UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA MECCATRONICA

TESI DI LAUREA

PROGETTAZIONE DI UN SISTEMA DI COLLAUDO AUTOMATICO IN AMBIENTE DI SVILUPPO LabVIEW

Relatore: Ch.mo Prof./Ing. **ROBERTO OBOE**

Correlatore: **OSCAR ZANIN**

Laureando: **LUCA PAMATO**

Matricola 581179-IMC

ANNO ACCADEMICO 2010-2011

Sommario

L'elaborato tratta le problematiche relative alla definizione, formalizzazione ed automatizzazione delle procedure di collaudo funzionale di una scheda elettronica. Nello specifico si vuole realizzare un collaudo automatico attraverso software NI LabVIEW sfruttando la documentazione esistente in azienda (diagrammi di flusso, istruzioni operative, procedure, ecc.) al fine di monitorare tutte le variabili interne del sistema, nascondere la complessità generale, diminuire le tempistiche di test ed evitare danni alla scheda e all'operatore. Il collaudo automatico genera inoltre segnali visivi e comanda segnali sonori per la facile interpretazione dell'esito del test, oltre a creare report dettagliati per l'identificazione immediata di valori d'interesse, guasti e anomalie su scheda all'interno di un piano di produzione.

Indice

Introduzione

Il problema che si presenta a fronte di un collaudo manuale è principalmente l'elevato numero di operazioni necessarie per portare a termine il test, indifferentemente dal suo esito, al quale fanno da cornice errati cablaggi e distrazioni dell'operatore. Con la creazione di un automatismo che verifichi tutte le soglie, i valori di interesse e le variabili interne ed esterne in gioco si vanno a ridurre notevolmente i rischi di errato collaudo causato da fattori non direttamente legati alla conformazione e natura della scheda, oltre a ridurre i tempi totali di test. Va specificato che il collaudo automatico non esclude in nessun modo l'operatore dal processo di test bensì semplifica le operazioni da svolgere e ne velocizza il lavoro, consentendo il testing di più schede in minor tempo. Occorre inoltre tener presente che la ricerca del collaudo ideale per uno specifico tipo di scheda è svolta a partire dal diagramma di flusso delle operazioni da eseguire e dalle specifiche tecniche dettate dall'azienda: la virtualizzazione del collaudo necessita quindi di particolare accuratezza in termini di tempi di attesa e verifica, sincronizzazione tra eventi, gestione degli errori e allarmi nonché scambio di dati real time tra computer e banco di collaudo.

Nei capitoli seguenti verranno affrontati tutti gli aspetti salienti dello stage a partire dalla descrizione delle funzionalità del banco di collaudo e l'hardware utilizzato, l'utilizzo del software con una breve descrizione della sua struttura, le specifiche stabilite a livello aziendale fino al codice creato in ambiente LabVIEW e la relativa stesura della procedura operativa di collaudo.

Capitolo 1

Presentazione del progetto

1.1 Il collaudo

Il collaudo di una scheda elettronica in ambito industriale deve avvenire in condizioni tali da garantire, salvo difetti interni dei dispositivi, un esito positivo. Ciò si ottiene se ad una scelta mirata della disposizione dei componenti sulla scheda si affianca un sistema di controllo veloce ed intuitivo per l'operatore preposto al collaudo. Ne deriva quindi la necessità di stabilire procedure specifiche per ogni operazione che si effettua sulla scheda in fase di test. Se ad una procedura manuale composta da più operazioni di cablaggio, accensione, disinserimento, ecc. si va a sostituire un controllo automatico del processo si ottengono vantaggi sia in termini di tempo test che in termini di precisione e controllo di tutte le variabili interne di processo. La documentazione presente in azienda consiste in una procedura di collaudo manuale per la UUT (Unit Under Test ovvero la scheda montata sul banco di collaudo) assieme all'istruzione operativa per il corretto utilizzo del banco di collaudo ospitante la scheda e agli schemi elettrici corrispondenti. L'obbiettivo del progetto è quello di partire dal flow chart delle operazioni da compiere e realizzare in ambiente LabVIEW tutte le verifiche e i controlli del caso, informando passo passo l'utente grazie a tabelle, messaggi e report finali.

1.2 L'hardware utilizzato

Nello specifico verrà trattato il testing di una scheda elettronica per teleruttori generali di corrente. Si sfrutterà poi un banco di collaudo creato in azienda appositamente per questa scheda e contenente un DAQ NI USB-6008 per l'interfacciamento diretto col PC, consentendo quindi il controllo e la gestione dei dispositivi tramite LabVIEW.

1.2.1 La scheda elettronica

La scheda elettronica presa in esame per il progetto comanda dei TGC/E (Teleruttori Generali di Corrente Elettronici) a 24[V] con stacco del polo positivo di alimentazione. Secondo la Norma CEI 17-3, un teleruttore è un dispositivo meccanico di manovra, generalmente previsto per un numero elevato di operazioni, avente una sola posizione di riposo, ad azionamento non manuale, capace di stabilire, sopportare ed interrompere correnti in condizioni di sovraccarico. La posizione di riposo corrisponde ordinariamente alla posizione di apertura dei contatti principali. Si distingue dal relè per il fatto che quest'ultimo è impiegato per il comando di potenze relativamente piccole o segnali in ambito elettronico, mentre il teleruttore è impiegato nel comando di potenze anche molto elevate.

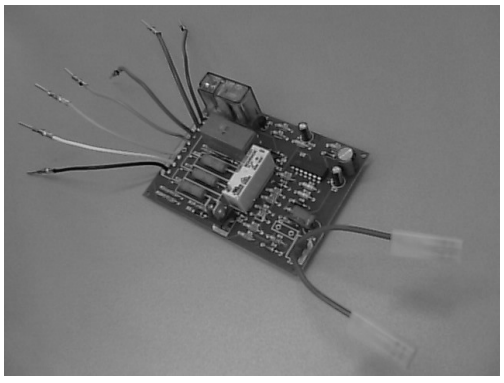


Fig. 1.1 La scheda da collaudare.

In Fig. 1.2 si mostra la specifica del cliente, riportante i collegamenti che la scheda deve avere per il normale funzionamento in fase di test.

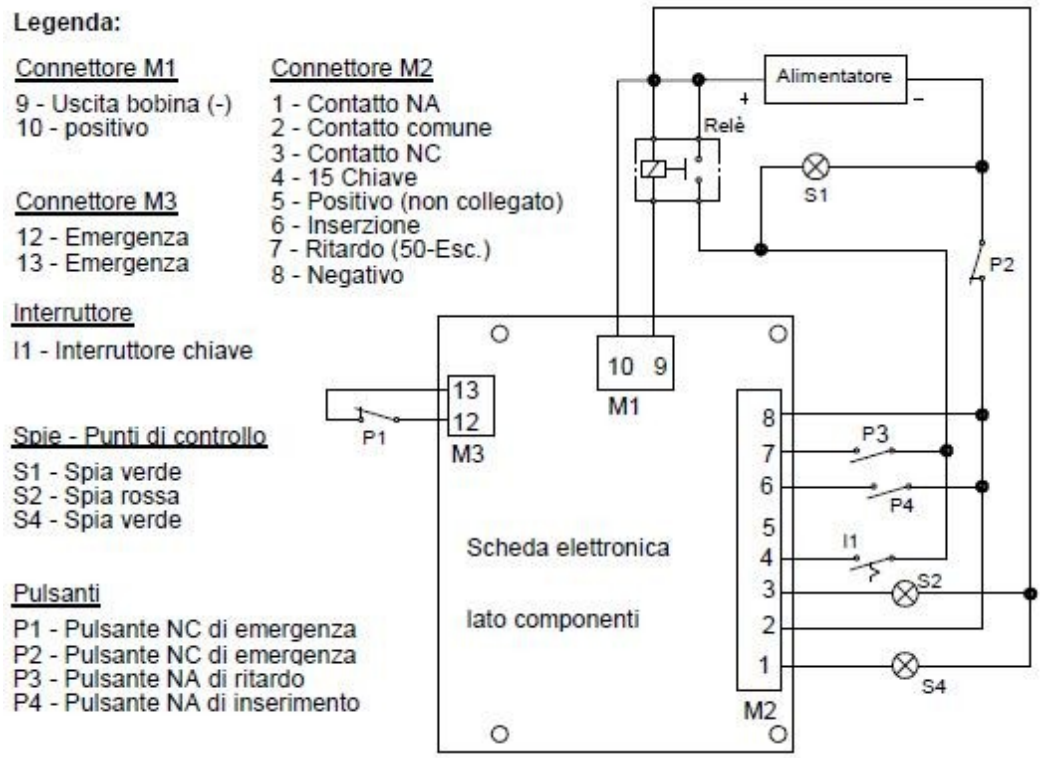


Fig. 1.2 La specifica del cliente per la scheda da collaudare.

Ciò che poi si verifica è il posizionamento della scheda sul banco di collaudo e l'avvio del test. I collegamenti sono interni al banco e l'operatore deve solo assicurarsi che la scheda sia bloccata correttamente.

1.2.2 Il banco di collaudo

Il banco di collaudo, studiato e progettato internamente all'azienda, ha il compito di simulare tutta la circuiteria che la UUT dovrà poi comandare una volta installata nel prodotto finito. Esso presenta nella parte superiore un letto ad aghi sul quale va adagiata e bloccata con l'apposito meccanismo la UUT. Gli aghi constano di un'estremità "a testa trasversale": una volta bloccata la scheda, tra le piazzole stagnate poste nella parte inferiore e la testa dell'ago si crea un contatto eccellente poiché le punte affilate penetrano nel metallo. Questi aghi sono l'ideale per apparecchiature automatiche di collaudo (ATE), maschere e via dicendo, laddove cioè sia necessario testare rapidamente gruppi di schede a circuito stampato. Nel banco di collaudo in questione gli aghi vengono montati a pressione in modo da creare una tenuta ermetica senza l'uso di collanti. Il vantaggio che ne deriva è la certezza della conduzione elettrica e la prevenzione di eventi indesiderati quali contatti accidentali verso piazzole e/o componenti vicini con conseguente danneggiamento della scheda. In Fig. 1.3 e Fig. 1.4 sono riportati rispettivamente il banco di collaudo e l'ago del tipo "a testa trasversale" mentre in Fig. 1.5 si mostra il sistema completo di scheda bloccata a banco (si notano i cavi di collegamento liberi inutilizzati per la presenza degli aghi).

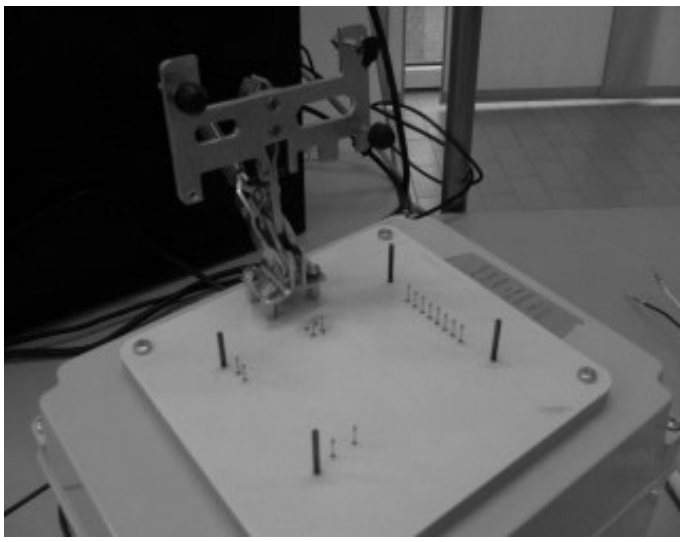


Fig. 1.3 Banco di collaudo utilizzato.

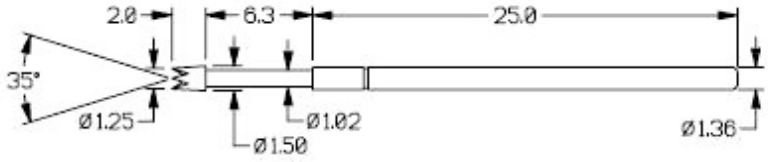


Fig. 1.4 Ago a testa trasversale.

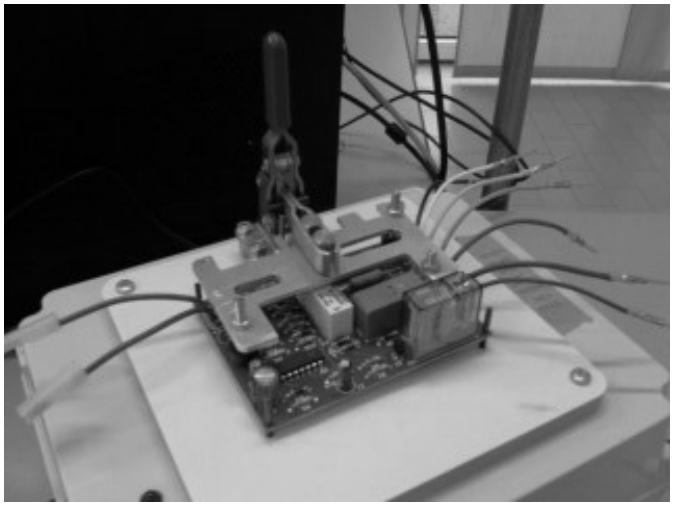
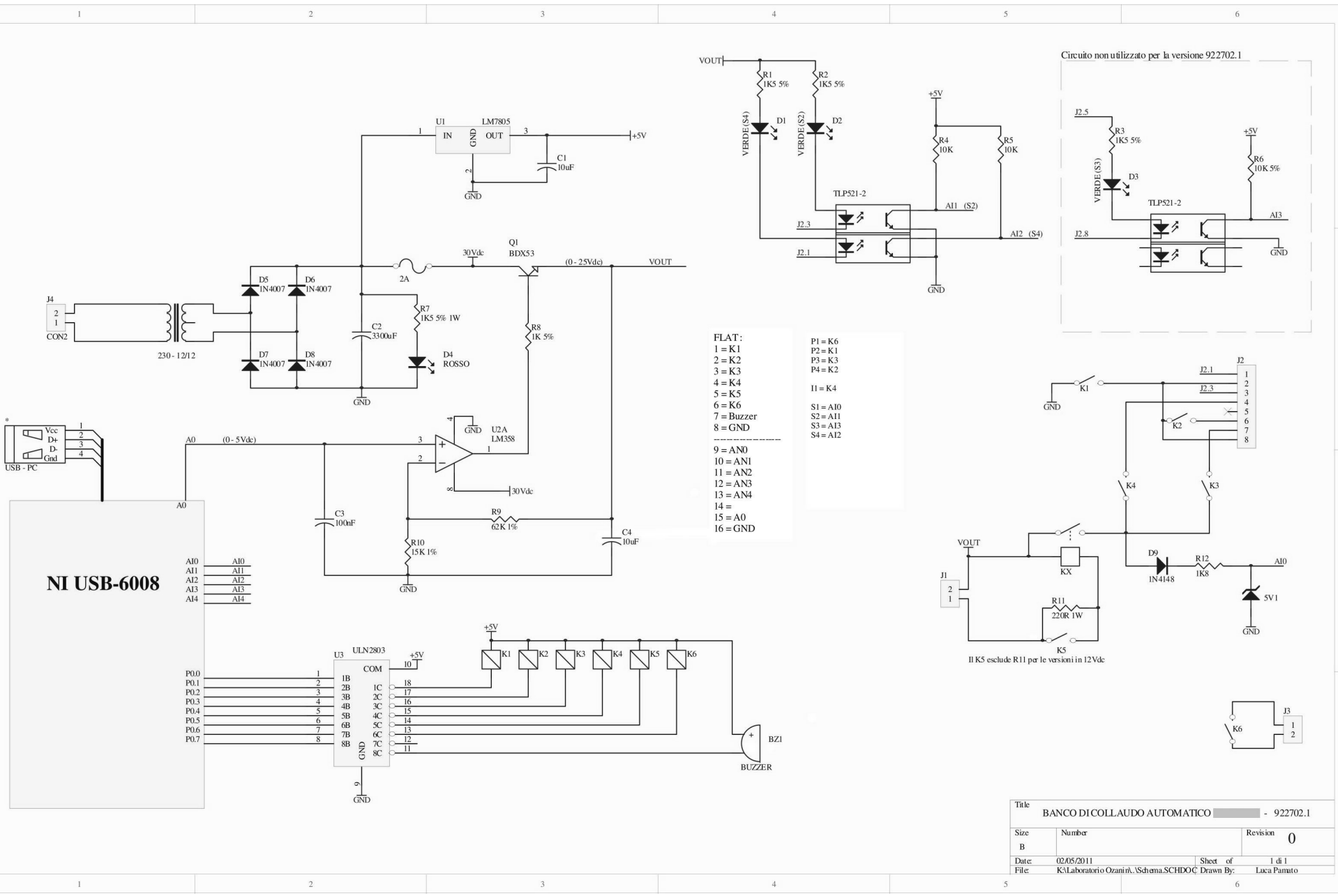


Fig. 1.5 La scheda sul banco di collaudo.

All'interno del banco di collaudo (Fig. 1.6) troviamo una parte per la trasformazione della tensione di rete, una parte di amplificazione tramite AO e la restante circuiteria, costituita principalmente da opto-isolatori, LED e relè, che simula tutto ciò che la UUT andrà a comandare una volta installata sul prodotto finito. Nel seguito verrà descritto il comportamento di tali configurazioni e saranno riportati i passaggi matematici svolti al fine di rendere più chiara la trattazione. Nella pagina seguente viene riportato lo schema elettrico del banco di collaudo in tutte le sue parti.


Circuito non utilizzato per la versione 922702.1
U1 LM7805 IN GND OUT +5V C1 10uF
Q1 BDX53
30Vdc (0 - 25Vdc) VOUT 2A
D5 1N4007 D6 1N4007 D7 1N4007 D8 1N4007
J4 CON2 230 - 12/12
R7 1K5 5% 1W C2 3300uF D4 ROSSO R8 1K 5%
GND
USB - PC Vcc D+ D- Gnd
A0 (0 - 5Vdc) U2A LM358 30Vdc
C3 100nF R9 62K 1% R10 15K 1% C4 10uF
NI USB-6008
AI0 AI1 AI2 AI3 AI4
P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7
U3 ULN2803 COM +5V
1B 2B 3B 4B 5B 6B 7B 8B GND
1C 2C 3C 4C 5C 6C 7C 8C
K1 K2 K3 K4 K5 K6
BZ1 BUZZER
VOUT R1 1K5 5% R2 1K5 5%
VERDE (S4) D1 VERDE (S2) D2
R4 10K R5 10K +5V
TLP521-2 J2.3 J2.1 AI1 (S2) AI2 (S4)
J2.5 R3 1K5 5% VERDE (S3) D3 R6 10K 5% TLP521-2 J2.8 AI3
FLAT :
1 = K1
2 = K2
3 = K3
4 = K4
5 = K5
6 = K6
7 = Buzzer
8 = GND
9 = AN0
10 = AN1
11 = AN2
12 = AN3
13 = AN4
14 =
15 = A0
16 = GND
P1 = K6
P2 = K1
P3 = K3
P4 = K2
I1 = K4
S1 = AI0
S2 = AI1
S3 = AI3
S4 = AI2
J2 J2.1 J2.3 K1 K2 K4 K3
VOUT KX J1 R11 220R 1W K5
D9 1N4148 R12 1K8 AI0 5V1
Il K5 esclude R11 per le versioni in 12Vdc
J3 K6
Title BANCO DI COLLAUDO AUTOMATICO - 922702.1
Size B Number Revision 0
Date: 02/05/2011 Sheet of 1 di 1
File: K:\Laboratorio Ozanin\..\Schema.SCHDOC Drawn By: Luca Pamato

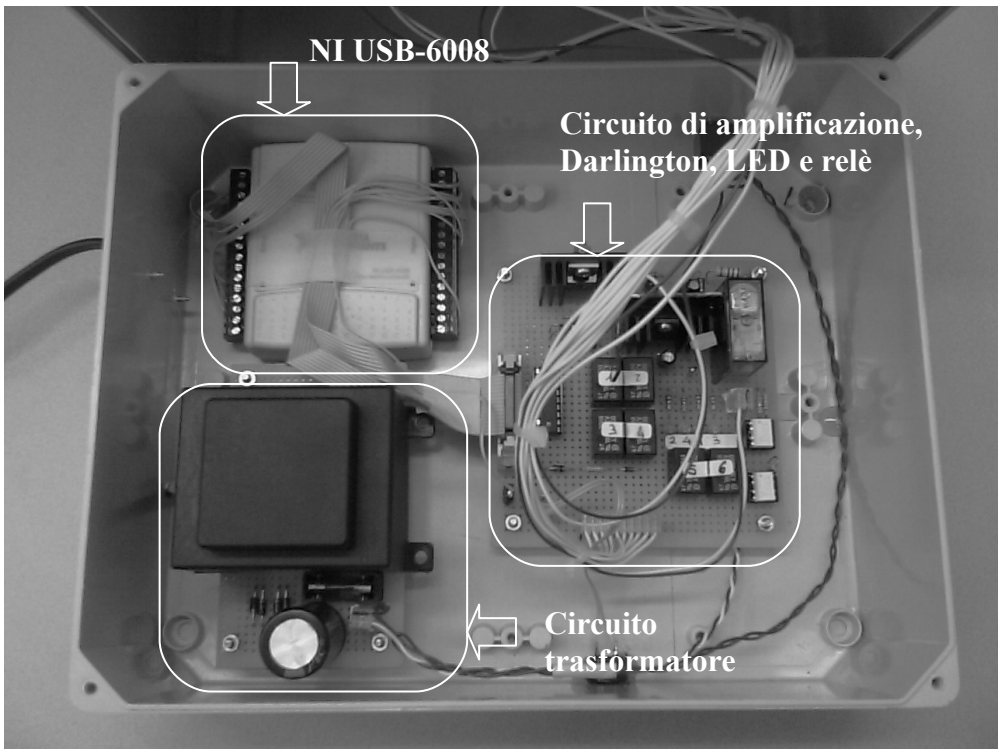


Fig. 1.6 Struttura interna del banco di collaudo.

