POLITECNICO DI TORINO

Facoltà di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Elettronica

Progetto:

Tester per bus di potenza

MONOGRAFIA

Studenti:

Relatore: Prof. Claudio Sansoè

Puglia Simone *matricola*: 148360
Strumbo Domenico *matricola*: 149847
Zafaro Francesco *matricola*: 150218

Anno Accademico:
2009/10

INDICE

1. FUNZIONAMENTO

1.1 Introduzione

Lo scopo di questo progetto è realizzare uno strumento che faciliti le operazioni di test dei principali parametri elettrici (correnti e tensioni) in un bus di potenza. Il dispositivo dovrà interfacciarsi con un calcolatore elettronico, utilizzando la scheda di acquisizione e controllo della *National Instruments NI USB 6008*.

Il sistema è suddiviso in due blocchi principali: il primo consiste in un circuito atto a effettuare le misure, il secondo , noto come *carico attivo*, ha lo scopo di variare la resistenza di carico sul bus.

1.2 Descrizione scheda di acquisizione e controllo

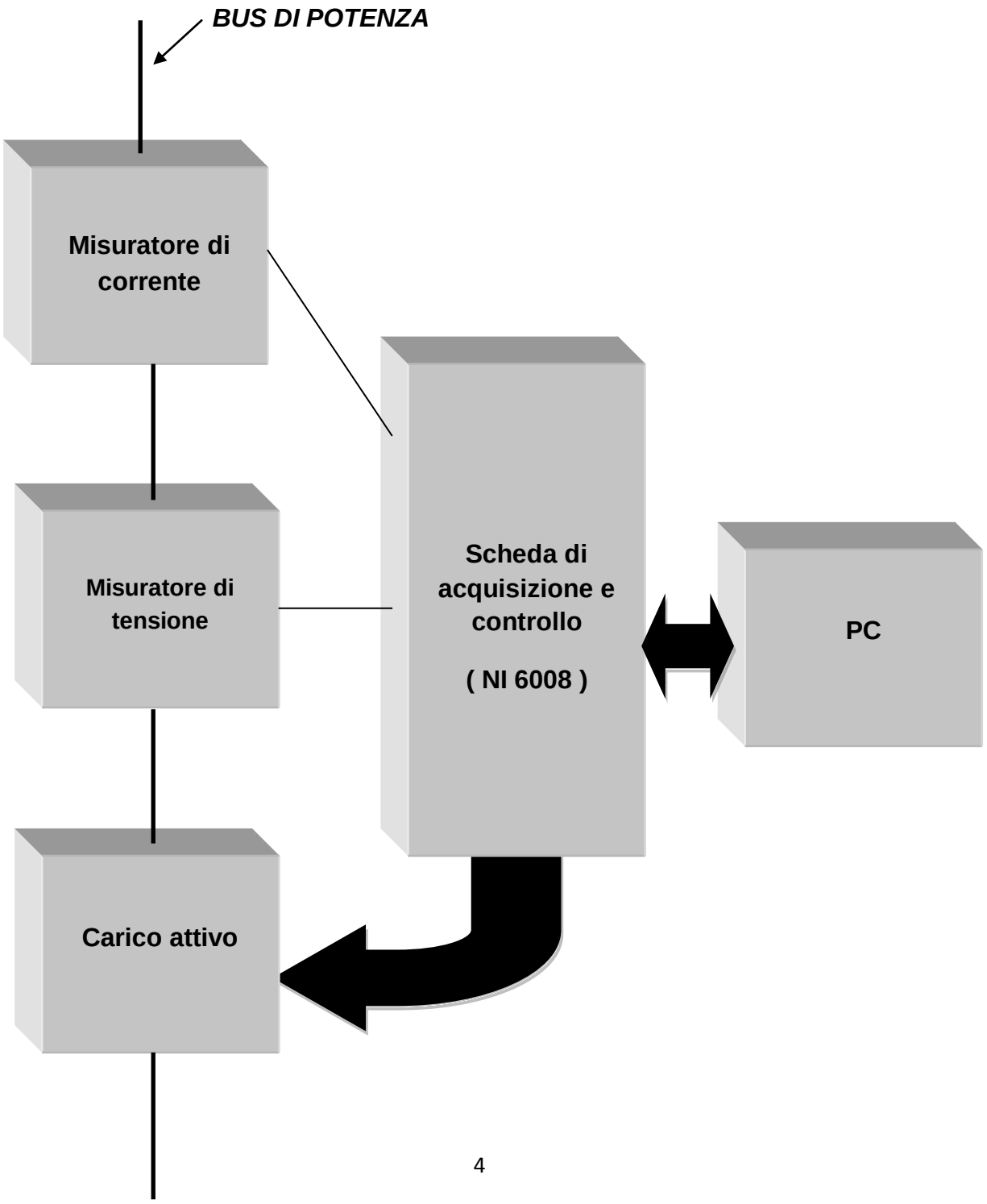
NI USB-6008

Per quanto riguarda l'interfacciamento tra il dispositivo ed il calcolatore, ci siamo serviti della scheda *NI USB 6008* della *National Instruments*. Questa ha le seguenti caratteristiche:

- 8 ingressi analogici (12-bit, 10 kS/s);
- 2 uscite analogiche (12-bit, 150 S/s);
- 12 I/O digitali;
- 32-bit counter;
- Bus-powered for high mobility;
- Built-in signal connectivity;
- OEM version available;

Le sue uscite digitali verranno utilizzate per comandare il carico attivo, mentre gli ingressi digitale/analogico verranno utilizzate per effettuare le misure di corrente e tensione.
La comunicazione con il PC verrà effettuata utilizzando la porta di comunicazione seriale presente nella board ed il tutto mediante l'ambiente di lavoro LabWindows/CVI 9.0.

1.3 SCHEMA A BLOCCHI



Descrizione schema a blocchi

1) Misuratore di corrente : il blocco per la misurazione di corrente deve essere in grado di rilevare correnti entranti e uscenti dal bus in analisi.

2) Misuratore di tensione: con questo blocco si deve effettuare una misura della tensione ai capi del carico attivo. Questo blocco dovrà essere costituito da un circuito di condizionamento per adattare la tensione del bus alla dinamica in ingresso della scheda di acquisizione.

3) Carico attivo: esso dovrà cambiare la resistenza ai suoi capi a seconda di quanto impostato tramite software. Per far ciò verrà utilizzato un circuito costituito un numero di resistenze collegabili in parallelo imposto dalle specifiche di progetto.

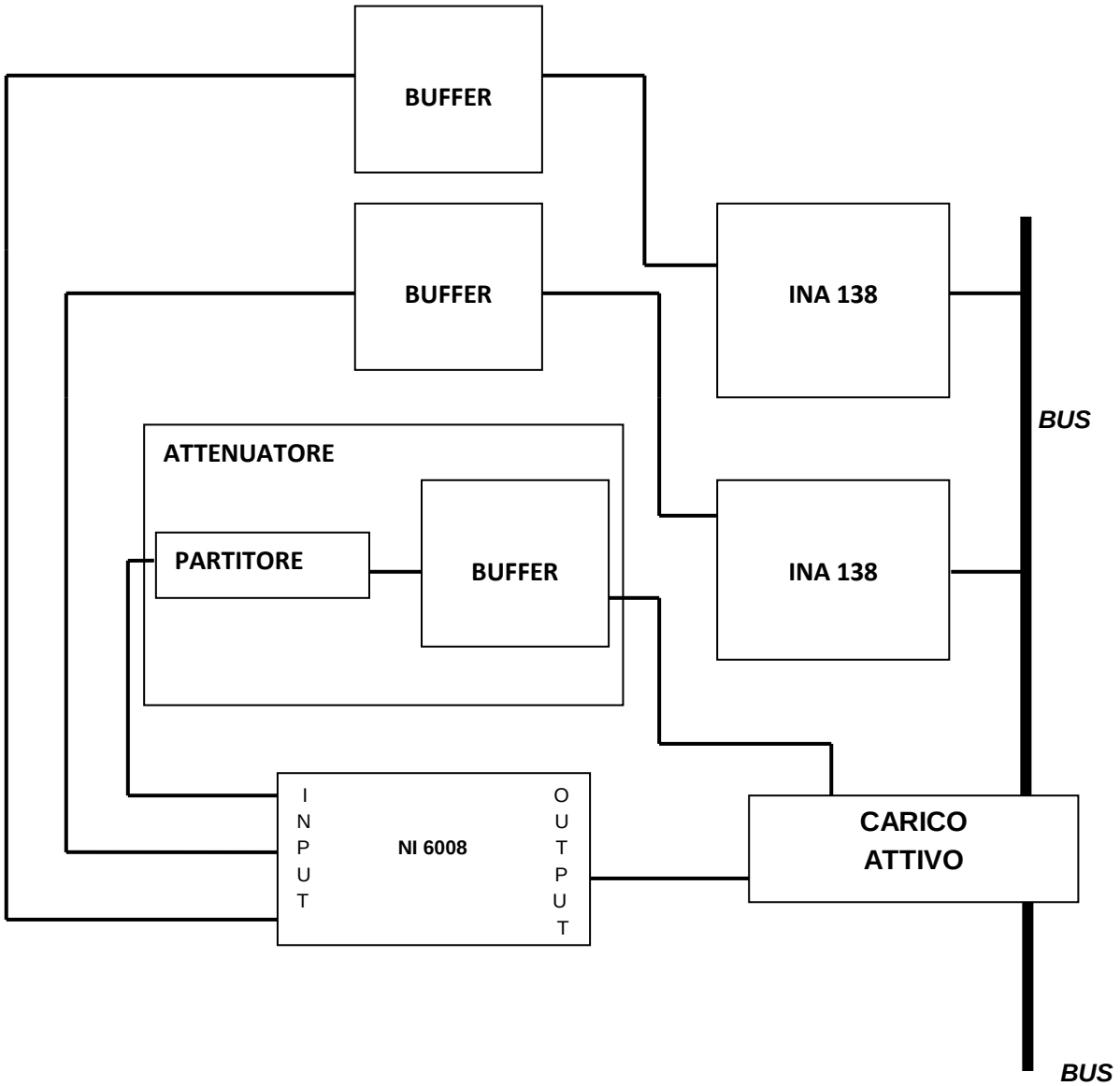
4) Scheda di acquisizione e controllo : scheda *National Instruments NI USB 6008*.

