in corrispondenza delle giunzioni degli *endcap*. A tal fine il tubo è gonfiato con una sovrapressione di 15-20 *mbar* d'aria ed immerso in una vasca d'acqua [F]. In caso di perdita, il limite tollerato è di una bolla ogni minuto.
6. *Condizionamento* [G]: la durata del test è di circa sette giorni e per assorbire il rate di produzione giornaliero, risulta necessaria una postazione in grado di ospitare almeno 75 camere. Per evitare accumuli e tenendo conto dell'eventualità che per alcuni tubi siano necessari due o più cicli di *Conditioning*, è stata allestita una postazione in grado di condizionare contemporaneamente 132 camere.
7. *Plateau* [G]: viene effettuato contemporaneamente su 11 camere, effettuando acquisizioni di conteggi ogni 20 secondi con incrementi di tensione di 50 Volt.
8. *Scan con sorgente radioattiva* [H]: i tubi che superano il test del Plateau vengono sottoposti alla scansione con sorgente collimata di Sr^{90} di intensità pari a 10 MBq. Lo scan viene effettuato per ciascuna cella sull'intera lunghezza del tubo. Il tavolo adibito a tale scopo permette di preparare per la scansione tre tubi alla volta, per i quali il test avviene in modo automatico. Il numero di tubi ispezionati giornalmente è comparabile, o leggermente superiore, al rate di produzione.
9. *Test di lungo periodo* [I]: la sua durata è di circa di un mese. Sono state allestite nove stazioni, ciascuna delle quali può ospitare sino a 48 camere. L'impianto complessivamente permette di effettuare il test su un totale di 432 camere.
10. *Misura di capacità* [L]: prima della spedizione di ciascun tubo negli U.S.A. viene eseguita una misura di capacità per verificare l'integrità delle connessioni e dei fili.

La deposizione della soluzione colloidale di carbone micronizzato (grafitatura), avviene all'interno d'una camera pulita costruita isolando una porzione del fabbricato con teli di nylon, entro i quali è creata una leggera sovrapressione che genera un flusso laminare discendente tale da impedire alla polvere di penetrare.
Inoltre, durante l'intero periodo di produzione, una stazione meteo [M] ha registrato ad intervalli di 30 minuti i valori di pressione, temperatura ed umidità locali.

In tutte le fasi di produzione è stato utilizzato un sistema di *codici a barre* (barcode), del tipo EAN13, pensato per semplificare e garantire il corretto inserimento dei dati nei programmi di test delle DAQ. In figura 1.6 ne viene mostrato il formato.

Figura 1.6. Codice a barre di tipo EAN13.

In pratica l'operatore, anziché inserire i dati ottenuti durante i test mediante la tradizionale tastiera del calcolatore, doveva leggere il barcode che rappresentava quel tipo di dato con un lettore di codici a barre. Il programma della DAQ, conoscendo il significato di ogni codice a barre, lo traduceva nel suo significato originale e lo memorizzava sui file di log. Ad esempio se l'operatore voleva terminare una sessione di test, invece di inserire un codice numerico o alfanumerico che doveva essere necessariamente prestabilito e ben noto, "cliccava" semplicemente sul codice a barre associato all'operazione di *Fine Sessione*. Per questo motivo ogni DAQ era fornita di un foglio A4 plastificato sul quale erano stampati tutti i barcode necessari a quel determinato test, limitando così al minimo l'interazione operatore-calcolatore e quindi la possibilità di errori o problemi di sorta. In figura 1.7 viene riportato un esempio di foglio A4 con possibili barcode associati alla DAQ del test di *Ispezione visiva della grafite*.

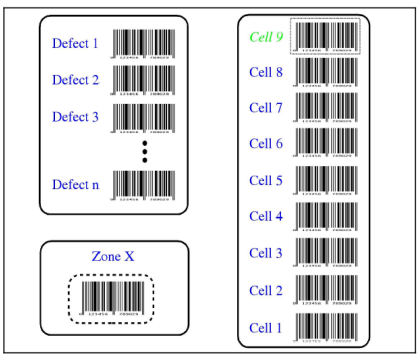

Figura 1.7. Esempio di foglio contenente i codici a barre per il test di *Visual graphite inspection*.

Per identificare il singolo LST durante tutte le sue fasi di vita, un barcode è stato stampato su etichette in due formati: uno da applicare al profilo e l'altro alla camicia esterna in corrispondenza dell'*endcap* dotato dei contatti per la connessione dell'HV. Il formato del barcode di un tubo e la nomenclatura completa dei codici EAN13 utilizzati dal *QC* è mostrata in tabella 1.2, dove:

Z = 6 - n zeri, dove n è il numero di caratteri dell'ID

ID = numero progressivo di identificazione tubo

cell = numero di celle del tubo

SIGNIFICATO	**IS** **(identificativo specie)**	**S** **(specie)**	**SS** **(sottospecie)**	**CS** **(checksum)**
Tubo	1	[Z][ID]	[cell]9999	random
Numero Cella	2	100001 - 100008	10000 - 90000	random
Tipo di Difetto	3	200001 - 200008	10000 - 30000	random
Lato Nord	5	100000	00000	2
Lato Sud	5	200000	00000	9
Bubbles	5	300000	00000	random
No Bubbles	5	400000	00000	random
Cambio Bobina	5	555555	55555	0
Numero Bobina	5	000001 - 000048	00000	random
Annulla	4	100000	00000	0
Fine Ispezione	4	200000	00000	0
Fine Sessione	4	300000	00000	7

Tabella 1.2. Nomenclatura completa dei codici EAN13.

Capitolo 2

LabVIEW

2.1 Introduzione

L'intero software di *quality control* è stato sviluppato impiegando LabVIEW, un ambiente di tipo visuale creato da National Instruments, fatta esclusione per la parte relativa alla comunicazione con i sistemi di High Voltage che è stata scritta in C. LabVIEW utilizza quello che National definisce un *linguaggio di programmazione grafico*, denominato G, al fine di creare programmi secondo una metodologia simile ad un diagramma a blocchi, particolarmente consueta in ambito tecnico-scientifico.
Nei prossimi paragrafi saranno approfondite le caratteristiche e i vantaggi del suo impiego, nonché una breve guida all'uso e all'installazione sul sistema operativo Linux.

2.2 L'ambiente di sviluppo visuale

La sigla LabVIEW sta per *Laboratory Virtual Instrument Engineering Workbench* cioè laboratorio di strumenti di lavoro virtuali, in quanto nel loro aspetto e nelle loro modalità operative imitano quelli reali. Uno *strumento virtuale* (VI, virtual instrument) può essere pensato come un software che interagisce con un hardware specifico, aggiunto ad un elaboratore, in modo tale che gli utenti possano interagire con l'elaboratore stesso considerandolo come se fosse uno strumento elettronico di tipo tradizionale. Quindi i VI permettono agli utenti di costruire il proprio sistema di strumentazione, utilizzando computer standard ed hardware di costo contenuto, per ottenere il controllo e l'analisi di fenomeni tecnico-scientifici.
LabVIEW può essere considerato un linguaggio general-purpose, nel quale la combinazione di oggetti sviluppati in proprio e provenienti dalle numerose librerie disponibili può risolvere problemi di ogni tipo. Tuttavia, LabVIEW dà il meglio di sé quando viene utilizzato per *acquisire ed elaborare dati sperimentali in tempo reale*, attraverso schede

di acquisizione della National o di altri produttori, interfacce seriali e GPIB (General Purpose Interface Bus). Per ciascuna di queste schede di acquisizione esistono appositi driver che nascondono la gran parte dei dettagli hardware, rendendo possibile per esempio la sostituzione di una scheda con una di nuova produzione senza modificare nulla del codice sviluppato, se non il driver di gestione.
Il concetto di programmazione è quello *data driven*, ovvero l'esecuzione non segue una sequenza predeterminata ma viene pilotato dai dati in arrivo. È fondamentale tenere in mente questo fin dall'inizio, per esempio nella rappresentazione grafica del programma in fase di sviluppo, in modo tale che i dati (e dunque l'esecuzione) scorrano per quanto possibile da sinistra a destra.
È da notare inoltre la presenza di potenti strumenti di temporizzazione che consentono di sincronizzare le operazioni all'interno di diverse "aree" del codice, e di efficaci metodi di debugging che consentono di seguire l'esecuzione, a partire dai dati di ingresso fino alla produzione dei risultati, in modo completamente visuale[5].
Una discreta collezione di esempi prefabbricati e configurabili sulle esigenze dell'utente consentono un rapido ingresso nella filosofia dell'ambiente di sviluppo.

2.3 Perché LabVIEW: vantaggi e scelte implementative

La facilità d'uso di LabVIEW e la semplificata manutenzione e riutilizzazione del software prodotto consentono una riduzione del tempo necessario allo sviluppo delle applicazioni, il che si traduce in un sostanziale aumento di produttività. LabVIEW, in questo modo, diventa lo strumento fondamentale per la progettazione e lo sviluppo di sistemi automatici di misura e di controllo.
Caratteristica rilevante di un approccio allo sviluppo software di questo tipo risiede nell'estrema semplicità della creazione di pannelli di interfaccia grafica con l'utente, consentendo il controllo interattivo del sistema in esame. LabVIEW possiede infatti le possibilità dei linguaggi standard, ma mentre questi ultimi sono di tipo *text-based*, esso è un linguaggio grafico; quest'aspetto è di grande aiuto per gli sviluppatori meno esperti in

[5] In realtà l'operazione di debug in LabVIEW, già di per se impegnativa, può diventare maggiormente difficoltosa proprio a causa delle caratteristiche visuali del linguaggio. È infatti difficile focalizzare il problema (bug) a livello concettuale se il programmatore viene eccessivamente distratto dalla necessità di seguire visivamente il fluire dell'esecuzione del programma e, più in generale, degli eventi associati. Questo è un chiaro esempio di come, spesso, le caratteristiche di maggior pregio di un sistema possano rappresentare il punto debole del sistema stesso.

programmazione di alto livello e contemporaneamente semplifica la vita nella creazione di front-end, il cui tempo di realizzazione è spesso elevato se confrontato a quello del programma "vero e proprio".
Si può comprendere quindi come il limite maggiore nello sviluppo software tramite questo ambiente risieda nelle caratteristiche intriseche del programma che si intende realizzare: tanto più questo si adatta ad un *linguaggio di tipo procedurale* e si allontana dalla semplice necessità di acquisire ed elaborare dati, tanto meno LabVIEW produrrà software efficiente e facile da debuggare.
Infine una nota critica potrebbe essere l'enorme impiego di risorse di cui necessita un'applicazione LabVIEW, se confrontata ad una sviluppata con linguaggi tradizionali di alto livello come il C++ o Java.