- **Circuito di alimentazione e trasformazione**

  Il banco di collaudo viene alimentato alla tensione di rete 220[V] e utilizza un trasformatore 220–24[V] e relativo ponte a diodi per il raddrizzamento. Nel circuito di Fig. 1.7 è stato inserito volutamente un condensatore da 100[μF] per far risaltare il ripple in uscita, sebbene poi nel circuito vero e proprio si utilizza un condensatore da 3300[μF] che migliora notevolmente tale inconveniente (Fig. 1.9), permettendo di ottenere una tensione stabilizzata a 32[V].

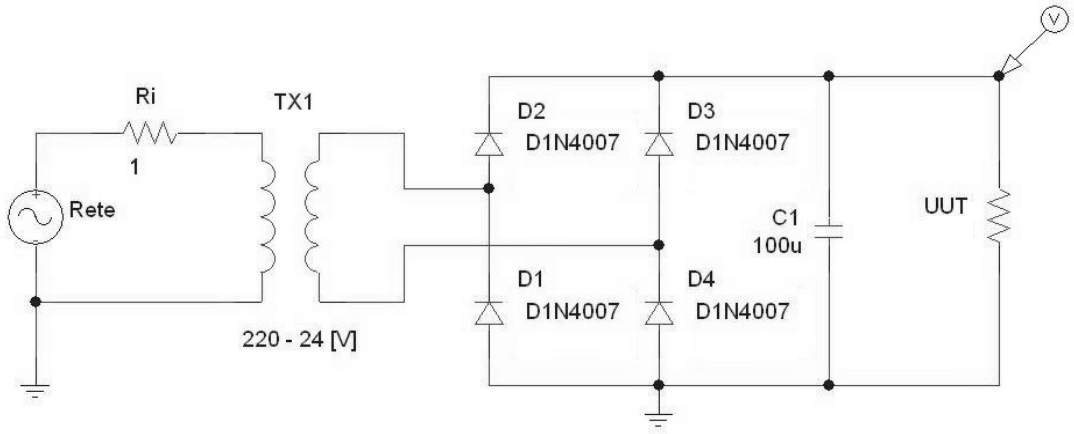


Fig. 1.7 Circuito di trasformazione e raddrizzamento della tensione di rete.

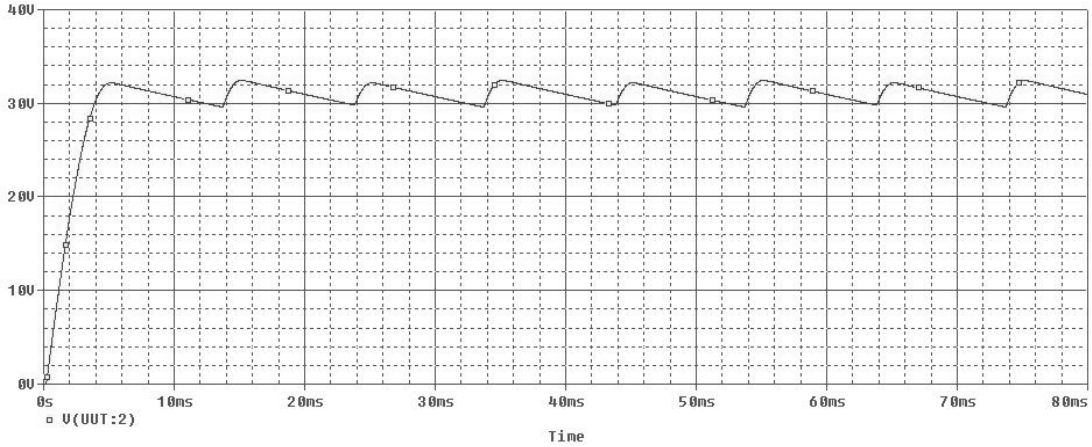


Fig. 1.8 Tensione sulla UUT con C=100[μF] e relativo ripple.

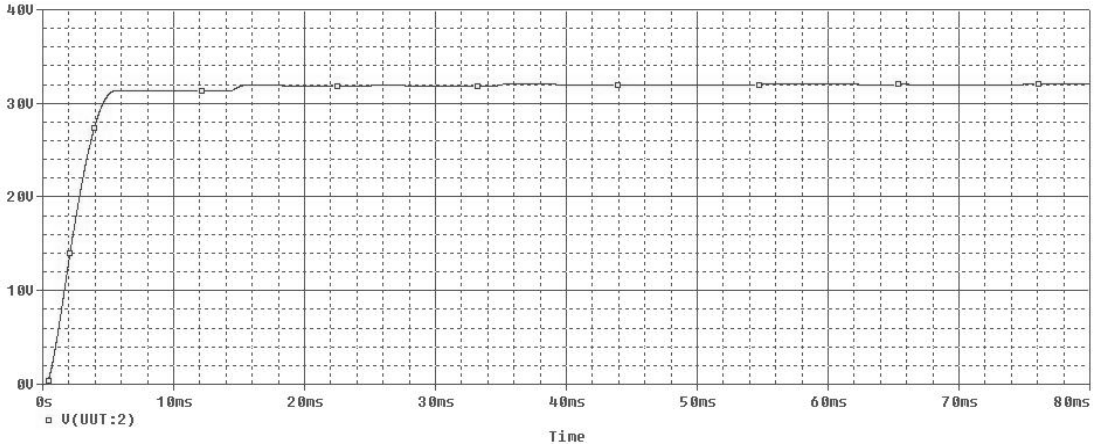


Fig. 1.9 Tensione 32[V] sulla UUT raddrizzata con C=3300[μF].

Il ripple della tensione stabilizzata è infatti ricavato dalla (1.1) e risulta

$$V_{ripple} = \frac{V_{max}}{f \cdot R_l \cdot C_l} = 0.194[V] \qquad (1.1)$$

dove V_{max} è la tensione riportata in Fig. 1.9, C_l è il condensatore da 3300[μF] e R è il carico (la UUT) considerando un assorbimento max da parte della scheda di 50[mA]. È inoltre presente un circuito regolatore di tensione che sfrutta un LM7805 per ottenere la tensione di 5[V], a partire dai 32[V] stabilizzati, per la logica di commutazione degli opto-isolatori che comandano gli ingressi digitali del DAQ USB-6008 e, di conseguenza, risultano essere gli ingressi che LabVIEW andrà ad interrogare in fase di test. L'opto-isolatore (Fig. 1.10) è un dispositivo che permette di trasferire un segnale fra due circuiti mantenendo l'isolamento galvanico fra gli stessi.

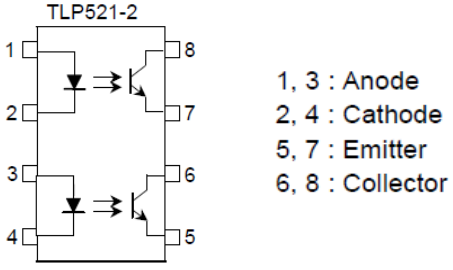


Fig. 1.10 La struttura dell'opto-isolatore TLP521-2.

Tale dispositivo viene normalmente realizzato con un accoppiamento ottico tra un LED ed un elemento fotosensibile: le variazioni di luminosità legate al segnale d'ingresso vengono rilevate dall'elemento fotosensibile ottenendo il trasferimento dell'informazione da un circuito all'altro senza che vi sia continuità elettrica (Fig. 1.11). Questa caratteristica risulta adatta al caso in esame poiché la tensione V_{out} fornita dal circuito di amplificazione non è una tensione fissa ma varia su tre livelli (20.5[V],22[V] e 24[V]) e l'informazione che interessa a livello di collaudo con LabVIEW è la semplice conduzione elettrica (e quindi un segnale ON-OFF) per stabilire se la scheda sia alimentata o meno.

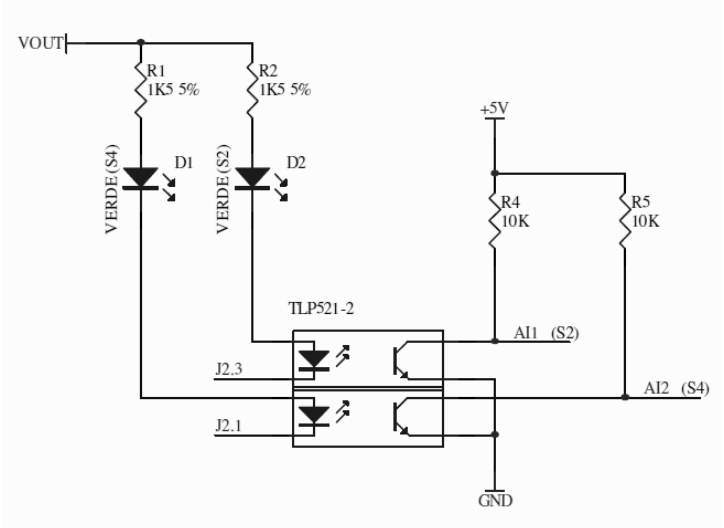


Fig. 1.11 Il circuito in cui è usato l'opto-isolatore: ci interessa l'informazione logica.

Si è verificato l'effettivo funzionamento del circuito su breadboard, collegando a V_{out} un alimentatore variabile 0-24[V] e connettendo $J_{2.3}$ e $J_{2.1}$ a GND. Variando tramite manopola la tensione di alimentazione si nota la commutazione 5-0[V] ai connettori A_{I1} e A_{I2} sullo schermo dell'oscilloscopio non appena si supera la soglia dei 3[V]. Ciò che si ottiene è praticamente un disaccoppiamento tra l'informazione d'interesse e il segnale d'ingresso: tale soluzione circuitale è stata scelta in sostituzione della configurazione diodo-resistore-Zener per la variabilità della tensione di ingresso, la quale può causare nello Zener un livello di tensione che non è quello desiderato (stabile) all'ingresso del DAQ. I led S_2 ed S_4 vengono quindi utilizzati solo in fase di debug a "banco aperto" per l'immediata verifica della loro attivazione e non per contenuto informativo ai fini del test LabVIEW.

- **Circuito di amplificazione**

il circuito di amplificazione è stato riportato in Fig. 1.13 tramite simulazione con software PSpice. Esso è composto da due alimentazioni DC a 30[V] e 5[V] che simulano rispettivamente la tensione proveniente dal circuito trasformatore e quella proveniente dal NI USB-6008. Troviamo poi un AO in configurazione non invertente che va a comandare un transistor di tipo Darlington: questo dispositivo ha la particolarità di essere composto da due transistor a giunzione bipolare in cascata, permettendo un elevatissimo guadagno in corrente pari al prodotto dei guadagni in corrente dei singoli transistor. Per contro presentano una caduta di tensione doppia rispetto i normali transistor ed è perciò necessario dimensionare il guadagno di retroazione dell'AO tenendone conto. Nel caso in esame il guadagno risulta essere 5.13, calcolato dalla formula (1.2)

$$A = \frac{V_o}{V_i} = \left(1 + \frac{R_2}{R_1}\right) = \left(1 + \frac{62}{15}\right) = 5.13 \qquad (1.2)$$

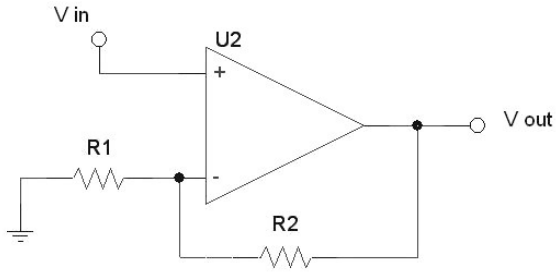


Fig. 1.12 AO configurazione non invertente.

in modo da avere un certo margine di manovra per la tensione finale, come riportato in Fig.1.13 in cui si vede che la tensione di base del Darlington è di 26.52[V] e quella finale sull'emettitore, e quindi sul carico ovvero la UUT da alimentare, è 25.68[V]. In questo modo attraverso l'operazionale ho la possibilità di comandare il Darlington per generare i tre livelli di tensione necessari per il collaudo.

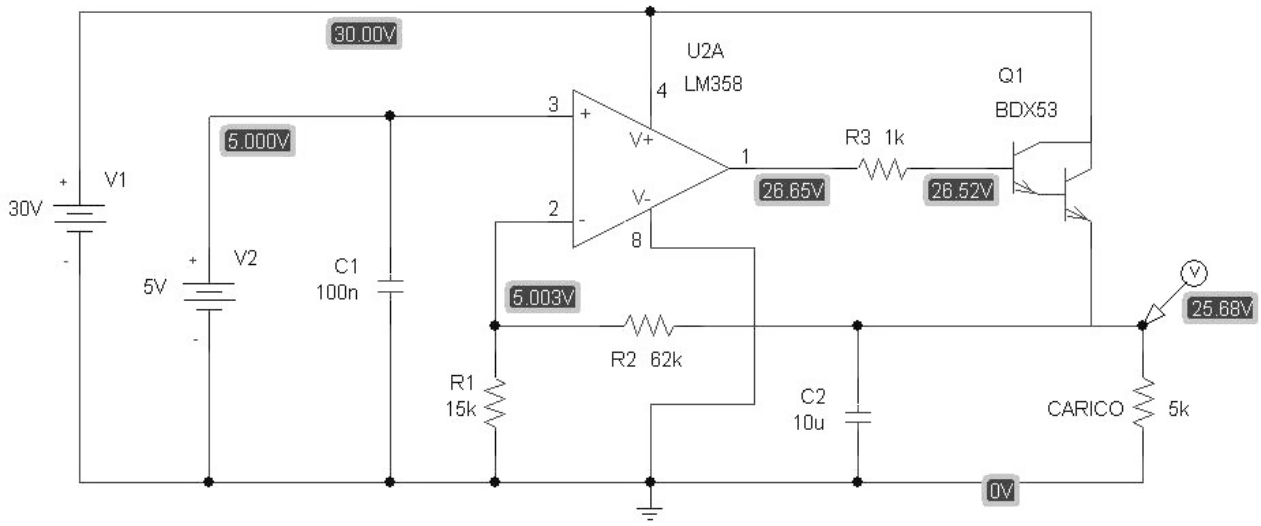


Fig. 1.13 Circuito di amplificazione con AO non invertente.

1.2.3 NI USB-6008

Il NI USB-6008 è un DAQ con 8 ingressi analogici (AI) ±10[V] single ended o ±20[V] differenziale, 2 uscite analogiche (AO), 12 ingressi/uscite digitali (DIO), un contatore a 32bit, risoluzione 12bit e frequenza di campionamento massima di 10[kS/s] collegabile al PC tramite interfaccia USB full-speed. Il suo compito è quello di comandare un buffer e quindi dei relè di potenza tramite le sue uscite digitali e di fornire in uscita una tensione analogica continua da 0÷5[V] che andrà poi amplificata per alimentare la UUT a tre differenti livelli di tensione secondo le specifiche di collaudo. In Fig. 1.14 è riportato lo schema interno del DAQ in cui si notano i due ADC ad elevata precisione per le uscite analogiche e i bassi valori di corrente in uscita del dispositivo.

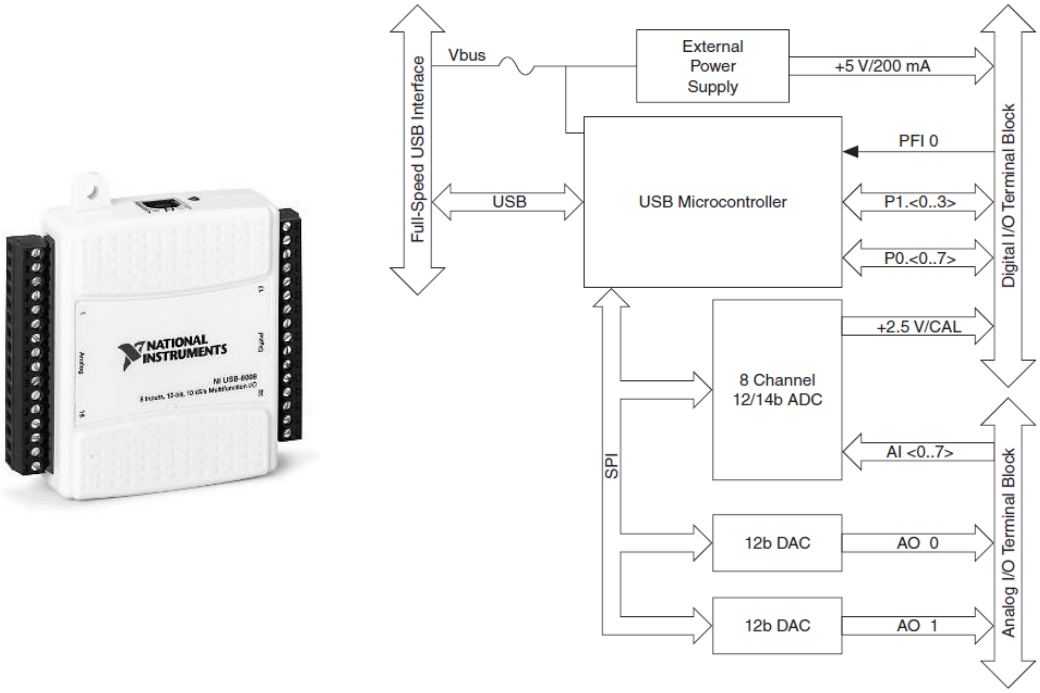


Fig. 1.14 NI USB-6008: componenti interni e relativi ingressi ed uscite.