5) PC : i dati acquisiti e le impostazioni del dispositivo verranno ricevuti e/o trasmessi tramite l'interfaccia software.

2. ARCHITETTURA

2.1 SPECIFICHE DELL'ARCHITETTURA

MODULI ED INTERCONNESSIONI:



2.2 DESCRIZIONE BLOCCHI ARCHITETTURA

NI 6008 :

La scheda per l'interfacciamento con il calcolatore è la *NI USB 6008* della *National Instruments*. Nel progetto vengono collegate 8 uscite digitali e tre ingressi analogici. Le uscite digitali sono collegate ai MOS del carico attivo mentre i tre ingressi analogici, due dei quali in configurazione differenziale, effettuano le misure desiderate.

CARICO ATTIVO

Per comandare il carico attivo vengono utilizzati interruttori, di tipo low-side, costituiti da MOS di potenza interfacciati alla scheda di acquisizione e controllo. Lo stato degli interruttori verrà imposto tramite software, in modo da poter modificare la corrente che scorre nel BUS ed effettuare le misure in tali condizioni.

INA 138

L'integrato INA138 permette di ottenere una tensione proporzionale alla corrente che scorre in una resistenza di misura detta di "Shunt". Questa tensione viene quindi misurata dalla scheda di acquisizione per ottenere il valore di corrente nel carico attivo.

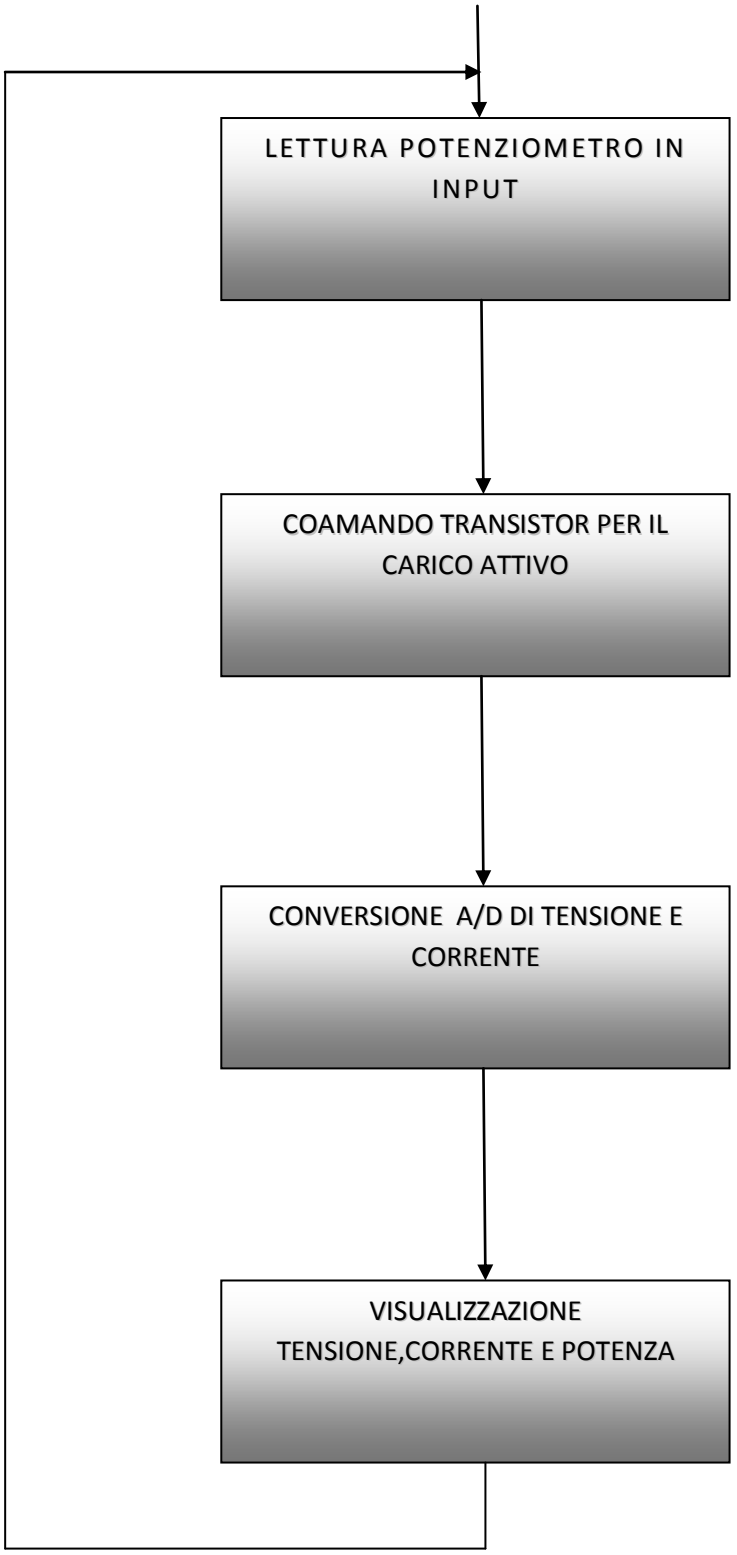
BUFFER

I buffer tra le uscite degli INA138 e la scheda di acquisizione, permettono la lettura corretta della tensione proporzionale alla corrente sul bus. I buffer risultano necessari in quanto gli ingressi della scheda NI6008 hanno una bassa impedenza di ingresso, provocando problemi di modo comune che portano ad acquisizioni prive di significato.

ATTENUATORE :

L'attenuatore è necessario al fine di riscalare la tensione ai capi del carico attivo con la dinamica in ingresso della scheda di acquisizione e controllo.
Il buffer, costituito da un amplificatore operazionale in configurazione voltage follower, ha lo scopo di separare l'impedenza tra il carico attivo e il partitore di tensione. In questa configurazione, la resistenza del carico non viene influenzata dalla resistenza equivalente del partitore di tensione.

