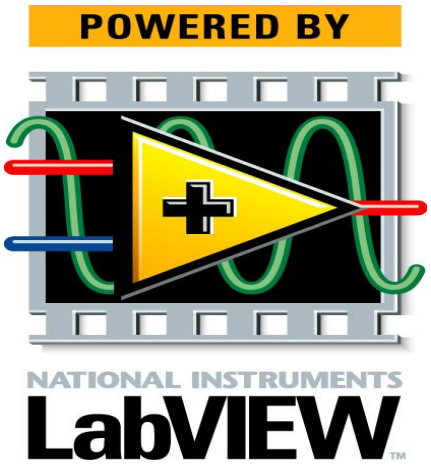
TECNICHE ACQUISIZIONE DATI 2

A.A.2009/2010

Esperienze in laboratorio



INTRODUZIONE

Questo documento spiega come sono state realizzate le esperienze di laboratorio assegnate durante il corso di Acquisizione Dati 2 dal Professore G. Ambrosi. Elementi utilizzati per la relazione:

1) NI LabView 6.0 for Windows installata nel laboratorio
2) Scheda Acquisizione dati NI 6024E
3) NI LabView 6.1 versione for Linux utilizzata nel PC personale
4) Scheda audio

Si parla di Acquisizione dati, ma che cosa significa e quanto è importante?

L'acquisizione dell'informazione è per l'uomo da sempre il fulcro della sua esistenza. Il nostro cervello recepisce informazioni e invia informazioni in modo continuo. Compie quindi continuamente operazioni di *acquisizione di dati*. Con lo sviluppo delle tecnologie si è portato questa "capacità" nei sistemi dell'informazione con ottimi risultati. Chiaramente l'uso di queste tecnologie è assai ampio. Tipicamente i sistemi di acquisizione dati si basano sullo studio di un fenomeno che porta all'acquisizione dell'informazione basata su un *segnale,* cioè una grandezza fisica variabile nel tempo a cui è associata un informazione. Il segnale acquisito, quindi, viene studiato ed elaborato.

Il caso più elementare e tipico di sistema d'acquisizione dati è quello composto da:

1) un **trasduttore** che "legge" il segnale e lo converte in segnale elettrico.
2) Un **Trigger** seleziona gli eventi e decide se fare la lettura del trasduttore.
3) **ADC** che ha il compito di trasformare l'informazione analogica in digitale.
4) un **sistema di elaborazione** che prende il segnale digitale in ingresso e provvede a gestirlo a seconda dello scopo del suo utilizzo (ad esempio: scrittura su file, masterizzazione su cd, etc..).

LABVIEW

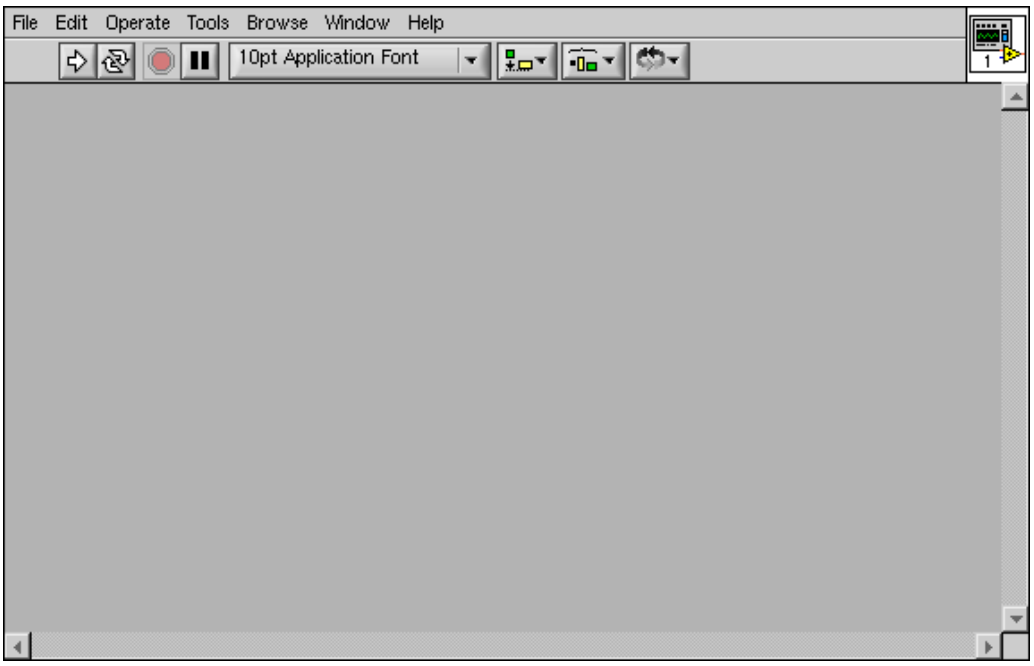
LabView (abbreviazione di **Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench) è l'ambiente di sviluppo per il linguaggio di programmazione visuale di National Instruments. Il programmatore che userà tale linguaggio non dovrà creare nemmeno una riga di codice in quanto è completamente grafico! Per questo viene chiamato *Linguaggio G (G sta per graphic)*.

Un fattore molto importante di LabView è che traduce in C le istruzioni che l'utente definisce durante la fase di stesura del codice in modo completamente trasparente.

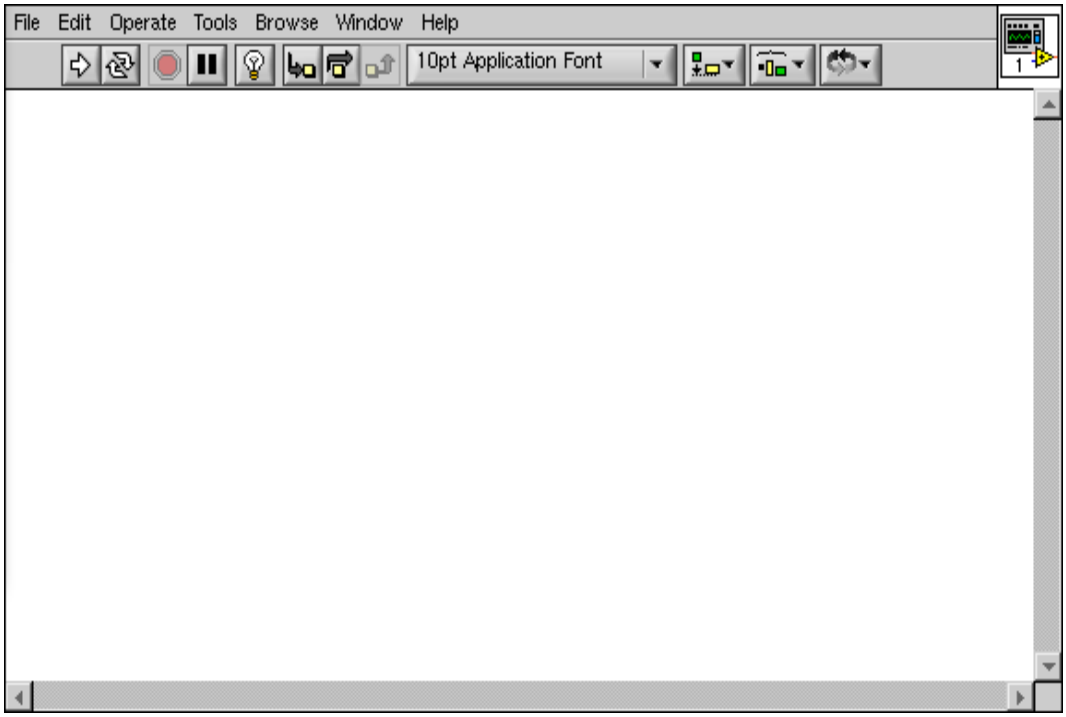
Il programma con estensione VI (**V**irtual **I**nstrument) è salvato come file binario ed è visualizzabile solo da LabView.

L'interfaccia è composta dal C*ontrol Panel* e dal *Block Diagram.*

Nel *Control Panel* si crea l'interfaccia grafica dove l'utente può inserire valori, cambiare lo stato delle variabili, leggere il risultato delle operazioni svolte dal programma, analizzare grafici, etc... .



Il *Block Diagram* è una pagina bianca dove il programmatore inserisce i vari oggetti. Come detto nulla è scritto, ma tutto ha una rappresentazione grafica.



I motivi per cui questo linguaggio ha preso piede sono molti:

_ la semplicità di programmazione

_ il suo editor completamente grafico

_ la semplicità di utilizzo

_ grande versatilità

_ la velocità di realizzazione di un programma

ESEPERIENZE IN LABORATORIO

Le esperienze assegnate sono 4 e sono riassunte sinteticamente qua sotto.

Esercitazione 1:

Scrivere un VI per sintetizzare un segnale di onda quadra a partire dai suoi coefficienti di Fourier.

Esercitazione 2:

Scrivere un VI per realizzare un registratore digitale di voce.

Esercitazione 3:

Scrivere un VI per realizzare lo studio in frequenza dei segnali sinusoidali, triangolari e quadratici

Esercitazione 4:

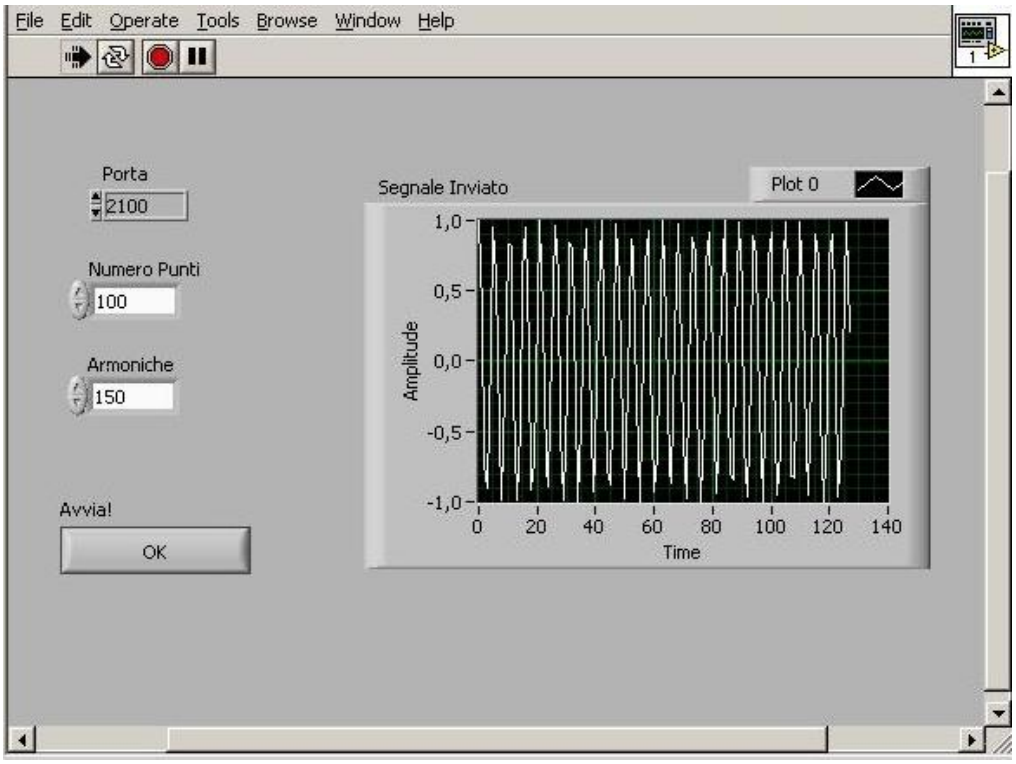
Scrivere un VI per realizzare una stazione termometrica basata su sensori LM35 e la scheda NI 6024E.

ESEPRIENZA 1

Si scriva un VI per sintetizzare un segnale di onda quadra a partire dai suoi coefficienti di Fourier. Rendere disponibile in rete i dati della forma d'onda e graficare la forma d'onda prodotta da un host remoto.