ANALOG INPUT	
inputs	8 single-ended, 4 differential, software selectable
resolution	12 bits differential, 11 bits single-ended
sampling rate	10 [kS/s]
Input range	Single-ended ±10[V]
	Differential ±20[V], ±10[V], ±5[V], ±4[V], ±2.5[V], ±2[V], ±1.25[V], ±1[V]
Input impedance	144 [kΩ]
Working voltage	±10[V]
ANALOG OUTPUT	
resolution	12 bits
Output range	0 to +5[V]
Output impedance	50[Ω]
Slew rate	1[V/µs]
DIGITAL I/O	
P0.<0..7>	8 lines
P1.<0..3>	4 lines
Direction control	Each channel individually programmable as input or output
Compatibility	TTL, LVTTL, CMOS
Absolute maximum voltage range	–0.5 to 5.8[V] with respect to GND
COUNTER	
Number of counters	1
Resolution	32 bits
Counter direction	Count up
BUS INTERFACE	
USB specification	USB 2.0 Full-Speed
USB bus speed	12[Mb/s]

Tab. 1.1 Caratteristiche salienti del NI USB-6008.

Come si può vedere dalla Tab. 1.1, tramite LabVIEW e la porta USB si ha la possibilità di interfacciarsi con il DAQ e controllare tutti gli ingressi e le uscite del sistema di collaudo in maniera istantanea.

1.3 Procedura di collaudo

La documentazione fornita in azienda è costituita da schede tecniche e linee guida del cliente per l'assemblaggio, le saldature, la marcatura di etichette e via dicendo. Il cliente specifica lo stato della scheda come da Tab. 1.2. Viene anche fornito un flow chart (Fig. 1.15) delle operazioni da eseguire per verificare l'idoneo funzionamento della scheda.

Stato scheda	*Verifica punti di controllo*
ON	Spie S1 e S4 accese Spia S2 spenta
OFF	Spie S1 e S4 spente Spia S2 accesa

Tab. 1.2 Stato della scheda da collaudare.

I pulsanti di Fig. 1.2 si devono intendere nella condizione stabile. Il loro azionamento deve essere momentaneo, della durata di 500[ms]. Nel diagramma di flusso le tensioni di alimentazione vengono indicate dalle sigle riportate nella tabella seguente.

Sigla tensione di alimentazione	*Descrizione caratteristica*
Vout=24[V]	Tensione nominale di funzionamento
Vout=22[V]	Tensione di soglia MAX per intervento automatico
Vout=20.5[V]	Tensione di soglia MIN per intervento automatico

Tab. 1.3 Tensioni di alimentazione scheda.

Ciò significa che tramite NI USB-6008 dovrò fornire in uscita le tensioni opportune secondo il coefficiente di amplificazione precedentemente calcolato con circuito non invertente. In Tab. 1.4 vengono riportati i valori di tensione che si devono fornire all'uscita analogica del DAQ per i tre livelli di tensione desiderati, calcolati con la semplice proporzione (1.3)

$$25 : 5 = 24 : x \quad [V] \tag{1.3}$$

in cui al primo membro si indicano il massimo valore in Volt ottenibile dal circuito amplificatore e il massimo valore in Volt dell'uscita analogica del DAQ mentre al secondo membro si specifica la tensione desiderata per alimentare la scheda, ottenendo quindi la tensione analogica che il DAQ dovrà fornire.

Tensione A0 USB-6008	*Tensione corrispondente*
VAO=4.8[V]	Vout=24[V]
VAO=4.4[V]	Vout=22[V]
VAO=4.1[V]	Vout=20.5[V]

Tab. 1.4 Tensioni di uscita del NI USB-

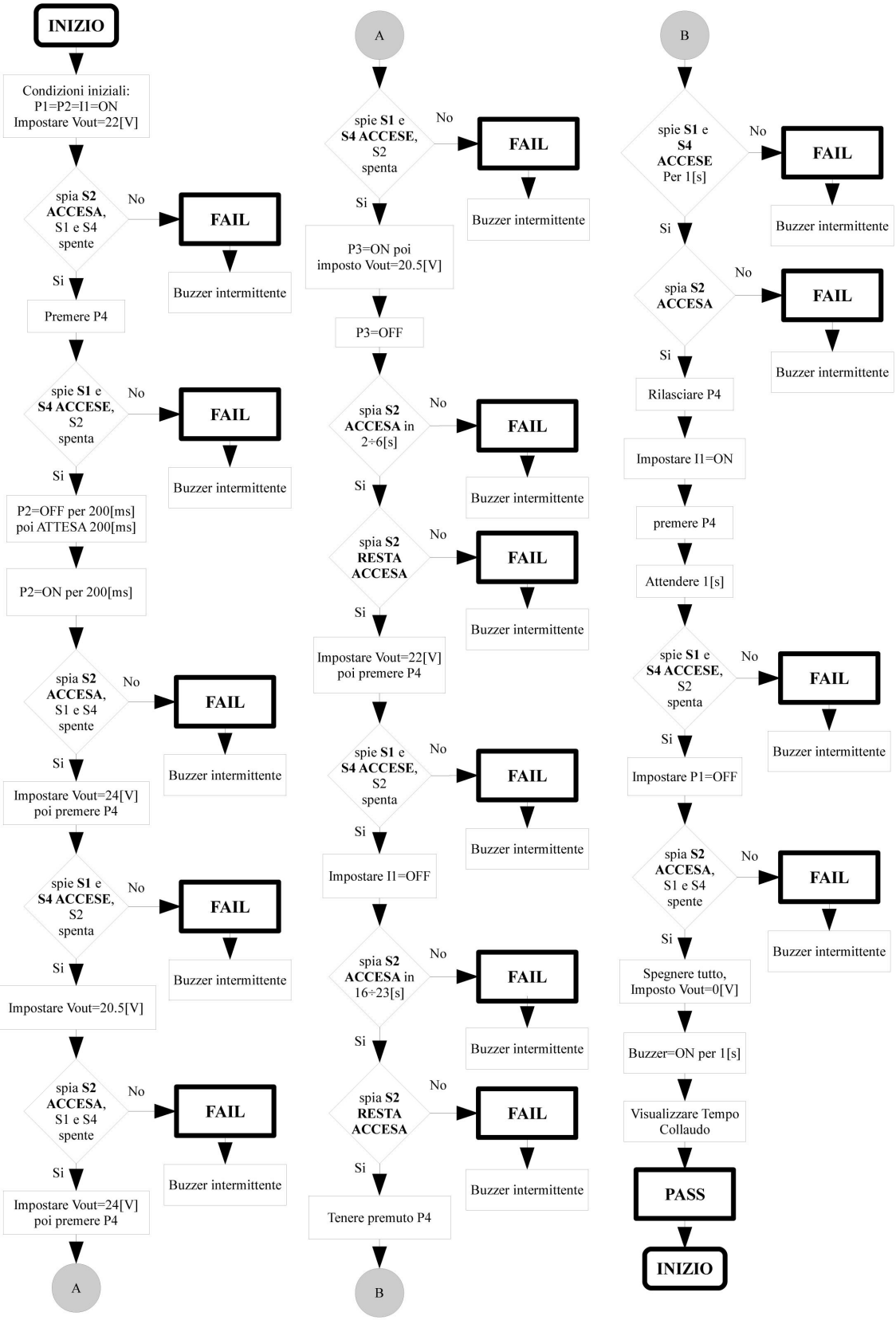


Fig. 1.15 Il flow chart dei controlli da eseguire in fase di test.

Capitolo 2

Interfaccia LabVIEW

2.1 LabVIEW

LabVIEW (abbreviazione di Laboratory Virtual Instrumentation Engineering Workbench) è l'ambiente di sviluppo integrato per il linguaggio di programmazione visuale della National Instruments nato nel 1983 dalla necessità di disporre di un software grafico, con il quale testare rapidamente gli apparati hardware prodotti internamente. NI ha rivoluzionato il modo di lavorare di tecnici e ricercatori attraverso l'utilizzo del personal computer. L'azienda sviluppa e produce centinaia di prodotti software e hardware integrati che, affiancati ad un computer, permettono di sostituire la strumentazione tradizionale o di comunicare con essa, monitorandone e controllandone i processi. LabVIEW è un ambiente di sviluppo per applicazioni principalmente orientate ad

- acquisizione, analisi, visualizzazione, elaborazione, memorizzazione di segnali e dati
- gestione di strumentazione elettronica
- simulazione ed emulazione di sistemi fisici.

Si presenta come un ambiente di programmazione grafico ad oggetti, che consente di realizzare i programmi in forma block diagram e per questa ragione è battezzato *G-Language* (*Graphic Language*). La semplicità di programmazione (intuitiva in quanto modellata sulla logica di un flow chart), la semplicità di utilizzo (l'utente finale dispone di una strumentazione virtuale disegnata a schermo) e la grande versatilità hanno reso LabVIEW molto impiegato e diffuso nell'ambito dell'acquisizione dei dati e nel loro controllo nei processi industriali. La definizione di strutture dati ed algoritmi avviene anch'essa con icone e oggetti grafici uniti da linee di collegamento (*wire*) o bus dati disegnate dal programmatore. Poiché i dati possono anche scorrere in parallelo attraverso blocchi e fili non consecutivi, il linguaggio realizza spontaneamente il multithreading senza bisogno di un'esplicita gestione. Il programmatore ha a disposizione una serie di librerie che possono essere richiamate ed utilizzate all'interno dei programmi, le quali comprendono:

- funzioni aritmetiche, statistiche, logiche, comparative e di manipolazione di stringhe
- funzioni specializzate per l'acquisizione e l'elaborazione dei segnali
- il controllo di strumentazione numerica via interfaccia IEEE-488.2 (GPIB), VXI o PXI
- la trasmissione di dati mediante l'uso di porte seriali (RS-232 o RS-485) oppure mediante il protocollo di comunicazione TCP/IP.

2.2 Il VI in LabVIEW

I programmi realizzati in LabView prendono il nome di VI (Virtual Instrument) poiché nell'aspetto e nel funzionamento sono simili a strumenti fisici, ma l'interazione avviene attraverso un programma in esecuzione, che simula il funzionamento del dispositivo fisico (oscilloscopio, multimetro, ecc.). L'utente può modificare il valore di alcune grandezze agendo su opportune manopole o interruttori visualizzati e osservare istantaneamente il risultato delle elaborazioni su display e grafici a schermo, molto simili a quelli che si trovano sulla strumentazione numerica/digitale. Un VI non è un codice sotto forma di testo, bensì un file binario visualizzabile e compilabile solo da LabVIEW.

I VI sono formati principalmente da tre parti:

- Front Panel (pannello frontale)
- Block Diagram (schema a blocchi)
- Riquadro Icona-Connettori

Il Front Panel è la finestra che rappresenta l'interfaccia per l'utente finale. Si realizza con controlli e indicatori, che costituiscono rispettivamente i terminali interattivi d'ingresso e d'uscita. Tra i controlli troviamo manopole, potenziometri, pulsanti, interruttori, quadranti ecc. i quali simulano i dispositivi d'ingresso degli strumenti di misura e forniscono dati verso il Block Diagram del VI. Gli indicatori invece possono essere LED, grafici, tabelle, termometri, indicatori di livello ecc. i quali simulano i dispositivi d'uscita degli strumenti e visualizzano i dati che il Block Diagram acquisisce o genera. Un esempio di controlli e indicatori è riportato in Fig. 2.1.

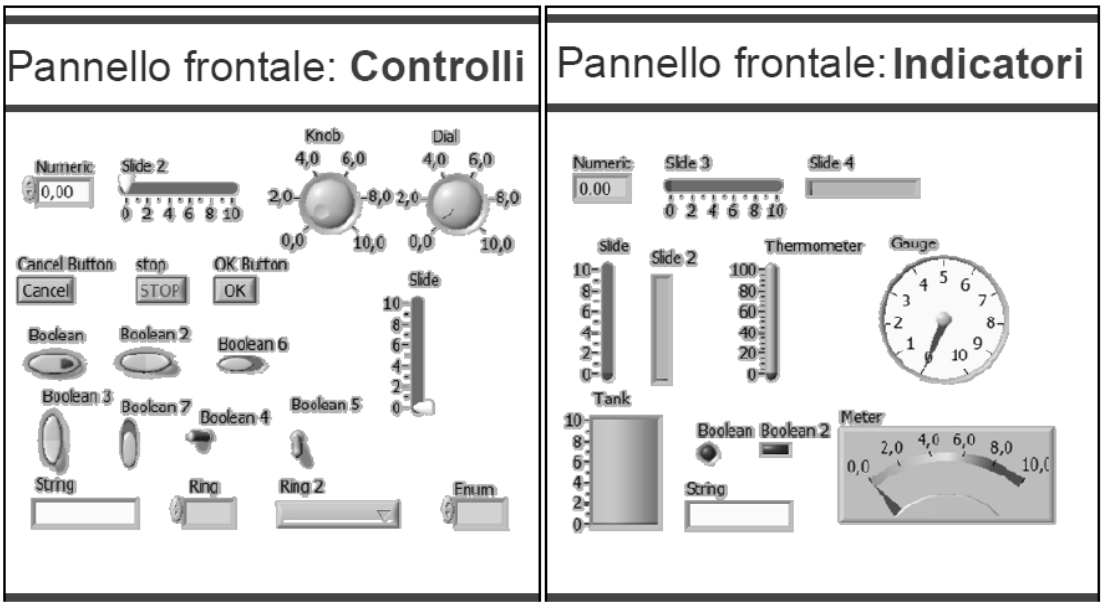


Fig. 2.1 Controlli e indicatori del Front Panel.

Il Block Diagram è il diagramma di flusso che rappresenta il codice sorgente in formato grafico. Gli oggetti utilizzati nel Front Panel appaiono qui come terminali, ovvero porte di ingresso e di uscita che scambiano informazioni tra i due "livelli" e sono del tutto analoghi alle variabili dei linguaggi di programmazione testuali. Le varie tipologie di terminali che si possono incontrare sono:

- ***terminali dei controlli e degli indicatori:***
  I dati immessi dall'utente nei controlli ubicati sul pannello frontale vengono trasmessi allo schema a blocchi grazie ai terminali creati in automatico dal programma. Una volta che i dati vengono elaborati si ha il procedimento inverso poiché il dato che parte dal Block Diagram viene trasmesso al Front Panel attraverso il terminale dell'indicatore (Fig. 2.2);

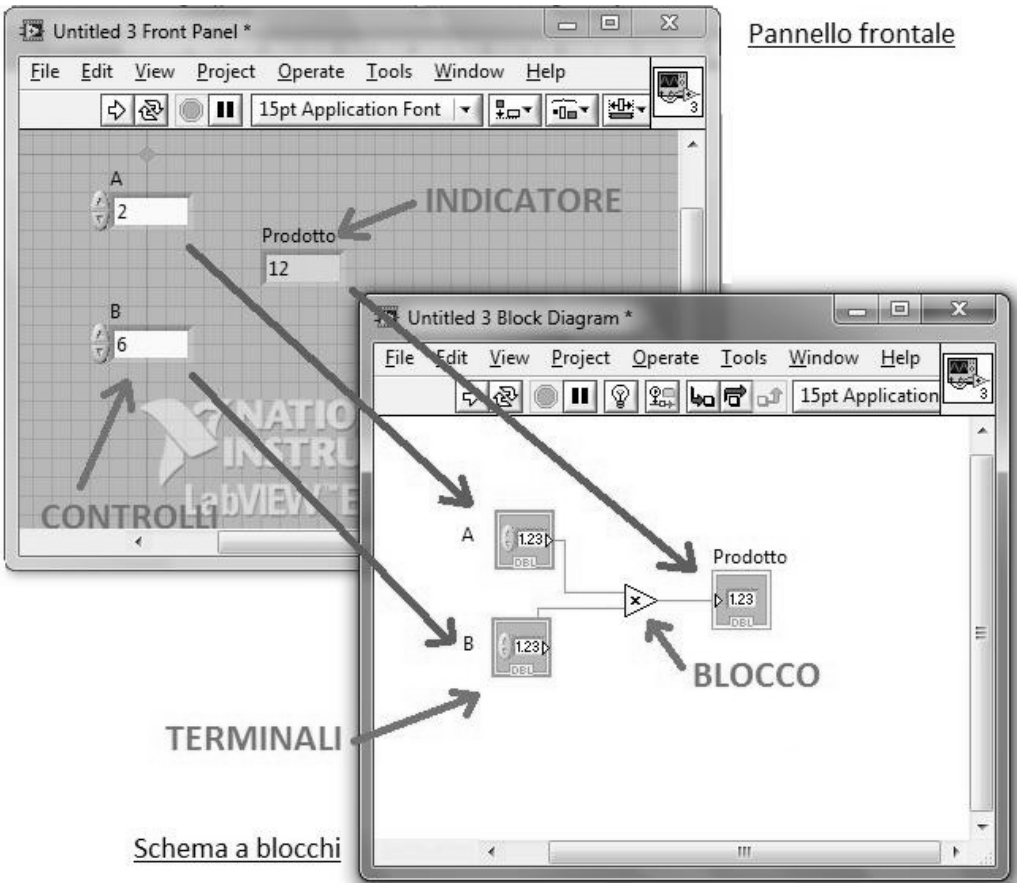


Fig. 2.2 Collegamento tra Block Diagram e Front Panel.

- ***funzioni o blocchi:***
  Sono gli elementi fondamentali costituenti il linguaggio di programmazione LabVIEW. Si possono selezionare da appositi menù a tendina e rappresentano le unità di elaborazione con le quali l'utente programmatore crea il codice. Le funzioni non hanno pannelli frontali o schemi a blocchi ma presentano solo connettori di ingresso e/o di uscita;

- ***fili di collegamento (wire):***
  Possono trasportare qualunque mole di dati e di qualunque tipo, anche aggregati (bundle), definiti dal programmatore. Il colore e lo spessore del filo cambiano in base alla tipologia e alla quantità di dati circolanti permettendone una facile identificazione. Ad esempio gli interi scorrono su fili blu e le stringhe su fili rosa, gli scalari su filo sottile, gli array su filo spesso e le matrici su filo doppio;

- ***subVI:***
  Sono VI costruiti per essere utilizzati all'interno di un altro VI. Ogni VI può essere potenzialmente utilizzabile come subVI a patto che lo si munisca di connettori ingresso-uscita tramite un apposito riquadro, detto riquadro connettori.

- ***commenti testuali:***
  Piccoli box testuali con cui si possono specificare particolari e modifiche, utili soprattutto per il programmatore in fase di progetto.

Il Riquadro Icona-Connettori, come precedentemente detto, è utilizzato in LabVIEW nel momento in cui un VI venga sfruttato come subVI in un altro VI. Per poter fare ciò occorre definire le variabili che il subVI riceve in ingresso e quelle che restituisce in uscita specificando i rispettivi controlli e indicatori a partire dal riquadro connettori del Front Panel. L'icona, completamente personalizzabile, è invece il simbolo che si applica al subVI per il suo riconoscimento all'interno del programma. In Fig. 2.3 è riportato un particolare di un VI in cui si vedono il riquadro icona e quello connettori applicati ad un subVI.

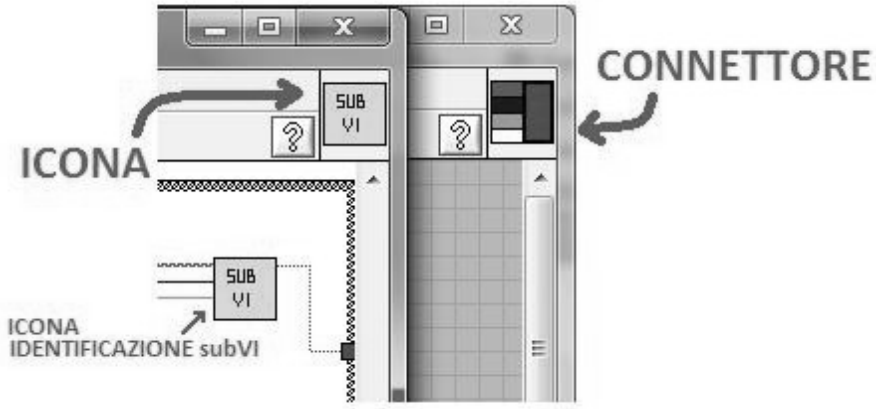


Fig. 2.3 Utilizzo del riquadro connettori di un subVI.