2.3 DIAGRAMMA DI FLUSSO



3.PROGETTO

3.1 SPECIFICHE

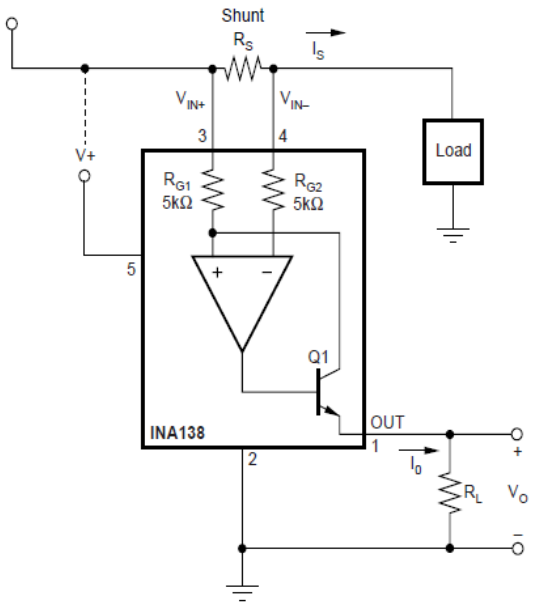
Le specifiche iniziali del sistema di controllo sono:

1. La tensione di alimentazione deve essere standard a *24V*.
2. La tensione minima sul carico è pari a V_{min} *=12,25V.*
3. La tensione massima sul carico è pari a V_{max} *=14V.*
4. Il dispositivo deve sopportare una corrente massima sul bus di $I_{bus,max}$ *= 20 A*.
5. Nel carico attivo deve scorrere una corrente $I_{bus,max}$ quando ai suoi capi si ha V_{max}.
6. Il carico attivo deve essere costituito da 8 resistori di potenza collegabili in parallelo.

3.2 DIMENSIONAMENTO COMPONENTI

3.2.1 MISURATORE DI CORRENTE

L'integrato INA 138 è un misuratore di corrente unipolare high-side. Tale dispositivo è stato scelto in quanto l'ampio range di tensione in ingresso di modo comune e l'alimentazione, tra loro indipendenti, sopportano un range da 2.7V a 36V.
La misura della corrente viene effettuata dal dispositivo, convertendo una tensione d'ingresso differenziale in una corrente in uscita. Questa viene, inseguito, riconvertita in una tensione utilizzando una resistenza di carico esterna dimensionata secondo le

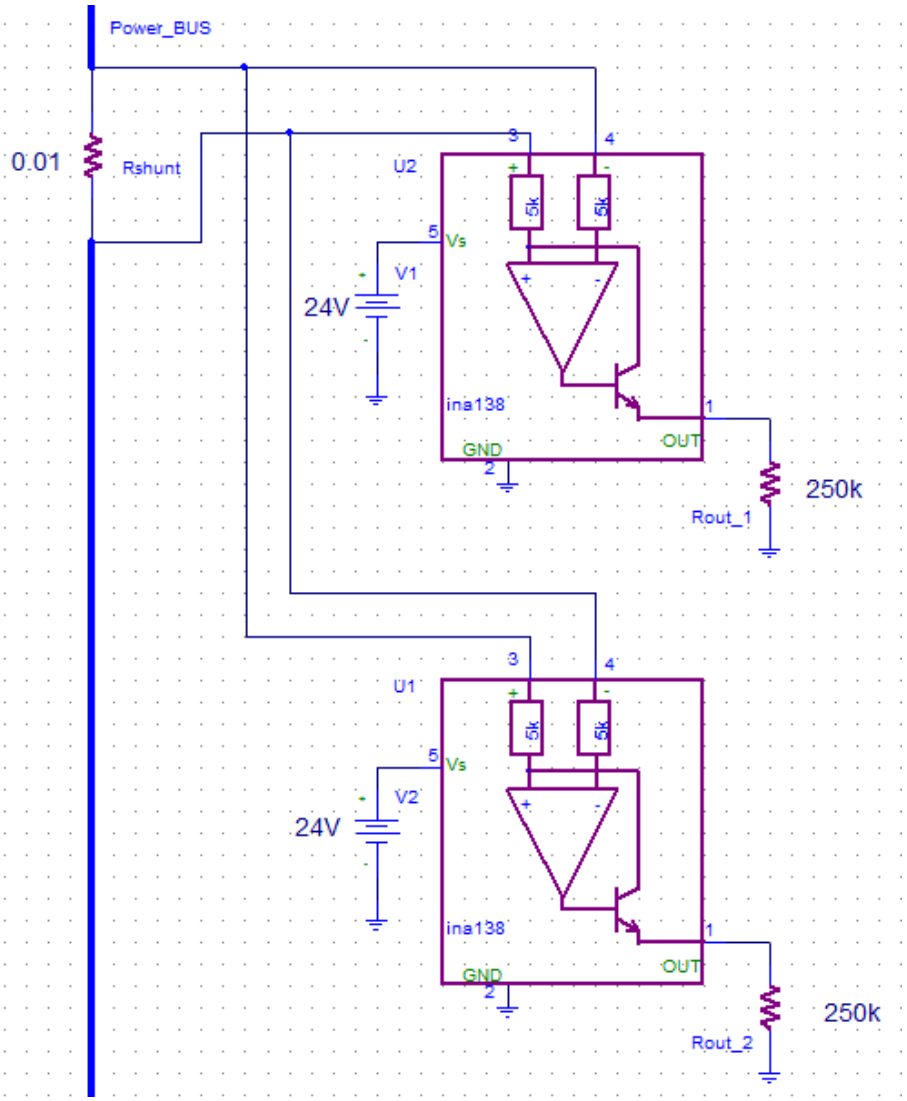


specifiche i cui calcoli sono riportati di seguito:

$$V_L = \frac{I_s * R_s * R_L}{5\ K\Omega}$$

Avendo scelto $R_{shunt} = 10\ m\Omega$ si ricava che $R_L = \frac{V_L * 5}{I_S * R_S} = 250\ K\Omega$.

Data l' unipolarità della misura, per effettuare misurazioni bipolari, sono state utilizzati due INA138 collegando la resistenza di shunt, all'ingresso differenziale, in maniera simmetrica.
Il dimensionamento in entrambi i casi rimane comunque invariato.



3.2.2 MISURATORE DI TENSIONE

Mediante il progetto di questo blocco, condizioniamo la tensione ai capi del carico attivo al fine di adattarla alla dinamica d'ingresso della scheda d'acquisizione, che vede come range analogico in ingresso da 0V a 10V, mentre la tensione ai capi del bus supera tale dinamica arrivando fino a 14V.
Per realizzare questa attenuazione è stato utilizzato un partitore di tensione:

$$V_{out} = \frac{R_{p2}}{R_{p1} + R_{p2}} V_{in}$$

Sostituendo con i valori di V_{in} e V_{out}:

$$10\ V = \frac{R_{p2}}{R_{p1}+R_{p2}} 14\ V; \qquad R_{p2} = \frac{10V}{14V-10V} R_{p1}$$

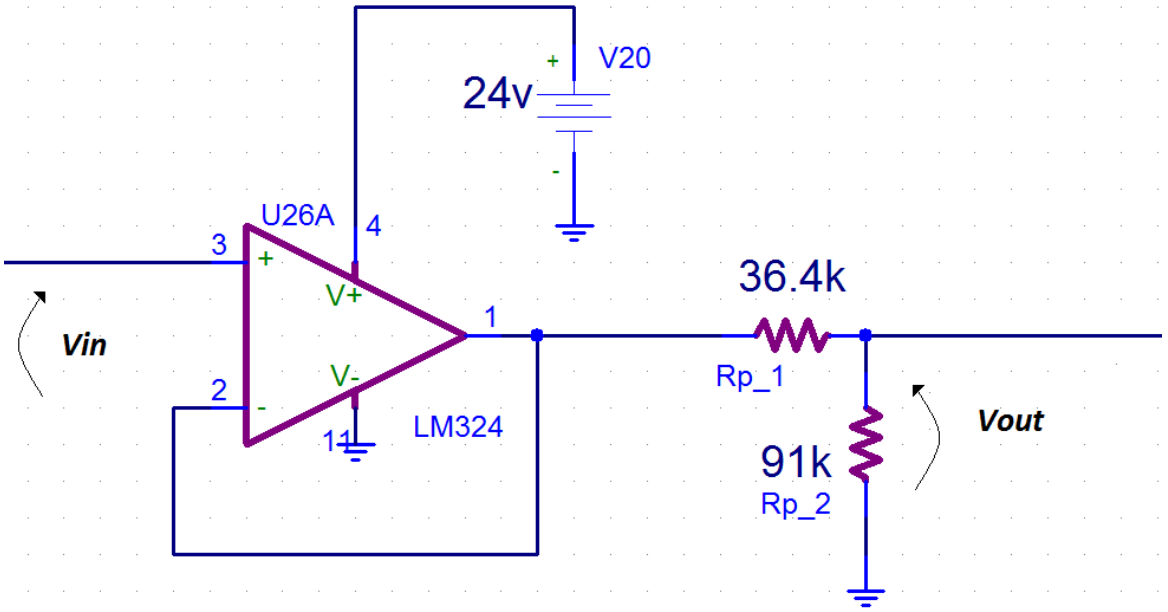
Dalla quale si ricava:

$$R_{p2} = 2{,}5 R_{p1}$$

Scegliendo dalla serie E24, le resistenze utilizzate sono:

$$R_{p1} = 36{,}4\ k\Omega; \qquad R_{p2} = 91\ k\Omega$$

Per evitare che l'impedenza del carico attivo venga influenzata dal partitore resistivo, è stato utilizzato un circuito voltage follower, che funge da separatore di impedenza. L'amplificatore operazionale scelto è l' LM324.
Tale scelta è dovuta al fatto che quest'amplificatore operazionale è un dispositivo rail-to-rail che può essere alimentato a singola alimentazione.