Nel caso specifico del sistema di QC, durante la fase di progettazione è risultata immediatamente chiara la necessità di avere una struttura informatica che fosse nel suo insieme semplice da usare, considerando che il personale addetto ad interagire con il software era privo di qualsiasi competenza informatica. Allo stesso tempo l'attività principale dei sistemi informatici adottati sarebbe stata quella di acquisire dati (e perciò d'ora in poi si parlerà di sistemi DAQ, cioè sistemi di acquisizione dati, intendendo con questo termine le singole postazioni dotate di un calcolatore fisso o portatile) attraverso strumenti di misurazione e/o con l'inserimento manuale diretto da parte dell'operatore. Infine, fattore di non minore importanza, il tempo a disposizione per lo sviluppo era preoccupantemente limitato.
La scelta più naturale, quindi, non poteva che ricadere su un sistema di sviluppo software come LabVIEW, in grado di soddisfare a pieno le necessità del progetto, eccezion fatta per un paio di sistemi di DAQ che soffrivano di una limitazione: la strumentazione che impiegavano non era direttamente supportata dal linguaggio LabVIEW ma richiedeva un software esterno che fosse in grado di farlo. Quest'ultimo era poi da integrare con quella parte, scritta comunque in LabVIEW, che permetteva all'operatore di interagire con il sistema e quindi acquisire le informazioni necessarie (si veda §3.6).

2.4 Tutorial

In questo paragrafo viene fornita una guida introduttiva all'uso dell'ambiente LabVIEW, descrivendone i componenti essenziali e i passi necessari per iniziare a sviluppare software.
Un programma LabVIEW è costituito da tre elementi principali: il pannello, il diagramma e l'icona/connettore.[5]

2.4.1 Il pannello

Permette di introdurre i valori di ingresso e vedere i risultati generati dal programma: rappresenta l'interfaccia utente. Esso altri non è che il pannello frontale di uno strumento (solo che in questo caso è virtuale) e come tale gli ingressi sono detti *controlli* e le uscite *indicatori*.

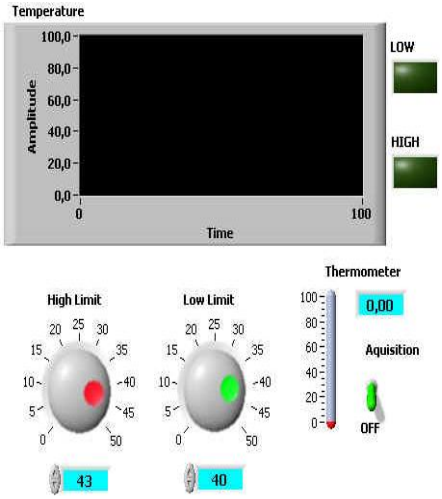

Figura 2.1. Esempio di front-panel con cui l'utente interagisce.

2.4.2 Il diagramma

Costituisce l'equivalente del codice di programmazione di un linguaggio ad alto livello e viene realizzato utilizzando il linguaggio grafico G. Per costruire il diagramma bisogna collegare tra loro, mediante dei fili (*wire*), le icone che rappresentano le funzioni di più basso livello e i sottoprogrammi.

Fanno parte del diagramma anche i *terminali* corrispondenti ai vari oggetti (controlli e indicatori) posizionati sul pannello frontale, consentendo il passaggio dei dati fra il pannello e il diagramma.
Uno o più elementi del diagramma possono essere inseriti all'interno di particolari strutture di controllo per condizionare l'esecuzione. Si tratta di cicli (*loop*), scelte (*case*) e sequenze (*sequence*).

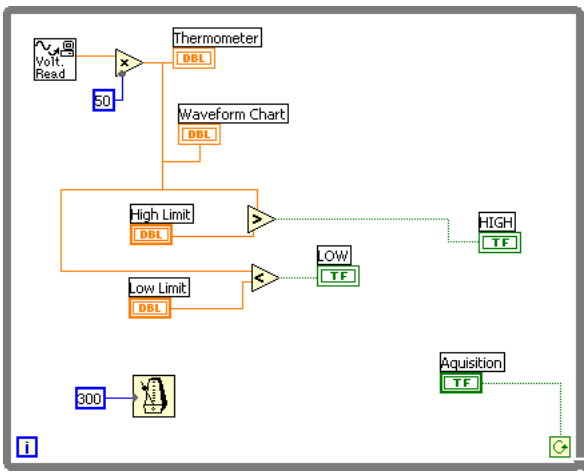

Figura 2.2. Il diagramma (codice di programmazione) relativo al pannello di figura 2.1.

2.4.3 L'icona/Connettore

Permette di visualizzare un VI come sottoprogramma all'interno di un altro VI. Mentre l'icona è solo un simbolo grafico che rappresenta il VI, il connettore dei terminali nascosto sotto l'icona permette di distinguere tutti i terminali, corrispondenti agli ingressi e alle uscite del sottoprogramma.

2.4.4 Lavorare con LabVIEW

All'avvio il programma presenta un'interfaccia tramite la quale è possibile aprire un VI salvato oppure iniziarne uno nuovo. Supponendo di aprire un nuovo VI (*New VI*), apparirà la finestra del pannello frontale sulla quale si distinguono la barra dei menù e quella degli strumenti.
Il programma presenta comandi in lingua inglese e non esiste al momento una versione in lingua italiana. La barra dei menù è come quella di un qualsiasi programma mentre quella degli strumenti contiene una serie di pulsanti che servono a control-

lare l'esecuzione dei VI, le opzioni per l'impostazione del testo e diversi comandi di allineamento.
Nel caso in cui non si ricordino con sicurezza le funzioni dei pulsanti, si può tenere il puntatore su ciascuno di essi fino a quando non compare la targhetta che ne indica la funzione (*Tool Tip Text*).
Se invece si apre un VI esistente, l'interfaccia che viene caricata per prima è sempre quella del pannello frontale, solo che questa volta esso non sarà vuoto ma riempito con i componenti che lo contraddistinguono.

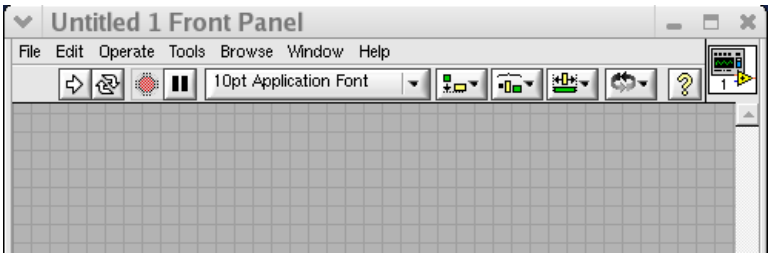

Figura 2.3. Aspetto dell'interfaccia di LabVIEW 7.

La palette degli strumenti

Per operare sui vari oggetti è necessario far apparire la palette degli strumenti selezionando *Window-Show Tools Palette*.

Figura 2.4. Struments Palette.

La palette dei controlli

La prima cosa da fare per costruire un VI è quella di definire cosa deve esserci sul pannello frontale. Per selezionare gli oggetti sul pannello serve la palette dei controlli che appare con il comando *Window-Show Controls Palette* oppure facendo click con il tasto destro del mouse in una zona libera del pannello.

Figura 2.5. Controls Palette.

La palette delle funzioni

Non appena un controllo o un indicatore viene posto sul pannello frontale, LabVIEW fa comparire automaticamente un terminale sul diagramma. Le elaborazioni che portano alla risoluzione del problema sono svolte dalle funzioni. Per selezionarle serve la palette delle funzioni che appare con il comando *Window-Show Function Palette* oppure con un click del pulsante destro del mouse in una zona libera del diagramma. Passando il puntatore sulle funzioni che si trovano sul diagramma, compare il loro nome e, se è stata attivata la voce *Help-Show Context Help*, anche la loro sintetica descrizione che può essere d'aiuto durante lo sviluppo del VI. Nella palette sono presenti anche le strutture (*Structures*) che permettono di eseguire le funzioni elaborative in modo subordinato al verificarsi di particolari condizioni, oppure ciclicamente proprio come accade nei linguaggi di programmazione tradizionali.

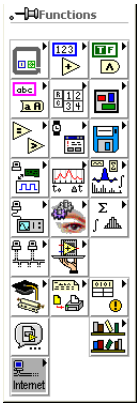

Figura 2.6. Functions Palette.

Collegamenti

Per realizzare il codice di programmazione bisogna tracciare dei collegamenti (*wires*) fra i terminali dei controlli e degli indicatori ed i terminali delle funzioni utilizzate. Essi assumono grafiche e colori diversi a seconda del tipo di dato che li attraversa.