L'esercizio è stato realizzato in due file separati. Uno è stato creato per il Server, l'altro per il Client. Innanzitutto vediamo come sono strutturati i Control Panel

Control Panel - *Server*



Nel Control Panel del Server troviamo:

Porta: indica il port number che serve per la connessione con il Client tramite tcp.

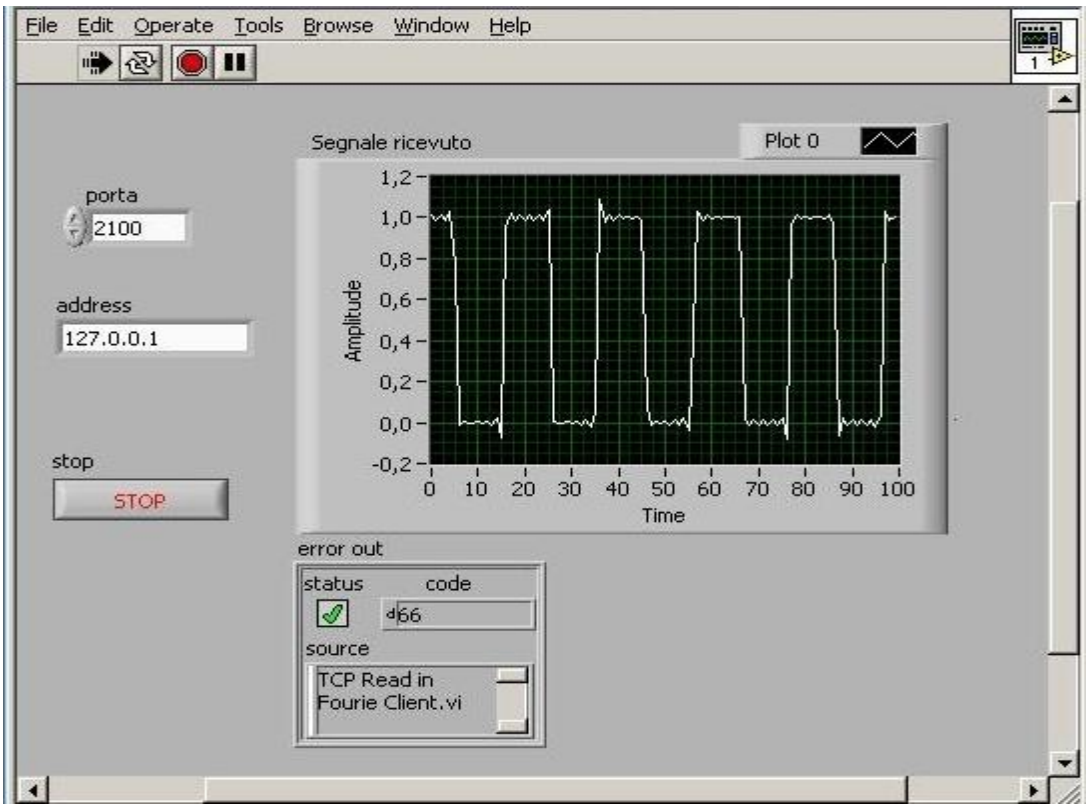
Numero punti: numero di valori che il programma utilizzerà per creare il segnale.

Armoniche: il numero delle armoniche. Più alto è il numero, più l'onda quadra ricostruita sarà precisa.

Bottone "OK": avvia e ferma il Server.

Segnale Inviato: mostra le armoniche che vengono generate e inviate.

Control Panel - *Client*



Nel Client invece troviamo:

Porta: indica il port number, deve essere lo stesso del Server per permettere l'effettiva connessione.

Address: indirizzo del Client, in questo caso è la stessa macchina del Server per cui sarà l'indirizzo di local host: 127.0.0.1

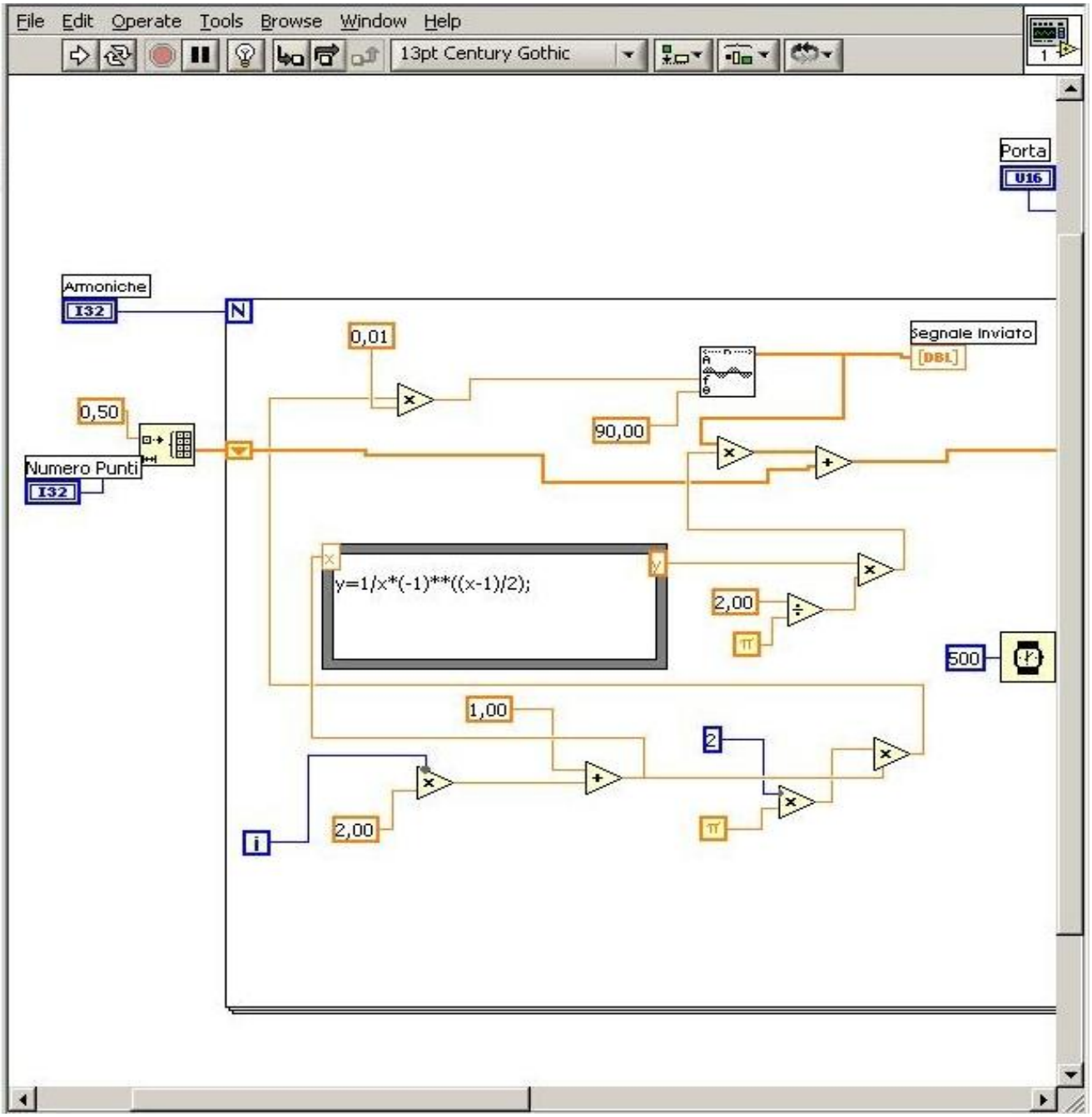
Bottone "STOP": termina l'esecuzione del Client.

Segnale Ricevuto: permette di vedere il segnale in ingresso.

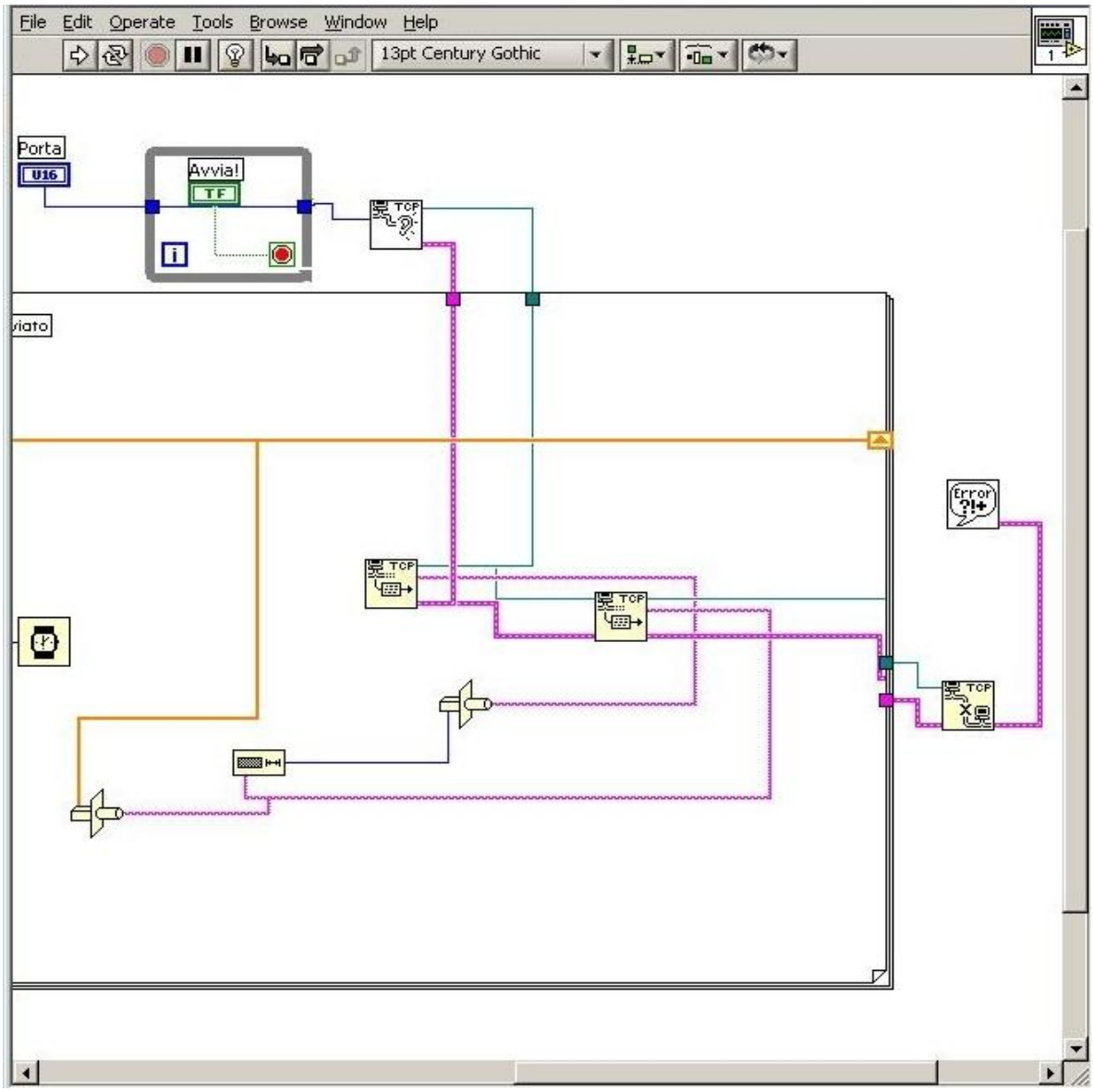
Error out: consente di vedere se la connessione con il Server ha generato qualche problema riportando eventualmente il codice dell'errore.