3.2.3 SCELTA DEL MOS

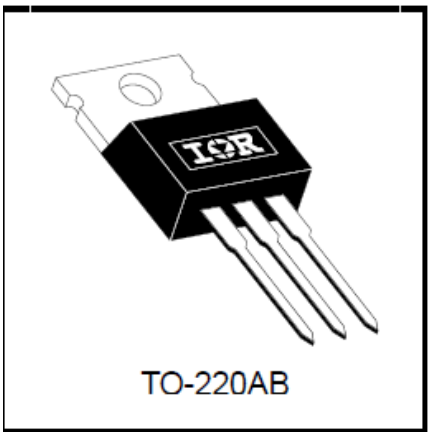
E' stato scelto di utilizzare interruttori in configurazione low-side in quanto, è possibile controllare lo dello stato dei MOS utilizzando direttamente la tensione alle uscite digitali della scheda *NI USB 6008*. Poiché questo sia possibile, è però necessario che sia soddisfatta la condizione $V_{GS} > V_{TH}$ quando viene applicata l'uscita digitale a 5V riferendo alla massa del BUS la scheda.

La scelta di tale MOS è stata effettuata, quindi, semplicemente trovando un transistor che rispetta le specifiche di progetto:

- Vth inferiore a 5 V.
- V_{DSS} maggiore di 14V
- Idrain max maggiore di 20A.
- R_{ON} con VGS a 5V molto bassa.

Un ***IRL2703*** è assolutamente in grado di soddisfare questo genere di richieste:

- Vth circa pari a 1 V.
- VDSS circa pari a 30V
- Idrain circa pari a *24A*.
- R_{ON} con VGS a 4.5V circa pari a *0.06Ω*.



3.2.4 CARICO ATTIVO

Come già in precedenza accennato, il carico attivo progettato varia la resistenza ai suoi capi in modo discreto. Esso è costituito da una rete a scala composta da 8 resistori, secondo le specifiche richieste, ed è interfacciata con la scheda di acquisizione e controllo utilizzando altrettanti interruttori low-side costituiti da transistor N-MOS di potenza.

1. DIMENSIONEMANTO VALORE RESISTORI
   Per quanto riguarda la scelta dei resistori di potenza, abbiamo tenuto conti delle seguenti considerazioni:

   sapendo che V_{MAX} = 14 V e I_{MAX} = 20 A , otteniamo

$$R = \frac{V}{I} = \frac{14\,V}{20\,A} = 0{,}7\ \Omega\,.$$

   Essendo le otto resistenze tutte in parallelo tra loro, quando si ha il worst case ricaviamo:

$$R = \left(\frac{1}{R} + \frac{1}{2R} + \frac{1}{4R} + \frac{1}{8R} + \frac{1}{16R} + \frac{1}{32R} + \frac{1}{64R} + \frac{1}{128R}\right)^{-1}$$

   da cui:

$$R = \left(\frac{1}{R_{min}} * \left(\frac{255}{128}\right)\right)^{-1} = 0{,}7 \ \rightarrow\ R_{min} = 1{,}4\ \Omega$$

Si sono scelti, quindi, i resistori tra i valori ancora disponibili dalla serie E12 (eventualmente utilizzando serie di resistori) :

- R1 = 1,1Ω (2x 2,2 Ω in parallelo);
- R2 = 2,2Ω;
- R3 = 4,40Ω (3,9 Ω +0,5 Ω);
- R4 = 8,76Ω (8,2 Ω + 0,56 Ω);
- R5 = 18Ω;
- R6 = 35,2Ω (33 Ω + 2,2 Ω);
- R7 = 70,2Ω (68 Ω + 2,2 Ω);
- R8 = 144Ω (120 Ω + 22 Ω).

2. *DIMENSIONEMANTO POTENZA RESISTORI*

Per quanto riguarda la scelta della massima potenza sopportata dai resistori abbiamo calcolato:

$$P_{max} = \frac{{V_{MAX\_BUS}}^2}{R_{MIN\_SCALA} + R_{SH}} = \frac{14\ V^2}{0{,}55 + 0{,}01} = 350\ W$$

- $P_{R1} = \frac{1}{2}\ P_{MAX} = 175\ W\ effettivi \rightarrow\ R_1 = R1' // R1'' con\ R1'\ e\ R1''\ da\ 100W$
- $P_{R2} = \frac{1}{4}\ P_{MAX} = 87{,}5W\ effettivi \rightarrow R_2\ da\ 100W$
- $P_{R3} = \frac{1}{8}\ P_{MAX} = 43{,}75W\ \ effettivi\ \rightarrow\ R_3\ da\ 50W$
- $P_{R4} = \frac{1}{16}\ P_{MAX} = 21{,}875W\ \ effettivi\ \rightarrow\ R_4\ da\ 25W$
- $P_{R5} = \frac{1}{32}\ P_{MAX} = 10{,}94W\ \ effettivi\ \rightarrow\ R_5\ da\ 25W\ (unica\ disponibile\ su\ RS)$
- $P_{R6} = \frac{1}{64}\ P_{MAX} = 5{,}47W\ \ effettivi\ \rightarrow\ R_6\ da\ 15W\ (unica\ disponibile\ su\ RS)$
- $P_{R7} = \frac{1}{128}\ P_{MAX} = 2{,}73W\ \ effettivi\ \rightarrow\ R_7\ da\ 15W\ (unica\ disponibile\ su\ RS)$
- $P_{RSH} = RI^2 = 0.01\Omega * (20\text{A})^2 = 4\ \text{W} \rightarrow\ R_{SH}\ da\ 25W\ (unica\ disponibile\ su\ RS)$

Dissipatore e resistenze scelte

3. DIMENSIONAMENTO DISSIPATORI

A. Dissipatori per MOS

Considerando:

Temperatura ambiente = Ta = 25 °C;

dal datasheet si trova:

- -55°C < T_j < 175°;
- $R_{thjcmax}$ = 3,3 °C/W;
- R_{thja} = 62 °C/W;
- R_{thcs} = 0,5°C/W;

dove T_j è la temperatura di lavoro della giunzione, R_{thjc} è la resistenza termica tra giunzione e involucro, R_{thja} è la resistenza termica tra giunzione e ambiente e R_{thcs} è la resistenza termica tra l'involucro e il dissipatore.

Supponiamo le seguenti condizioni di lavoro:

- Sistema a regime termico;
- V_{BUS} = 14V;
- R_{LOAD} = 0.55Ω;
- R_{SH} = 0.01Ω;
- $R_{ON,max}$ = 0.08Ω.

I transistor che assorbono più potenza sono i primi tre, nei quali passa ¼ della corrente del BUS.

$$I_{MAX} \cong \frac{V_{BUS}}{R_{LOAD} + R_{SH}} = \frac{14\ V}{0.55\Omega + 0.01\Omega} = 25\ A$$

$$P_{MOS,max} = R_{ON,max} I^2_{MOS,max} = 0{,}08 * 6{,}25^2 = 3{,}125\ W$$

Senza dissipatore si avrebbe:

$$T_j = P_{MOS,max} Rth_{ja} + T_a = 219°C$$

Dal valore di temperature ottenuto, risulta necessario un dissipatore per non uscire dalla temperatura di lavoro ammessa per il MOS.

Si impone:

$T_j \leq 90°C$;

$$Rth_{sa} \leq \frac{(T_j - T_a)}{P_{MOS,max}} - Rth_{jc} - Rth_{cs} = 17°C/W$$

Per gli altri transistor scegliamo il dissipatore considerando la IMOS,max inferiore alla metà di quella precedentemente calcolata:

$$P_{MOS,max} = R_{ON,max} I^2_{MOS,max} = 0{,}08 * 3{,}125^2 = 0{,}8\ W$$

Si impone una temperatura di giunzione inferiore:
$T_j \leq 55°C$;

$$Rth_{sa} \leq \frac{(T_j - T_a)}{P_{MOS,max}} - Rth_{jc} - Rth_{cs} = 33{,}7°C/W$$

B. Dissipatore per le resistenze di potenza

Il dissipatore scelto per il progetto è unico per tutte le resistenze di potenza.