2.4.5 Costruzione di un VI

Il modo di procedere è il seguente:

1. si apre un nuovo pannello
2. anche se non indispensabile, conviene visualizzare una sull'altra le finestre pannello e diagramma impostando il comando *Window-Tile Up and Down*
3. si inseriscono sul pannello i controlli e gli indicatori necessari: i relativi terminali compariranno sul diagramma
4. si introducono sul diagramma le funzioni necessarie
5. si eseguono i collegamenti utilizzando lo strumento *Collega* della *Tools Palette*. Quando lo strumento si trova su un componente, inizia a lampeggiare indicando che è il momento di effettuare la connessione. È possibile visualizzare i terminali di ciascun componente tramite l'uso di un menù a discesa (pop-up), cliccando con il pulsante destro sull'icona e selezionando *Visible-Terminals*.

Salvare lo strumento VI

Un VI può essere salvato come file singolo (.VI) oppure in gruppo costituente una libreria (.LLB). Per creare una nuova libreria, quando si salva la prima volta un VI, selezionare *File-Save* e poi cliccare sul pulsante *New VI Library*. Per eliminare i singoli VI da una libreria, si deve utilizzare *File-Edit VI Library*. Se si vuole salvare un VI come singolo file, selezionare *File-Save as*.

Editing degli oggetti

Per selezionare un oggetto usare lo strumento *Posiziona* cliccando su di esso. Se si vogliono selezionare più oggetti, premere contemporaneamente il tasto *Shift*.
Si possono spostare gli oggetti selezionati usando i tasti freccia; per vincolare la direzione di un oggetto selezionato in modo orizzontale o verticale, premere *Shift* mentre lo si sposta.

In caso di errore si possono cancellare gli oggetti (dopo averli selezionati) premendo il tasto *Canc* oppure dal menù *Edit-Clear*.
Se si vuole annullare l'ultima operazione eseguita, procedere con il menù *Edit-Undo*. Possono essere impostate il numero di azioni su cui effettuare l'*Undo* tramite il menù *Tool-Options-Block Diagram*.

Etichette

LabVIEW comprende due tipi di etichette: vincolate e libere. Quelle vincolate appartengono ad un determinato oggetto e si spostano con esso mentre quelle libere non sono collegate ad alcun oggetto e quindi possono essere mosse e cancellate in modo indipendente. Per creare un'etichetta libera si usa lo strumento *Testo*.

Collegamenti

Per selezionare un collegamento si utilizza lo strumento *Posiziona* cliccando una sola volta sul filo (*wire*). Se si clicca due volte si seleziona un ramo intero mentre se si clicca tre volte, l'intera connessione. Un collegamento tratteggiato rappresenta un collegamento errato. Per rimuovere tutti i collegamenti indesiderati usare *Edit-Remove Broken Wires*.

Font, stile e dimensioni testo

Utilizzando il menù a tendina che si trova sulla barra degli strumenti, qualsiasi tipo di testo può essere modificato.

Dimensioni oggetti

Per modificare le dimensioni degli oggetti si usano le maniglie di ridimensionamento, tramite lo strumento *Posiziona*. Quando si ridimensiona, le dimensioni del font rimangono inalterate. Se si vogliono mantenere le proporzioni dell'oggetto durante il ridimensionamento, premere il tasto *Shift*. Per allineare un gruppo di oggetti lungo gli assi, usare il menù a tendina *Align Objects* sulla barra degli strumenti dopo averli selezionati. Se invece si vogliono spaziare uniformemente gli oggetti, utilizzare dal menù a tendina *Distribute Objects*. Infine gli oggetti possono essere copiati e incollati, dopo averli selezionati, con il menù *Edit-Copy* e *Edit-Paste*.

Colore

Si possono cambiare i colori di molti oggetti (non di tutti) utilizzando lo strumento *Colora* e cliccando con il pulsante destro del mouse sull'oggetto. Inoltre possono essere modificati i colori di default della maggior parte degli oggetti, selezionando *Tools-Options-Colors* dal menù a tendina. Infine per rendere trasparenti gli oggetti del pannello frontale, cliccare con il tasto destro su uno di essi mediante lo strumento *Colora* e selezionare il riquadro con una T all'interno.

2.4.6 Creazione di un subVI

Un subVI corrisponde ad una subroutine dei linguaggi di programmazione testuale. Gli accorgimenti per realizzare un subVI sono:

Creare l'icona

Essa è una rappresentazione grafica del VI. Per default contiene un numero che indica quanti nuovi VI sono stati aperti all'avvio di LabVIEW. Cliccare con il tasto destro nell'angolo superiore destro del pannello frontale, o del diagramma, e selezionare *Edit Icon* dal menù pop-up. Utilizzando gli strumenti di editing messi a disposizione si può creare un'icona personalizzata.

Figura 2.7. L'icona di default di un subVI.

Definire il connettore

Per utilizzare un VI come subVI occorre definire i terminali corrispondenti ai controlli e agli indicatori. Il loro numero dipende da quello degli indicatori e dei controlli che compaiono sul pannello frontale. Cliccando con il tasto destro sull'icona in alto a destra del pannello e selezionando *Show Connector*, il connettore sostituisce l'icona.
Si può modificare lo schema dei terminali cliccandone il riquadro e selezionando *Patterns*. Per modificare la disposizione spaziale degli schemi dei riquadri dei connettori, cliccare con il tasto destro sul loro riquadro e selezionare *Flip Horizontal*, *Flip Vertical* o *Rotate 90 Degrees* dal menù pop-up.

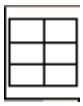

Figura 2.8. Schema dei terminali di un subVI.

Assegnare i terminali a controlli e indicatori

Portarsi sullo schema dei terminali e cliccarne uno, poi spostarsi sul pannello e cliccare sul controllo o indicatore che gli si vuole associare.

2.4.7 Uso di un subVI

Per inserire un subVI in un VI usare la palette *Functions* oppure, dal menù *Functions-Select a VI*, scegliere direttamente il VI che interessa e inserirlo nel nuovo diagramma. Tramite il comando *Help-Show Context Help* si accede alla finestra *Context Help* che contiene un pulsante *Simple/Detailed* nell'angolo inferiore sinistro; qui vengono mostrate tutte le informazioni relative ad un subVI.
I collegamenti necessari sono descritti in grassetto, quelli raccomandati in testo normale mentre quelli opzionali in testo opaco, se è stata selezionata la visualizzazione *Detailed*. Si possono impostare gli ingressi e le uscite come necessarie, raccomandate o opzionali, cliccando con il tasto destro sul riquadro dei connettori e selezionando *This Connection Is*; un contrassegno indica le impostazioni *Required*, *Recommended* ed *Optional*. LabVIEW imposta per default *Recommended*.

2.4.8 Strutture e grafici

Il flusso dei dati in un VI è controllato dalle strutture. LabVIEW ne include cinque: ciclo *While*, *For*, *Case*, *Sequence* e *Formula Node*. Esse si trovano nella palette delle *Functions*, sotto la voce *Structures*.

Il ciclo WHILE

Esegue una parte di codice fintanto che una condizione viene soddisfatta. Il comportamento del *terminale condizione* è per default *Continue If True*. La struttura esegue ciclicamente la porzione di schema contenuta al suo interno finché il terminale condizione riceve il valore booleano "true". Il VI controlla questo terminale al termine di ciascuna iterazione, perciò il ciclo while verrà eseguito almeno una volta.

Il terminale iterazione contiene il numero di cicli completati. Il conteggio parte sempre da 0. Si può modificare il *terminale condizione* cliccando con il destro su di esso o sulla cornice del ciclo while, poi selezionare *Stop If True*.

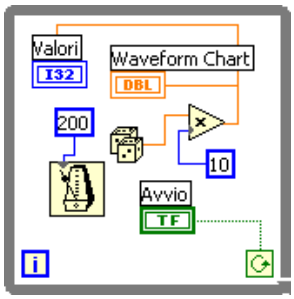

Figura 2.9. Esempio di ciclo WHILE.

Il ciclo FOR

Esegue la parte di codice contenuta al suo interno un numero prefissato di volte come i for dei linguaggi di programmazione testuali. Il valore N del *terminale di conteggio* indica quante volte il ciclo è ripetuto. Il *terminale di iterazione* contiene il numero di volte completate. Il conteggio parte da zero.

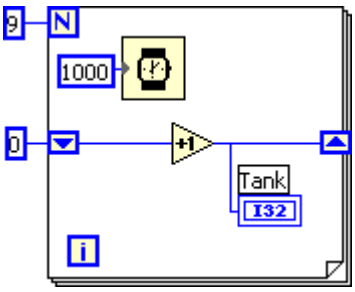

Figura 2.10. Esempio di ciclo FOR con registri a scorrimento.

Registri a scorrimento

Anch'essi sono usati nei cicli while e for per trasferire valori da un'iterazione del ciclo a quella successiva.
Un registro appare come una coppia di terminali, contrapposti l'un l'altro, sui due lati verticali del bordo struttura. Il terminale di destra memorizza il dato al termine di ogni iterazione: il dato riappare nel terminale di sinistra all'inizio dell'iterazione successiva. Per creare un registro si clicca con il pulsante destro sul bordo destro o sinistro della

struttura e si seleziona *Add Shift Register*. I dati gestiti da un registro devono essere dello stesso tipo. Per trattare dati diversi, si creano registri a scorrimento multiplo su una stessa struttura mentre per avere in memoria più valori precedentemente trattati, si crea più di un terminale a sinistra.
Per evitare comportamenti imprevisti all'avvio del programma, è opportuno inizializzare sempre i registri a scorrimento. Per fare ciò si collega il valore desiderato all'esterno del ciclo sul terminale di sinistra. Se il dato del registro è booleano, il valore iniziale è `false`, se numerico il valore iniziale è `zero`.