2.3 Tipi di dato e strutture

LabVIEW mette a disposizione un'ampia gamma di tipologie di dati (Fig. 2.4). Ad ogni tipo di dato è associato uno specifico colore (verde per i booleani, blu per gli interi, arancio per i double, ecc.) così come ai collegamenti che uniscono oggetti e funzioni che operano con dati dello stesso genere. I vari dati si possono a loro volta organizzare in matrici, vettori o Cluster (un insieme eterogeneo di dati). Va inoltre specificato che nella filosofia di LabVIEW i dati viaggiano in pacchetti (data packets) per cui se un pacchetto, per portare l'informazione dovuta, ha la necessità di unire un array con una stringa il programma realizza in automatico un cluster formato da tali elementi.

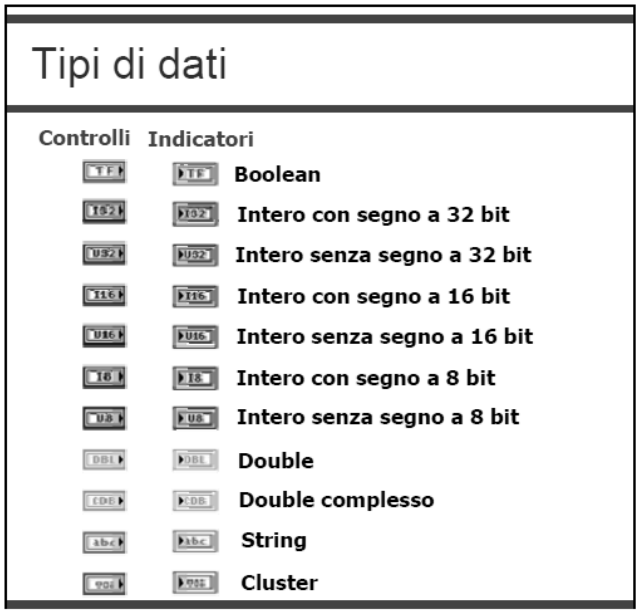


Fig. 2.4 Utilizzo del riquadro connettori di un subVI.

Oltre ai vari tipi di dato in LabVIEW sono presenti anche diversi tipi di strutture, molto simili ai cicli dei linguaggi testuali, sfruttabili dall'utente per la programmazione. Le principali sono riportate in Fig. 2.5.

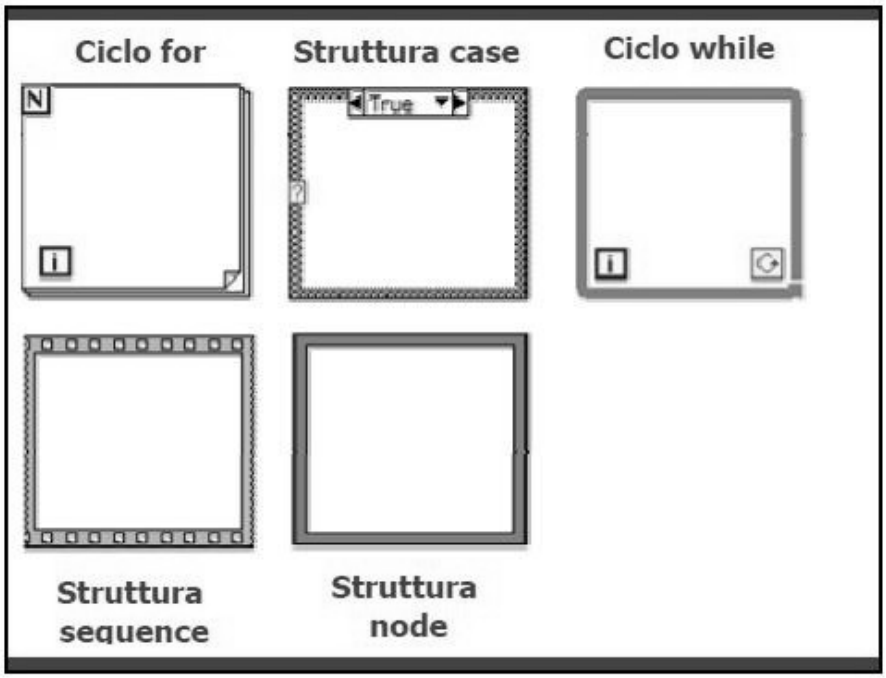


Fig. 2.5 Principali strutture di LabVIEW.

Ogni struttura o ciclo viene rappresentata con una diversa veste grafica: l'area delimitata dal bordo dimensionale appartenente al corpo del ciclo, mentre i terminali per mezzo dei quali le strutture si connettono ad altri blocchi o strutture sono detti tunnel. Se al tunnel d'ingresso sono disponibili dati validi la struttura viene eseguita automaticamente e al termine dell'elaborazione essa fornisce il risultato al tunnel di uscita. Occorre precisare che i dati d'ingresso vengono letti una sola volta all'inizio del ciclo e che i dati in uscita vengono aggiornati solo quando il ciclo ha termine (come per i linguaggi testuali). Si descrivono brevemente i principali tipi di strutture:

- ***Ciclo For***
  Struttura di controllo che esegue iterativamente una porzione di programma per un numero di volte stabilito dall'utente. Esso si distingue dalle altre strutture per mezzo dei terminali “N” e “i” chiamati rispettivamente Count Terminal, indicante il numero di iterazioni da eseguire e quindi variabile d'ingresso, e Iteration Terminal, indicante il numero di iterazioni completate e quindi variabile d'uscita.

- ***Ciclo While***
  Ripete una porzione di codice fino al verificarsi di una determinata condizione. Il ciclo viene eseguito fino a quando il Conditional Terminal (variabile di ingresso booleana) assume un determinato valore. L’Iteration Terminal contiene invece il numero di iterazioni completate ed è identico a quello visto per il ciclo For.

- ***Case o If***
  Eseguono una parte di codice piuttosto che un’altra a seconda del valore assunto dalla variabile di controllo detta Selector Terminal. Per ottenere una struttura If è sufficiente collegare il Selector Terminal ad una variabile di tipo Boolean. Una struttura Case, invece, si crea con una variabile di tipo Integer, String o di un altro tipo comunque enumerabile.

- ***Struttura Sequence***
  Permettono di controllare il flusso di esecuzione del programma. Essa contiene uno o più sotto-diagrammi (detti frame) che vengono eseguiti in sequenza. Esistono due tipi di queste strutture, quelle Flat e quelle Stacked, dove le prime visualizzano le operazioni da svolgere in sequenza tutte insieme e una a fianco dell'altra, mentre le seconde una alla volta.

- ***Struttura Node***
  consente di sfruttare formule ed espressioni derivate dal linguaggio C all'interno del Block Diagram invece di utilizzare i blocchi e le normali funzioni presenti in LabVIEW.

2.4 Acquisizione dei dati e comunicazione con NI-USB6008

Per testare il corretto funzionamento del NI USB-6008 prima della sua applicazione all'interno del banco si utilizza il software NI MAE (Measurement & Automation Explorer). Esso è praticamente un riconoscitore di periferica nel quale vengono mostrati tutti i dispositivi NI collegati al PC. In Fig. 2.6 è riportata la schermata di Test Panel del MAE che permette di controllare la periferica collegata e settarne eventuali ingressi od uscite per controlli preliminari. Occorre tener presente che l'utilizzo del MAE in questa parte di collaudo ha il solo scopo di verificare che il DAQ sia in buono stato. Per la comunicazione tra LabVIEW e USB-6008 si utilizzeranno poi degli oggetti presenti in una libreria specifica di LabVIEW, la NI-DAQmx Data Acquisition, aggiornabile tramite il software fornito da NI con il DAQ.

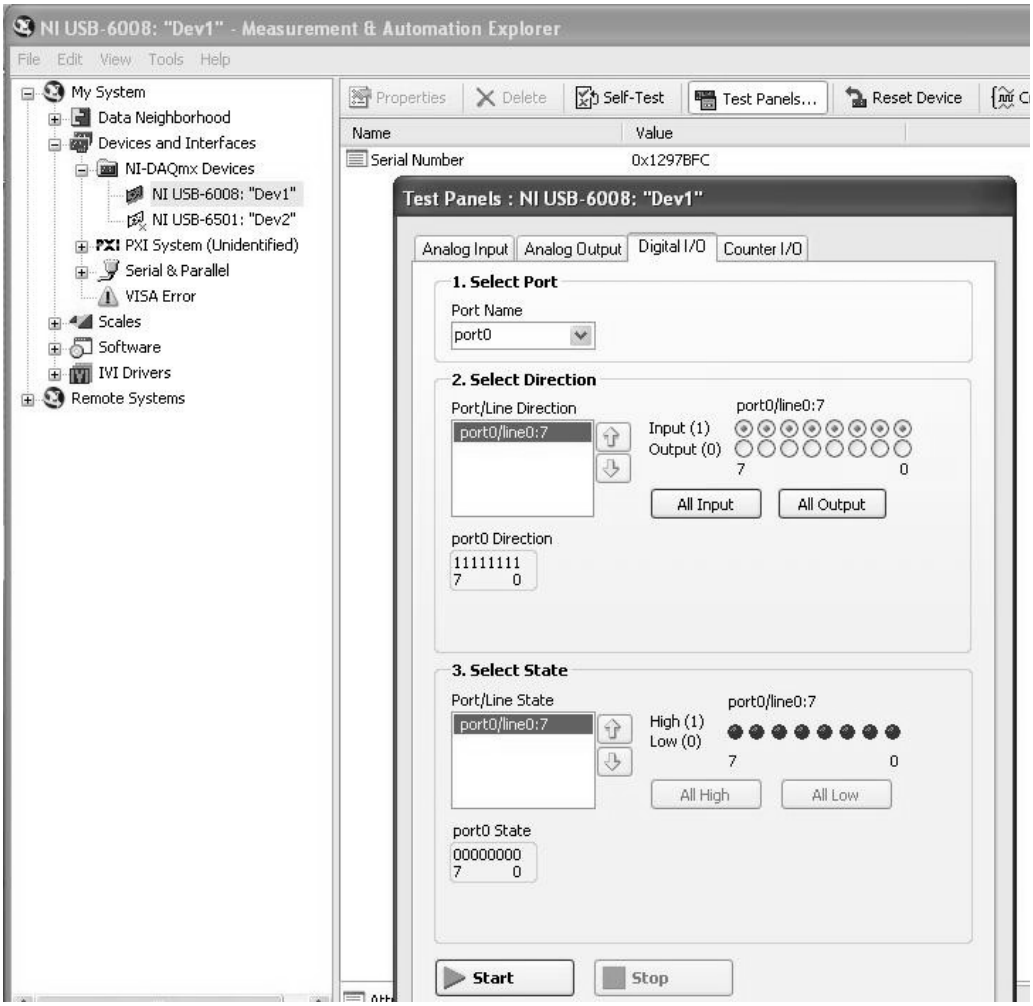


Fig. 2.6 Il Test Panel del NI MAE.

Tale libreria contiene al suo interno oggetti per la lettura e scrittura di dati sul DAQ, la gestione di tempistiche, pause e ritardi nonché livelli di trigger oltre a funzioni speciali per la taratura e il reset del dispositivo. Questi oggetti sono completamente personalizzabili in termini di indirizzamento a porte o bit, quantità e tipo di dati e quindi estremamente versatili. I tempi di risposta del dispositivo ai comandi di LabVIEW sono praticamente immediati e l'acquisizione delle tensioni è precisa al decimo di Volt, più che sufficiente per questo tipo collaudo dove non si richiede una precisione assoluta. Tali oggetti verranno descritti e mostrati nel seguito della trattazione.

Capitolo 3

Descrizione del codice in LabVIEW

3.1 Impostazione del codice

Per cominciare col programma che andrà a comandare l'intero collaudo occorre stabilire una linea guida di riferimento che resterà sempre la stessa per tutto il codice creato. Dal Flow Chart di Fig. 1.15 si capisce che il test è di tipo sequenziale e quindi la struttura preferenziale che andremo a trattare con LabVIEW sarà la Case (si riveda il paragrafo 2.3). Con essa è possibile eseguire uno step alla volta in successione e comandare a piacimento i salti agli step desiderati. Grazie alla caratteristica di enumerabilità propria di questa struttura basterà aggiornare il Case Selector una volta terminato lo step in esecuzione. Altra importante questione affrontata è stata quella dell'annidamento del codice: come per tutti i linguaggi di programmazione si è reso necessario l'inserimento di strutture le une dentro le altre, in modo da creare un codice comunque complesso ma estremamente versatile soddisfacente le casistiche di ciascuno step. Nota di merito per LabVIEW è proprio la possibilità di inserire senza alcun limite apparente (tranne forse una minima lentezza di esecuzione del codice) più strutture per così dire “a matriosca”, diversamente da altri linguaggi quali ad esempio il C, nel quale è piuttosto complicato, se non impossibile, eseguire una porzione di codice che contempli l'esecuzione di un ciclo While internamente ad un Case, internamente ad una struttura Sequence, internamente ad una struttura Case, quest'ultima all'interno di un altro ciclo While (tale stratagemma è davvero presente nel codice creato per questo collaudo, non è un'invenzione!). Si sfrutterà inoltre il parallelismo, altra grande qualità di LabVIEW, che permette l'esecuzione simultanea di due parti di codice distinte che possono eseguire compiti completamente differenti previo inserimento di ritardi uguali nelle due per creare un sincronismo di esecuzione. Occorre specificare poi che l'idea di fondo nella creazione di ciascun subVI è quella di generalizzare al massimo, intendendo con ciò che un subVI che riesce a funzionare in vari case con ingressi differenti producendo la stessa tipologia di risultati è da preferire a singoli case adattati a ciascun caso di elaborazione.

3.2 Il file dei parametri di collaudo

Come base di partenza per il collaudo si definiscono tutte le soglie di accettazione per le varie misurazioni effettuate in ogni step. Nello specifico tali soglie si riferiscono allo stato dei led di controllo del banco ed alla misurazione dei tempi di risposta della UUT. Si noti che non vengono trattate soglie di tensione poiché tramite multimetro si è verificato che il NI USB-6008 fornisce una tensione analogica di uscita molto precisa per i vari livelli di alimentazione della scheda richiesti. Con la possibilità tramite LabVIEW di acquisire, riconoscere ed elaborare specifiche stringhe nei file .txt si è deciso di creare un file di testo (Fig. 3.1) contenente tutte le soglie di collaudo. Un primo accorgimento si ha nello stabilire una formattazione comune a tutte le fasi di test senza che vi sia omissione di informazione e possibilità di errori in fase di lettura. Si rende quindi necessario l'uso di separatori, i punti e virgola in questo caso, affinché il programma riconosca come dato utile tutto ciò che vi è compreso. Nella formattazione di Fig. 3.1 è presente inoltre uno specifico ordine in modo da stabilire in maniera univoca l'informazione da elaborare (numero step; descrizione step;limite min;limite max) con l'opportunità quindi di lavorare tramite indicizzazione delle stringhe acquisite. Si noti che in Fig. 3.1, per quanto riguarda la verifica dei LED, i limiti min e max coincidono volutamente. Tale espediente sarà utile nel seguito quando verranno trattati i blocchi per la verifica della tolleranza di lettura poiché si è preferito farli generici e compatibili per ogni step.

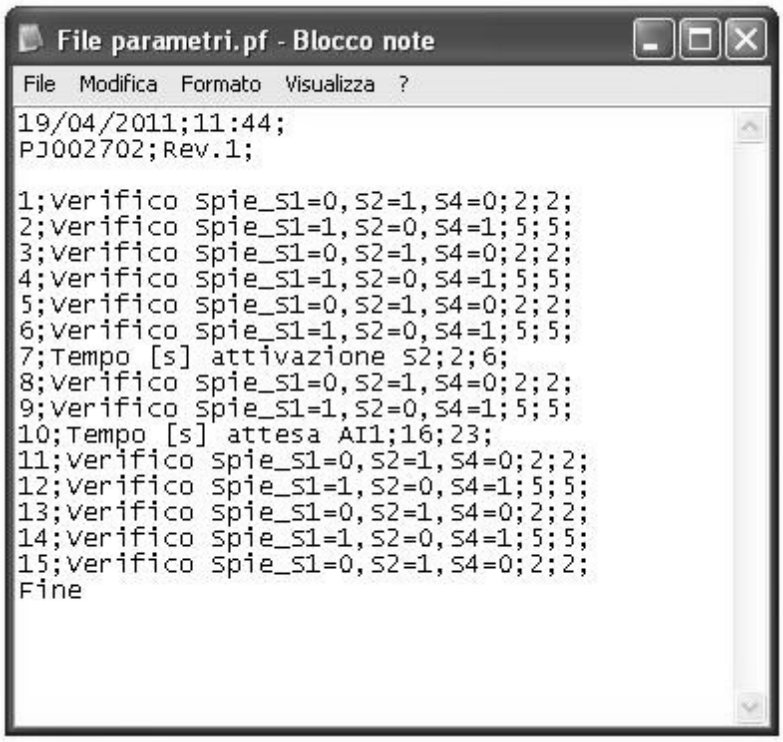


Fig. 3.1 Il file .txt dei parametri di collaudo.

Per leggere tale file occorre come prima cosa sapere la sua posizione all'interno del disco. Per far ciò si è creato un subVI (Fig. 3.2 e Fig. 3.3) contenente un particolare oggetto di LabVIEW che fornisce il Path, cioè il percorso, della cartella in cui si trova il VI generale in esecuzione. Per comodità si è scelto poi di posizionare tutti gli altri file e sottocartelle utilizzati nella stessa cartella così da permettere al programma l'identificazione automatica previa specificazione del nome file o cartella.

Fig. 3.2 Il subVI creato che restituisce il percorso del programma generale.

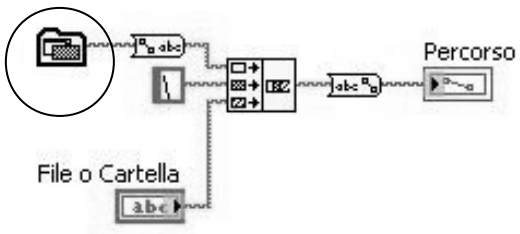


Fig. 3.3 Il Block Diagram del subVI appena citato.

In evidenza l'oggetto di LabVIEW che tiene traccia della posizione del VI generale nel disco. In uscita ho direttamente il percorso del file nel formato tipico C:\ (a seconda del disco logicamente).

La successiva creazione di un subVI (Fig. 3.4) atto all'estrazione delle informazioni è stata possibile sfruttando appositi blocchi di indirizzamento file, blocchi per riconoscimento, frazionamento e sostituzione di stringa e blocchi per l'inserimento in Array (contenitore) dei dati elaborati. Cercando quindi il carattere ";" nel file si compie un primo passo verso l'automatizzazione dell'acquisizione dati per il collaudo. Il file .txt inoltre è accessibile solo a livello di Block Panel tramite un apposito blocco (Fig. 3.4) e si avvia in maniera "silenziosa" all'avvio del test senza che l'operatore debba comandarne la lettura. Si noti da Fig. 3.5 la complessità logica e circuitale del subVI: questo ci permette di capire come un semplice oggetto per la lettura di un file possa risultare comunque molto elaborato.

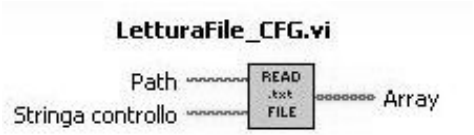


Fig. 3.4 L'icona del subVI ed i suoi collegamenti visibili nel Block Diagram di LabVIEW.