Dalla lettura del datasheet delle resistenze si trova:

$$Rth_{diss} \leq \frac{T_{max} - T_{amb}}{Pmax} - (Rth_1 + Rth_3);$$

dove Rth_{diss} è la resistenza termica che deve avere il dissipatore, Rth_1è la resistenza termica tra l'elemento a filo e l'involucro in allumino, Rth_3è la resistenza termica tra l'involucro di alluminio e il dissipatore.

Sempre dal datasheet si trova che per il corretto funzionamento dei resistori T_{max} non deve essere superiore a 220 °C.

Per ogni resistore vengono quindi calcolati i valori della resistenza termica necessaria e successivamente sommati in parallelo:

R1',R1'',R2:

$$Rth_{diss,1} \leq \frac{220 - 25}{87.5} - (1.03 + 0.07) = 1.12 \text{ °C/W};$$

R3:

$$Rth_{diss,2} \leq \frac{220 - 25}{43.75} - (1.9 + 0.06) = 2.49 \text{ °C/W};$$

R4:

$$Rth_{diss,3} \leq \frac{220 - 25}{21.875} - (3.2 + 0.06) = 5.65 \text{ °C/W};$$

R5:

$$Rth_{diss,4} \leq \frac{220 - 25}{10.94} - (3.2 + 0.06) = 14.56\ °C/W;$$

R6:

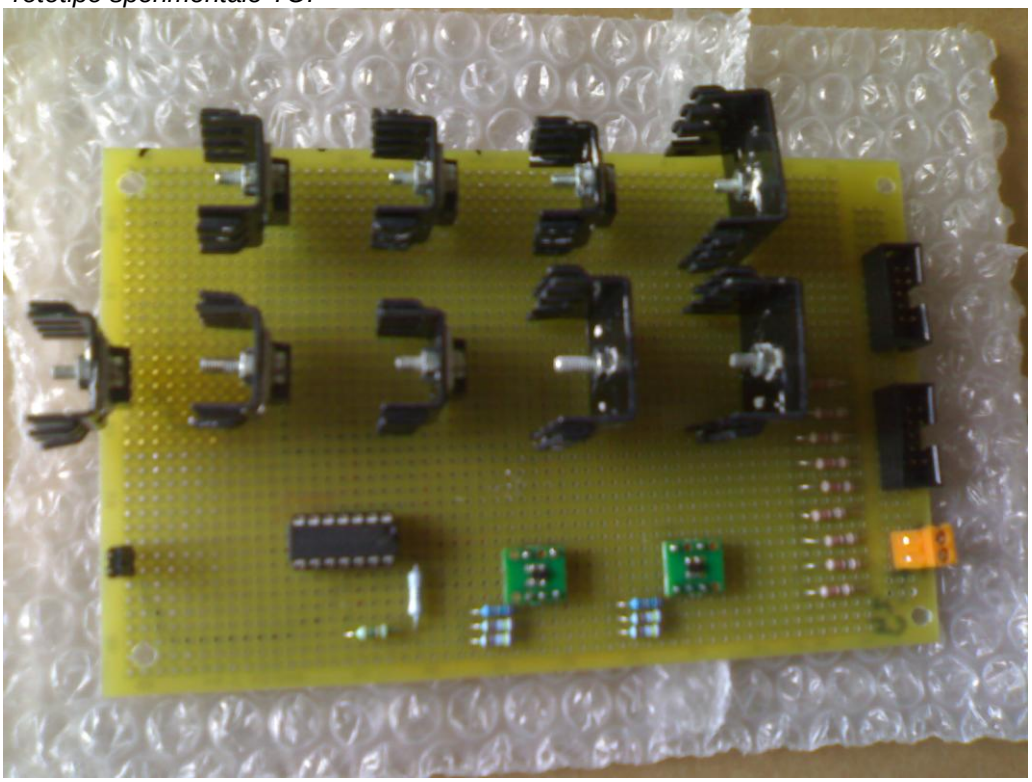
$$Rth_{diss,5} \leq \frac{220 - 25}{4} - (3.2 + 0.06) = 45.49\ °C/W;$$

R7,R8: dissipatore non necessario.

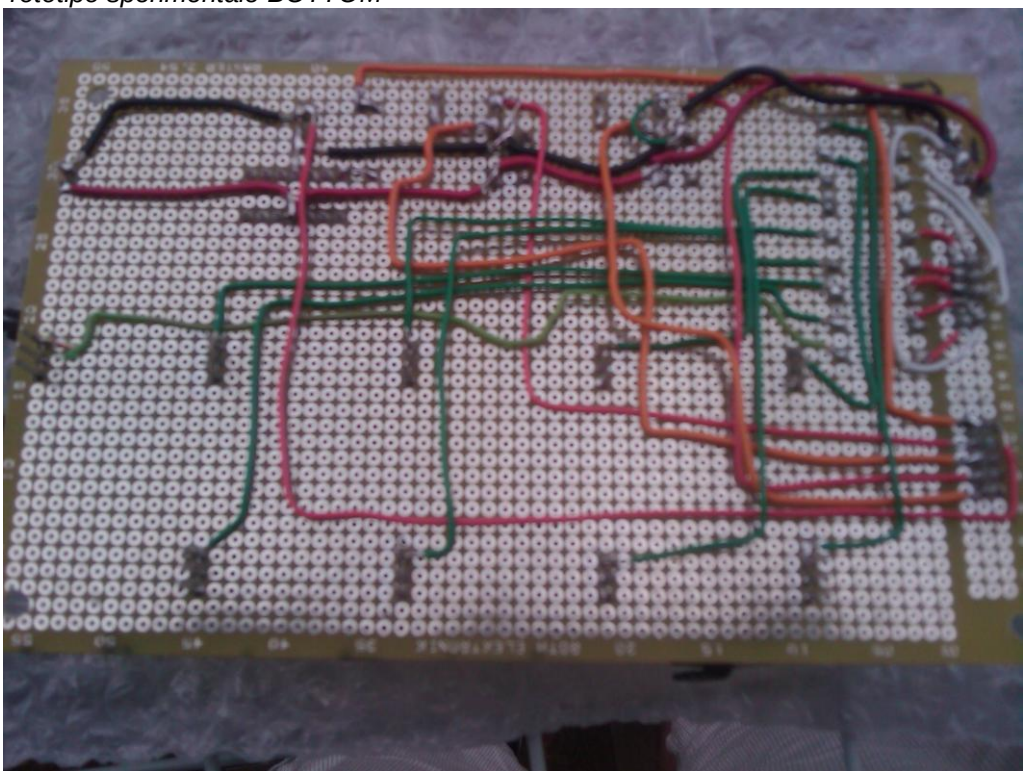
$$Rth_{diss} = \left(\frac{3}{Rth_{diss,1}} + \frac{1}{Rth_{diss,2}} + \frac{1}{Rth_{diss,3}} + \frac{1}{Rth_{diss,4}} + \frac{1}{Rth_{diss,5}}\right)^{-1} = 0.3\ °C/W$$

La scelta ricade su un dissipatore da 0.33 °C, ma è necessario ridurre ulteriormente la sua resistenza termica. Per effettuare tale riduzione, è stata introdotta una ventola, con un flusso di $61.2\ m^3/h$ di aria, che permette di abbassare di circa il 20% il valore della Rth_{diss}.