Struttura CASE

È simile alle istruzioni If - Then - Else - Endif dei linguaggi di programmazione testuali; è caratterizzata da due o più riquadri all'interno di ciascuno dei quali è collocata una porzione di schema. È visibile solo un riquadro alla volta e la struttura esegue uno schema alla volta. Un valore d'ingresso determina quale sottodiagramma deve essere eseguito. Nella parte in alto della struttura è presente (al centro) l'etichetta di selezione contenente il valore di identificazione della condizione relativa al riquadro visibile. Le condizioni possibili sono visibili tramite l'azione sui relativi pulsanti.
Il *terminale di selezione* è un terminale di ingresso che riceve il valore della condizione corrispondente al riquadro che deve essere eseguito. A questo terminale si può collegare un numero intero, un booleano o una stringa. Se il terminale è booleano, la struttura ha solo due condizioni: TRUE e FALSE.
Cliccando con il tasto destro del mouse sul bordo della struttura, si possono aggiungere, duplicare, rimuovere o risistemare le condizioni e selezionare quella di *default*. Si possono specificare elenchi di valori o intervalli per selezionare la condizione. Per un elenco di valori, separare gli elementi con le virgole (1,3,5,8) oppure con due punti singoli (5..10).
Per questa struttura si possono creare tunnel di ingresso e di uscita multipli. Quando si predispone il tunnel di uscita di una condizione, esso compare in tutti gli altri casi. Bisogna collegare il tunnel di uscita per ogni condizione in cui non è collegato, cliccando sul tunnel ogni volta.
Gli ingressi sono disponibili in tutti i riquadri ma *non devono necessariamente essere utilizzati da tutti i sottoprogrammi*.

Struttura SEQUENCE

Essa garantisce l'ordine di esecuzione ed impedisce operazioni parallele. È caratterizzata da un certo numero di riquadri all'interno dei quali è collocata una porzione di schema. I riquadri (*frame*) sono eseguiti in un certo ordine e non termina l'esecuzione né fornisce dati fino a quando l'ultimo frame non è stato eseguito. Per passare i dati da un frame al successivo, si usa un terminale locale della sequenza (*sequence local*). Nel terminale che contiene i dati compare una freccia verso l'esterno mentre nel terminale dei frame successivi ne compare una rivolta verso l'interno. Cliccando con il tasto destro del mouse sulla cornice e selezionando *Add Sequence Local* si crea un terminale di sequenza.

Formula Node ed Expression Node

La prima è una struttura, la seconda una funzione: entrambe eseguono operazioni matematiche.

1. FORMULA NODE: appare come un riquadro che consente di inserire formule algebriche direttamente nel codice del diagramma (utile quando la funzione da calcolare ha più variabili ed è piuttosto complessa). Bisogna creare i terminali di ingresso e di uscita cliccando con il pulsante destro del mouse sulla cornice del nodo e selezionando *Add Input* oppure *Add Output*. L'equazione sarà scritta all'interno della struttura e dovrà terminare con un punto e virgola (;).
2. EXPRESSION NODE: si trova nella palette *Functions-Numeric* e si usa per calcolare funzioni che contengono una sola variabile.

2.4.9 Matrici

Quando si lavora con i cicli for, spesso si incontrano variabili strutturate dette *Matrici* o *Array*. Un array è costituito da un insieme di elementi, tutti dello stesso tipo, che sono individuati tramite *indici* (detti anche dimensioni). Le dimensioni possono essere lunghezza, altezza e profondità nel caso di matrice tridimensionale. Se la matrice ha una sola dimensione si chiama *vettore*.
Gli elementi di una matrice sono ordinati e per accedere ad uno di essi occorre un indice. Ciascun indice inizia da 0 e quindi appartiene all'intervallo [0, n-1] dove n è il numero

degli elementi contenuti in quella dimensione. Le matrici possono essere *numeriche*, *booleane*, *path*, *stringhe*, *forme d'onda* e *cluster*.

Creazione di matrici di controlli e di indicatori

Selezionare la finestra delle matrici sulla palette *Controls-Array & Cluster* inserendo sul pannello l'oggetto *Array Shell*. Prima di inserire un oggetto nella struttura della matrice, il terminale della matrice stessa appare in nero con una parentesi vuota. Usando lo strumento *Posiziona* in prossimità di un angolo della finestra, apparirà una maniglia di ridimensionamento che, trascinata con il pulsante sinistro premuto, permette di aumentare o diminuire la visibilità degli elementi. Se lo strumento *Posiziona* viene posto in prossimità di un angolo dell'indicatore o del controllo, il ridimensionamento permetterà di cambiare le dimensioni alla matrice.

Matrici bidimensionali

Esse memorizzano i dati su una griglia. Per aggiungere dimensioni ad una matrice, una alla volta, cliccare con il tasto destro sul display indice e selezionare *Add Dimension*. Si può creare una matrice costante selezionando *Functions-Array-Array Constant* dalla palette, inserendola nel diagramma e trascinando dentro di essa una costante. Utilizzando l'autoindicizzazione si può utilizzare un ciclo while o for per generare automaticamente gli elementi di una matrice. Per creare una matrice bidimensionale, si possono usare due cicli for annidati: quello esterno crea le righe mentre quello interno le colonne.

Funzioni che operano sulle matrici

Si trovano selezionando la palette *Functions-Array*:

 ARRAY SIZE: fornisce il numero di elementi presenti in ogni dimensione.
 INIZIALIZE ARRAY: crea una matrice n-dimensionale in cui ogni elemento viene inizializzato al valore della costante scelta.
 BUILD ARRAY: concatena più elementi dello stesso tipo per formare una matrice, concatena più matrici per formare una matrice a più dimensioni oppure aggiunge elementi ad una matrice n-dimensionale esistente. Se ad esempio due vettori si collegano ad un *Build Array*, si ottiene una matrice bidimensionale con 2 righe ed n colonne. Se si attiva la funzione *Concatenate Input*, si ottiene un vettore con $n_1 + n_2$ elementi.

 ARRAY SUBSET: fornisce una parte di un array che inizia da *index* e contiene un numero di elementi pari ad una certa variabile assegnata.
 INDEX ARRAY: fornisce in uscita l'elemento di una matrice individuato dall'*index*. Si può utilizzare la funzione *Index array* per estrarre una riga o una colonna da una matrice bidimensionale oppure per ricavare una sottomatrice da quella originale. È possibile disporre di due terminali *index*: quello superiore indica la riga mentre il secondo la colonna. Collegando entrambi gli indici si può estrarre un solo elemento della matrice.

2.4.10 Cluster

È una struttura che raggruppa dati in modo simile alla matrice ma, a differenza si essa, può contenere dati di tipo diverso. Un cluster è simile ad un record dei linguaggi di programmazione testuali. Permette di raggruppare (*Bundle*) diversi elementi di dati in una sola finestra alla quale corrisponde un solo terminale che si collega con un unico filo. Come per le matrici, un cluster può essere un controllo o un indicatore. I diversi elementi di un cluster possono essere ottenuti estraendoli (*Unbundle*) tutti insieme oppure prelevando solo quelli che interessano. Utilizzando un cluster si evita la presenza di numerosi terminali e collegamenti sul diagramma.

Creazione di cluster per controlli e indicatori

Per creare un cluster di questo tipo, selezionare *Controls-Array & Cluster* e posizionare la finestra sul pannello: in seguito trascinare un controllo/indicatore dentro la finestra. Si può creare sul diagramma un cluster di costanti selezionando dalla palette *Functions-Cluster* la finestra *Cluster Constant* e trascinando dentro di essa una costante.

Ordine dei cluster

Quando si trattano cluster di dati sono importanti sia la tipologia dei dati dei singoli elementi, sia l'ordine degli elementi nel cluster. Il primo oggetto che viene inserito, indipendentemente dalla posizione che occupa nella finestra del cluster, è l'elemento 0 (quando viene cancellato un elemento il nuovo ordine si imposta da solo). L'ordine può essere modificato cliccando con il pulsante destro sulla cornice del cluster e selezionando *Reorder Controls Cluster*. Non si possono collegare fra loro dei cluster dove l'ordine degli oggetti non è corrispondente.

Funzioni che operano sui cluster

Si trovano sulla palette *Functions-Cluster*:

 BUNDLE: raggruppa elementi di ingresso individuali in un singolo cluster oppure modifica i valori di elementi individuali in un cluster già esistente.

 BUNDLE BY NAME: sostituisce o accede ad elementi di un cluster già esistente. Equivale alla funzione *Bundle* ma invece di riferirsi agli elementi tramite il loro ordine, si riferisce ad essi tramite le loro etichette. Questa funzione si utilizza nel caso in cui i dati possono cambiare durante lo sviluppo.

 UNBUNDLE: serve ad individuare i singoli elementi del cluster.

 UNBUNDLE BY NAME: individua gli elementi del cluster di cui si specifica il nome.

2.4.11 Waveform Chart, Waveform Graph e XY Graph

Questi tipi di grafici si collegano molto bene con i cluster di dati. *Waveform Graph* (grafici di forme d'onda) e *XY Graph* (grafici XY) differiscono dal *Waveform Chart* in quanto quest'ultimo visualizza i dati (scalari) in modo interattivo, man mano che vengono prodotti.

Waveform Chart

È un particolare indicatore numerico, spesso associato al ciclo while, è il Waveform Chart che si trova nella palette *Controls-Graph* e può ricevere tranquillamente un'uscita scalare. Esistono tre modi per rappresentare i dati sul grafico:

1. STRIP CHART: mostra i dati facendoli scorrere continuamente da sinistra a destra.
2. SCOPE CHART: mostra un insieme di dati per volta, scorrendoli da sinistra a destra.
3. SWEEP CHART: mostra i dati più vecchi sulla destra e quelli più nuovi sulla sinistra di una linea di separazione verticale.

Queste modalità possono essere selezionate cliccando con il pulsante destro sul grafico per scegliere *Advanced-Update Mode*. Se l'indicatore deve visualizzare più tracce simultaneamente, i dati devono essere raggruppati insieme con la funzione *Bundle* che si trova nella palette *Functions-Cluster*.