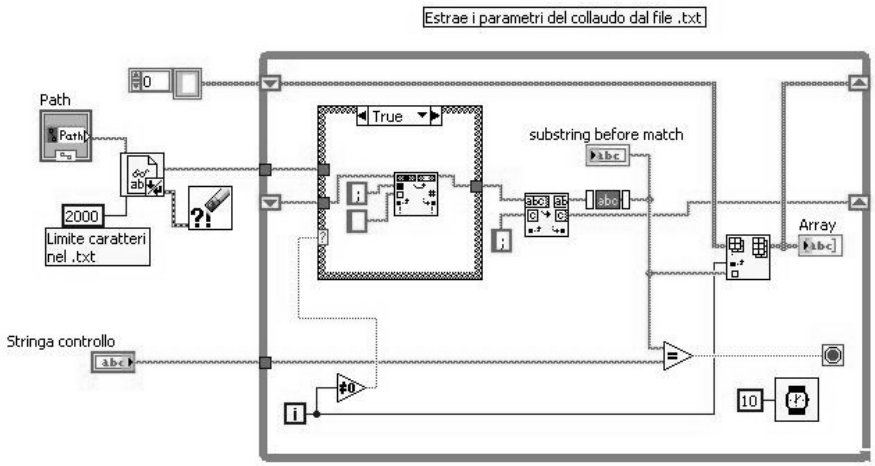


Fig. 3.5 Il subVI per la lettura del file .txt

3.3 Elaborazione dei dati dell'Array

Per estrarre i dati di interesse contenuti nell'Array di Fig. 3.5 si è creato un subVI apposito che va a scansionare per indice tutta l'informazione racchiusa nel contenitore e ne estrae, tramite "filtri numerici", le sole stringhe d'interesse. I filtri numerici altro non sono che degli accorgimenti logici e matematici, ben visibili in Fig. 3.7, per la lettura in blocco di più stringhe (quattro nel nostro caso dato il file .txt creato in precedenza) con i quali posso decidere quante e quali stringhe andare a leggere.

Fig. 3.6 Il subVI per l'elaborazione dei dati contenuti nell'Array.

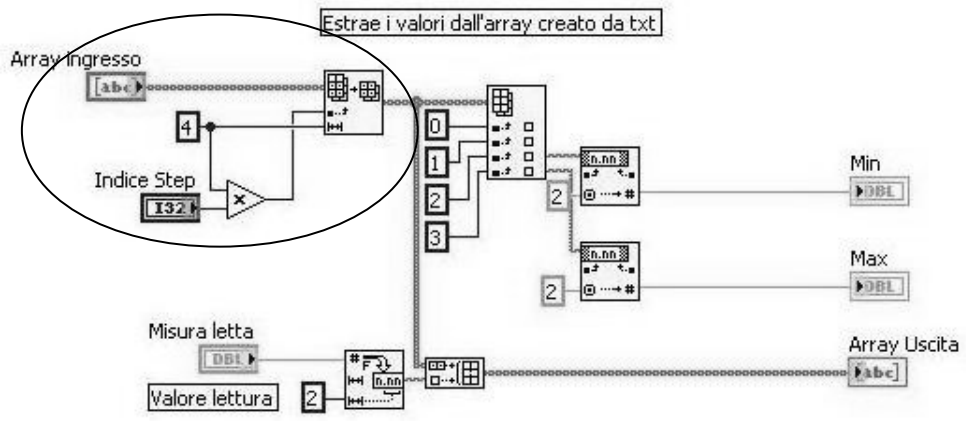


Fig. 3.7 La logica di controllo del subVI per estrazione dei dati da Array.

La Parte di codice evidenziata in Fig. 3.7 rappresenta proprio il filtro numerico utilizzato per la scansione dell'Array. I dati arrivano dal controllo d'ingresso e procedono in un blocco di elaborazione detto "Array Subset", il quale ritorna in uscita una porzione di Array a partire dall'indice specificato e di lunghezza desiderata (in questo caso quattro elementi). L'indice step è legato all'indice di ciclo d'esecuzione e col fattore moltiplicativo applicato va a puntare le stringhe dati a multipli di 4. Il resto del codice si divide in due parti: la prima estrae dalla porzione di Array i limiti min e max che verranno utilizzati nel blocco di controllo tolleranze, la seconda va ad aggiungere al sub-Array il valore numerico (di tipo Double) estratto dalla lettura. Tale sub-Array d'uscita verrà poi utilizzato per aggiornare la tabella informativa di Fig. 3.8, posizionata nel Front Panel e quindi visibile all'operatore, al solo scopo di visualizzare in modo chiaro e immediato in quale punto del test ci troviamo. Ciò che l'operatore nota guardando la tabella è che essa si aggiorna aggiungendo righe d'informazione man mano che si avanza con gli step di collaudo. Si è scelta tale configurazione per rendere user-friendly l'interfaccia grafica e per evitare all'operatore un continuo controllo visivo sulla scheda.

STEP	DESCRIZIONE	MIN	MAX	LETTURA	CRITERIO
1	Verifico Spie_S1=0,S2=1,S4=0	2	2	2.00	PASS
2	Verifico Spie_S1=1,S2=0,S4=1	5	5	5.00	PASS
3	Verifico Spie_S1=0,S2=1,S4=0	2	2	2.00	PASS
4	Verifico Spie_S1=1,S2=0,S4=1	5	5	5.00	PASS
5	Verifico Spie_S1=0,S2=1,S4=0	2	2	2.00	PASS
6	Verifico Spie_S1=1,S2=0,S4=1	5	5	5.00	PASS
7	Tempo [s] attivazione S2	2	6	2.68	PASS
8	Verifico Spie_S1=0,S2=1,S4=0	2	2	2.00	PASS

Fig. 3.8 La tabella informativa del Front Panel creata dal sub-Array.

3.4 Blocco di controllo lettura e tolleranza

Affinché vi sia un controllo preciso sulla lettura dei valori provenienti dal DAQ è stato creato un blocco (Fig. 3.9) di verifica tolleranza in cui si inseriscono i valori limite max e min estratti precedentemente dal blocco di elaborazione del sub-Array, il valore di lettura ottenuto da NI-USB6008 e una costante di tolleranza fissata al 5% modificabile esclusivamente in fase di debug dal programmatore.

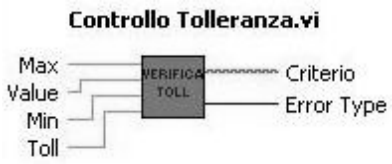


Fig. 3.9 La tabella informativa creata dal sub-Array.

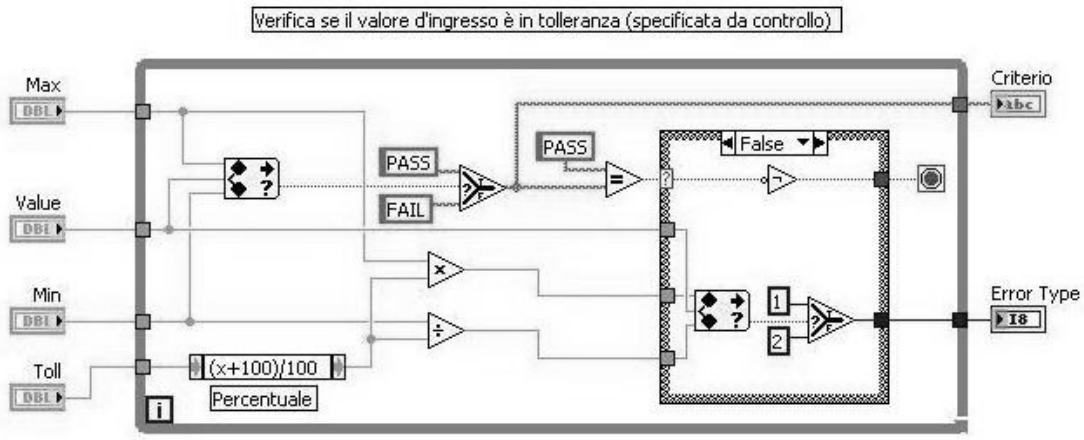


Fig. 3.10 Il codice per controllare la misura letta da DAQ.

Il blocco decide praticamente che tipo di errore dare in uscita (Fig. 3.10) a seconda della verifica sulla misura letta: se il valore letto è in range il criterio d'uscita è un PASS e l'Error Type fornito si aggiorna a 0 mentre se la misura è esterna al range scatta la verifica con una percentuale di tolleranza del 5%. Se la misura è esterna al range di tolleranza il criterio sarà un FAIL ovviamente e l'Error Type sarà aggiornato a 2. Se invece la misura è compresa in questo nuovo range di tolleranza il criterio sarà comunque un FAIL ma, come da Fig. 3.10 si andrà ad aggiornare l'Error Type a 1. Tali valori di Error Type vengono sfruttati per informare l'operatore, tramite messaggi a schermo, della condizione di FAIL in tolleranza su una misura, consentendogli di decidere se continuare o terminare il test (Fig. 3.12).

Fig. 3.11 Il subVI per il messaggio utente.

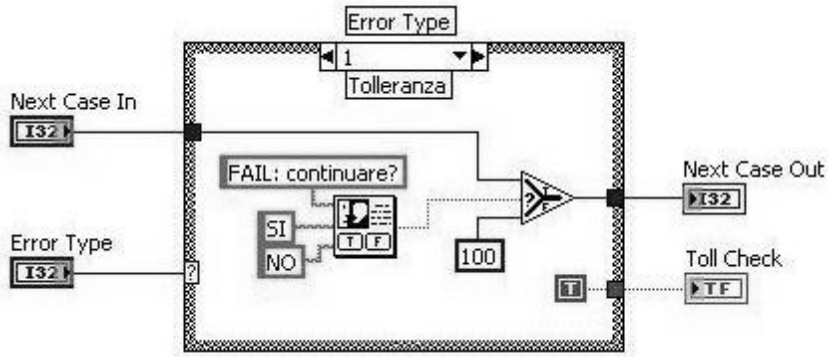


Fig. 3.12 Il controllo sull'Error Type permette la decisione dell'utente.

In uscita abbiamo l'indicatore preferenziale di salto step (Next Case Out) che è diverso a seconda dell'Error Type e un indicatore di tipo Boolean (Toll Check) utile nella fase in cui si riporta l'esito del test: secondo le linee guida dell'azienda si preferisce che la scheda che presenta una misura entro tolleranza permetta in fase di test di continuare o meno la verifica dei parametri dando comunque esito negativo. Si è deciso di applicare questo espediente affinché l'operatore abbia il controllo sull'esito del collaudo anche in fase intermedia di test: notando molteplici FAIL in tolleranza esso può decidere di terminare il collaudo ritenendo la scheda non idonea.

3.5 Comunicazione con NI USB6008

Per comunicare con il NI USB6008 si sfrutta la libreria inclusa nel CD driver in allegato col dispositivo. La libreria si chiama NI DAQmx e contiene molteplici oggetti e logiche per l'interfacciamento diretto tra LabVIEW e dispositivo di acquisizione. Ci sono oggetti per la lettura dei dati provenienti dal DAQ, oggetti per il set della porte e oggetti per il sincronismo, oltre a oggetti complessi per il controllo di routine della scheda, per la verifica del suo stato o per il suo reset. In Fig. 3.13 è riportata la libreria come visualizzata nel Block Panel di LabVIEW.

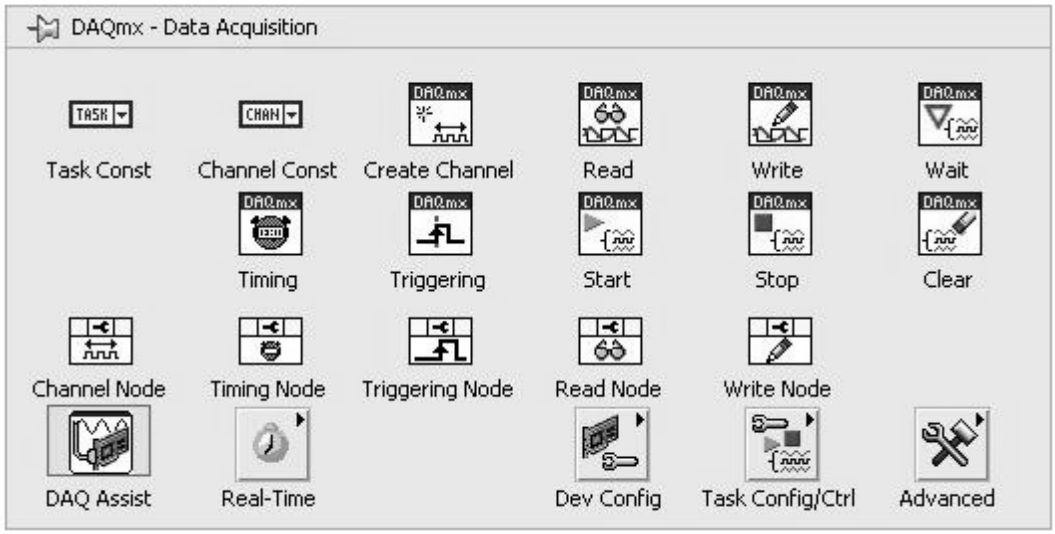


Fig. 3.13 La libreria per la comunicazione con NI USB6008.

3.6 Lettura e scrittura con NI USB6008

Ricordando che il DAQ per questo collaudo è impostato con otto uscite digitali,un'uscita analogica ed tre ingressi analogici occorre creare singolarmente degli oggetti preposti rispettivamente alla scrittura digitale, alla scrittura e alla lettura analogica. Combinando gli oggetti della libreria di Fig. 3.13 in determinate configurazioni si ottengono subVI che permettono il controllo preciso e sincronizzato del dispositivo. Nelle figure seguenti verranno riportati i blocchi creati a tale scopo e ne verranno descritti il funzionamento, già intuitivo dai nomi dei blocchi, e i comandi creati per l'interfaccia col codice del test.

3.6.1 Set delle linee digitali d'uscita del DAQ

In Fig. 3.15 è riportato il Block Diagram del subVI creato per la scrittura sulle porte digitali del dispositivo.

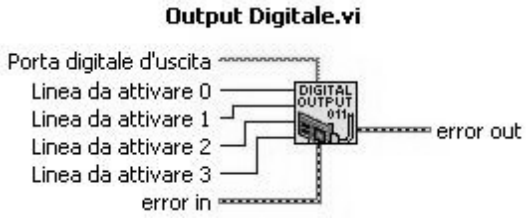


Fig. 3.14 L'oggetto creato per settare le uscite digitali a livello alto o basso.

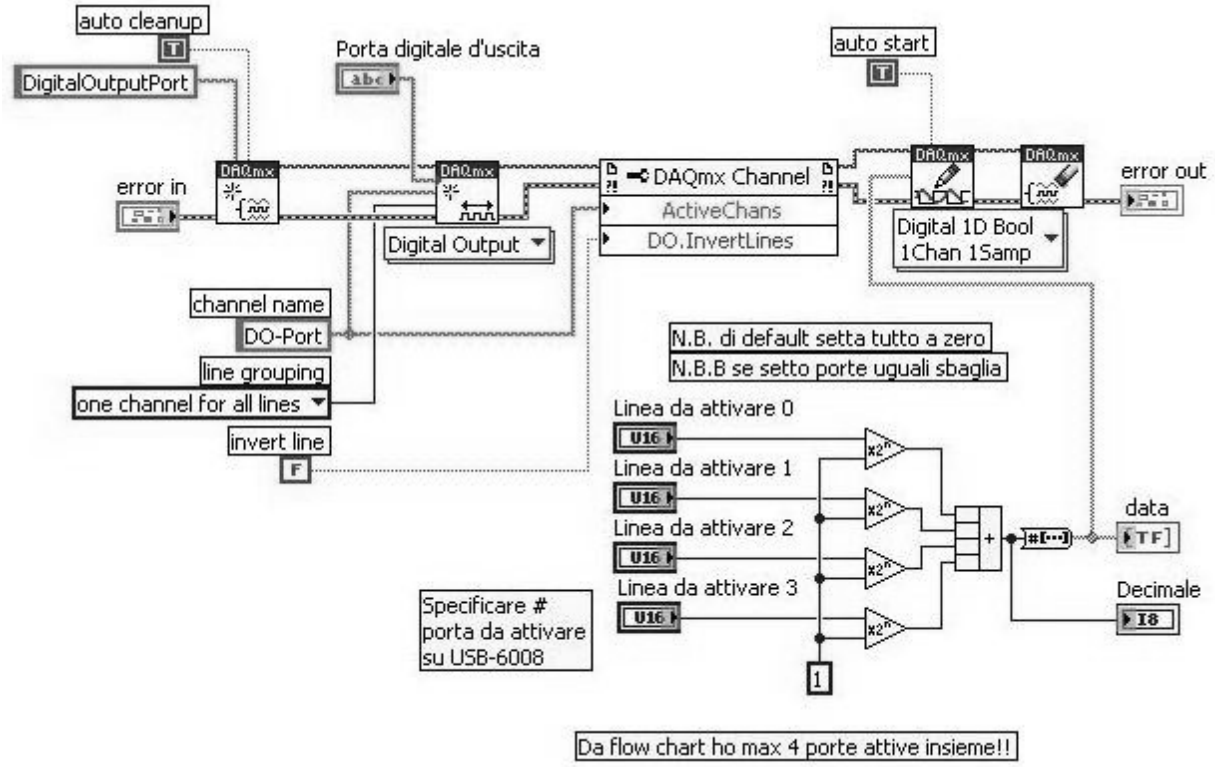


Fig. 3.15 La logica di controllo del subVI Output Digitale.

Si noti come solo per la lettura dei valori dal DAQ la complessità di programmazione aumenti in maniera esponenziale. I quattro blocchi della parte alta del subVI sono infatti a loro volta dei subVI (l'annidamento vale quindi anche tra VI e subVI) al cui interno si vanno a modificare direttamente i

registri di comunicazione tra programma e dispositivo. I primi due blocchi, posizionati in alto a sinistra nella Fig. 3.15, permettono la creazione di un Task, ovvero una raccolta di uno o più canali, temporizzazioni, attivazioni e altre proprietà che si applicano al NI USB6008. Concettualmente il Task rappresenta una misurazione o una generazione che si desidera eseguire. Tramite questi blocchi è quindi possibile specificare il nome del Task, il nome del canale (channel name) e la porta che intendiamo comandare. Nella parte alta di Fig. 3.15 troviamo invece un Property Node ovvero una particolare proprietà legata al canale (DAQmx Channel) che permette, in questo caso, l'attivazione della linea specificata e a richiesta l'inversione di tutti i suoi valori logici (al canale sono applicabili anche altre proprietà, da questo punto di vista LabVIEW è molto ricco di opzioni). In alto a destra troviamo, come intuibile dalle icone, gli oggetti per la scrittura a dispositivo e la cancellazione del Task appena creato. Si noti che nell'oggetto per la scrittura a dispositivo il software dà l'opportunità di specificare con comodi menù a tendina il tipo di dato da scrivere (Boolean), la sua quantità (1D Array, ovvero Array ad una dimensione) il suo indirizzamento (1 Channel) e il tipo di scrittura (1 Sample, ovvero un singolo campione). Nella parte bassa invece è presente la logica di controllo delle linee da attivare sulla porta specificata. La particolarità di tale configurazione è che si ha la possibilità di attivare fino ad un massimo di quattro linee in contemporanea come richiesto dal collaudo (il valore è comunque modificabile per adattarsi alle esigenze). Tale configurazione fornisce in uscita un numero decimale, ottenuto dalla conversione binaria delle linee attivate considerate come logica ON-OFF, ed un Array di dati Booleani, i quali hanno il compito di portare alte o basse le linee corrispondenti sul dispositivo. I terminali di Error In ed Error Out riportati in Fig. 3.14 sono necessari per il corretto funzionamento dei blocchi della libreria poiché la NI ha predisposto il dispositivo affinché fornisca sempre un report sugli eventuali errori che si verificano in fase di comunicazione.

3.6.2 Set dell'uscita analogica del DAQ

In Fig. 3.16 è riportata la logica utilizzata per il comando dell'uscita analogica del DAQ.