Passiamo ora a controllare più in dettaglio il programma andando a visualizzare i **Block Diagram** iniziando dal Server

Block Diagram1 - *Server*



Block Diagram2 - *Server*

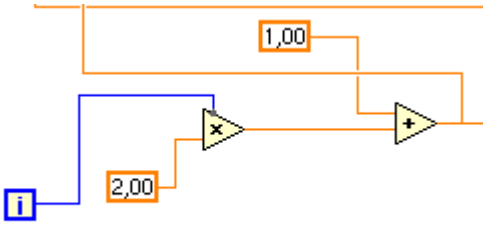


Purtroppo ho dovuto restringere e suddividere in due parti il Block Diagram del Server per motivi di spazio. Ho tentato di suddividere le parti in modo tale da avere nella prima immagine la parte di sintesi, mentre nella seconda la parte, la connessione al Client.

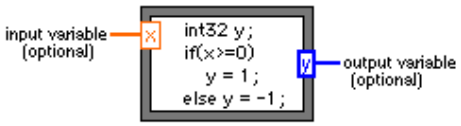
Nella parte di sintesi andiamo a vedere come ho sviluppato Fourier in LabView.

$$x(t) = \frac{1}{2} + \frac{2}{\pi}\sum_{k=1}^{\infty}\frac{1}{k}(-1)^{(k-1)/2}\cos(2\pi f_0 t) \qquad k\ dispari$$

Per fare in modo che *k* sia dispari, prendo la “i” (variabile del ciclo for che sale linearmente) la moltiplico per 2 e aggiungo 1, come posso vedere nell'immagine sottostante



L'argomento della sommatoria lo ottengo grazie al *Formula Node*, infatti questo oggetto mi permettere di scrivere al suo interno formule matematiche in linguaggio simil C.

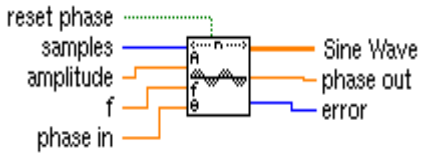


Formula Node

Evaluates mathematical formulae and expressions similar to C on the block diagram. The following built-in functions are allowed in formulas: abs, acos, acosh, asin, asinh, atan, atanh, ceil, cos, cosh, cot, csc, exp, expm1, floor, getexp, getman, int, intrz, ln, lnp1, log, log2, max, min, mod, rand, rem, sec, sign, sin, sinc, sinh, sqrt, tan, tanh.

Il risultato viene poi sommato con gli elementi dell’array inizializzato appositamente ad ½. Per completare la formula di Fourier manca il termine cosinusoide e la sommatoria.

Il primo problema lo risolviamo grazie all'oggetto *Sine Wave.vi* il quale inserendo gli adeguati parametri crea ciò che vogliamo



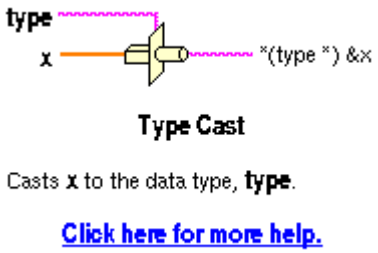
Sine Wave.vi

Generates an array containing a sine wave.

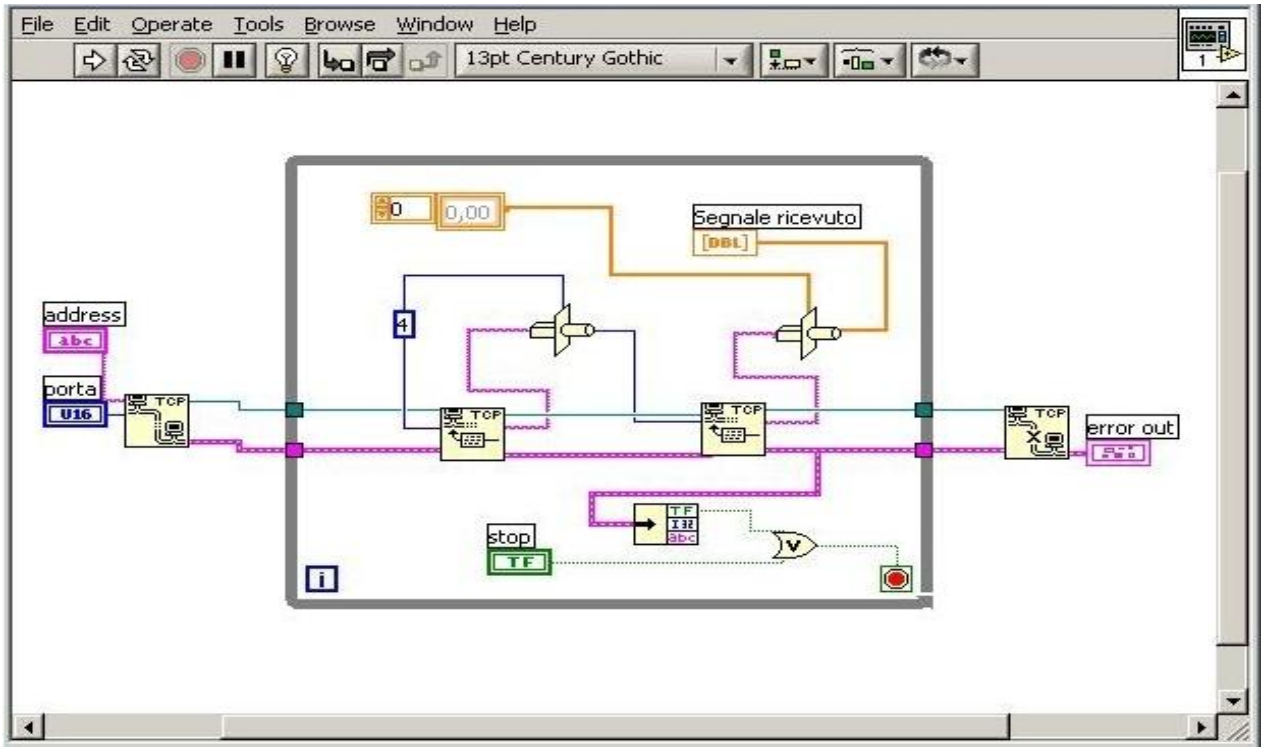
Il secondo problema lo risolviamo andando a creare un registro di scorrimento che rimanda gli output in input, mettendo poi un oggetto che “somma”, ottengo così la sommatoria.

La fase di sintesi sembra così conclusa, andiamo ora ad analizzare la seconda immagine, quella cioè del collegamento con il Client.

In alto nell'immagine vediamo un ciclo while con all'interno un bottone booleano che avvia e ferma il server. Una volta che il Server è partito, questi prende l'informazione elaborata dalla parte di sintesi e la invia al Client. L'informazione inviata è stata opportunatamente modificata dall'operatore *Type Cast:*



Block Diagram – *Client*



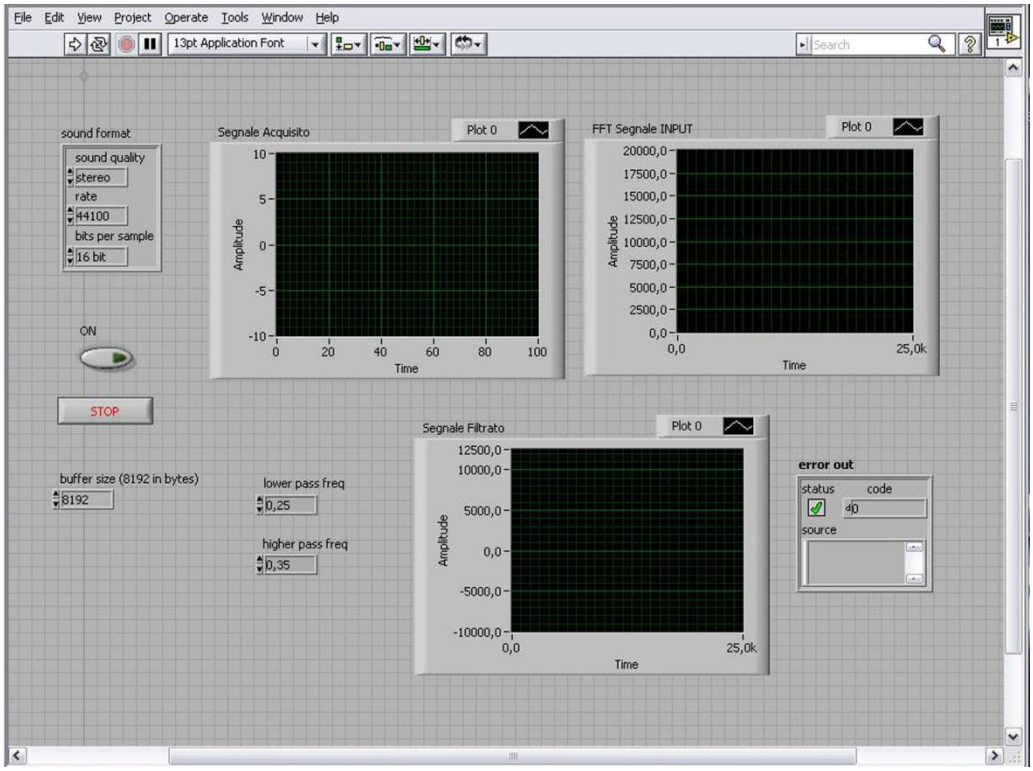
Come possiamo vedere il Block Diagram del Client è molto semplice in quanto consiste solo nei blocchi di TCP che servono alla connessione, i *type cast* che mi ritraducono l'informazione ricevuta e un WaveForm Graph che mi fa vedere il segnale che mi arriva.

Anche qui ho inserito un bottone (stop) che se premuto interrompe il Client.

ESPERIENZA 2

Si scriva un VI per realizzare un registratore digitale di voce, il quale deve essere in grado di salvare dati su file solo in presenza di voce umana; riprodurre il segnale registrato con opportuno filtro per ottimizzare l'ascolto; salvare su file i valori delle frequenze principali che compongono il segnale.

Control Panel



Nel Control Panel troviamo:

Sound Format: permette di scegliere le caratteristiche del segnale audio da acquisire.

Bottone "ON": fa partire l'acquisizione del segnale audio.

Bottone "STOP": ferma il programma.

Buffered Size: permette di scegliere la dimensione del buffer in byte, più è grande, più a lungo verrà registrato l'audio.

Lower/Higher pass freq: sono i controlli per il filtro Band-pass. Sono impostati in modo tale da far passare frequenze che appartengono solo alla mia tonalità di

voce.