WaveForm Graph

Rappresenta funzioni ad un valore come *y = f(x)* con punti uniformemente distribuiti lungo l'asse *x*. È l'ideale per rappresentare vettori di dati nei quali i punti sono distribuiti uniformemente. Accetta come input:

1. un vettore, interpretando i dati come punti sul grafico iniziando da *x = 0* e incrementando l'indice di 1.
2. un cluster contenente un valore iniziale x_0, un incremento Δx ed un vettore di dati *Y*. I dati vengono interpretati come punti sul grafico iniziando da $x = x_0$ ed incrementando l'indice *x* di Δx.
3. una matrice bidimensionale *Y* in cui ogni riga della matrice contiene dati relativi ad una singola curva. Il grafico interpreta i dati come punti sul grafico, iniziando da *x = 0* ed incrementando l'indice di 1. Se i dati sono rappresentati per colonna, selezionare il grafico con il pulsante destro del mouse e scegliere *Transpose Array.*
4. un cluster con un valore x_0, un incremento Δx e una matrice bidimensionale *Y*. Il grafico interpreta i dati *Y* come punti sul grafico iniziando da $x = x_0$ con un incremento dell'indice di Δx.
5. una matrice contenente due cluster ciascuno con un valore iniziale, un incremento Δx ed un vettore *Y*. In questo caso *Bundle* unisce x_0, Δx e ciascun vettore in un cluster mentre *Build Array* serve per costruire la matrice di dati.

XY Graph

Rappresenta diagrammi cartesiani. È quindi adatto a visualizzare forme d'onda acquisite ad intervalli di tempo variabili. Questo tipo di grafico ha lo stesso aspetto di *WaveForm Graph* ma richiede differenti tipi di dati in ingresso.

2.5 Installazione e configurazione su sistemi Red Hat Linux 9.0

La piattaforma di sviluppo e produzione impiegata nell'intera infrastruttura informatica del sistema di QC è rappresentata dal sistema operativo Linux, in particolare le sue distribuzioni Red Hat Linux 9.0 e Mandrake 9.1. Tale scelta è motivata dalle note caretteristiche di stabilità e sicurezza operativa che tale sistema garantisce, oltre alla sua natura *Open Source*[6], entrambi fattori che lo rendono strumento ideale in ambiti tecnico-scientifici.

La procedura di installazione di LabVIEW su un sistema Linux non differisce molto da quella di un qualsiasi altro software: una volta inserito il cdrom di installazione è sufficiente eseguire lo script "`INSTALL`", da utente *root*, e seguire passo passo le informazioni mostrate a video o, in alternativa, installare manualmente i pacchetti che lo compongono servendosi del consueto tool *rpm*. Ogni componente dell'ambiente LabVIEW verrà installato di default nel percorso "`/usr/local`" o altrove se specificato dall'utente; un'installazione completa richiede approssimativamente 300 MB di spazio libero su disco rigido.

[6] I termini *open source* e *software libero* vengono normalmente utilizzati per identificare software il cui codice sorgente può essere liberamente studiato, copiato, modificato e ridistribuito. In particolare, la definizione di *software libero* proposta dalla *Free Software Foundation* (FSF) recita testualmente:
"L'espressione *software libero* si riferisce alla libertà dell'utente di eseguire, copiare, distribuire, studiare, cambiare e migliorare il software. Più precisamente, esso si riferisce a quattro tipi di libertà per gli utenti del software:
- Libertà di eseguire il programma, per qualsiasi scopo.
- Libertà di studiare come funziona il programma e adattarlo alle proprie necessità.
- Libertà di ridistribuire copie in modo da aiutare il prossimo.
- Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio."
I termini *open source* e *free software* sono da taluni considerati sinonimi. In realtà essi si riferiscono a filosofie ed approcci diversi:
Free software: secondo la FSF di Richard Stallman il software deve essere libero non in quanto gratuito, ma per una questione etica e di principio. Esistono una serie di diritti dell'utente del software (indicati nella definizione proposta in precedenza) che devono essere adeguatamente tutelati; il software deve essere libero per questi motivi prima ancora che per motivi di carattere economico e di mercato.
Open Source: la comunità del software OS condivide in larga misura le posizioni del mondo del software libero, ma deenfatizza gli aspetti etici, fondando le proprie scelte e motivazioni su considerazioni di carattere tecnico-economico. Secondo i sostenitori del software OS, tali motivazioni sono sufficienti a giustificare la necessità del software aperto/libero.

Per quanto riguarda i requisiti minimi richiesti, prendendo in esame la versione LabVIEW 7.0 impiegata nel presente progetto, il produttore dichiara:

- sistema Linux con kernel 2.0.x, 2.2.x, o 2.4.x per architetture Intel x86.
- libreria GNU C (glibc) versione 2.1.3 (raccomandata la 2.2.4 o successiva).
- processore Intel Pentium III/Celeron 600 Mhz o equivalente e 128 MB di RAM.

Normalmente le operazioni necessarie per rendere operativo l'ambiente di sviluppo LabVIEW sono pressoché nulle. Tuttavia non è affatto raro desiderare di poter accedere a dispositivi esterni tramite le ben note porte seriali e/o parallele, per le quali è previsto un apposito strumento (*tool*) di configurazione chiamato *visaconf*. Il suo impiego è reso estremamente semplice grazie ad una chiara interfaccia grafica che consente di impostare tutte le porte alle quali si pensa di voler accedere. Nel caso specifico, si è provveduto ad abilitare la porta seriale e quella USB (universal serial bus); quest'ultima verrà impiegata simulando la prima su alcuni computer portatili che ne erano sprovvisti (si veda §3.3). Per realizzare ciò è necessario modificare il file nascosto "`.labviewrc`" presente nella home directory dell'utente, inserendovi le seguenti linee:

```
labview.serialDevices:  "/dev/ttyS0"
labview.serialDevices:  "/dev/ttyUSB0"
```

Successivamente avviare `visaconf` e modificare la risorsa "`ASRL1::INSTR`" in modo che punti a "`/dev/ttyS0`" e "`ASRL2::INSTR`" a "`/dev/ttyUSB0`". Riavviando LabVIEW, le risorse da utilizzare all'interno del codice di programmazione saranno, rispettivamente, la prima per la porta seriale e la seconda per quella USB.

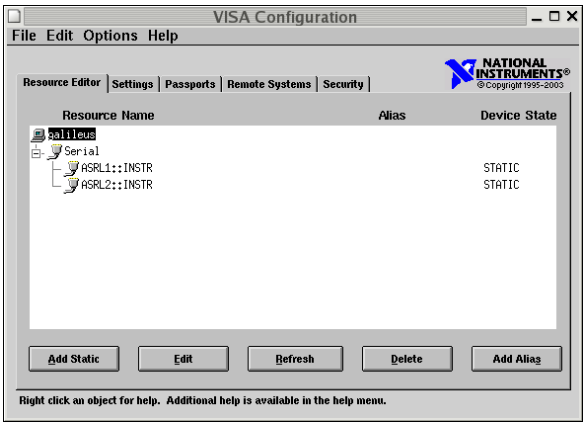

Figura 2.11. L'interfaccia del tool "Visaconf".

Capitolo 3

Il software di QC (Quality Control)

3.1 Introduzione

Nei prossimi paragrafi verranno mostrati dettagliatamente i cinque software per il sistema di controllo di qualità attivato presso il sito di produzione. Ogni sistema di DAQ è stato pensato come tipo *dedicato*, e quindi adibito ad un unico compito, o *multipurpose* cioè uno strumento universale con diverse interfacce grafiche specializzate su diverse operazioni. L'idea era che ogni *TSm* (Test Station multipurpose, vedere figura 1.5) fosse utilizzabile indifferentemente in tutti i compiti di QC che avevano frequenza saltuaria ma che potevano anche avvenire simultaneamente nell'arco della giornata. Per queste ragioni l'interfaccia utente ricopriva un ruolo rilevante ed era necessario che:

1. avessero tutte una coerenza grafico-funzionale in modo da essere immediatamente comprensibili dopo la prima esperienza con una qualsiasi di esse.
2. riducessero a zero ogni possibilità di interazione operatore-computer diversa da quella programmata.
3. automatizzassero la creazione dei file di dati (corretta convenzione dei nomi, dislocazione, etc..).
4. gestissero l'inserimento dei dati assicurando la consistenza dei dati immessi.
5. gestissero il trasferimento automatico dei dati al server a fine lavorazione.

A causa delle caratteristiche del linguaggio di programmazione grafico di LabVIEW, risulta impossibile mostrare per intero il codice di programmazione dei software sviluppati, né inserirlo in appendice; verrà perciò mostrata solo una minima parte rispetto al totale, in particolare le parti "salienti" o quelle ritenute di maggiore interesse per approfondimenti sulla struttura ed il funzionamento. Complessivamente, il software progettato e sviluppato sotto forma di VI e subVI (escludendo quindi alcune parti scritte in linguaggio C) ammonta a circa 15 MB di codice.

3.2 I principali subVI realizzati

Durante la fase di sviluppo si è palesata la necessità di creare un ampio numero di subVI che fossero facilmente riutilizzabili in tutti i test di DAQ; infatti operazioni come l'apertura e la scrittura di file, la decodifica dei codici a barre e molte altre ancora, sarebbero state comuni a tutti i test. Normalmente tutti i subVI devono essere installati nella directory "`/usr/local/lv70/user.lib/`" (nel caso di una installazione standard di LabVIEW come mostrato al §2.5) per essere facilmente accessibili durante la fase di creazione del diagramma.
Quello che segue è un elenco dei principali subVI sviluppati e delle loro caratteristiche. Per uno schema completo dei tipi di codici a barre utilizzati nel QC e del loro significato, si veda §1.4.2.