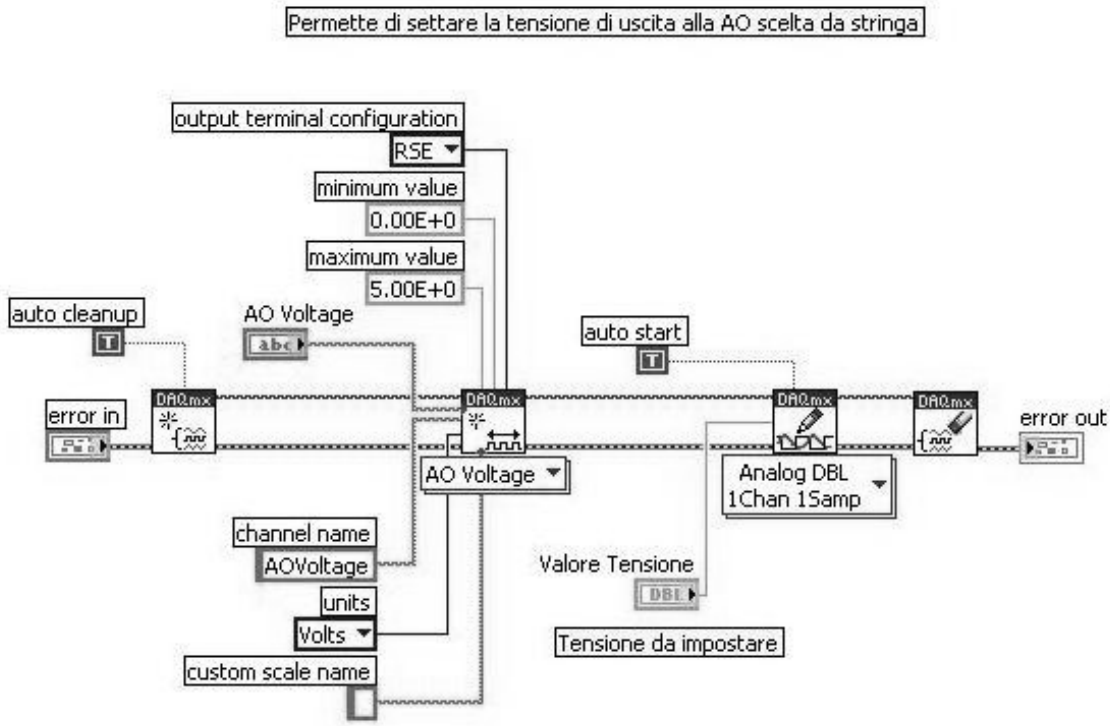


Fig. 3.16 Il suo Block Diagram di riferimento.

Fig. 3.17 Il subVI per l'impostazione della tensione d'uscita analogica.

Come si nota i blocchi utilizzati sono gli stessi di Fig. 3.15. Ciò che cambia è la scelta di quale Task attivare (AO Voltage in questo caso). Una volta attivata questa tipologia di Task l'oggetto modifica i suoi collegamenti d'ingresso e d'uscita permettendo la scelta del tipo di riferimento (RSE), il valore min e max oltre all'unità di misura preferita. Il principio di funzionamento di questo oggetto è molto simile a quello del subVI precedente con la differenza sostanziale che nel blocco di scrittura si specifica l'uscita analogica con un singolo sample di tensione, quest'ultimo fornito dal controllo d'ingresso double "Valore Tensione". Una volta deciso il valore di tensione a cui si vuole settare l'uscita il dispositivo riconosce i comandi e porta l'uscita analogica a tale livello con una precisione fino al decimo di Volt.

3.6.3 Lettura dagli ingressi analogici

In Fig. 3.19 è invece riportato il subVI creato per la lettura degli ingressi analogici provenienti dal banco di collaudo e collegati quindi alla logica di funzionamento della scheda.

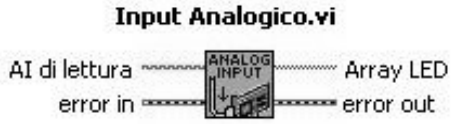


Fig. 3.18 L'oggetto creato per leggere la tensione dal banco di collaudo.

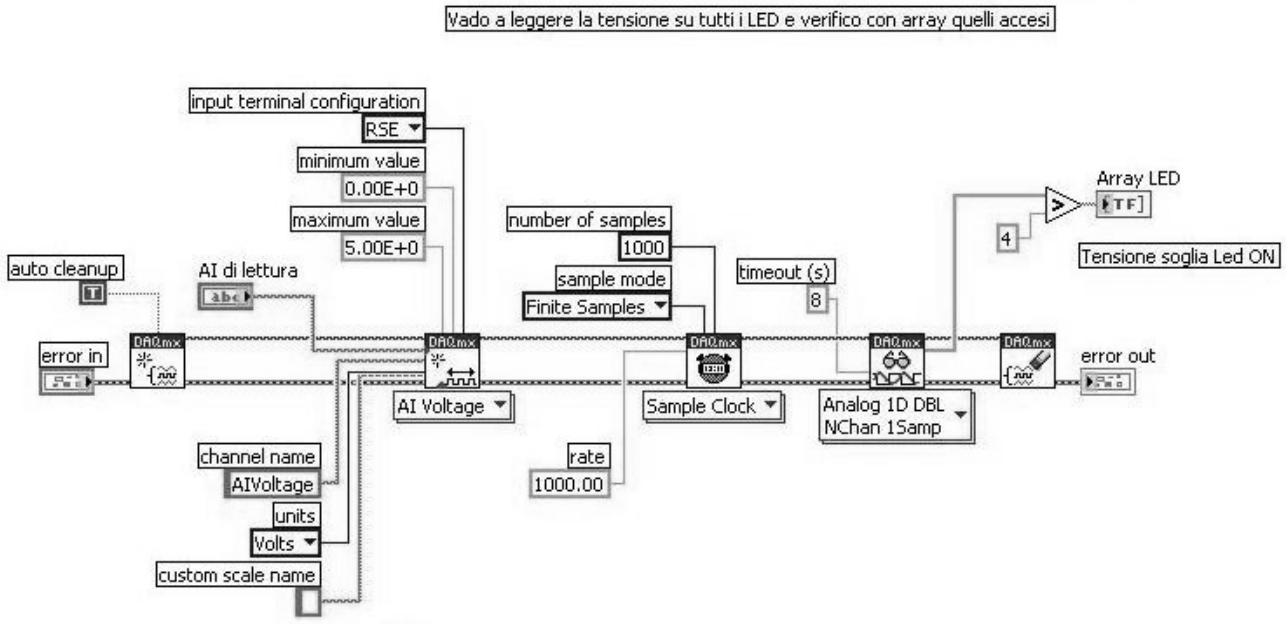


Fig. 3.19 La logica interna al subVI di lettura.

Tale subVI ha gli stessi principi di funzionamento degli altri: una volta impostati i parametri desiderati i blocchi cambiano la loro configurazione di collegamenti adattandosi di volta in volta. Notiamo due blocchi, situati nella parte di destra del block Diagram, che non appaiono nei precedenti subVI e che hanno rispettivamente il compito di impostare l'orologio di campionamento per il NI USB6008 e di leggere i valori presenti alle sue porte d'ingresso (in questo caso attendendo per 8[s] la ricezione di un dato valido). Occorre specificare che l'informazione d'interesse ottenuta dalla lettura delle tensioni è l'attivazione o meno dei LED collegati agli ingressi del DAQ. Tali LED si considerano accesi al superamento della soglia di 4[V] pertanto si è deciso di creare un Array di LED che fornisca come risultato grafico un gruppo di 3 LED ad attivazione indipendente comandati ovviamente dalla logica della scheda attraverso gli optoisolatori, ma considerati attivi al solo superamento della soglia limite. Due dei tre subVI appena trattati presentano uscite di tipo Array 1D (ad una dimensione). Per collegare tali uscite al resto del codice, affinché vi sia la verifica dei dati letti tramite il subVI di verifica tolleranza precedentemente visto, occorre estrarre dall'Array i singoli valori Booleani dei LED. LabVIEW offre degli oggetti preposti a tale compito di facile utilizzo che estraggono il dato specificato tramite indicizzazione numerica come indicato in Fig. 3.20. Si noti la porta NOT applicata ai LED S2 ed S4: come detto al paragrafo 1.2.2 dove si è parlato degli opto-isolatori la NOT ha il solo compito riportare il contenuto informativo al livello corretto dopo la negazione logica ottenuta dal passaggio dei dati negli isolatori galvanici.

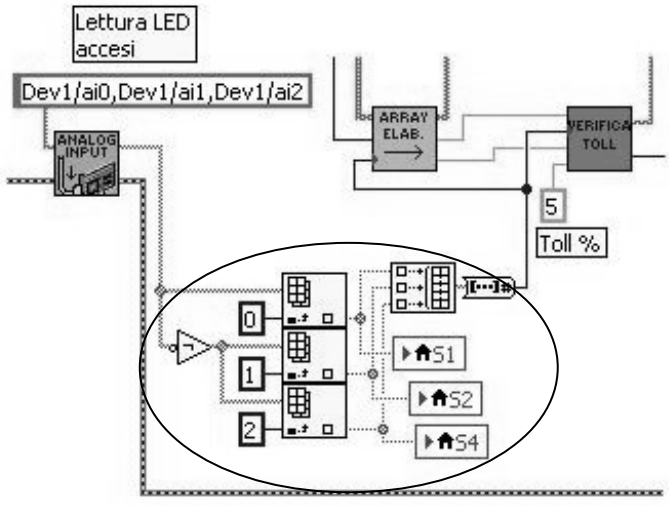


Fig. 3.20 Estrazione del contenuto informativo dall'Array d'uscita.

3.6.4 La finestra temporale di lettura

Il flow chart di Fig. 1.15 è costituito da alcuni step che hanno il compito di verificare la risposta dalla scheda in un tempo predefinito. Si rende quindi necessaria la creazione di un oggetto che legga ripetutamente una linea d'ingresso analogica della scheda e verifichi che la risposta da parte dell'hardware sia contenuta in un certo lasso di tempo, pena il FAIL del test. La finestra temporale di accettazione abilita quindi la lettura di un eventuale segnale d'ingresso mentre esternamente alla finestra qualsiasi segnale in arrivo provoca un esito negativo poiché la scheda ha risposto in anticipo o ritardo a seconda del caso. In Fig. 3.22 è riportata la logica di controllo del subVI creato appositamente per questa funzione. Esso è diverso dagli altri oggetti creati sebbene contenga al suo interno (annidamento) il subVI per la lettura di un ingresso analogico. Anche qui si nota come LabVIEW permetta di richiamare diversi subVI all'interno di altri senza alcun limite e riportando tutti i collegamenti di ingresso e uscita come se fossero oggetti qualsiasi (si ricorda che per creare i collegamenti personalizzati si agisce a livello di Front Panel nella sezione Show Connector).

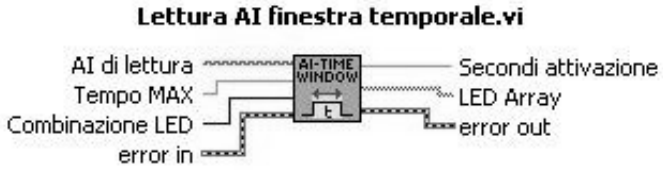


Fig. 3.21 L'oggetto creato per la lettura del tempo di risposta della scheda.

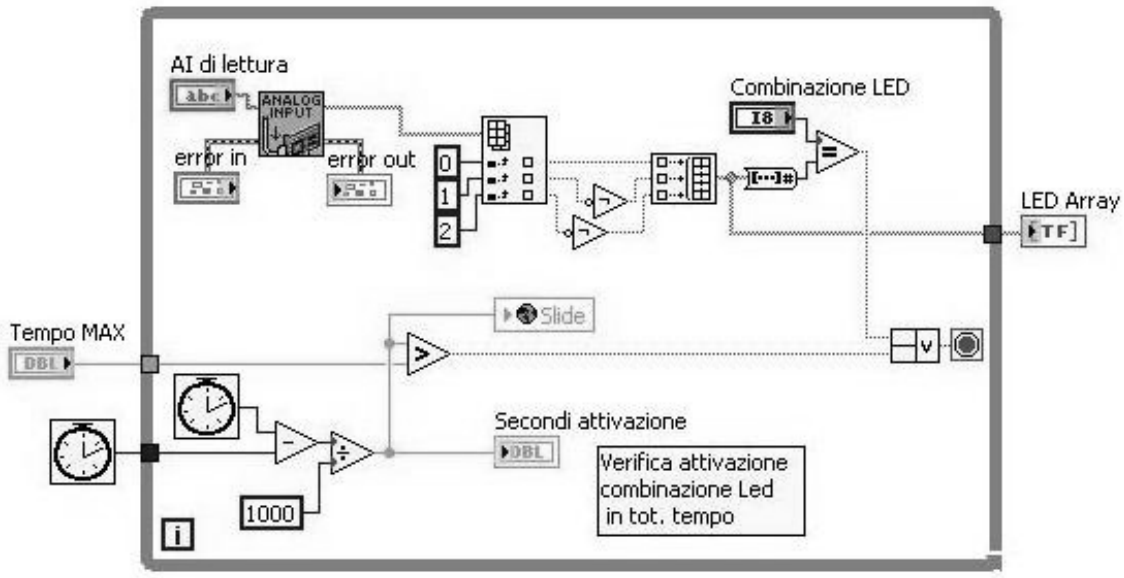


Fig. 3.22 Il codice che comanda il subVI della finestra temporale.

Come si può notare l'oggetto risulta essere piuttosto complesso anche solo per il fatto di contenere al suo interno il subVI del paragrafo precedente. Il cliente specifica che in questo step del collaudo la scheda risponde in un certo tempo con una determinata configurazione dei LED. Si è deciso quindi di sfruttare lo stato dei LED come controllo all'interno del subVI, con la possibilità di specificarlo esternamente, e facendo partire in contemporanea un conteggio dei secondi che trascorrono dal momento in cui si va a richiamare il subVI al momento in cui lo si ferma. Specificando un Tempo MAX utile solo nel caso in cui la UUT sia guasta e non fornisca alcuna risposta ed estraendo inoltre i secondi di attivazione si ha un controllo totale sul tempo di risposta della scheda (si noti che in questo subVI l'informazione d'interesse – Secondi attivazione – è fornita tramite logica di arresto, ovvero il conteggio continua ad aggiornarsi fino a quando si raggiunge una tra le due condizioni di arresto, cioè configurazione LED o superamento Tempo MAX).In Fig. 3.23 viene mostrata la porzione di codice in cui si trova l'oggetto appena descritto.

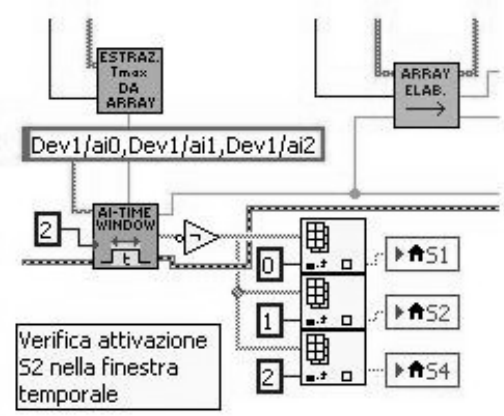


Fig. 3.23 L'applicazione del subVI nel codice.

Una volta specificata la configurazione dei LED (2 ovvero la combinazione binaria voluta con 3 LED), le linee d'ingresso dalle quali si aspetta una qualche tipo di segnale e il tempo massimo (estratto dall'Array creato dopo lettura dal .txt) si ottiene un oggetto che fornisce in uscita un valore Double contenente il tempo in cui la scheda risponde ed un Array di LED con la configurazione effettivamente letta dalla UUT tramite il DAQ.

3.7 Generazione del report .HTML a fine collaudo

La politica aziendale è quella di creare un resoconto a fine test che contempli tutti i passaggi effettuati, tutti i dati misurati, tutti le informazioni inserite dall'operatore e quelle deducibili dall'elaboratore o dall'esito del Test. Poiché LabVIEW ha una libreria specifica per la generazione di report (Fig. 3.24) si è deciso di utilizzare un tipo di file che possa essere letto a prescindere dalla

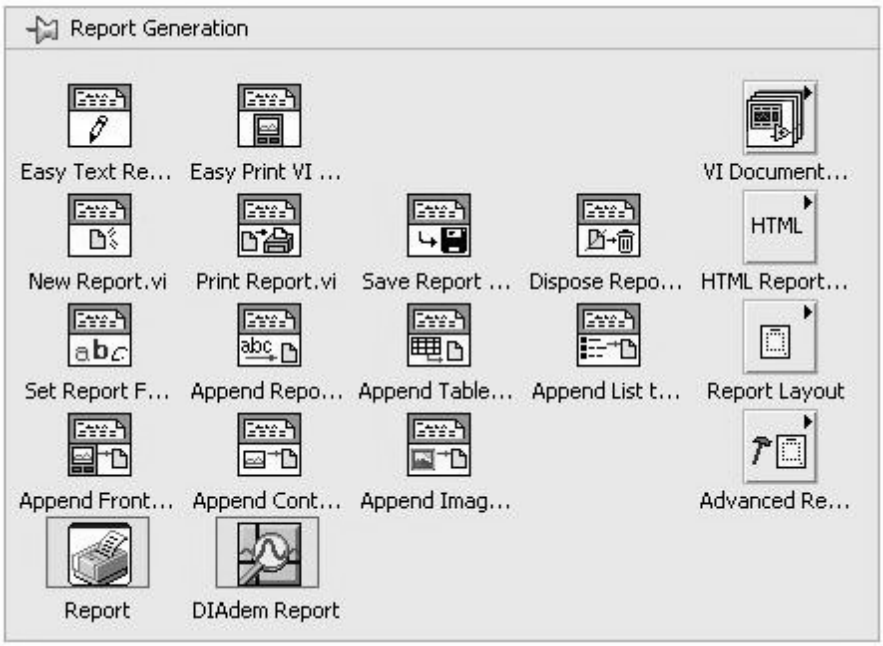


Fig. 3.24 La libreria per la generazione dei report contenuta in LabVIEW.

presenza o meno sul calcolatore di pacchetti software per ufficio. La scelta quindi è ricaduta sul report in formato HTML poiché qualsiasi computer al giorno d'oggi ha un Browser per l'accesso ad Internet integrato. Nel seguito verrà riportato il codice del subVI creato per la generazione del report e ne verrà descritta ampiamente la logica di controllo: questo oggetto è il più complesso tra quelli creati per il collaudo e la comprensione di tutti i suoi blocchi interni è stata possibile sfruttando il Context Help di LabVIEW, potente strumento di aiuto, informazione e descrizione integrato nel software.

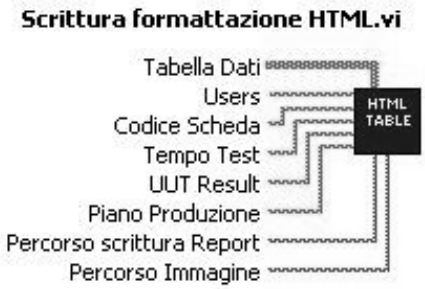


Fig. 3.25 Il subVI a livello di Block Panel per la formattazione HTML.