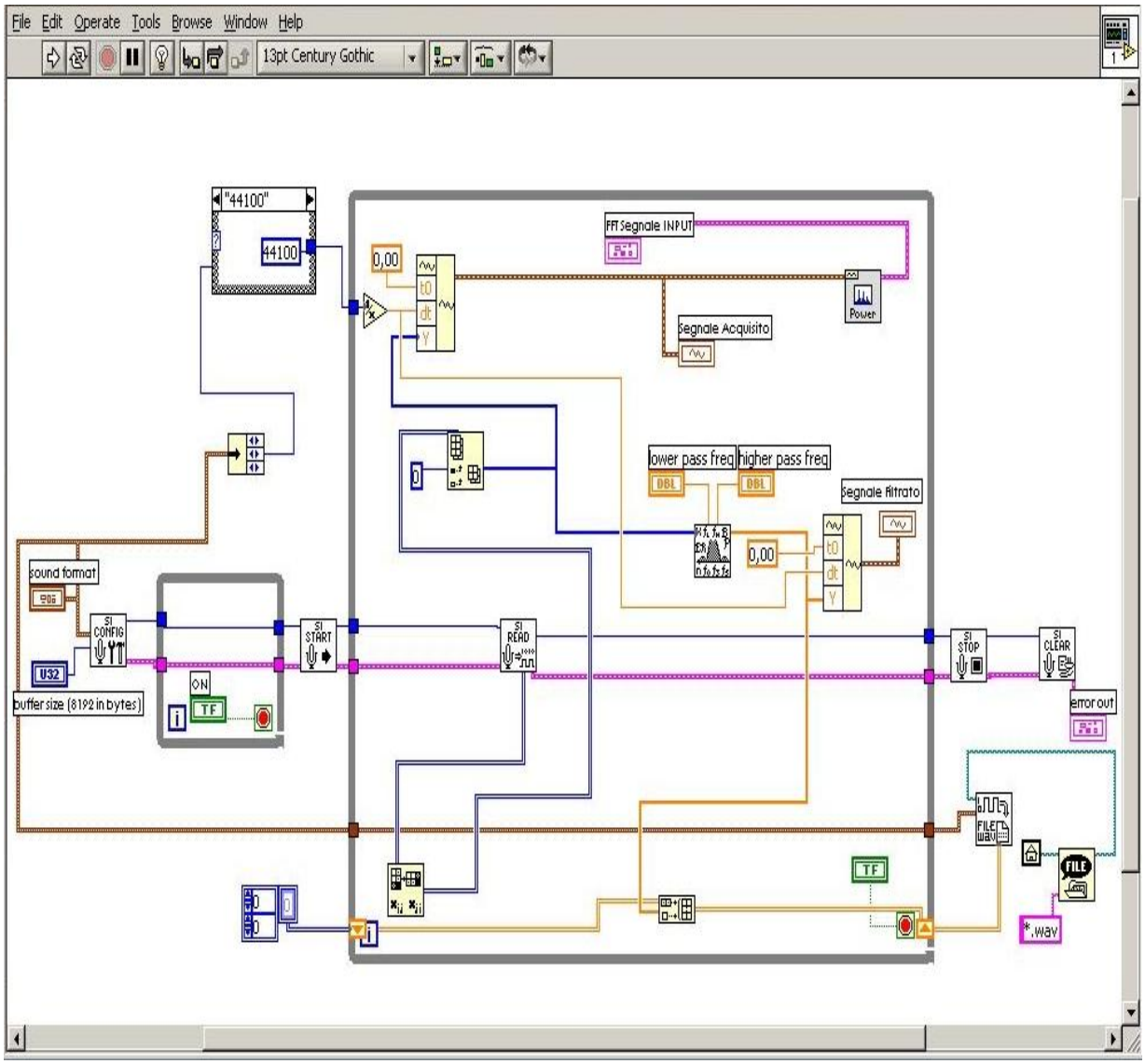
Error out: mi segnala, ove avvenga, il tipo di errore.

Segnale Acquisito: mostra visivamente il segnale acquisito.

FFT segnale input: mostra visivamente la trasformata di Fourier del segnale acquisito.

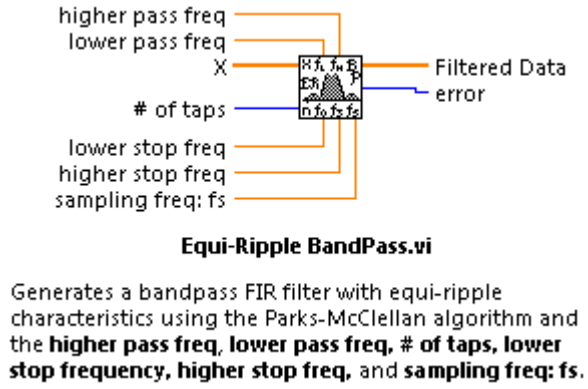
Segnale Filtrato: mostra visivamente il segnale filtrato tramite appunto il filtro band-pass.

Block Diagram



Nell'immagine possiamo notare innanzitutto i blocchi di acquisizione del segnale audio

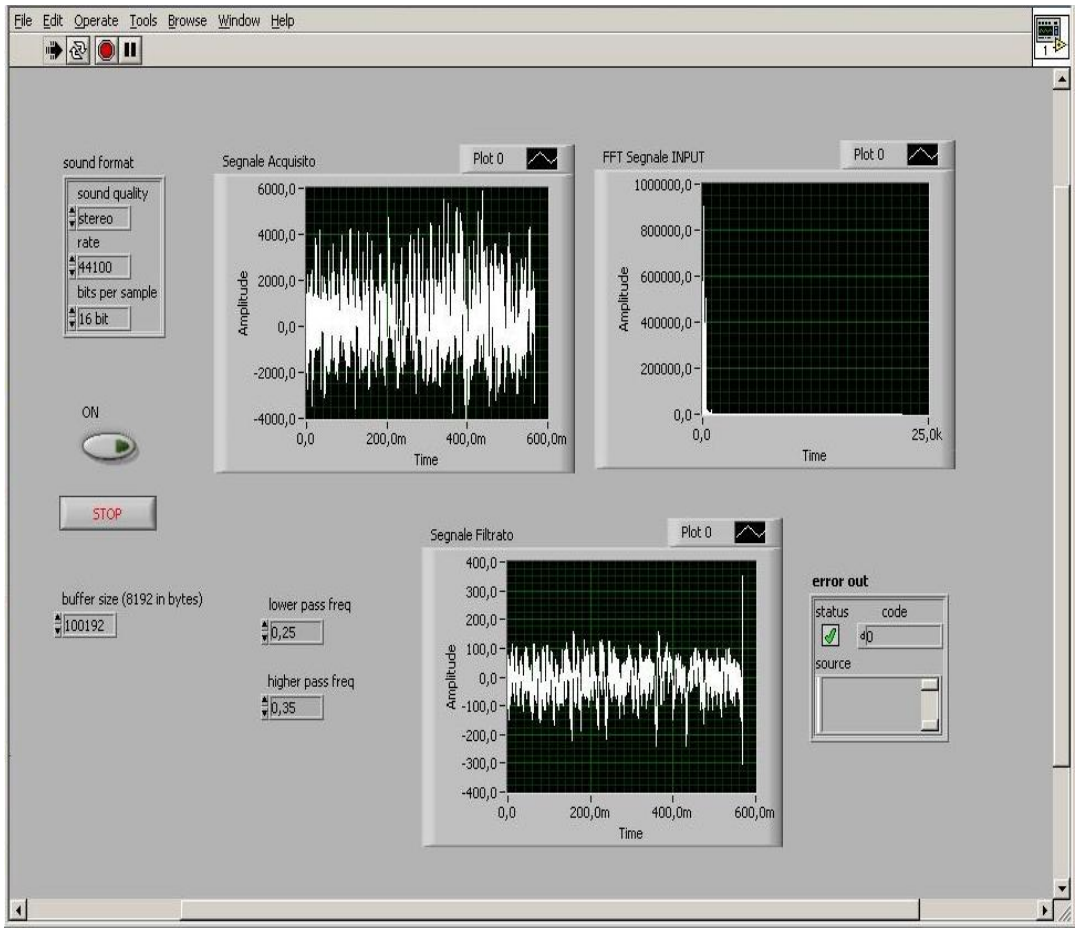
che appunto permettono di configurare, inizializzare, far leggere e resettare la scheda audio del dispositivo. Una volta che il segnale viene acquisito, passa attraverso il Filtro Passa Banda



che mi permette di catturare il segnale audio solo se le frequenze del segnale sono comprese fra i limiti inseriti (lower/higher pass freq). Da notare che la frequenza di campionamento l'ho fissata a 44100 Hz, quindi l'utente non è molto "libero" di scegliere quella che vuole; ciò è stato fatto per un motivo fondamentale, dovevo infatti fare in modo che la condizione di Nyquist venisse rispettata ($f_c >= 2B$) e dato che la voce umana si espande su un range di frequenze che va dai 500 Hz ai 2 Khz è indispensabile che la frequenza di campionamento sia 44100 Hz.

Una volta terminato il programma, il buffer si svuota e il contenuto viene salvato su file.

Vediamone ora un esempio



Prima di introdurre le esperienze 3 e 4 vorrei fare una precisazione. Per l'esperienza 3 volevo sottolineare il fatto che non sono riuscito a collegare il generatore di onde con la scheda acquisizione. Ho modificato quindi il programma creando un VI Server che invia tramite tcp i diversi tipi di onda e un VI Client che appunto riceve e "riconosce" il segnale in ingresso.

Invece per l'esperienza 4 sono riuscito a far funzionare la scheda acquisizione tuttavia ho avuto dei problemi con il collegamento del sensore. Mi scuso per l'inesattezza ma ho fatto il massimo che ho potuto.

ESPERIENZA 3

Si scriva un VI per realizzare lo studio in frequenza dei seguenti segnali:

_onda sinusoidale a 1Khz, 10 Khz, 100Khz

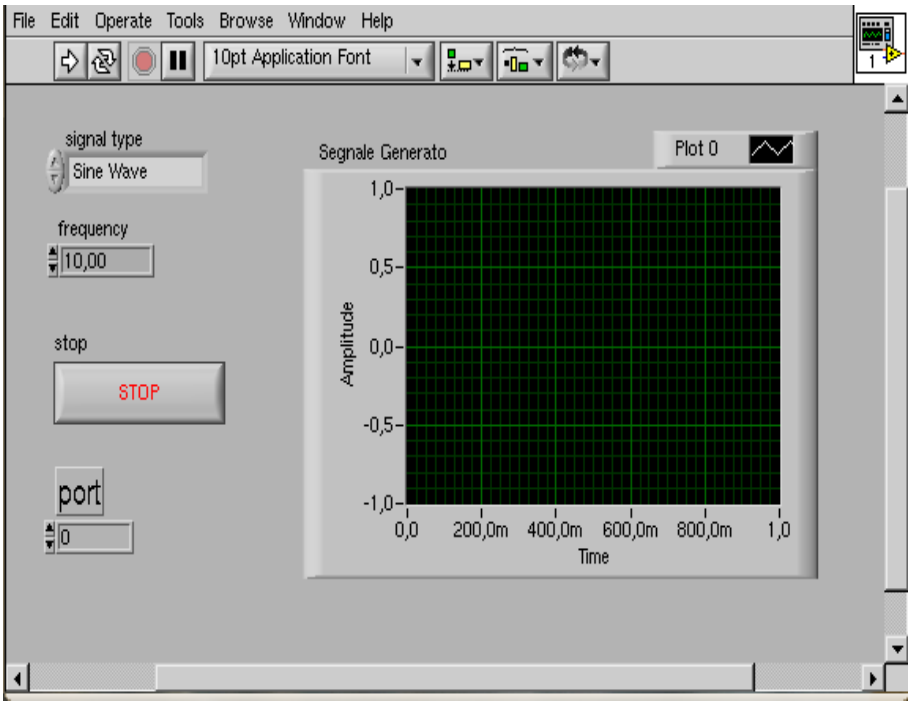
_onda triangolari a 1 Khz, 10 Khz, 100Khz

_onda quadra a 1Khz, 10 Khz, 100Khz

Si sviluppi un algoritmo per il riconoscimento automatico del tipo di segnale.