3.2.1 Ean13

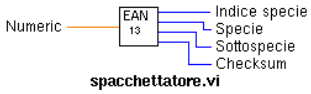

Figura 3.1. Icona e terminali del subVI *EAN13*.

Si occupa di "spacchettare" il codice a barre. È un subVI fondamentale perché impiegato praticamente da tutti gli altri (*Decode*, *Check*, *CellZoneDefect*, *EndCodes*). La sua struttura è semplice: ricevuto il dato numerico di tipo *double* come input, lo converte in una stringa di testo e successivamente lo suddivide nelle informazioni base come mostrato al §1.4.2; ogni sottostringa viene poi riconvertita in numero intero e passato in output.

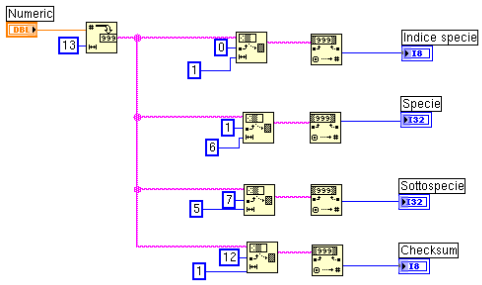

Figura 3.2. Il diagramma di *EAN13*.

3.2.2 Check

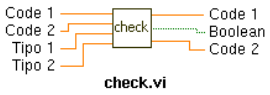

Figura 3.3. Icona e terminali del subVI *Check*.

La sua funzione è quella di verificare che un codice a barre (o al massimo due contemporanemente) inserito in input, corrisponda al tipo desiderato; ciò è molto utile per verificare la consistenza dei dati inseriti, ad esempio controllando che sia stato effettivamente inserito un codice di identificazione di un tubo (ID tubo), piuttosto che un altro, quando richiesto dal programma. Impiega a sua volta il subVI *EAN13*, e tratta tutti i dati come numeri *double* tranne un dato *booleano* in uscita che indica l'esito finale della verifica.

3.2.3 Decode

Figura 3.4. Icona e terminali del subVI *Decode*.

Questo subVI impiega entrambi i precedenti, *Check* ed *EAN13*, per decodificare un codice a barre (*barcode*) passatogli come input e mostrare in uscita tutte le informazioni che quel codice rappresentava. In particolare è utilizzato per i test di *Visual graphite inspection* e *Gas tightness* e di seguito ne viene mostrata una parte del diagramma.

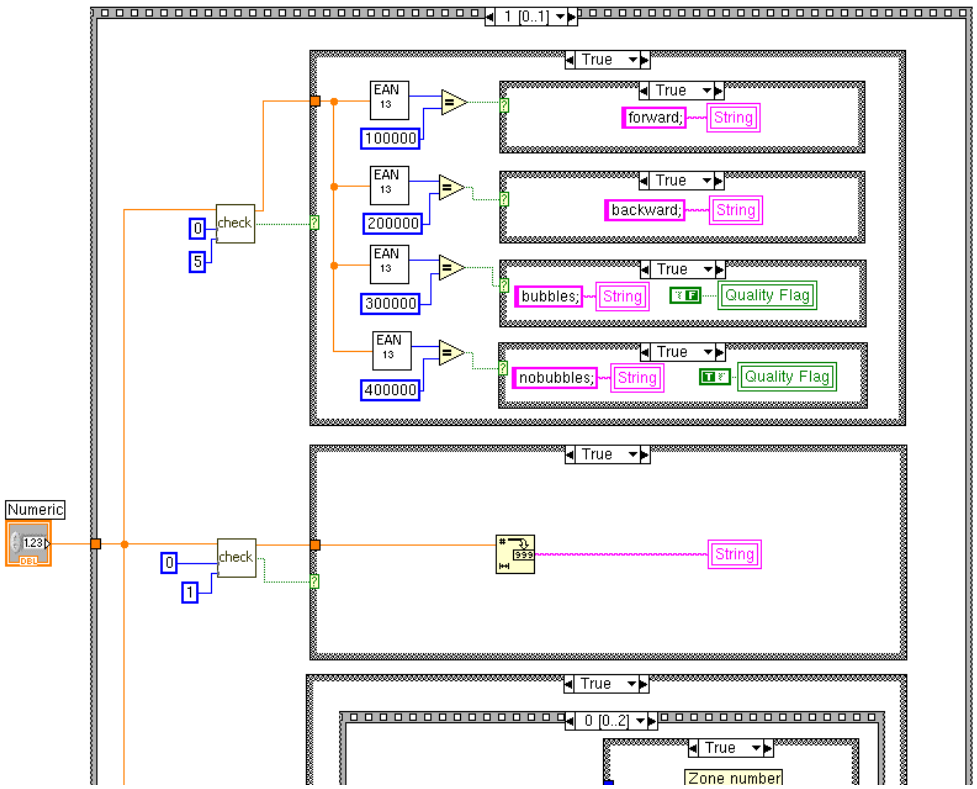

Figura 3.5. Parte del diagramma del subVI *Decode*.

3.2.4 CellZoneDefect

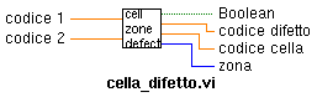

Figura 3.6. Icona e terminali del subVI *CellZoneDefect*.

L'operazione compiuta da questo subVI è quella di estrarre dal barcode inserito, impiegando il subVI *EAN13*, le informazioni relative al tipo di difetto, numero di cella e zona impiegati nel test di *Visual graphite inspection*. Una variabile booleana conferma la correttezza dell'operazione.

3.2.5 EndCodes

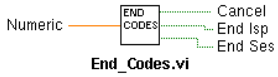

Figura 3.7. Icona e terminali del subVI *EndCodes*.

Il semplice compito assolto dal subVI *EndCodes* è quello di decodificare, con l'ausilio di *EAN13*, il codice a barre inserito e verificare se questo rappresenta uno dei seguenti possibili codici: *Cancel* (annulla l'ultimo dato inserito e torna un passo indietro), *End Isp* (fine ispezione del tubo corrente) e *End Ses* (fine sessione, termina l'esecuzione del test sulla stazione DAQ).

3.2.6 OpenFile

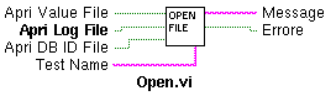

Figura 3.8. Icona e terminali del subVI *OpenFile*.

Questo pratico subVI gestisce in maniera completa la creazione dei file necessari per tutti i test di QC. Ad esempio si occupa di *creare la directory per i log* (i file che contengono tutti i dati relativi al test eseguito) nella home dell'utente assegnadogli come nome il nome del test stesso (questo evita di mescolare log quando si eseguono test diversi su di una stessa stazione DAQ), *crea il file di log* rinominandolo in base al tipo, data, ora, minuti e secondi di esecuzione del test, *crea il file dei valori* (value) usato da alcuni test per registrare i valori di alcune grandezze ottenute mediante misurazioni. Infine *crea il file di identificazione per il database* (DB) di produzione che necessita, per alcuni test, di un numero incrementale utile per le operazioni di normalizzazione[8]. Due tipi di uscite completano il subVI: una stringa di testo contenente dettagliati messaggi di errore o di conferma sulle operazioni compiute e un booleano che risulta vero in caso di errore, falso in caso contrario.

[8] Il processo di normalizzazione di uno schema relazionale consiste in una serie di raffinamenti successivi dello schema del database atti a ridurne al minimo la *ridondanza* o almeno consentirne una gestione controllata.

Segue un esempio del suo utilizzo nel test di *Ispezione visiva della grafite*, in cui si può notare come il subVI venga impiegato per creare contemporaneamente tutti e tre i tipi di file di log necessari (i tre booleani in ingresso sono tutti inizializzati a TRUE).

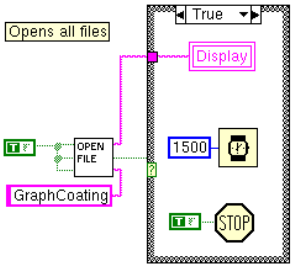

Figura 3.9. Diagramma di utilizzo del subVI *OpenFile*.

Oltre ai subVI *CloseFile* e *WriteFile* mostrati più avanti, *OpenFile* ha la necessità di condividere insieme al VI chiamante le variabili contenenti i *byte stream* dei file creati; questo consente a entrambi di poter accedere ai file, sia in lettura che in scrittura, in modo indipendente e sicuro. Per realizzare ciò, a livello di programmazione con LabVIEW, si rende necessario definire delle *variabili globali* (il cui impiego è bene rimanga limitato il più possibile) all'interno di un VI fittizio, in questo caso "`My_Global.vi`", che andrà salvato nella directory radice contenente tutti i VI utilizzati[9].

3.2.7 CloseFile

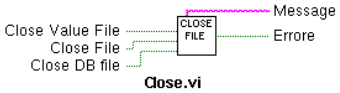

Figura 3.10. Icona e terminali del subVI *CloseFile*.

Lo scopo di questo subVI è evidente: è il complementare del precedente, *OpenFile*. A seconda di come vengono impostate le tre variabili booleane di input, verranno chiusi i rispettivi byte stream dei file di log e verrà prodotto in uscita un messaggio di conferma o di errore e impostata di conseguenza la variabile booleana in uscita.

[9] Consultare lo *User Manual* di LabVIEW al capitolo 10-2 per i dettagli sull'uso delle variabili globali.

3.2.8 WriteFile

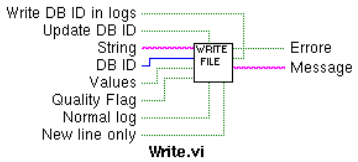

Figura 3.11. Icona e terminali del subVI *WriteFile*.