Prototipo sperimentale TOP



Prototipo sperimentale BOTTOM



3.3 CIRCUITO ELETTRICO

4.SOFTWARE

```
#include <ansi_c.h>
#include <stdlib.h>
#include <cvirte.h>
#include <userint.h>
#include <math.h>
#include <NIDAQmx.h>
#include <DAQmxIOctrl.h>
#include "Acq-IntClk.h"

#define DAQmxErrChk(functionCall)

    if( DAQmxFailed(error=(functionCall)) )
        goto Error;
    else
        static int panelHandle;


static int plotColors[12] = {VAL_RED, VAL_GREEN, VAL_BLUE, VAL_CYAN,
VAL_MAGENTA, VAL_YELLOW, VAL_DK_RED, VAL_DK_BLUE, VAL_DK_GREEN,
VAL_DK_CYAN, VAL_DK_MAGENTA, VAL_DK_YELLOW};


// MAIN

int main(int argc, char *argv[])
{

      if( InitCVIRTE(0,argv,0)==0 )
                return -1;  /* out of memory */
        if( (panelHandle=LoadPanel(0,"Acq-IntClk.uir",PANEL))<0 )
                return -1;

SetCtrlAttribute(panelHandle,PANEL_DECORATION_BLUE,ATTR_FRAME_COLOR,VAL_BLUE);

SetCtrlAttribute(panelHandle,PANEL_DECORATION_GREEN,ATTR_FRAME_COLOR,VAL_GREEN);

NIDAQmx_NewPhysChanDOLineCtrl(panelHandle,PANEL_CHANNEL_2,1);


        DisplayPanel(panelHandle);
        RunUserInterface();
        DiscardPanel(panelHandle);

        return 0;
}
```

// FUNZIONI

```
int CVICALLBACK PanelCallback(int panel, int event, void *callbackData, int eventData1, int eventData2)

{
	if( event==EVENT_CLOSE )
		QuitUserInterface(0);

	return 0;
}

int CVICALLBACK RangeCallback(int panel, int control, int event, void *callbackData, int eventData1, int eventData2)
{
	if( event==EVENT_COMMIT ) {
		double  min,max;

		GetCtrlVal(panel,PANEL_MINVAL,&min);
		GetCtrlVal(panel,PANEL_MAXVAL,&max);

		if( min<max )

	SetAxisScalingMode(panel,PANEL_GRAPH,VAL_LEFT_YAXIS,VAL_MANUAL,min,max);
		return 1;
	}
	return 0;
}

int CVICALLBACK AcquireCallback(int panel, int control, int event, void *callbackData, int eventData1, int eventData2)
{

	TaskHandle  taskHandle=0;
	char        chan[256];
	char        errBuff[2048]={'\0'};
	char        tensione[256];
	float mediaV = 0;
	float64     min,max,min1,min2,max1,max2,rate;
	float64     *data=NULL;
	uInt32      sampsPerChan;
	uInt32      i;
	uInt32      numChannels;
	int32       numRead;
	int32       error=0;
	int         log;

	if( event==EVENT_COMMIT )
	{

		strcpy(chan,"Dev1/ai0");        //Impostazione canale di acquisizione (Tensione)

		GetCtrlVal(panel,PANEL_MINVAL,&min);
		GetCtrlVal(panel,PANEL_MAXVAL,&max);
```

```
GetCtrlVal(panel,PANEL_MINVAL_2,&min1);
GetCtrlVal(panel,PANEL_MAXVAL_2,&max1);
GetCtrlVal(panel,PANEL_SAMPSPERCHAN,&sampsPerChan);
GetCtrlVal(panel,PANEL_RATE,&rate);

// riscala il grafico
SetCtrlAttribute(panel,PANEL_GRAPH,ATTR_XAXIS_GAIN,1.0/rate);

// riscala il grafico
SetCtrlAttribute(panel,PANEL_GRAPH_2,ATTR_XAXIS_GAIN,1.0/rate);
log = (int)log10(rate);

// riscala il grafico
SetCtrlAttribute(panel,PANEL_GRAPH,ATTR_XPRECISION,log);
SetCtrlAttribute(panel,PANEL_GRAPH_2,ATTR_XPRECISION,log);

// Pulisce il grafico
DeleteGraphPlot(panel,PANEL_GRAPH,-1,VAL_IMMEDIATE_DRAW);

DeleteGraphPlot(panel,PANEL_GRAPH_2,-1,VAL_IMMEDIATE_DRAW);


/*********************************************/
// DAQmx Configure Code
/*********************************************/

SetWaitCursor(1);
DAQmxErrChk (DAQmxCreateTask("",&taskHandle));


DAQmxErrChk(DAQmxCreateAIVoltageChan(taskHandle,chan,"",DAQmx_Val_Diff,min,max,D
AQmx_Val_Volts,NULL));

DAQmxErrChk(DAQmxCfgSampClkTiming(taskHandle,"",rate,DAQmx_Val_Rising,DAQmx_Val_Finite
Samps,sampsPerChan));


DAQmxErrChk (DAQmxGetTaskAttribute(taskHandle,DAQmx_Task_NumChans,&numChannels));


if( (data=malloc(sampsPerChan*numChannels*sizeof(float64)))==NULL )
 {
        MessagePopup("Error","Not enough memory");
        goto Error;
}


/*********************************************/
// DAQmx Start Code
/*********************************************/

DAQmxErrChk (DAQmxStartTask(taskHandle));

SetCtrlAttribute(panel,PANEL_ACQUIRE,ATTR_DIMMED,1);
```

```
		ProcessDrawEvents();

		/*********************************************/
		// DAQmx Read Code
		/*********************************************/
DAQmxErrChk(DAQmxReadAnalogF64(taskHandle,sampsPerChan,10.0,DAQmx_Val_GroupByChannel,data,sampsPerChan*numChannels,&numRead,NULL));


		for(i=0;i<sampsPerChan;i++)		// Riscalamento segnale acquisito
		{

			data[i] = (data[i]-0.195)*1.62 ;

			mediaV = mediaV + data[i];

		}


		mediaV = mediaV/(i-1);

		sprintf(tensione, "%.3f V", mediaV);	// Calcolo e stampa della media

		SetCtrlVal (panel, PANEL_STRING_2, tensione);

		if( numRead>0 )
			for(i=0;i<numChannels;i++)

PlotY(panel,PANEL_GRAPH,&(data[i*numRead]),numRead,VAL_DOUBLE,VAL_THIN_LINE,VAL_EMPTY_SQUARE,VAL_SOLID,1,plotColors[i%12]);
	}

Error:

	SetWaitCursor(0);
	if( DAQmxFailed(error) )
		DAQmxGetExtendedErrorInfo(errBuff,2048);

	if( taskHandle!=0 ) {
		/*********************************************/
		// DAQmx Stop Code
		/*********************************************/
		DAQmxStopTask(taskHandle);
		DAQmxClearTask(taskHandle);

		SetCtrlAttribute(panel,PANEL_ACQUIRE,ATTR_DIMMED,0);
	}
	if( data )
		free(data);
	if( DAQmxFailed(error) )
		MessagePopup("DAQmx Error",errBuff);
	return 0;
}

int CVICALLBACK RangeCallback2 (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
```

```
{
	switch (event)
	{

	case EVENT_COMMIT:

	double  min,max;

		GetCtrlVal(panel,PANEL_MINVAL_2,&min);
		GetCtrlVal(panel,PANEL_MAXVAL_2,&max);
		if( min<max )

	SetAxisScalingMode(panel,PANEL_GRAPH_2,VAL_LEFT_YAXIS,VAL_MANUAL,min,max);

			break;
	}
	return 0;
}

int CVICALLBACK AcquireCallback2 (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)

{	int32       error=0;
	TaskHandle  taskHandle=0;
	char        chan[256];

	float64     min,max,rate;
	uInt32      sampsPerChan;
	int32       numRead;
	uInt32      numChannels;
	float64     *data=NULL;
	int         log;
	char        errBuff[2048]={'\0'};
	uInt32      i;
	char corrente[256];
	float mediaI = 0;

	if( event==EVENT_COMMIT )
	 {

		strcpy(chan,"Dev1/ai1");        //Impostazione canale di acquisizione (Corrente OUT)


		GetCtrlVal(panel,PANEL_MINVAL_2,&min);
		GetCtrlVal(panel,PANEL_MAXVAL_2,&max);
		GetCtrlVal(panel,PANEL_SAMPSPERCHAN,&sampsPerChan);
		GetCtrlVal(panel,PANEL_RATE,&rate);

		// riscala il grafico
		SetCtrlAttribute(panel,PANEL_GRAPH_2,ATTR_XAXIS_GAIN,1.0/rate);
		log = (int)log10(rate);

		SetCtrlAttribute(panel,PANEL_GRAPH_2,ATTR_XPRECISION,log);

		DeleteGraphPlot(panel,PANEL_GRAPH_2,-1,VAL_IMMEDIATE_DRAW);
```

```
		/*********************************************/
		// DAQmx Configure Code
		/*********************************************/

		SetWaitCursor(1);
		DAQmxErrChk (DAQmxCreateTask("",&taskHandle));
		DAQmxErrChk
(DAQmxCreateAIVoltageChan(taskHandle,chan,"",DAQmx_Val_Diff,min,max,DAQmx_Val_Volts,NULL
));
DAQmxErrChk(DAQmxCfgSampClkTiming(taskHandle,"",rate,DAQmx_Val_Rising,DAQmx_Val_Finite
Samps,sampsPerChan));

DAQmxErrChk (DAQmxGetTaskAttribute(taskHandle,DAQmx_Task_NumChans,&numChannels));


		if( (data=malloc(sampsPerChan*numChannels*sizeof(float64)))==NULL ) {
			MessagePopup("Error","Not enough memory");
			goto Error;
		}


		/*********************************************/
		// DAQmx Start Code
		/*********************************************/

		DAQmxErrChk (DAQmxStartTask(taskHandle));

		SetCtrlAttribute(panel,PANEL_ACQUIRE,ATTR_DIMMED,1);
		ProcessDrawEvents();

		/*********************************************/
		// DAQmx Read Code
		/*********************************************/

DAQmxErrChk(DAQmxReadAnalogF64(taskHandle,sampsPerChan,10.0,DAQmx_Val_GroupByChan
nel,data,sampsPerChan*numChannels,&numRead,NULL));


		for(i=0;i<sampsPerChan;i++){  // Riscalamento valore acquisito

			data[i] = -2.40*(data[i]);

		media = media + data[i];

		}


		media = media/(i-1);			// Calcolo e stampa della media

		sprintf(corrente, "%.3f A", media);

		SetCtrlVal (panel, PANEL_STRING_3, corrente);


		if( numRead>0 )
			for(i=0;i<numChannels;i++)
```

```
PlotY(panel,PANEL_GRAPH_2,&(data[i*numRead]),numRead,VAL_DOUBLE,VAL_THIN_LINE,VAL_
EMPTY_SQUARE,VAL_SOLID,1,plotColors[i%12]);
        }

Error:
        SetWaitCursor(0);
        if( DAQmxFailed(error) )
                DAQmxGetExtendedErrorInfo(errBuff,2048);
        if( taskHandle!=0 ) {

                /*********************************************/
                // DAQmx Stop Code
                /*********************************************/
                DAQmxStopTask(taskHandle);
                DAQmxClearTask(taskHandle);

                SetCtrlAttribute(panel,PANEL_ACQUIRE,ATTR_DIMMED,0);
        }

        if( data )
                free(data);
        if( DAQmxFailed(error) )
                MessagePopup("DAQmx Error",errBuff);
        return 0;
}

int CVICALLBACK AcquireCallback3 (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{       int32       error=0;
        TaskHandle  taskHandle=0;
        char        chan[256];
        float64     min,max,rate;
        uInt32      sampsPerChan;
        int32       numRead;
        uInt32      numChannels;
        float64     *data=NULL;
        int         log;
        char        errBuff[2048]={'\0'};
        uInt32      i;
        char                corrente[256];
        float mediaI = 0;

        if( event==EVENT_COMMIT ) {

                strcpy(chan,"Dev1/ai2");        //Impostazione canale di acquisizione (Corrente IN)

                GetCtrlVal(panel,PANEL_MINVAL_2,&min);
                GetCtrlVal(panel,PANEL_MAXVAL_2,&max);
                GetCtrlVal(panel,PANEL_SAMPSPERCHAN,&sampsPerChan);
                GetCtrlVal(panel,PANEL_RATE,&rate);

                // riscala il grafico
                SetCtrlAttribute(panel,PANEL_GRAPH_2,ATTR_XAXIS_GAIN,1.0/rate);
                log = (int)log10(rate);
                SetCtrlAttribute(panel,PANEL_GRAPH_2,ATTR_XPRECISION,log);
```

```
DeleteGraphPlot(panel,PANEL_GRAPH_2,-1,VAL_IMMEDIATE_DRAW);

		/*********************************************/
		// DAQmx Configure Code
		/*********************************************/
		SetWaitCursor(1);
		DAQmxErrChk (DAQmxCreateTask("",&taskHandle));

DAQmxErrChk(DAQmxCreateAIVoltageChan(taskHandle,chan,"",DAQmx_Val_Diff,min,max,DAQmx_
Val_Volts,NULL));

DAQmxErrChk(DAQmxCfgSampClkTiming(taskHandle,"",rate,DAQmx_Val_Rising,DAQmx_Val_Finite
Samps,sampsPerChan));

DAQmxErrChk (DAQmxGetTaskAttribute(taskHandle,DAQmx_Task_NumChans,&numChannels));

		if( (data=malloc(sampsPerChan*numChannels*sizeof(float64)))==NULL ) {
			MessagePopup("Error","Not enough memory");
			goto Error;
		}

		/*********************************************/
		// DAQmx Start Code
		/*********************************************/

		DAQmxErrChk (DAQmxStartTask(taskHandle));

		SetCtrlAttribute(panel,PANEL_ACQUIRE,ATTR_DIMMED,1);
		ProcessDrawEvents();

		/*********************************************/
		// DAQmx Read Code
		/*********************************************/

DAQmxErrChk(DAQmxReadAnalogF64(taskHandle,sampsPerChan,10.0,DAQmx_Val_GroupByChan
nel,data,sampsPerChan*numChannels,&numRead,NULL));


		for(i=0;i<sampsPerChan;i++){ // Riscalamento valori acquisiti

			data[i] = 2.40*(data[i]);

		media = media + data[i];

		}

		media = media/(i-1); // Calcolo e stampa della media

		sprintf(corrente, "%.3f A", media);

		SetCtrlVal (panel, PANEL_STRING_3, corrente);

		if( numRead>0 )
			for(i=0;i<numChannels;i++)
```