In questa esperienza di laboratorio ho scelto di creare un VI Server e un VI Client come nell'esperienza 2. Andiamo a vedere il Control Panel del Server

Control Panel – *Server*



Nel Control Panel troviamo (ometto le funzioni già descritte nelle precdenti esperienze):

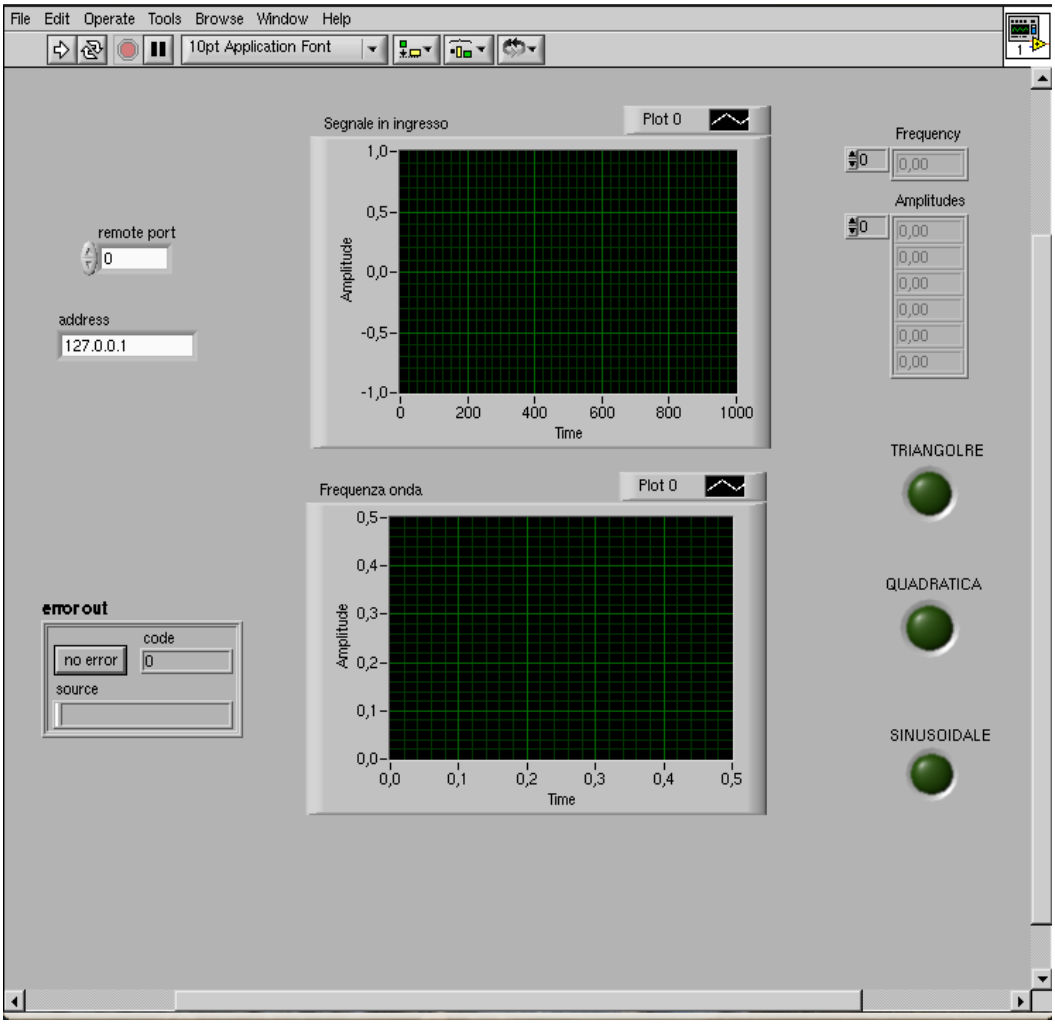
Signal Type: fa scegliere quale tipo di onda generare.

Frequency: permette di definire la frequenza, chiaramente è legata a *Signal Type.*

Segnale Generato: mostra visivamente il segnale generato.

Andiamo ora a vedere il Control Panel del Client.

Control Panel – *Client*



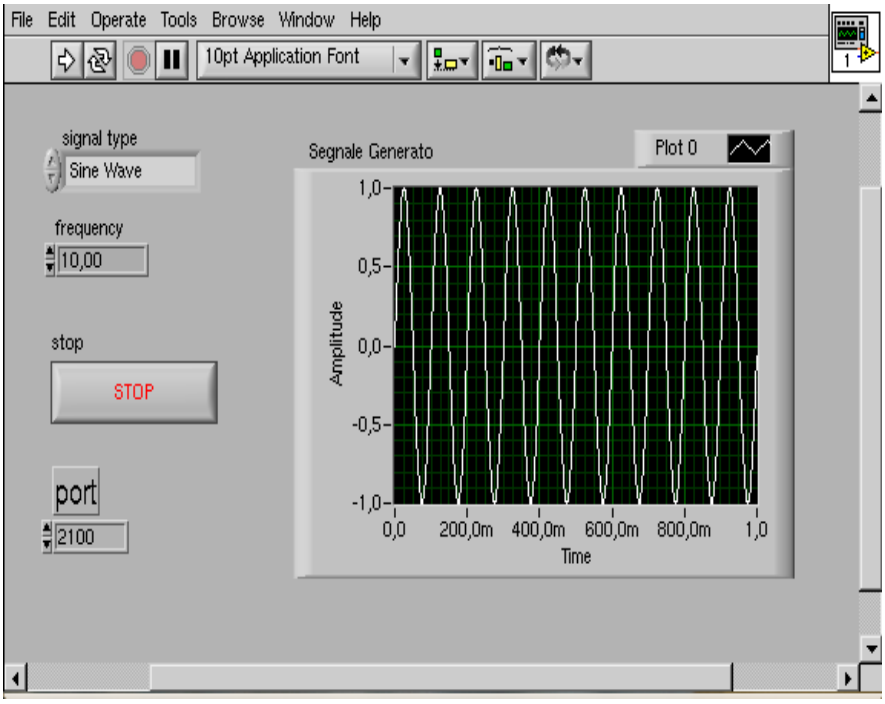
Nel Control Panel del Client Troviamo:

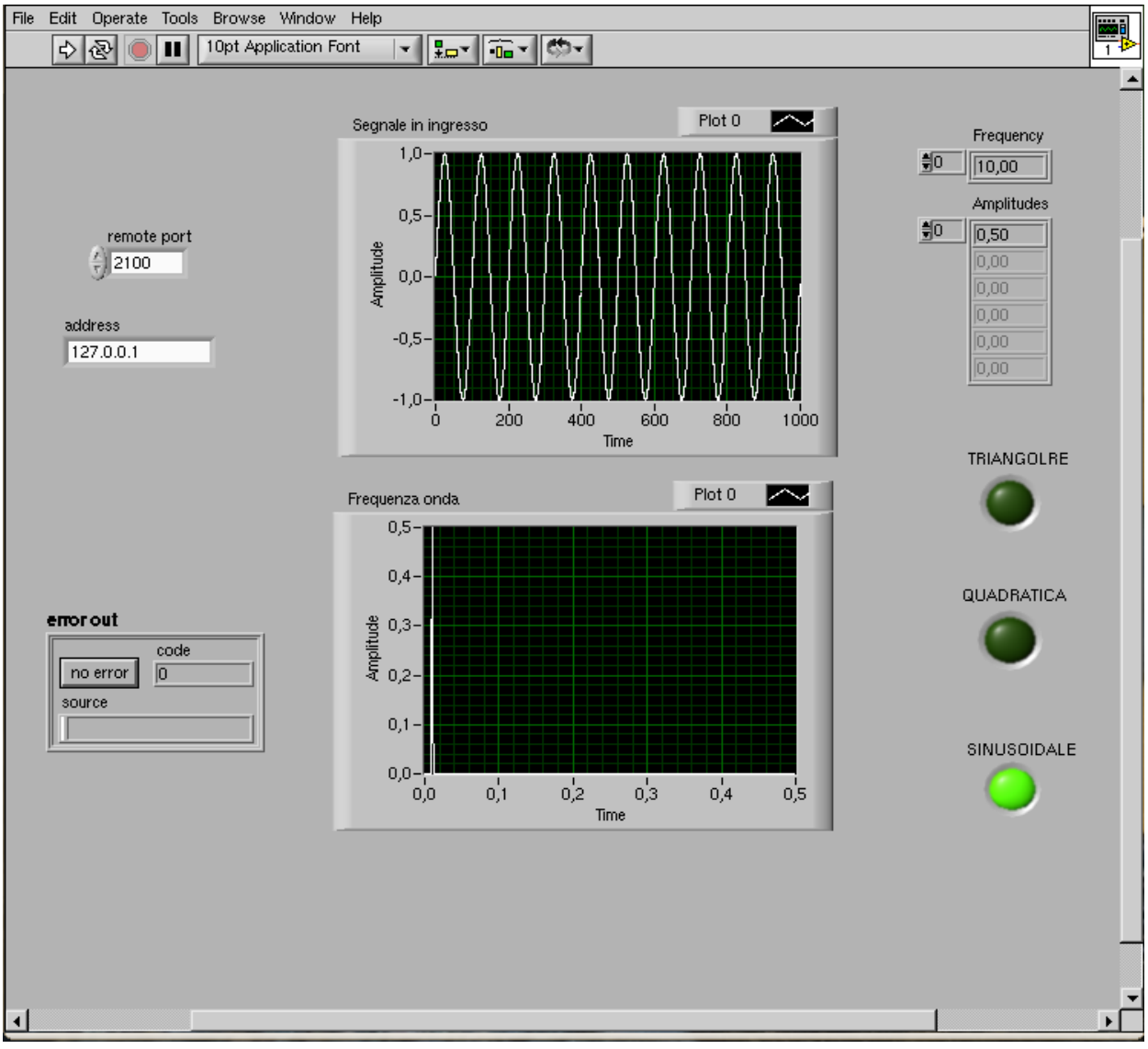
Frequency: indica la frequenza del segnale in ingresso, chiaramente coincide con quella inviata dal Server.

Amplitude: indica il valore più grande dell'ampiezza dei picchi, questo dato mi permette di riconoscere il segnale ricevuto in quanto è diverso per ogni segnale. Confrontando sotto determinate regole questo valore posso accendere i tre LED nel modo corretto.

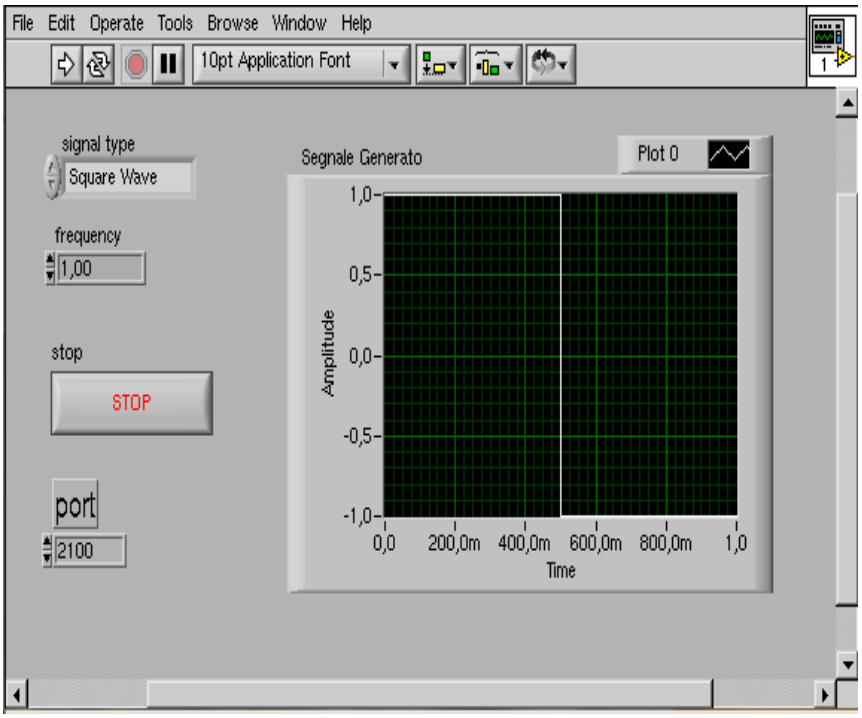
Di seguito le immagini del funzionamento.