È il più sofisticato dei subVI, essendo in grado di gestire le operazioni di scrittura su tutti i tipi di file di log che sono stati usati nei test di QC. Consente di scrivere su di essi semplicemente attivando la relativa variabile booleana (impostandola a TRUE) ed inserendo il dato da scrivere nella variabile di tipo stringa oppure inserendo caratteri speciali come il *new line* o il *quality flag*. Essendo il formato dei log standardizzato, un sistema di scrittura così centralizzato evitava di doversi assicurare ogni volta che tale operazione producesse log nel formato stabilito, semplificando la programmazione e al contempo limitando la possibilità di errori. Qualsiasi operazione di scrittura sui file di log andava quindi eseguita mediante questo subVI. Le uscite di tipo booleano e stringa, identiche a quelle dei subVI precedenti, ne completano il funzionamento.

3.3 Visual graphite inspection

L' *ispezione visiva della grafite* è il primo test di QC eseguito direttamente dall'operatore non appena la verniciatura risultava perfettamente asciutta. Il suo scopo era quello di verificare che tutta la superficie grafitata non presentasse difetti o imperfezioni visibili ad occhio nudo e, in caso affermativo, registrare il tipo di difetto e il punto in cui si trovava. Allo scopo è stato predisposto un tavolo (figura 3.12) diviso in 8 zone, ognuna identificata da specifici codici a barre, sul quale veniva posto il tubo; quando l'operatore, scorrendo il tubo, incontrava un difetto della superficie grafitata lo comunicava al software della DAQ "cliccando" con il lettore di codice a barre sul codice relativo al tipo di difetto per quella zona.

Figura 3.12. Tavolo adibito all'ispezione visiva della grafite.

Il test produce due tipi di file di log contenenti tutte le informazioni relative al tubo ispezionato, i cui formati sono i seguenti:

```
NUM_INCREM;DATA;ORA;ID_TUBO;QUALITY_FLAG
;1471;2004-06-08;03:37:07;1002209899996;1
```

```
NUM_INCREM;ZONA;CELLA;DIFETTO
;1471;16;1;Well Repaired;
;1471;1;8;undefined all zones;
```

Il primo log contiene il codice identificativo della camera e un flag che ne indica il superamento del test dopo eventuali correzioni dei difetti rilevati; il secondo file di log contiene invece la posizione, determinata dal codice di zona e di cella, e il tipo di difetto riscontrato. Il numero incrementale è stato introdotto per necessità di normalizzazione del database (si veda §3.2.6), ma torna utile anche per associare il primo log al secondo. Quest'ultimo infatti può riportare un numero indefinito di difetti per una singola camera e quindi un numero indefinito di entry che dovranno iniziare tutte con lo stesso numero incrementale; in questo modo l'associazione difetti-camera risulta corretta. È chiaro che il VI si dovrà occupare anche di conservare una copia locale del numero incrementale che verrà maggiorato di 1 ad ogni nuova ispezione di una camera.

L'interfaccia grafica del test, riportata in figura 3.13, mostra chiaramente i passi da seguire in fase di inserimento dati: codice di identificazione del tubo (id-tubo), codice cella o difetto (il sistema si adatta all'ordine di inserimento dati dell'operatore: se viene inserito per primo un codice cella, verrà successivamente richiesto un codice difetto,

l'inverso nel caso opposto), infine uno dei codici di uscita o fine ispezione.
In basso un pannello riproduce la situazione dell'intero tubo mostrandone in tempo reale le zone difettose: all'aumentare del numero di difetti di una zona, il suo colore vira verso il rosso. Questo sistema è stato implementato con l'intenzione di fornire all'operatore un'anteprima sullo stato generale del tubo durante e a fine ispezione.

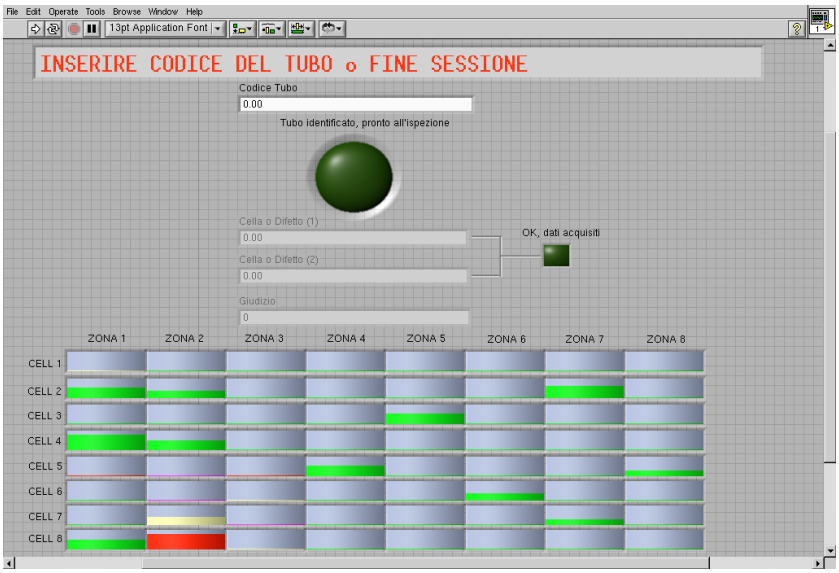

Figura 3.13. Pannello del test di *Visual graphite inspection*.

In questo test e nei prossimi due è stato previsto il supporto ai lettori di codici a barre di tipo *wireless*, ovvero senza filo. Questi dispositivi sono composti da una pistola laser che legge il barcode e lo trasmette ad una stazione ricevente collegata al calcolatore attraverso la porta seriale. Poiché molti portatili erano sprovvisti di tale porta, si è impiegato un cavo convertitore seriale-USB che risultava correttamente supportato dal kernel Linux. Rimaneva però da risolvere il problema più grosso: l'*emulazione tastiera*, cioè il modo operativo con cui i lettori di codici a barre "fanno credere" al calcolatore di avere a che fare con una normalissima tastiera (evitando così l'impiego di driver specifici), non funzionava e generava comportamenti imprevedibili. Dulcis in fundo, LabVIEW non era in grado di "vedere" la porta USB come una porta seriale (trasformazione ottenuta mediante il cavo convertitore di cui sopra) impedendo l'impostazione dei necessari parametri per la comunicazione (velocità, bit di parità, etc..) e quindi la corretta lettura dei barcode inseriti. La soluzione più semplice e al tempo stesso efficace proveniva dalla nota caratteristica dei sistemi Unix per i quali *tutto è un file*; ciò significa che anche un

dispositivo o una periferica esterna possono essere visti come semplici file[10]. Grazie a questa concezione architetturale, tipica anche dei sistemi Linux, una banale operazione di lettura della prima porta USB mediante `cat /dev/usb/ttyUSB0` consentiva di leggere il codice a barre proveniente dal lettore wireless; infine, essendo il barcode una semplice stringa alfanumerica non veniva richiesta alcuna interpretazione dei dati letti. In figura 3.14 è possibile vedere una parte del diagramma che implementa il sistema di lettura dei barcode: in pratica questo codice, eseguito in parallelo a quello del test vero e proprio e contenuto in un ciclo che terminava solo quando il booleano *CloseUSB* risultava vero (in genere al termine dell'esecuzione del programma), reindirizzava l'output del comando di shell `cat` su un file chiamato `usb.txt` che avrebbe contenuto tutti i codici a barre inseriti dall'operatore, disposti uno per riga. Leggendone l'ultima riga e verificando che il codice avesse una lunghezza di 14 byte, si otteneva il barcode che veniva poi memorizzato nella variabile *BarcodeNumber* (previa conversione da stringa a double) e contemporaneamente veniva impostata a `TRUE` la variabile booleana *BarcodeOK*, a significare la corretta esecuzione delle operazioni. Quando il programma di test in esecuzione richiedeva l'inserimento di un dato, impostava a `TRUE` la variabile *ReadNow* informando della necessità di avere un codice a barre e lo otteneva solamente se *BarcodeOK* risultava vera.

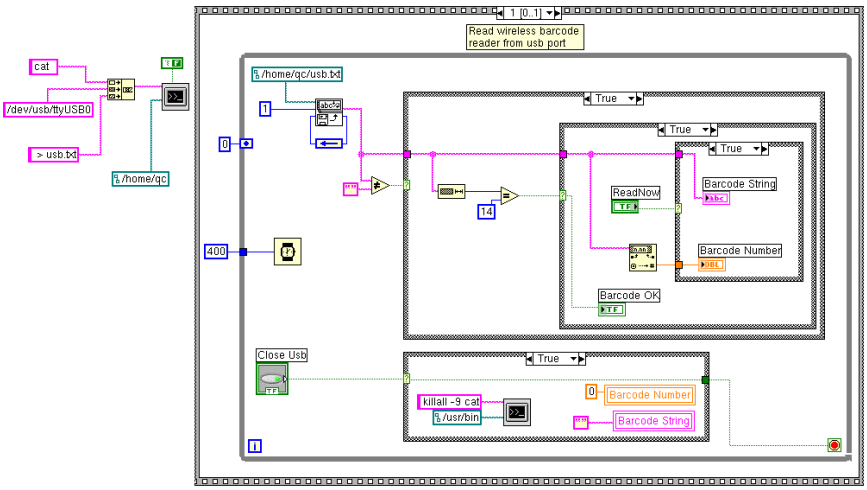

Figura 3.14. Diagramma che introduce il supporto ai lettori di codici a barre di tipo wireless.

[10] Uno degli aspetti più semplici ed eleganti dell'architettura di Linux è il modo in cui ogni cosa viene rappresentata sotto forma di file. Anche i dispositivi stessi sui quali i file vengono memorizzati sono rappresentati sotto forma di file. I dispositivi hardware sono associati a dei driver che forniscono un file di interfaccia; questi file speciali che rappresentano i dispositivi hardware (detti semplicemente dispositivi) vengono mantenuti nella directory `/dev`.[7]

Mediante ulteriori piccoli affinamenti e opportune temporizzazioni, il sistema si è rivelato molto affidabile e sono state quindi preparate due versioni per ognuno dei test di *Visual graphite inspection*, *Spool / tube association* e *Gas tightness*, una per l'uso con lettori di tipo wireless e l'altra per normali lettori a filo collegati alla porta PS/2 normalmente impiegata dalla tastiera del calcolatore.