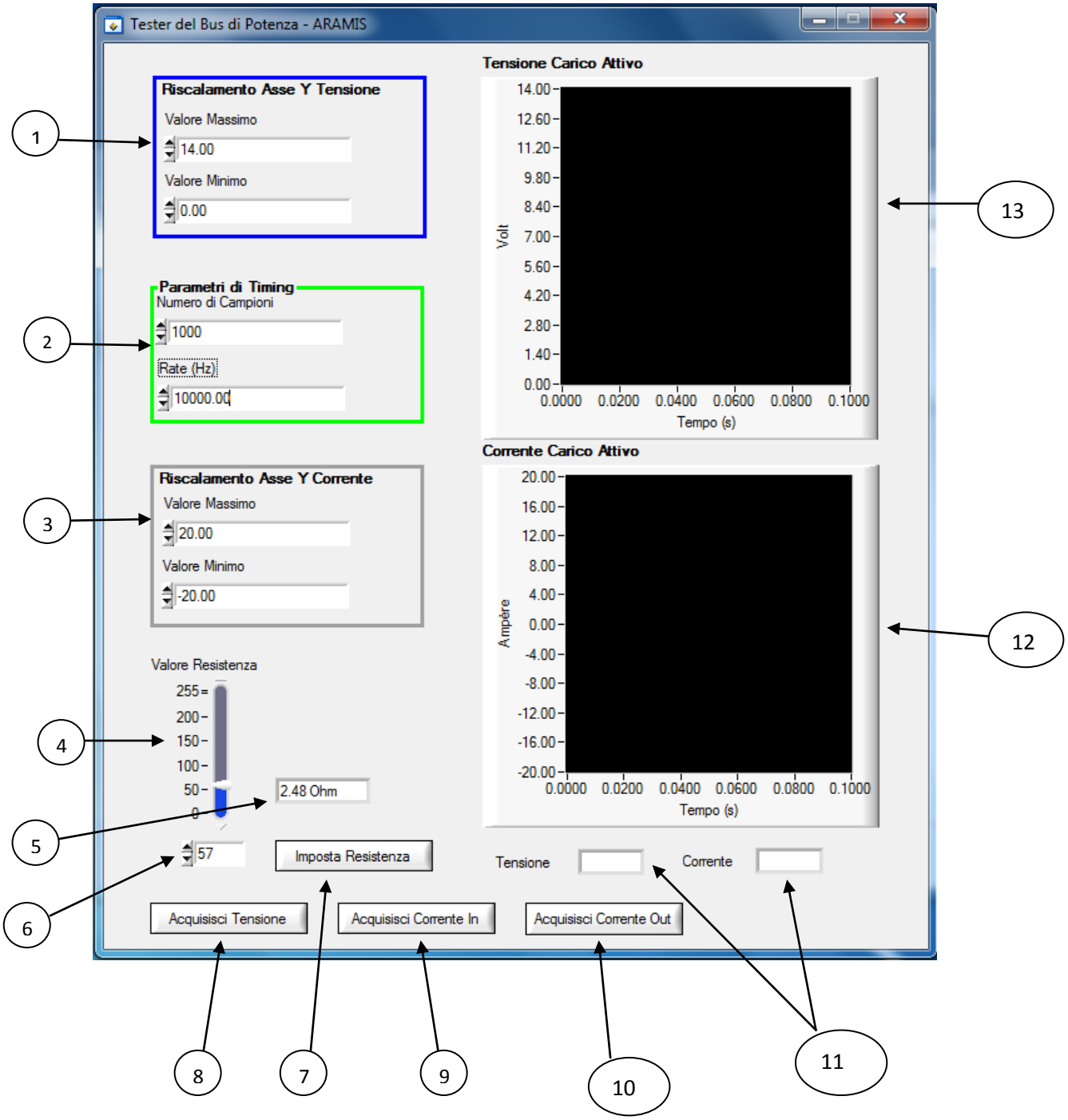
```
PlotY(panel,PANEL_GRAPH_2,&(data[i*numRead]),numRead,VAL_DOUBLE,VAL_THIN_LINE,VAL_
EMPTY_SQUARE,VAL_SOLID,1,plotColors[i%12]);

        }

Error:
        SetWaitCursor(0);
        if( DAQmxFailed(error) )
                DAQmxGetExtendedErrorInfo(errBuff,2048);
        if( taskHandle!=0 ) {

                /*********************************************/
                // DAQmx Stop Code
                /*********************************************/
                DAQmxStopTask(taskHandle);
                DAQmxClearTask(taskHandle);

                SetCtrlAttribute(panel,PANEL_ACQUIRE,ATTR_DIMMED,0);
        }


        if( data )
                free(data);

        if( DAQmxFailed(error) )
                MessagePopup("DAQmx Error",errBuff);
        return 0;
}
int CVICALLBACK WriteCallback(int panel, int control, int event, void *callbackData, int eventData1, int
eventData2)
{
        int        error=0;
        TaskHandle    taskHandle=0;
        char       chan[256];
        uInt8      data[8];
        int                        resistenza;
        char       errBuff[2048]={'\0'};
        int i;


        if( event==EVENT_COMMIT ) {

          strcpy(chan,"Dev1/port0/line0:7");  //Impostazione dei canali di outpit (Controllo Mos)


                /*********************************************/
                // DAQmx Configure Code
                /*********************************************/
                SetWaitCursor(1);
                DAQmxErrChk (DAQmxCreateTask("",&taskHandle));

DAQmxErrChk (DAQmxCreateDOChan(taskHandle,chan,"",DAQmx_Val_ChanForAllLines));

                /*********************************************/
                // DAQmx Start Code
                /*********************************************/
```

```
		DAQmxErrChk (DAQmxStartTask(taskHandle));

		/*********************************************/
		// DAQmx Write Code
		/*********************************************/

		GetCtrlVal(panel,PANEL_NUMERICSLIDE,&resistenza);

		for (i = 0; i<8 ; i++){  // Calcolo sequenza di bit da inviare ai MOS

		data[7-i] = resistenza%2;
		resistenza = resistenza /2;
		}

		DAQmxErrChk
(DAQmxWriteDigitalLines(taskHandle,1,1,10.0,DAQmx_Val_GroupByChannel,data,NULL,NULL));

	}

Error:

	SetWaitCursor(0);
	if( DAQmxFailed(error) )

		DAQmxGetExtendedErrorInfo(errBuff,2048);
	if( taskHandle!=0 ) {
		/*********************************************/
		// DAQmx Stop Code
		/*********************************************/
		DAQmxStopTask(taskHandle);
		DAQmxClearTask(taskHandle);
	}
	if( DAQmxFailed(error) )
		MessagePopup("DAQmx Error",errBuff);

return 0;
}

int CVICALLBACK CAMBIA (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{


	float		res[8]= {1.1, 2.2, 4.4,8.76,18,35.2,70.2,142};
	uInt8		data[8],temp;
	int		resistenza,i;
	float		somma =0;
	float		risultato;
	char		resparal[80];
```

```
switch (event)
	{
		case EVENT_COMMIT:
			GetCtrlVal(panel,PANEL_NUMERICSLIDE,&resistenza);


		for (i = 0; i<8 ; i++){		// Calcolo del parallelo delle resistenze selezionate

		data[7-i] = resistenza%2;
		resistenza = resistenza /2;
		if (data[7-i] == 1) {
		somma = somma + 1/(res[7-i]);
		}

		}

		risultato = 1/somma;

		sprintf(resparal, "%.2f Ohm", risultato);

			SetCtrlVal (panel, PANEL_STRING, resparal);	// Scrittura del valore della
									resistenza di carico

			break;
	}
	return 0;
}


			// FINE SOFTWARE
```