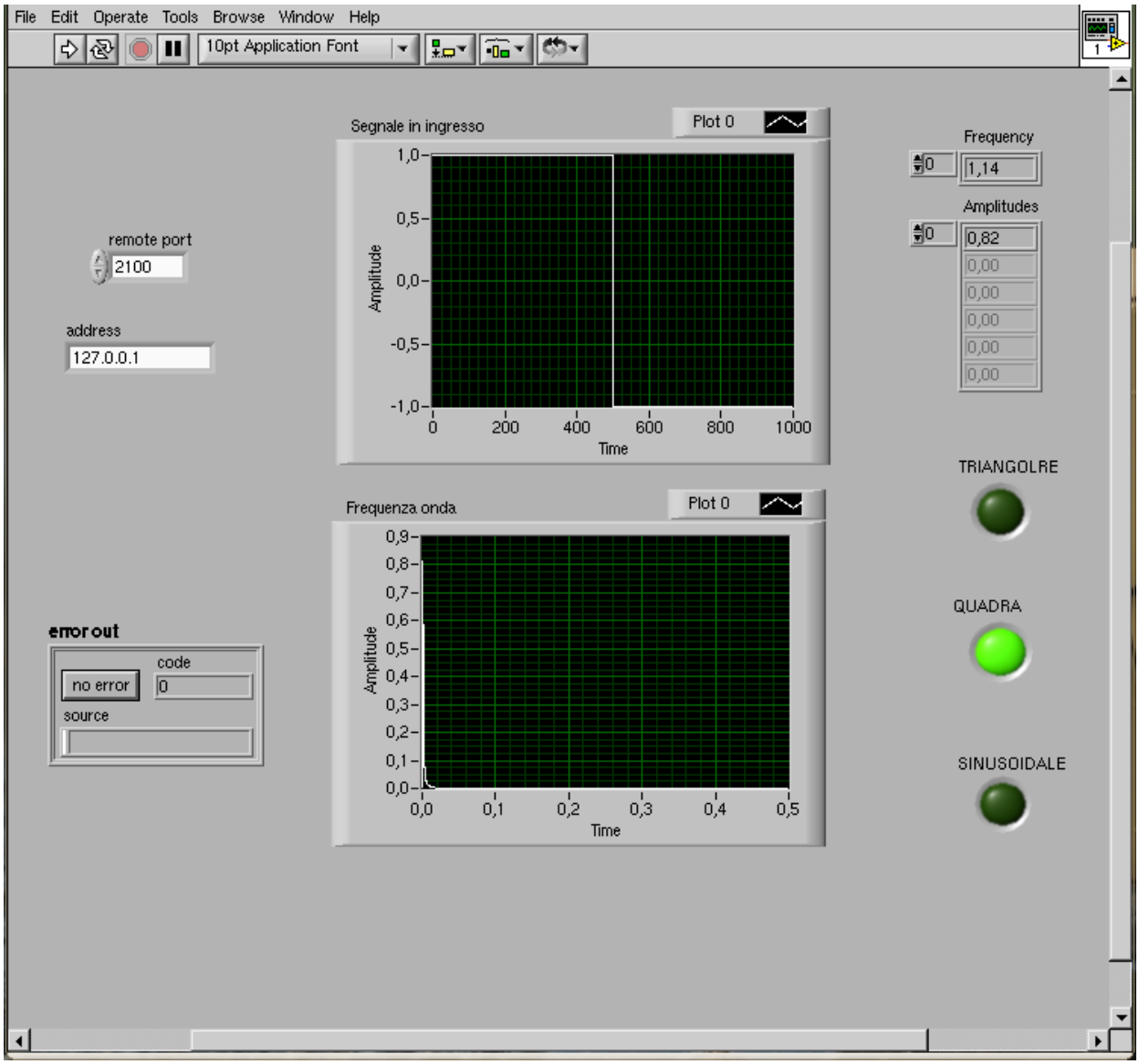
ONDA SINUSOIDALE 10 KHz



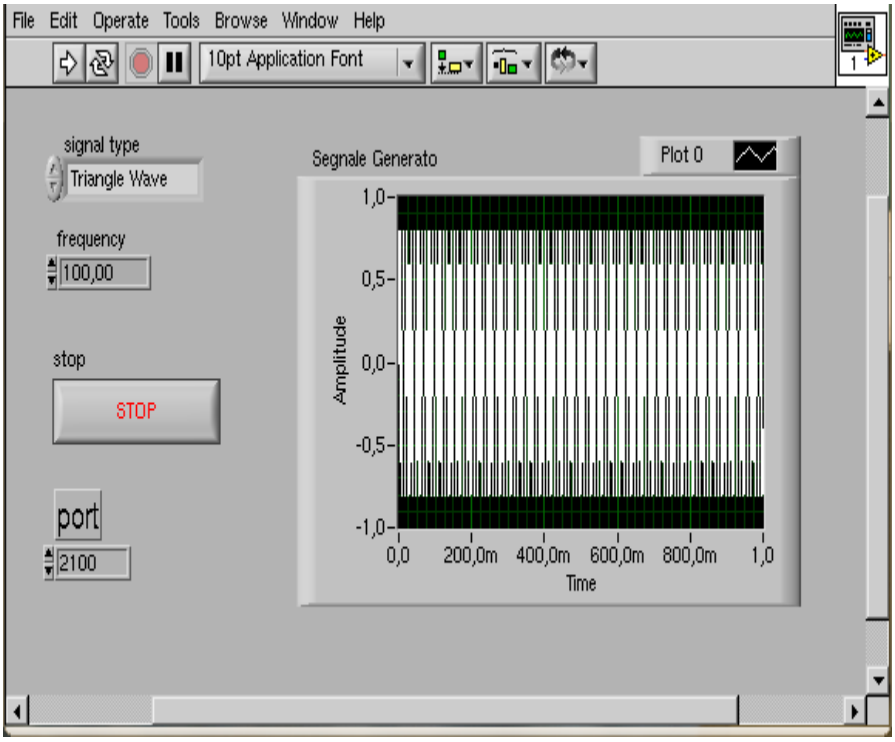


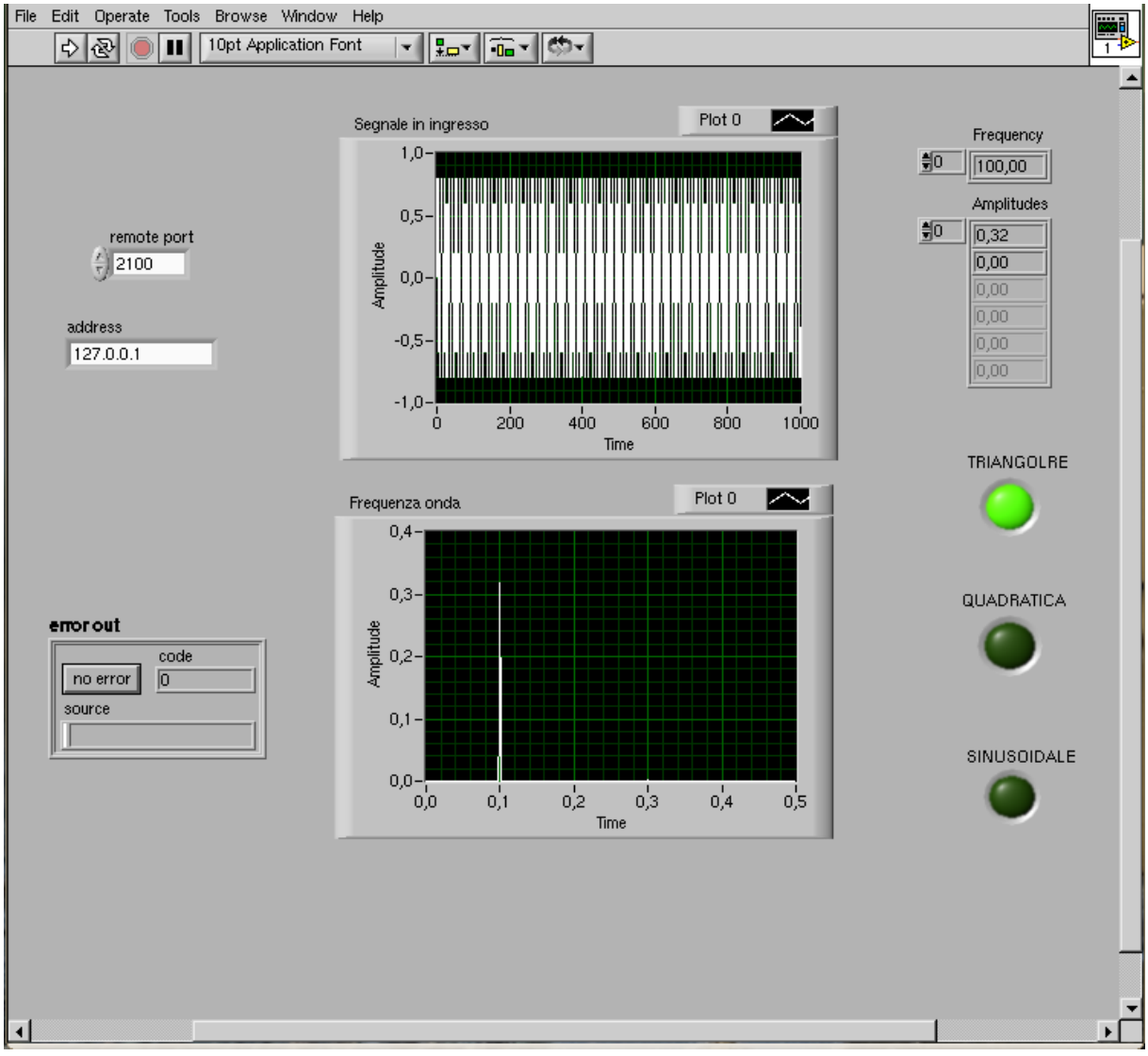
ONDA QUADRA 1 KHz





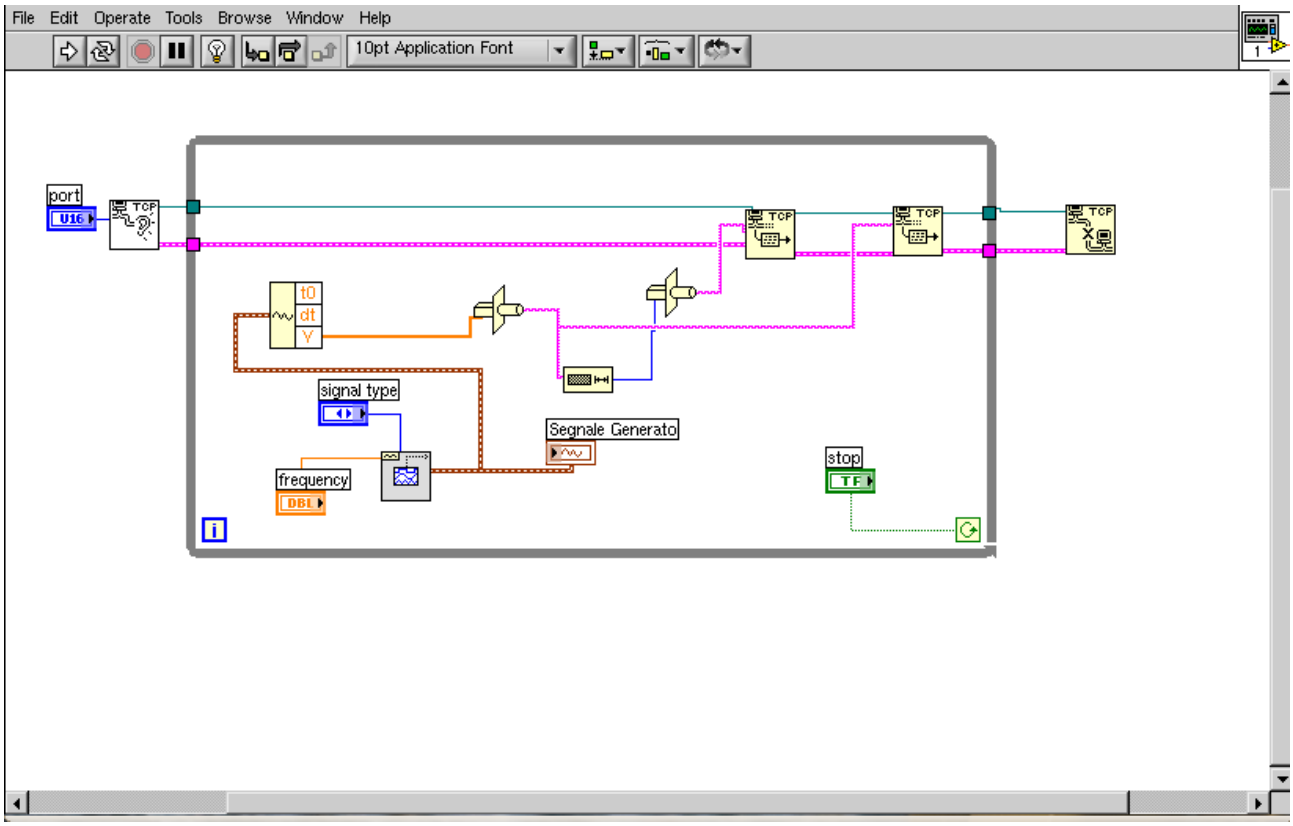
ONDA TRIANGOLARE 100 Khz



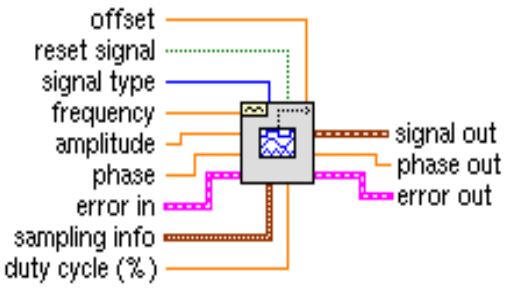


Andiamo a vedere ora il Block Diagram del Server

Block Diagram - *Sever*



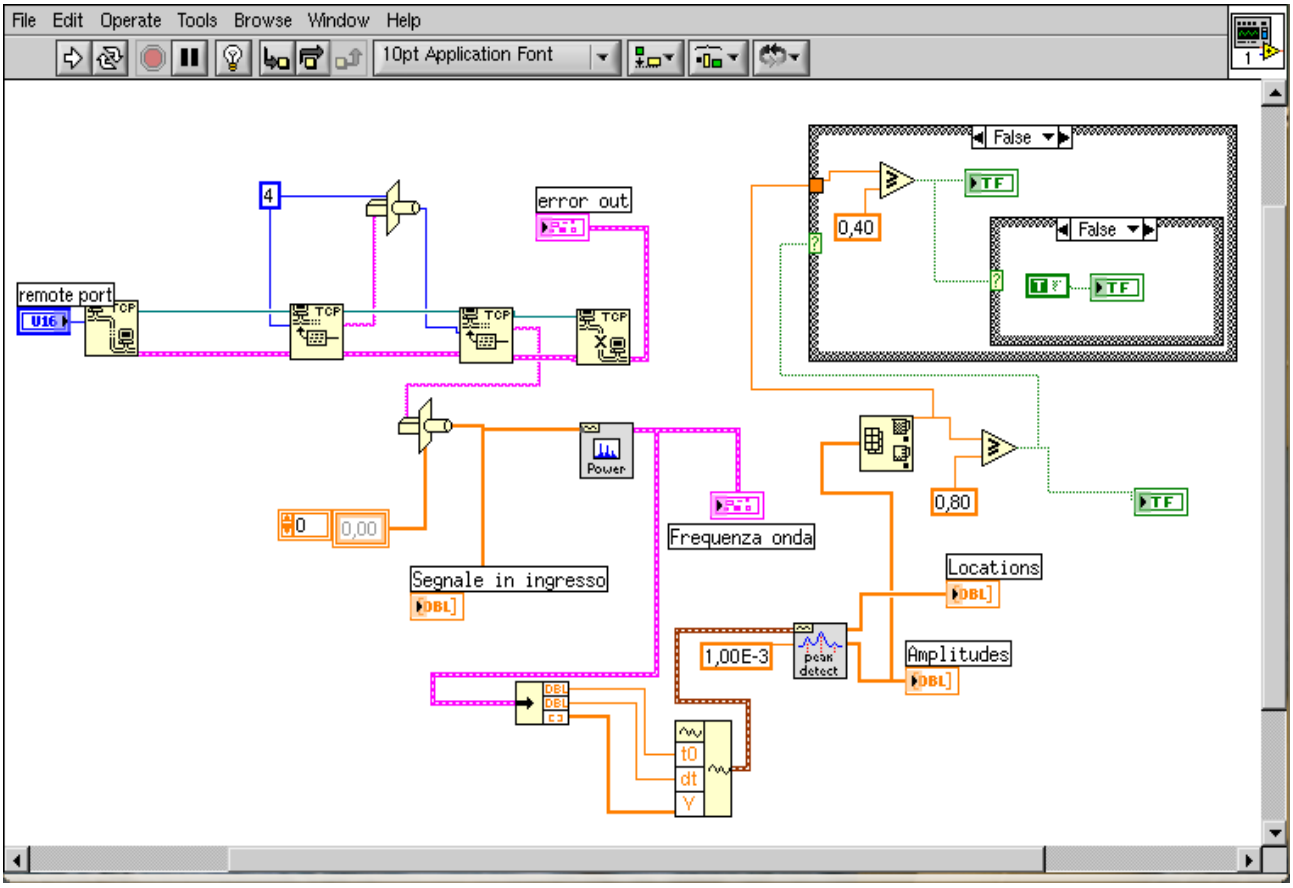
Il Block Diagram del Server è molto semplice, consiste infatti solamente dei blocchi di connessione TCP e del blocco *Generatore di Funzion.* Tramite questo oggetto l'utente può scegliere sia la frequenza che il tipo di segnale da inviare.



Basic Function Generator.vi

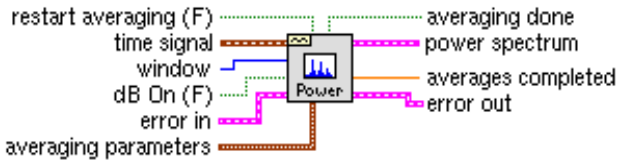
Creates an output waveform that is one of the following common signal types: Sinewave, Triangle, Sawtooth, or Squarewave.

Block Diagram – *Client*



Il Block Diagram del Client è l'algoritmo vero e proprio.

E' composto dai blocchi per la connessione TCP che ricevono il segnale, il quale una volta ricevuto va in input al blocco *FFT power spectrum*

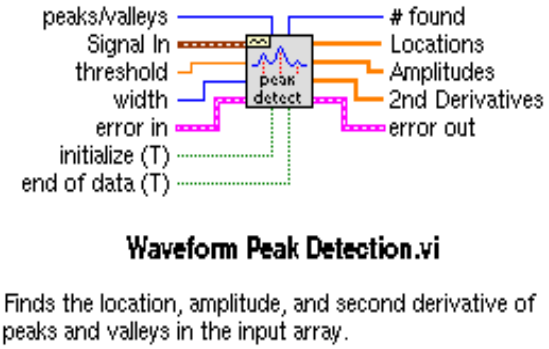


FFT Power Spectrum.vi

Computes the averaged auto power spectrum of the input signal. The VI computes the FFT of the input time waveform, forms its power spectrum, and then averages this power spectrum with those computed from previous time waveforms.