L'algoritmo qui esposto ha il pregio di integrarsi facilmente in qualsiasi programma, limitandone al minimo le modifiche necessarie. È infatti sufficiente aggiungere il diagramma di figura 3.14 a quello del programma che ne vuole far uso e utilizzare una variabile booleana, *ReadNow* in questo caso, che funge da interruttore e una variabile stringa o double che conterrà il dato richiesto; di seguito viene mostrato il diagramma di una piccola parte del test di ispezione visiva della grafite incaricata proprio dell'inserimento dei dati attraverso un lettore di codici a barre di tipo wireless e che quindi fa uso delle due variabili sopracitate.

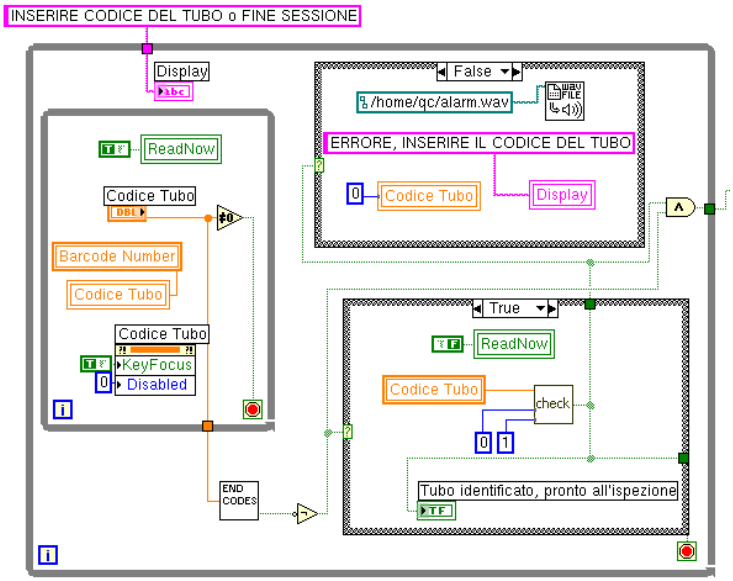

Figura 3.15. Parte del diagramma del test di ispezione visiva della grafite.

A fine ispezione, cioè dopo l'inserimento del relativo codice da parte dell'operatore, il programma si occupa di inviare al server i file di log prodotti stabilendo una connessione protetta tramite *SCP* (Secure CoPy) nonché di crearne una copia di backup sul disco rigido locale.

3.4 Spool/tube association

L'obiettivo di questo test era quello di associare il codice identificativo della camera (ID-tubo) al set di bobine di filo usate per "filarla", così come esposto al §1.4.2. L'operatore, prima della filatura, inseriva tramite il lettore di barcode il codice dell'etichetta della camera e a seguire quello delle bobine.
Poiché non era prevedibile il numero di bobine impiegate per ciascuna camera, il programma continuava ad accettare codici a barre delle bobine fino a quando l'operatore non inseriva il codice di *Fine Associazione* (in questo caso coincidente a quello di *Fine Ispezione*), dopo di che il VI si occupava di inviare al server il log prodotto e ne eseguiva una copia locale di backup. Di seguito viene riportato un esempio di file di log e del suo formato.

```
;DATA;ORA;ID_TUBO;COD_BOBINA;COD_BOBINA;...
;2004-06-10;08:13:57;1001075799997;40;45;06;04;42;39;17;28;
```

Questo VI ha una struttura simile a quella dell' *Ispezione visiva della grafite*, sebbene sia più semplice grazie al limitato numero di tipi di barcode utilizzati e al formato meno elaborato dei file di log. Il suo front panel è mostrato in figura 3.16.

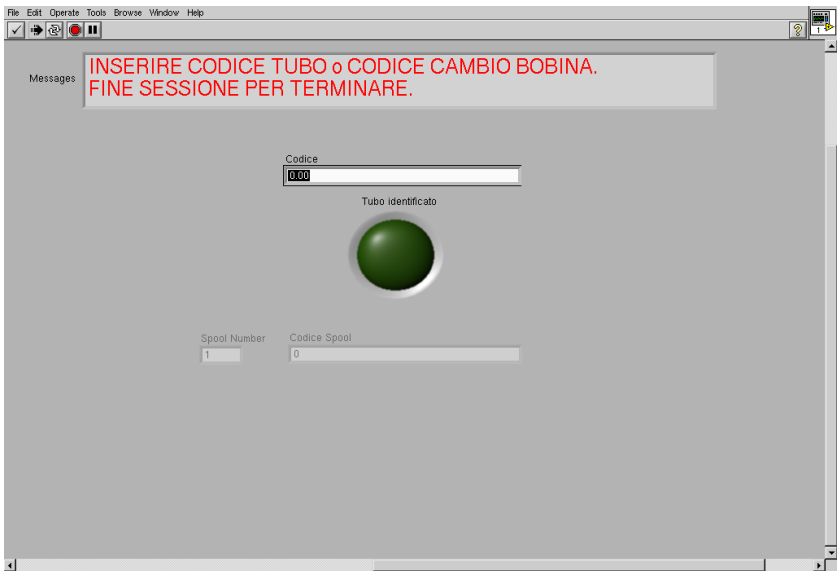

Figura 3.16. Pannello del test di *Spool/tube association*.

3.5 Gas tightness

In questo test l'operatore immergeva il tubo, a cui era stata applicata una sovrapressione di 15-20 mbar, in una vasca piena d'acqua e osservava l'eventuale formazione di bolle, segno che la sigillatura era imperfetta. Il VI sviluppato aveva quindi il semplice compito di assistere l'operatore e inserire nei log i risultati dell'operazione. Come si può vedere dal pannello riportato in figura 3.17, il programma per prima cosa chiedeva il codice identificativo del tubo, quindi il codice del lato che andava osservato (*nord* o *sud*); a questo punto doveva trascorre un minuto di tempo, la cui progressione era indicata dalla barra blu, durante il quale non veniva tollerata più di una bolla. Un segnale acustico avvertiva l'operatore che il tempo era trascorso e che doveva inserire il "giudizio" per quel lato del tubo osservato: due codici a barre diversi identificavano le due uniche possibilità, *Bubbles* o *NoBubble*s.
Il VI proseguiva ripetendo la stessa cosa per l'altro lato del tubo; al termine il test finiva e, come per i precedenti, inviava i log al server di produzione e ne eseguiva il backup locale. Di seguito ne viene riportato il formato dei log.

```
;DATA;ORA;ID_TUBO;LATO;GIUDIZIO;
;2004-06-01;00:37:53;1009000080320;backward;nobubbles;
```

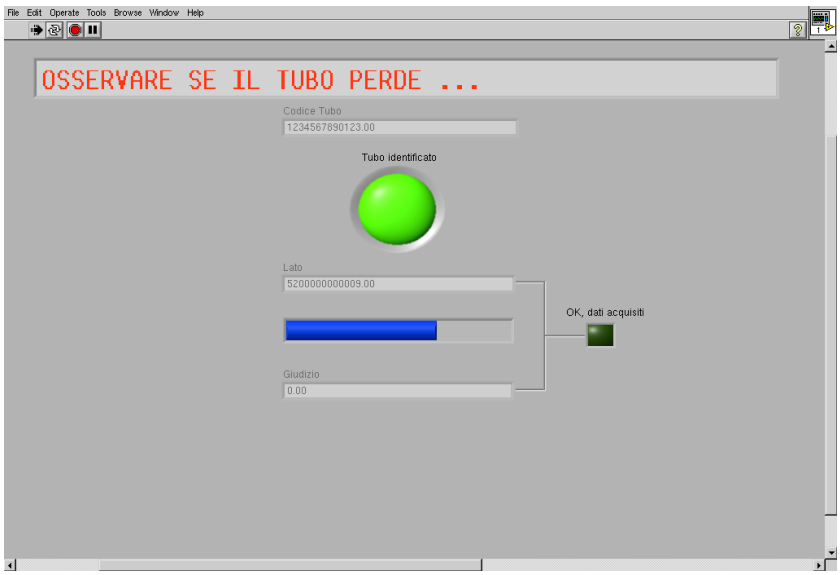

Figura 3.17. Pannello del test di *Gas tightness*.

3.6 High voltage conditioning

3.6.1 C.A.E.N. high voltage system SY546

3.6.2 High speed CAENET PCI controller A1303

3.6.3 Il software di comunicazione per il network HS CAENET

3.6.4 Il VI per il test di condizionamento

3.7 Long term test

Epilogo

Ringraziamenti

Appendice

Bibliografia

[1] John D. Barrow, *Il Mondo dentro il Mondo*, Adelphi, Milano, 1991.

[2] Homepage of the BaBar Group of the Perugia Section of INFN & Physics Department of the University of Perugia, *http://www.pg.infn.it/ricerca/esperimenti/babar/introduction.html*

[3] *The Particle Adventure*: copyright 1995-6 by The Contemporary Physics Education Project. Revised August 1996 by Charles GroomProject supervised by Michael Barnett. Ed. italiana: *L'avventura delle particelle*, copyright 1998, Istituto Nazionale di Fisica Nucleare. Curatore del progetto: Alessandro Pascolini.

[4] *Le Scienze.*

[5] National Instruments, *LabVIEW User Manual*, july 2000 Edition.

[6] Eduard Simioni, *Realizzazione del nuovo rivelatore di muoni con tubi a streamer per l'esperimento BABAR*, Università di Padova, luglio 2004.

[7] David Pitts e Bill Ball, *Red Hat Linux 6 Tutto&Oltre*, Apogeo, Milano, 2000.