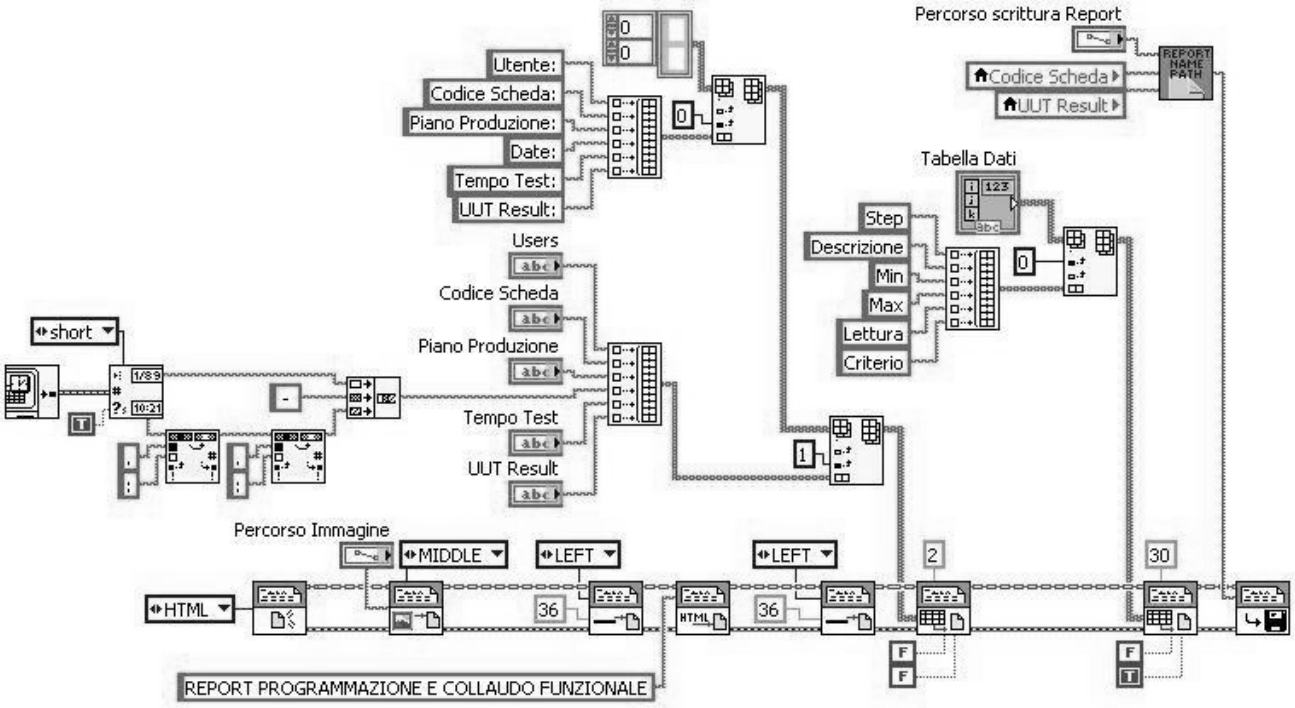


Fig. 3.26 La logica per la generazione del report.

In Fig. 3.25 sono riportati due collegamenti nella parte bassa del subVI generato. Essi rappresentano i percorsi da specificare al blocco (tramite il subVI visto al paragrafo 3.2) affinché esso crei il report e legga il logo dell'azienda nella posizione desiderata. Si distinguono subito due parti nel codice riportato in Fig. 3.26. Nella parte alta troviamo tutte le stringhe e le variabili che vanno inserite nel report e appariranno come voci d'interesse nel file finale mentre nella parte bassa troviamo tutti gli oggetti predefiniti di LabVIEW combinati opportunamente per ottenere la formattazione desiderata. Nel seguito verranno divise per chiarezza le trattazioni delle due parti.

3.7.1 Le stringhe, le variabili e le tabelle (parte alta)

Come prima cosa si intuisce che il report HTML è in ogni caso creato “per composizione” dal programmatore. Non esistono blocchi già predisposti per l'inserimento di voci in tabella: anche qui il programmatore ha piena libertà d'azione e può creare righe, colonne, spaziature e quant'altro come meglio crede rispettando, come unico vincolo, la dimensione dei dati e la loro tipologia. Secondo le specifiche dettate dall'azienda il report deve risultare chiaro, di immediata comprensione e ricco di informazioni relative al test effettuato. Informazioni quali la data e l'ora di test, l'esito, codici di autenticazione utente e della scheda e via dicendo sono quindi necessari per un report completo ed esauriente allo stesso tempo. Si è quindi deciso di dividere in due parti il report, una per i dati di lettura immediata (esito,data/ora,codici,ecc.) ed una contenente la tabella che appare all'utilizzatore nel Front Panel, completa di tutte le misure effettuate in tutti gli step. In Fig. 3.27 viene mostrato il report completo generato dal codice di Fig. 3.26. Per riportare le voci quali Utente, Codice Scheda, ecc. in colonna si è creato un Array di 6 elementi, analogamente a quanto fatto per le voci riportanti i valori delle variabili, la data e l'ora, l'esito e così via. Come si vede da Fig. 3.26 per accoppiare le due colonne si è scelto di creare un ulteriore Array posizionandole secondo gli indici: in questo modo è stato creato un Array di 6 righe e 2 colonne contenente la prima parte di dati del report. Ciò che interessa poi è l'immissione nel report della tabella del Front Panel: si va quindi a creare un Array che alla riga 0 comprende l'intestazione di tabella (composta da Step, Descrizione, Min, ecc.) e gli si va poi ad aggiungere la tabella del Front Panel. Anche qui notiamo come due tipi di dato di dimensione diversa (per lo spessore dei fili) vengano elaborati dal software senza alcuna difficoltà e anzi vengano amalgamati esattamente come il programmatore desidera.

EAS ITALY ELETTRONICA s.p.a.

REPORT PROGRAMMAZIONE E COLLAUDO FUNZIONALE

Utente:	485
Codice Scheda:	26521452
Piano Produzione:	65484568
Date:	17/05/2011 - 10:56:04
Tempo Test:	30 [s]
UUT Result:	PASS

Step	Descrizione	Min	Max	Lettura	Criterio
1	Verifico Spie_S1=0,S2=1,S4=0	2	2	2.00	PASS
2	Verifico Spie_S1=1,S2=0,S4=1	5	5	5.00	PASS
3	Verifico Spie_S1=0,S2=1,S4=0	2	2	2.00	PASS
4	Verifico Spie_S1=1,S2=0,S4=1	5	5	5.00	PASS
5	Verifico Spie_S1=0,S2=1,S4=0	2	2	2.00	PASS
6	Verifico Spie_S1=1,S2=0,S4=1	5	5	5.00	PASS
7	Tempo [s] attivazione S2	2	6	2.67	PASS
8	Verifico Spie_S1=0,S2=1,S4=0	2	2	2.00	PASS
9	Verifico Spie_S1=1,S2=0,S4=1	5	5	5.00	PASS
10	Tempo [s] attesa AI1	16	23	17.96	PASS

Fig. 3.27 Il report generato a fine Test.

3.7.2 I blocchi di formattazione HTML (parte bassa)

Nella parte bassa di Fig. 3.26 è presente tutta la formattazione per la creazione del report. Si parte creando un nuovo report specificando che tipo di file si vuole generare tramite un controllo a tendina (HTML in questo caso), si prosegue con l'inserimento nel report di un'immagine da un determinato percorso (in questo caso è stata scelto il logo dell'azienda) e l'inserimento poi di una linea divisoria di lunghezza specificata. Si continua poi con l'inserimento di un'intestazione per il report di collaudo (REPORT PROGRAMMAZIONE...) e a seguire una tabella specificando il numero di colonne che la costituiscono. Per riempirla si trasferisce la tabella contenente tutte le informazioni creata precedentemente nella parte alta di Fig. 3.26. Le due costanti di tipo Boolean presenti sotto il blocco di immissione tabella permettono di separare le pagine multiple, che si possono creare con l'immissione di molti dati, e di mostrare o meno nel report la griglia della tabella. Si nota poi un secondo blocco di immissione tabella identico al precedente nel quale si inserisce la tabella informativa completa d'intestazione creata precedentemente specificando la larghezza delle colonne (30 in questo caso) e ponendo a True la variabile Boolean che controlla la visualizzazione della griglia di tabella. Si ha poi un oggetto per il salvataggio del report che fornisce svariate possibilità di personalizzazione tra le quali l'immissione di una password per l'apertura del report, la sovrascrittura del file a scelta dell'utente e la specifica del percorso di destinazione (solo quest'ultima è d'interesse per il collaudo).

3.7.3 Specifica del nome file .HTML

L'azienda vuole che il nome del report contenga il codice scheda, la data, l'ora e l'esito del Test (tutto questo per una immediata identificazione dei PASS o FAIL in una cartella che può contenere anche centinaia di report di un lotto di produzione). Per fare ciò è stato creato un ulteriore subVI (Fig. 3.29) che crea un nome automatico per il report e sfrutta il percorso creato dal blocco di Fig. 3.2.

Fig. 3.28 Il subVI per il nome automatico del report.

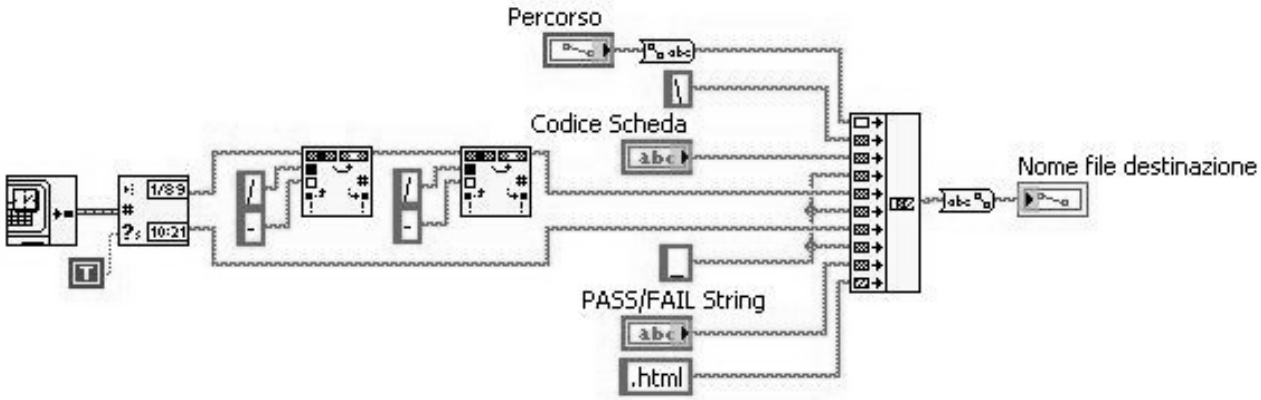


Fig. 3.29 Il codice per la generazione di un nome secondo le specifiche.

Si noti che tutti questi parametri sono nascosti all'operatore. Sarà il programmatore che in fase di debug andrà a specificare il Percorso mentre gli altri parametri, quali Codice Scheda e PASS/FAIL String verranno acquisiti automaticamente una volta avviato il subVI. In Fig. 3.30 si mostra l'utilità di tale oggetto: la verifica dei Test, siano essi PASS o FAIL, è effettivamente immediata.

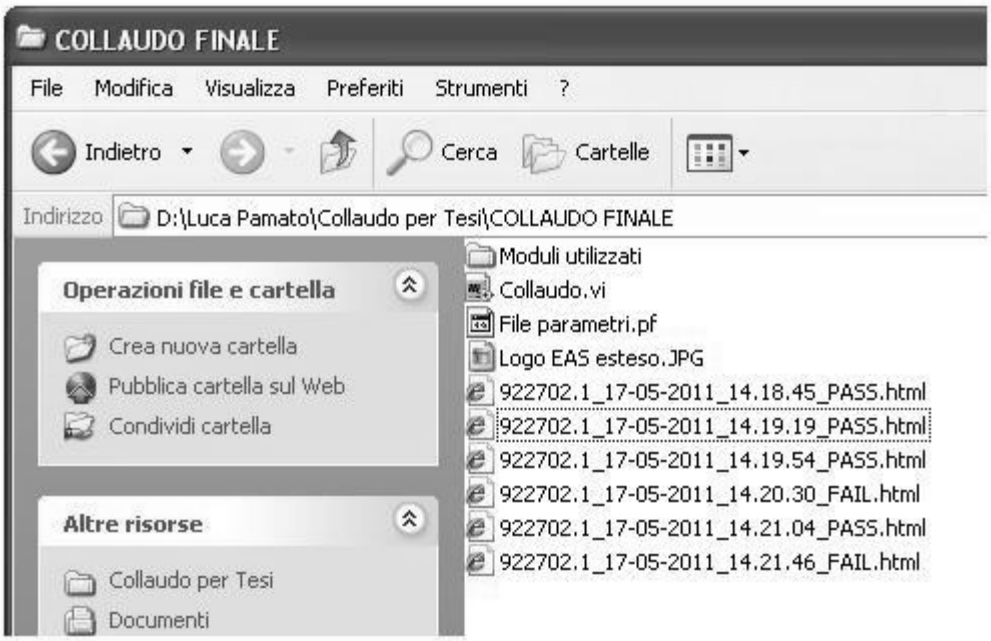


Fig. 3.30 La formattazione del nome permette una verifica immediata.

3.8 subVI aggiuntivi

Per finire con la descrizione del codice si trattano dei blocchi aggiuntivi che completano l'informazione trasmessa e danno la possibilità all'operatore di immettere dati successivamente elaborati nel codice. Si è deciso quindi di creare due subVI con i compiti, rispettivamente, di calcolo del tempo di test e di immissione codici utente, scheda e piano produzione.

3.8.1 Calcolo tempo di Test

Necessità espressa dall'azienda è che un collaudo automatico sia il più veloce possibile in modo da permettere all'operatore il controllo di un maggior numero di schede elettroniche nel minor tempo. Per tener traccia di ciò si è pensato di creare un subVI apposito (Fig. 3.31) che estrapoli dall'orologio di sistema l'ora e la visualizzi in secondi.

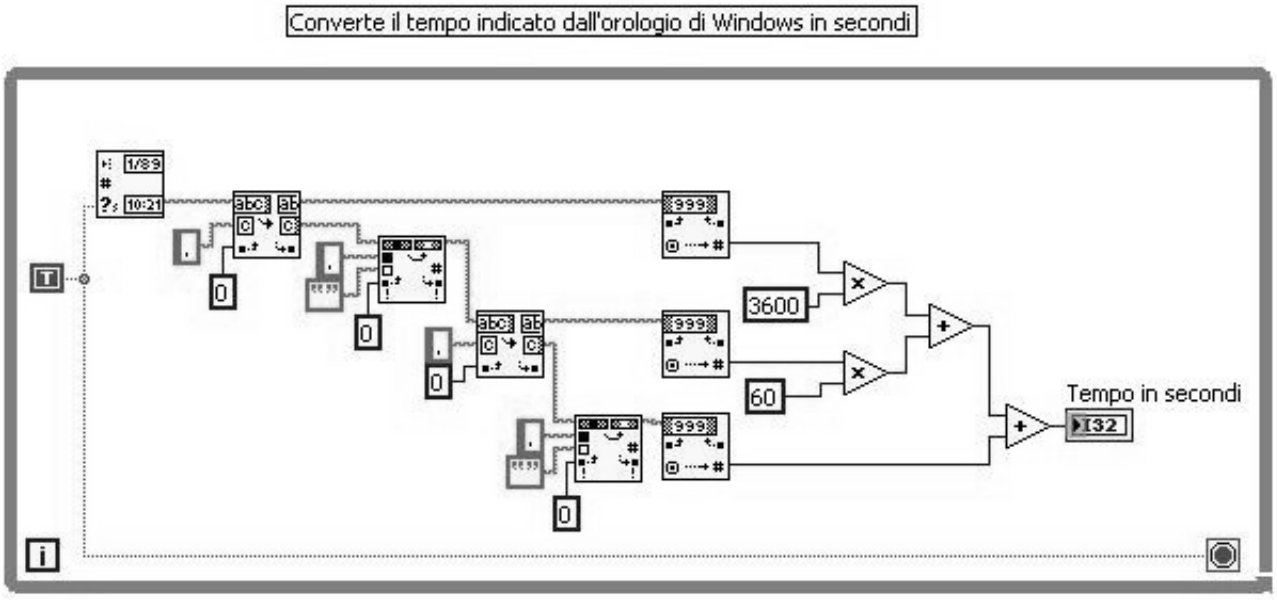


Fig. 3.31 Il codice che elabora il tempo in secondi.

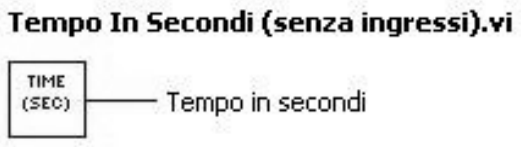


Fig. 3.32 L'oggetto per il calcolo del tempo.

Come si può vedere la prima elaborazione effettuata è l'estrazione del tempo dall'orologio di sistema. L'oggetto, già contenuto tra le librerie di LabVIEW, estrae l'ora in formato hh.mm.ss come dato unico e le successive elaborazioni hanno lo scopo di acquisire ore, minuti e secondi separatamente. Una semplice elaborazione numerica fornisce poi il tempo dell'orologio di sistema in secondi. Si tratta ora di stabilire come effettuare la misurazione del tempo di Test: si è deciso di settare un riferimento iniziale del tempo richiamando il subVI nello Step di inizializzazione del programma. Dopo tale Step infatti il Test inizia con il controllo e l'acquisizione dati della scheda. In Fig. 3.33 si riporta tale semplice collegamento.

Fig. 3.33 Il riferimento temporale iniziale.

Alla fine del test, sia l'esito positivo o negativo, si va a calcolare il tempo trascorso semplicemente come differenza tra tempo iniziale e tempo attuale di orologio di Windows. In Fig. 3.34 è mostrato il calcolo effettuato per la misura del tempo di Test.

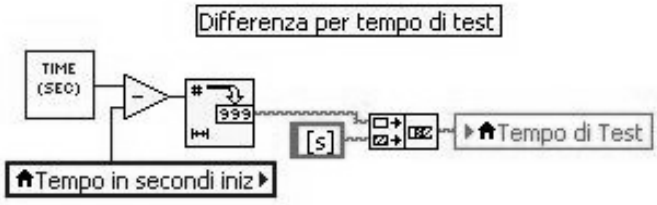


Fig. 3.34 Il calcolo del tempo trascorso.

Il blocco intermedio è sfruttato unicamente per convertire il numero in uscita dal blocco differenziale in una stringa, alla quale viene poi concatenata l'unità di misura.

3.8.2 Immissione dati e codici

L'azienda ha specificato che l'utente che compie il collaudo deve inserire il codice della scheda, del piano di produzione e la propria matricola una volta soltanto all'inizio della sessione di collaudo. Ciò significa che l'operatore che inizia il primo collaudo della sessione dev'essere interrotto da una schermata per l'immissione dei dati. In LabVIEW è presente un'apposita libreria per i messaggi a schermo (Fig. 3.35) sfruttabile proprio comunicare con l'utente.

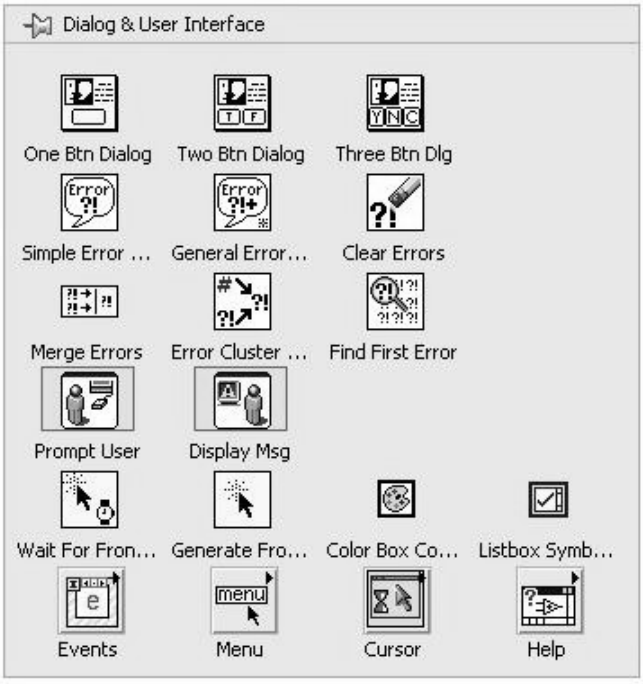


Fig. 3.35 La libreria per l'interfaccia con utente.

Tramite l'oggetto Prompt User riportato come icona in Fig. 3.35 e mostrato in Fig. 3.37 si ha la possibilità di creare una finestra di sistema come quella di Fig. 3.34 che resta in attesa dell'immissione dei dati finché l'utente preme il tasto OK.