che ha la funzione di applicare Fourie. Successivamente, i valori del segnale

vengono messi in input al blocco *peak detection*



che mi rileva le ampiezze e la frequenza. Dei valori delle ampiezze prendo solamente quello maggiore, poi con questo, faccio dei confronti per far in modo di accendere i LED nel modo corretto. Precisamente se il valore è maggiore o uguale a 0,80 allora l'onda ricevuta è un onda quadra; altrimenti se il valore è minore di 0,80 ma è maggiore di 0,40 allora è sinusoidale, se le seguenti relazioni sono entrambe false, significa che l'onda ricevuta è triangolare. I valori 0,80 e 0,40 non sono altro che indicatori del range di azione dei picchi delle frequenze del segnale sinusoidale.

ESPERIENZA 4

Si scriva un VI per realizzare una stazione termometrica basata su sensori LM35 e la scheda di acquisizione NI 6024E, con le seguenti funzionalità:

_visualizzare le temperature correnti

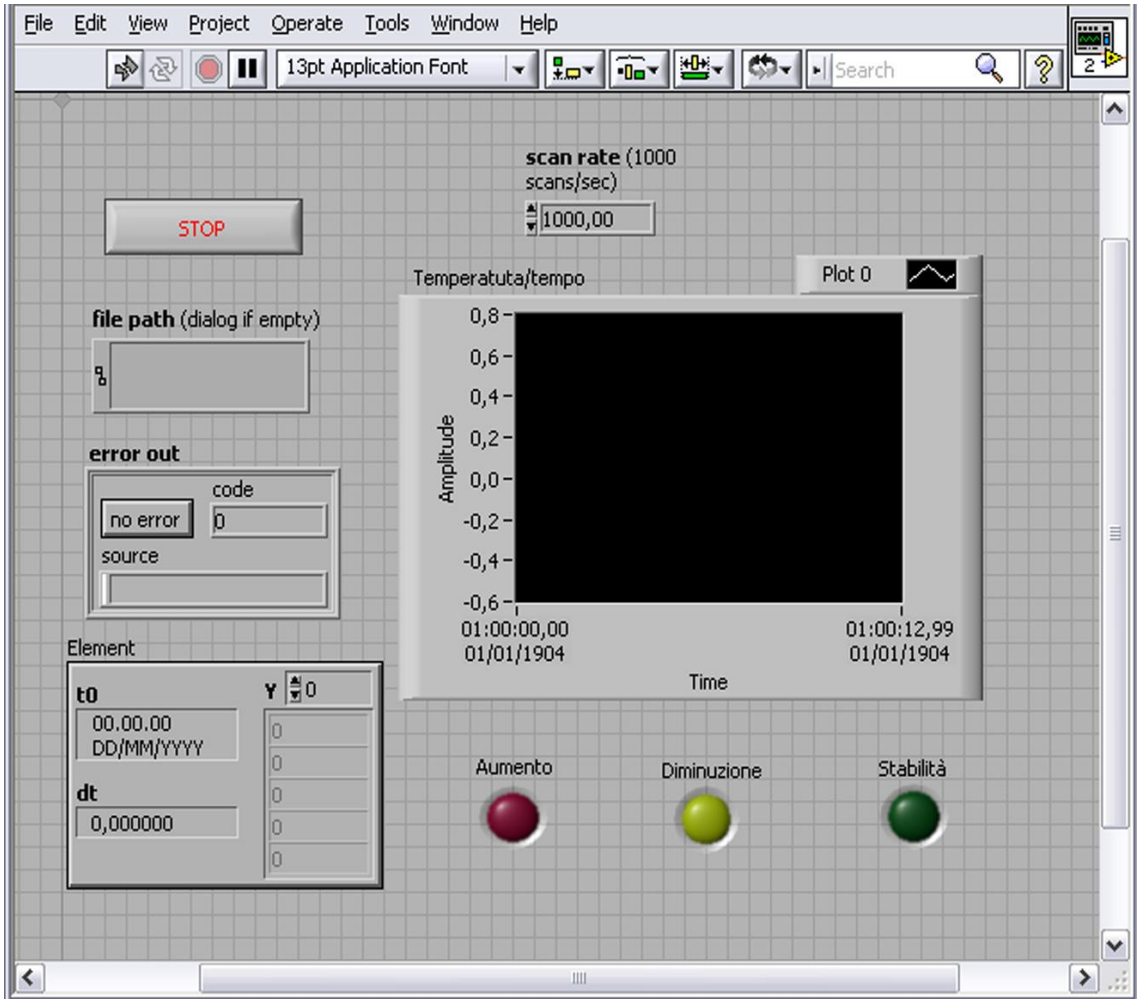
_visualizzare, tramite l'accensione di LED l'andamento delle temperature (aumento, diminuzione, stabilità)

_visualizzare l'andamento delle temperature in funzione del tempo

_salvare i dati su file

_visualizzare i dati del file

Control Panel



Nel Control Panel troviamo (ometto le funzioni già descritte nelle precdenti esperienze):