4.1 ISTRUZIONI SOFTWARE

Tramite l'interfaccia grafica del software realizzato è possibile controllare la scelta della resistenza di carico e visualizzare le misure di corrente e tensione acquisite dal dispositivo.
Tali misure vengono riportate su due differenti grafici, e la media delle acquisizioni in due caselle di testo sottostanti come si può vedere dalla seguente immagine:



1) Con questa maschera si impostano i *valori massimi e minimi* visualizzabili nel grafico della *tensione del carico attivo* in modo da avere una visualizzazione della forma d'onda ottimale sull'asse delle *y*.

2) Grazie a questa maschera, invece, si impostano i *numeri di campioni* ed il *rate* per i parametri di timing per impostare l'asse delle *x* di entrambi i grafici di tensione e corrente

3) Allo stesso modo della tensione, si impostano i valori massimi e minimi visualizzabili sull'asse delle *y* nel grafico della *corrente del carico attivo*.

4) Slider per impostare l'equivalente della resistenza di carico in bit.

5) Visualizzazione resistenza di carico espressa in *Ohm.*

6) Finestra per inserire manualmente il valore della resistenza nell'equivalente di bit.

7) Pulsante per impostare nel circuito la resistenza di carico scelta tramite i punti 4) o 5).

8) Pulsante per l'acquisizione della tensione ai capi del carico impostato.

9) Pulsante per l'acquisizione della corrente (verso entrante) nel bus di potenza.

10) Pulsante per l'acquisizione della corrente (verso uscente) nel bus di potenza.

11) Valori medi della tensione e della corrente acquisiti.

12) Grafico della corrente del carico attivo acquisita.

13) Grafico della tensione del carico attivo acquisita.

5. LISTA DEI COMPONENTI E COSTO

Num	***Quantità***	***Particolare***	***Codice Componente***	***P.U. x1Pz €***	***P.U. x10Pz €***	***P.U. x100Pz €***
1	2	INA138NA	660-7275	2,77	2,48	1,84
2	1	LM324N	533-8215	0,6	0,53	0,51
3	9	IRL2703PBF	541-2177	1,56	1,14	0,52
4	8	15 Ω 0.6W 1%	477-7574	3,2	2,3	1,55
5	1	10 mΩ 25W 10%	160-629	2,32	2,1	1,36
6	3	2,2Ω 100W 5%	188-093	7,7	6,85	4,21
7	1	3,9Ω 50W 5%	107-4125	2,81	2,58	1,87
8	1	0,5Ω 50W 5%	158-452	2,81	2,58	1,87
9	1	8,2Ω 25W 5%	107-3908	2,32	2,1	1,36
10	1	0,56Ω 25W 5%	615-0397	2,32	2,1	1,36
11	1	18Ω 25W 5%	160-742	2,32	2,1	1,36
12	1	33Ω 25W 5%	160-764	2,32	2,1	1,36
13	1	2,2Ω 25W 5%	157-538	2,32	2,1	1,36
14	1	120Ω 15W 5%	160-584	2,13	1,84	1,32
15	1	2,2Ω 15W 5%	159-736	2,13	1,84	1,32
16	1	120Ω 15W 5%	160-584	2,13	1,84	1,32
17	1	22Ω 15W 5%	159-770	2,13	1,84	1,32
18	1	36,5KΩ 0,6W 1%	683-3578		0,056	
19	1	91KΩ 0,6W 1%	148-966	0,39	0,29	0,22
20	2	250KΩ 0,6W 1%	683-3578	0,39	0,29	0,22
21	1	Ventola assiale 80x80x20mm 24V Maglev	191-682	7,44	6,62	5,94
22	1	Dissipatore 0.33°C/W EM/B/250	490-7056	27,43	24,01	22,76
23	1	LCD alfanumerico display 164B-CC-BC-3LP	532-6375	9,91	7,93	5,95
			TOT P.U. €	142,95	119,31	86,45

Tutti i componenti sono stati acquistati dal fornitore RS.

6.CONCLUSIONI

Dopo aver montato il prototipo, abbiamo effettuato alcuni test riguardo al suo funzionamento. Il circuito si comporta come ci si aspettava con la sovrapposizione di alcuni offset nella misurazione di correnti e tensioni, adeguatamente corretti tramite software. Per apportare tali correzioni, sono state confrontate le misurazioni eseguite dal prototipo, con quelle effettuate con strumenti industriali tarati.

La tolleranza lavorando a fondo scala e non, risulta inferiore allo 0.6%.

Prototipo sperimentale finale