Fig. 3.36 La finestra di comunicazione con l'utente.

La programmazione dell'oggetto Prompt User riportato in Fig. 3.37 permette la visualizzazione di massimo dieci voci e caselle per immissione, specificando per ognuna che tipo di dato si va a trattare, e consente la modifica del nome finestra oltre all'aggiunta di un'eventuale intestazione (per specificare l'operazione da compiere in questo caso).

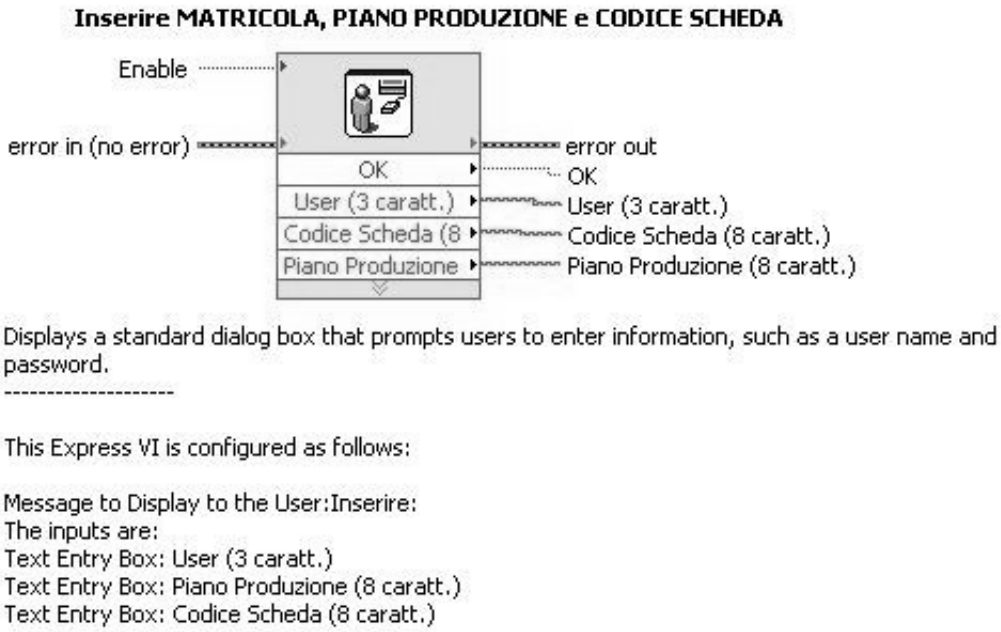


Fig. 3.37 L'oggetto per la comunicazione a schermo.

Si è poi creato un subVI anche per questa delicata funzione. Delicata poiché l'utente può sempre digitare caratteri errati e non conformi, o per la fretta lasciare i campi d'immissione vuoti. In Fig. 3.38 si riporta solo l'icona del subVI creato poiché il codice da riportare risulterebbe estremamente ingombrante.

Fig. 3.38 L'icona del subVI per inserimento codici.

Come si nota il subVI non presenta alcun ingresso poiché come già detto il messaggio utente è visualizzato direttamente a schermo. Le verifiche effettuate al suo interno sono su campi lasciati vuoti o sulla lunghezza dei dati inseriti (poiché è specificato dall'azienda che il codice scheda ha lunghezza fissa,così come il piano di produzione e la matricola utente). Non vengono eseguiti controlli su caratteri non alfanumerici poiché occorrerebbe creare un file di testo contenente tutti i caratteri ASCII non ritenuti necessari e fare un confronto per singolo carattere. L'azienda non ha ritenuto necessario tutto ciò, affidandosi al solo controllo “Empty & Length String”. Nel caso in cui l'utente inserisca tramite tastiera una stringa alfanumerica di lunghezza non idonea o vuota il programma reagisce mostrando a schermo, sempre grazie all'oggetto Prompt User, singole finestre per l'immissione dei dati risultati errati. Una volta che i dati sono stati inseriti correttamente si porta a True una variabile di tipo Boolean (DATI INSERITI). Nel programma principale si farà poi un controllo sul valore di tale variabile e se sarà True si salterà logicamente la fase di immissione dati.

3.9 La creazione dell'eseguibile del programma

A questo punto si vuole che il programma sia avviabile da qualsiasi calcolatore presente nel reparto. Per comodità di condivisione si potrebbe sfruttare la rete LAN con l'unico vincolo di installare su ogni macchina il software di casa NI. Per ovvie ragioni di costo licenza si intende evitare tale soluzione. La scelta è quindi ricaduta sulla possibilità offerta da LabVIEW di creare un file eseguibile funzionante da subito su una macchina che installi i pacchetti RunTime e NI-DAQmx. La procedura di creazione dell'eseguibile è accessibile dal menù Tools di LabVIEW (Fig. 3.39).

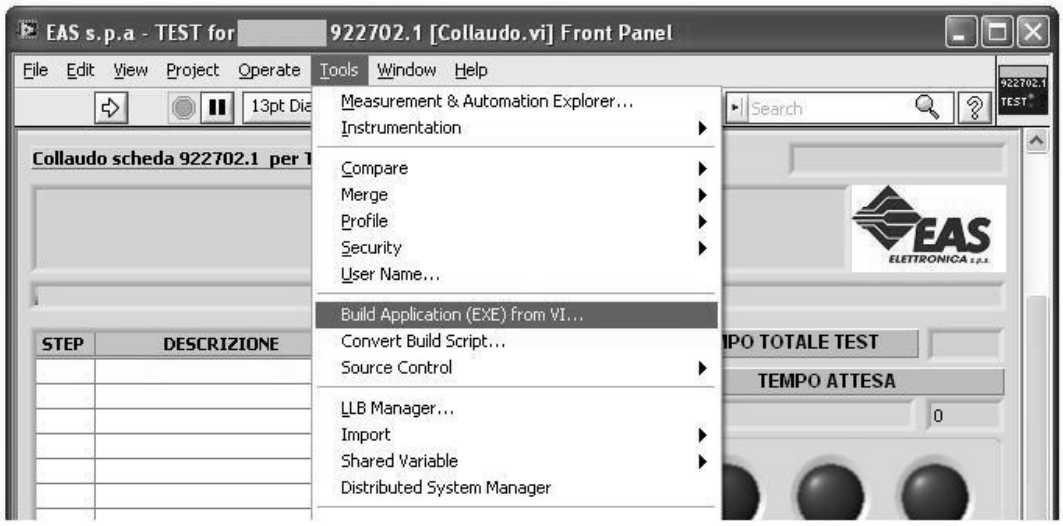


Fig. 3.39 La creazione del .EXE è avviabile dal menù Tools.

Una volta avviato il "Build Application (EXE) from VI" il software mostra una schermata nella quale è possibile scegliere le preferenze per il file .EXE tra le quali cartelle di destinazione, nomi e descrizioni, file aggiuntivi e simili. Alla fine del procedimento di creazione ciò che otteniamo è un insieme di file composto da un .EXE (il VI per il collaudo), un file .ini ed un file .aliases (Fig. 3.40) necessari per il funzionamento dell'eseguibile.

Fig. 3.40 I file creati col Build dell'eseguibile.

All'avvio dell'eseguibile si apre una finestra per l'utente preposto al collaudo che riproduce esattamente il Front Panel di LabVIEW (Fig. 3.41). Tale finestra non è modificabile dall'utente e funziona esclusivamente secondo la logica del programma creato quindi nessun comando esterno, eccetto quelli richiesti dal codice, può interferire con la sua esecuzione.

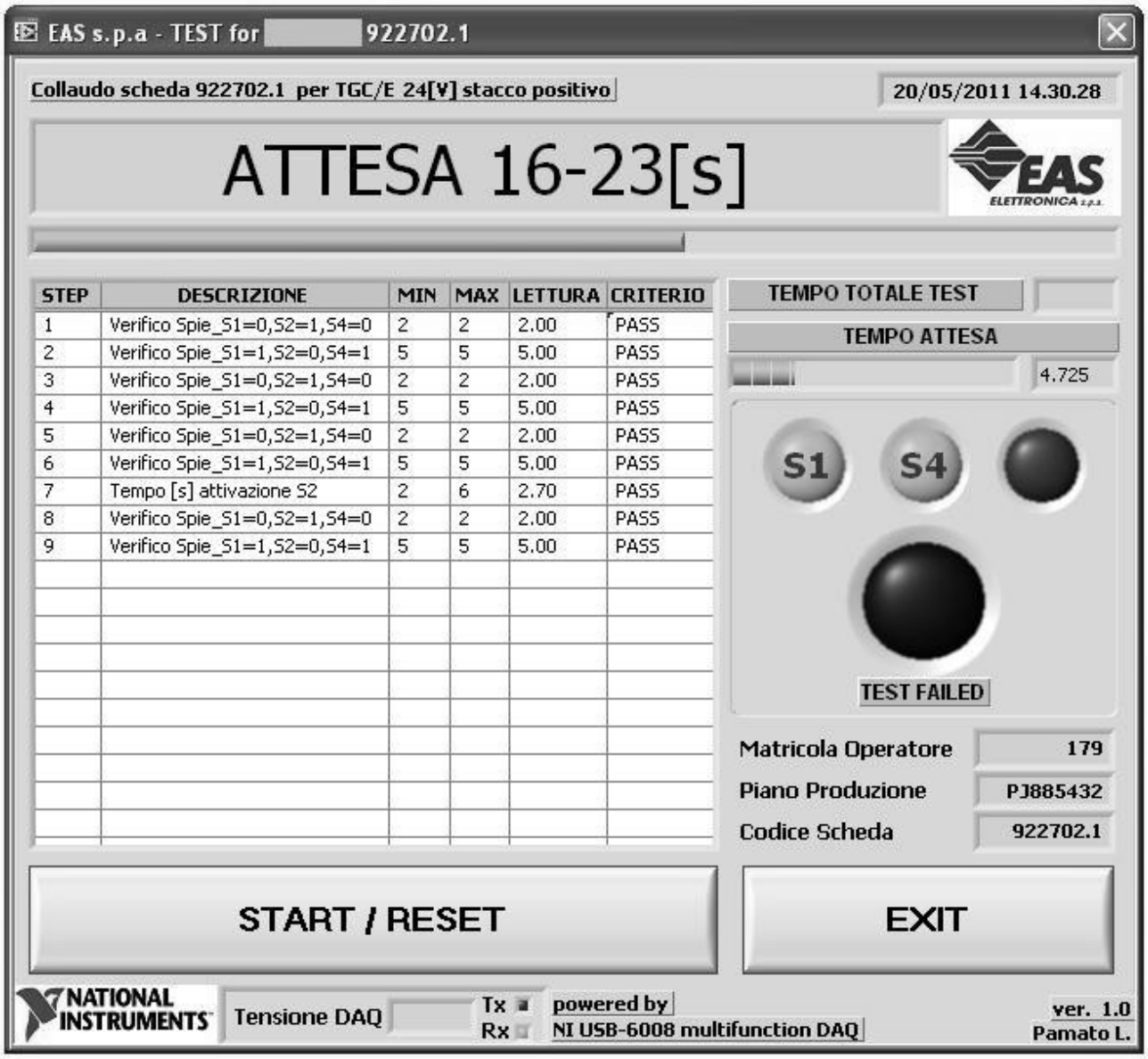


Fig. 3.41 La schermata del Test visualizzata dall'utente.

Capitolo 4

Definizione della procedura operativa

4.1 La procedura operativa

Come ultimo passo si è voluto redigere un documento tecnico, nel seguito riportato, per il collaudo appena creato. Il documento segue gli standard definiti internamente all'azienda e descrive le procedure da compiere per lo svolgimento del collaudo della scheda.

EAS	**ISTRUZIONE OPERATIVA COLLAUDO FUNZIONALE SCHEDA TGC/E 24[V] POS.**	**IO641_01**

CLIENTE	**CODICI PRODOTTI INTERESSATI**
Riservato	922702.1;

TABELLA DELLE DIFFERENZE TRA REVISIONI SUCCESSIVE		
REV.	**DESCRIZIONE**	**DATA**
00 01	Emissione Creazione nuova sequenza di collaudo	27/07/2010 25/05/2011

REV.	**EMISSIONE**	**APPROVAZIONE**	**DATA**
00 01	TEST – OSCAR ZANIN TEST – OSCAR ZANIN	Ufficio Tecnico Ufficio Tecnico	27/07/2010 25/05/2011

1 - SCOPO E APPLICABILITÀ

Scopo di questa istruzione è quello di formalizzare le fasi operative del collaudo funzionale. Questa istruzione deve essere applicata tutte le volte che, tra le operazioni di un Piano di produzione delle suddette schede è prevista la fase del collaudo funzionale. Se prevista, la fase deve essere svolta sul 100% delle schede del Piano.

2 - DOCUMENTI CORRELATI CON L'ISTRUZIONE

Sono correlati i seguenti documenti:

- Piano di montaggio schede PM92270201;
- Distinta base;
- Schema elettrico del cliente SE92270201_0 [78096512-A].pdf;
- Pagina "Etichette da applicare" in "Stato Rev. Documenti" sul sito Intranet.

3 - ATTREZZATURE E STRUMENTI NECESSARI

- **Banco Collaudo** M25500183 (Fig.1);
- **PC** con porta USB disponibile;
- Programma di collaudo in ambiente **LabVIEW 2010** denominato “922702.1 TEST.exe”.

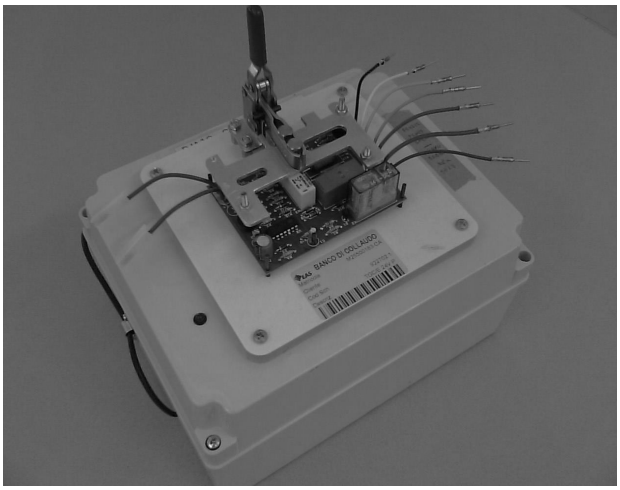


Fig. 1 - Banco M25500183

4 - ALLESTIMENTO DEL SISTEMA

- **Interruttore** POWER sul retro del banco posizionato verso il basso;
- Collegare la spina del banco alla tensione di rete 230Vac;
- Collegare la presa USB sul retro del banco al PC tramite apposito cavo USB;
- Lanciare sul PC il file eseguibile “922702.1 TEST.exe”;
  NOTA: l'eseguibile è rintracciabile al percorso: \\Eassrv20\Ate3500\Eseguibili.
  NOTA: I computer di test deve avere i pacchetti software LabVIEW Runtime Engine 2010 e NI-DAQmx installati. In caso contrario è necessario farne eseguire l'installazione.
- Inserire i dati richiesti: Matricola operatore, Piano Produzione e Codice Scheda (922702.1);
- **Interruttore** POWER sul retro del banco posizionato verso l'alto.

5 – COLLAUDO FUNZIONALE

5.1 Inserire la scheda nella Fixture ed abbassare il piano di contrasto tramite la leva rossa.
5.2 Avviare il Test automatico cliccando sul pulsante "**START / RESET**".

NOTA: Il Test è completamente automatico e quindi non serve visionare i passaggi. Alla fine del Test il programma e il banco avvisano l'utente tramite un segnale visivo ed acustico.

- In caso di PASS il programma avvisa l'utente con un messaggio ed il buzzer del banco si attiva per 1[s].

- In caso di FAIL il LED rosso del programma si accenderà ad intermittenza unitamente al buzzer intermittente del banco. In tal caso cliccare sul pulsante "**START / RESET**" per resettare.

5.3 Al termine del Test alzare il piano di contrasto e rimuovere la scheda dal banco.
5.4 Ripetere dal punto 5.1 per le schede successive.

6 - COMPORTAMENTO DA TENERE ALL'ESITO DEL COLLAUDO

Registrare l'esito del collaudo della scheda sul modulo M020 "Scheda registrazione collaudi".
Caso di esito positivo
Segnare un puntino con pennarello indelebile sulla casella "F" dell'etichetta di identificazione e rintracciabilità. Riporre l'assieme in un contenitore destinato a quelli collaudati con esito positivo.
Caso di esito negativo
Isolare la scheda in un contenitore destinato a quelle non conformi in attesa di un sollecito intervento di riparazione. Nel caso in cui la numerosità delle schede non conformi risulti elevata, allertare il Controllo Qualità. Il comportamento da tenere è quello descritto in PG12 "Non conformità".

7 - TERMINE DEL COLLAUDO

Spegnere il banco tramite l'interruttore POWER (posizione verso il basso) sul retro e scollegare il cavo USB. Spegnere il PC.

FINE ISTRUZIONE

Conclusioni

Il progetto sfrutta la velocità del calcolatore per il collaudo della scheda con risultati più che soddisfacenti in termini di tempo di Test. La complessa logica del programma permette un controllo totale della UUT con operazioni estremamente ridotte da parte dell'operatore quali innesto della scheda, accensione del banco di collaudo e avvio del programma di test. L'utilizzo del compilatore nel processo di verifica risulta utile nel momento in cui i compiti da svolgere per portare a termine un Test siano estremamente ripetitivi e possibilmente dannosi sia per l'operatore sia per la scheda. Le maggiori difficoltà riscontrate in fase di sviluppo software sono state l'interfacciamento tra LabVIEW e mondo esterno, la comprensione della logica d'esecuzione del programma non sempre immediata e la scelta di una struttura base con la quale comandare l'intero processo. Sono state poi affrontate particolari configurazioni software per comandare i pulsanti di start e stop Test in modi diversi da quelli proposti da LabVIEW facendo quindi leva su quella che è la completa personalizzazione dell'ambiente di sviluppo, oltre alla generazione di report HTML dettagliati i quali permettono un'immediata verifica delle condizioni della scheda. Ne deriva un ambiente assolutamente user-friendly ed estremamente adattabile alle esigenze del programmatore unitamente ad una potenza di calcolo ed elaborazione impensabile in altri linguaggi di programmazione. Nell'elaborato si è trattata marginalmente la parte hardware proprio per dare risalto al software di sviluppo non dimenticando che essa rappresenta il cuore del banco di collaudo ai fini di alimentazione del sistema e adattamento del segnale da e verso il NI-USB6008. Eventuali modifiche apportabili al codice potrebbero essere la generazione di report separati indicanti il numero progressivo delle schede controllate giornalmente, il numero di schede totale controllate ed il loro esito, la creazione di un database modificabile in cui si tenga traccia delle schede non idonee con possibilità che risultino idonee ad una successiva verifica e l'opportunità di scegliere la lingua per il Test (data la presenza di una filiale aziendale europea).

Ringraziamenti

Ringrazio Il sig. Giorgio Geronazzo e tutto lo staff di EAS s.p.a. per l'opportunità offertami di lavorare in un ambiente estremamente dinamico ed innovativo nel settore elettronico.

Ringrazio l'intero reparto Test Engineering, in particolar modo il sig. Oscar Zanin per la collaborazione e la disponibilità accordatami durante tutta la permanenza dello stage.

Ringrazio i sig. Roberto Oboe e Carlo Fongaro per l'intermediazione tra università e azienda.

Ringrazio in particolar modo la mia famiglia che mi ha sempre sostenuto moralmente ed economicamente in tutti i momenti cruciali della carriera universitaria.

Sitografia

NI Discussion Forums (http://forums.ni.com)

National Instruments :Test, Measurement and Embedded System (http://www.ni.com)

MENBER'S Connecting System (http://www.menbers.it)

Datasheet catalog for integrated circuits (http://www.datasheetcatalog.com)

CEI – Comitato Elettrotecnico Italiano (http://ceiweb.it)