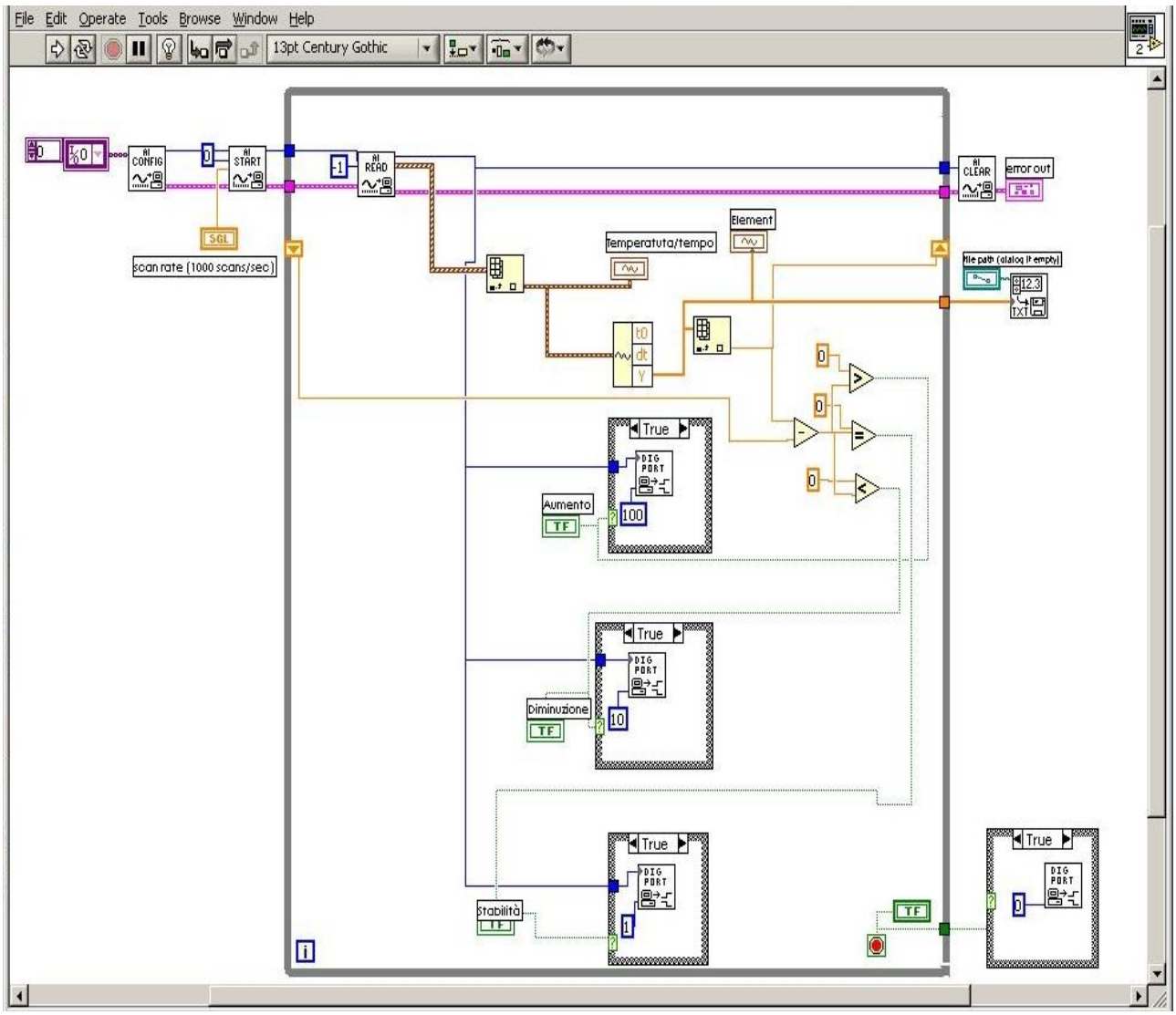
Scan Rate: Indica il numero di scansioni al secondo.

Element: mostra le temperature misurate.

Temperatura/Tempo: mostra visivamente il segnale della temperatura acquisita rispetto al tempo.

In basso troviamo tre LED che si accendono quando ho rispettivamente un *aumento*, *diminuzione*, *stabilità* di temperatura.

Block Diagram

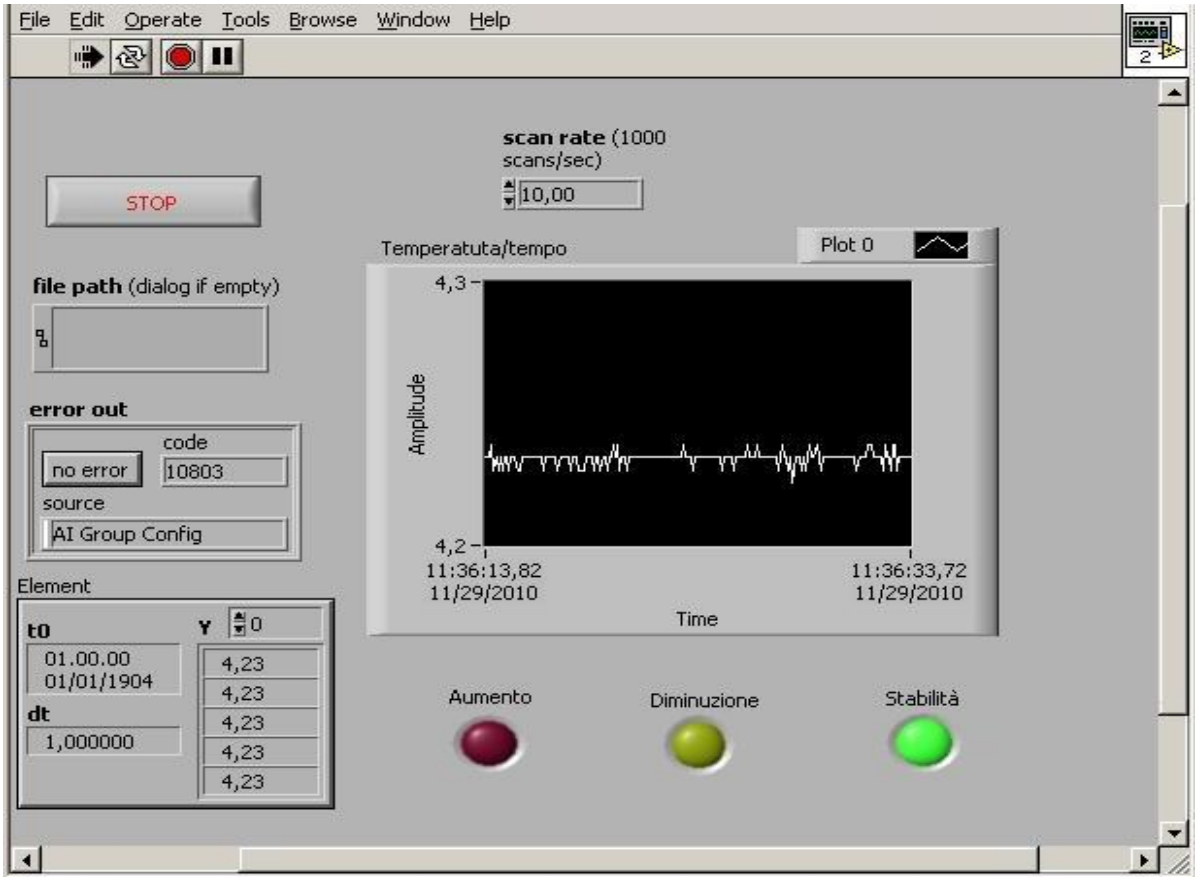


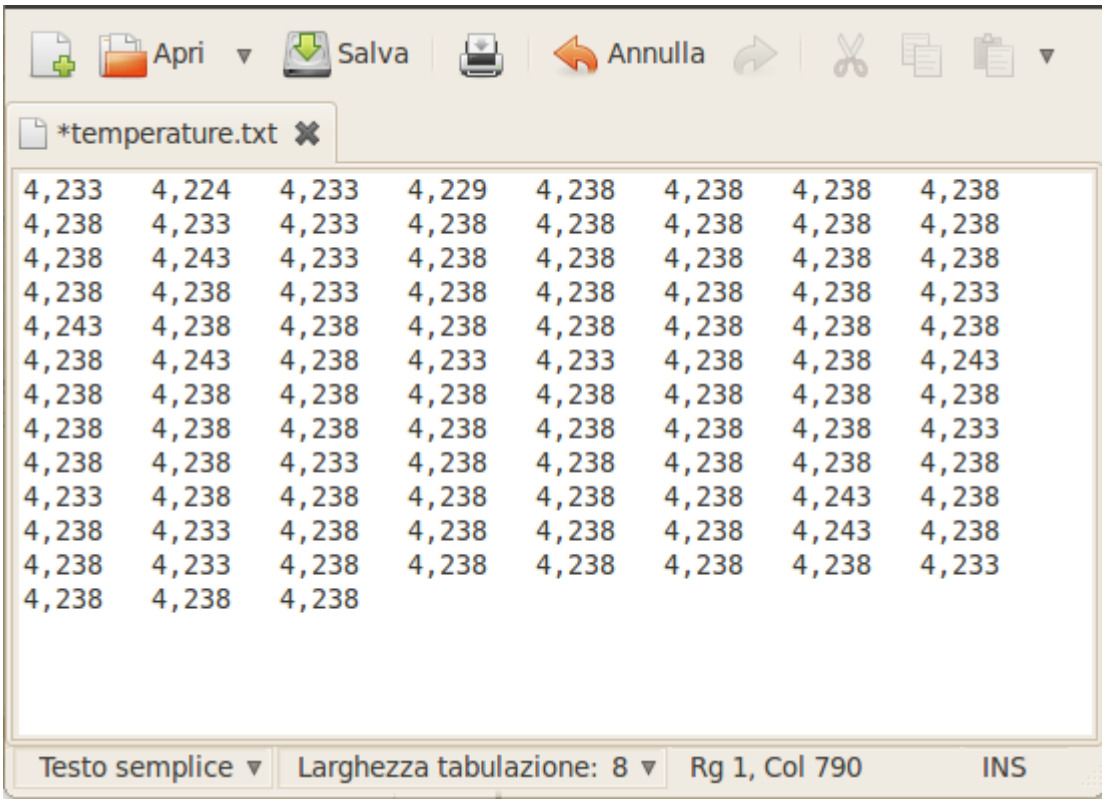
Il segnale viene acquisito dalla scheda acquisizione grazie ai blocchi

Quindi viene confrontata la temperatura corrente con quella precedete utilizzando un registro a scorrimento e a seconda che la temperatura sia maggiore minore o uguale a quella precedente, vengono accessi i LED. Anche se può sembrare folle controllare continuamente i valori della temperatura, in realtà per applicazioni convenzionali funziona discretamente. Questo perchè la temperatura in un ambiente chiuso non varia molto nel tempo, quindi facendo campionamenti sporadici, il programma acquisisce un senso.

Una volta terminato, il programma svuota il buffer e memorizza i valori in un file e i LED vengono spenti.

Ecco un esempio del programma e consecutivamente il file con i valori salvati.




Apri
Salva
Annulla
*temperature.txt
4,233 4,224 4,233 4,229 4,238 4,238 4,238 4,238
4,238 4,233 4,233 4,238 4,238 4,238 4,238 4,238
4,238 4,243 4,233 4,238 4,238 4,238 4,238 4,238
4,238 4,238 4,233 4,238 4,238 4,238 4,238 4,233
4,243 4,238 4,238 4,238 4,238 4,238 4,238 4,238
4,238 4,243 4,238 4,233 4,233 4,238 4,238 4,243
4,238 4,238 4,238 4,238 4,238 4,238 4,238 4,238
4,238 4,238 4,238 4,238 4,238 4,238 4,238 4,233
4,238 4,238 4,233 4,238 4,238 4,238 4,238 4,238
4,233 4,238 4,238 4,238 4,238 4,238 4,243 4,238
4,238 4,233 4,238 4,238 4,238 4,238 4,243 4,238
4,238 4,233 4,238 4,238 4,238 4,238 4,238 4,233
4,238 4,238 4,238
Testo semplice
Larghezza tabulazione: 8
Rg 1, Col 790
INS