UNIVERSITÀ DEGLI STUDI DI PISA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA ELETTRONICA

Tesi di Laurea Specialistica

Progettazione e collaudo di un sistema di elaborazione ed acquisizione dati da un interferometro ottico

Anno Accademico 2007/2008

Primo relatore: ***Prof. Roberto Saletti***

Secondo relatore: ***Prof. Luigi Lazzeri***

Candidato: ***Paolo Rocchiccioli***

Indice

Introduzione

Questo lavoro si pone l'obiettivo di realizzare un sistema elettronico per caratterizzare la cinetica del meccanismo di reticolazione di un idrogelo: una matrice polimerica reticolata che trattiene una certa quantità d'acqua al suo interno che mostra proprietà intermedie fra il solido ed il liquido. Il lavoro è stato svolto in collaborazione con il Dipartimento di Ingegneria Chimica, Chimica Industriale e Scienza dei Materiali della Facoltà di Ingegneria dell' Università di Pisa. L'interesse per gli idrogeli in generale, e per quelli di alginato (polisaccaride) in particolare, è notevole, soprattutto in campo biomedico, ed è legato alla possibilità di realizzare sistemi le cui applicazioni vanno dal rilascio controllato di farmaci all' ingegneria dei tessuti data anche l'elevata biocompatibilità.

Il meccanismo di reticolazione degli idrogeli di alginato richiede la presenza di ioni di calcio all'interno della soluzione, i quali si legano stabilmente lungo le molecole del polisaccaride. Si ha così la creazione di legami fra le catene polimeriche e la formazione di una rete tridimensionale le cui proprietà dipendono, oltre che dalla natura del polimero, dal numero dei legami, cioè dal cosiddetto grado di reticolazione.

Consapevoli del fatto che una caratteristica macroscopica come l'indice di rifrazione dipende dalla natura microscopica di un materiale è possibile, quindi, studiare il grado di reticolazione correlando l'indice di rifrazione del materiale alla concentrazione di ioni calcio attraverso l'impiego di un interferometro di Mach-Zehnder.

In questo lavoro di tesi viene progettato, realizzato e collaudato l'intero sistema di misura, costituito da:

- Trasduttore (sensore + circuito di condizionamento del segnale).
- Scheda di acquisizione dati (DAQ).
- Programma di elaborazione, visualizzazione ed archiviazione dei dati, realizzato con LabVIEW.

Nel Capitolo 1 si spiegano le ragioni che ci portano ad utilizzare l'interferometro di Mach-Zehnder ed i metodi possibili per la valutazione delle variazioni dell'indice di rifrazione nel tempo. Date poi le specifiche di progetto viene scelta l'architettura del sistema di misura e ne viene fatto un dimensionamento di massima. Nel Capitolo 2, attenendoci alle specifiche del sistema, viene scelto il sensore e progettato il circuito di condizionamento del segnale. Dopo aver effettuato le simulazioni con SPICE, quest' ultimo è stato realizzato su circuito stampato, collaudato ed inscatolato. Dato che la scheda di acquisizione dati si interfaccia con il PC attraverso il bus USB nel Capitolo 3 si forniscono i requisiti, sul protocollo e le caratteristiche dei segnali, necessari alla comprensione e alla realizzazione del sistema di misura. Nel Capitolo 4 viene illustrata più nel dettaglio la struttura del DAQ e i criteri di scelta dei componenti (controller USB, MCU, ADC), non solo in funzione delle prestazioni ma anche del tempo di progettazione, del costo finale e di realizzazione del PCB. Si spiega come è possibile raggiungere la massima velocità di comunicazione tra il DAQ e il PC, in modo da rispettare le specifiche di progetto e le funzioni contenute nella libreria del driver del controller USB, che ci permettono di eseguire le operazioni di lettura dei dati(e non solo) provenienti dal DAQ. Dopo aver illustrato il criterio di stesura del firmware e la possibilità di aggiornamento del DAQ tramite una programmazione *In-System*, nella parte finale del capitolo sono riportate le foto del DAQ assemblato su PCB e i risultati del collaudo (*debugging firmware*).
Nel Capitolo 5 ,si spiegano le varie funzioni svolte dall' interfaccia software (VI), realizzata con LabVIEW, tra cui:

- Inizializzazione della comunicazione con il DAQ.
- Acquisizione dei dati dal DAQ.
- Elaborazione ottimizzata dei dati finalizzata alla valutazione della variazione dell'indice di rifrazione attraverso lo studio dei massimi.
- Visualizzazione della variazione dell'indice di rifrazione (Dn) in funzione del tempo trascorso.
- Archiviazione su un file di testo, su colonne separate, dei punti che compongo il grafico Dn-t.
- Visualizzazione del segnale proveniente dal trasduttore.

- Visualizzazione del segnale per la calibrazione (allineamento) dell' interferometro.

Si spiega inoltre come installare il driver del DAQ, il *LabVIEW run-time engine 8.0* necessario e ed il VI attraverso l'*installer (SetUp_Strumento.exe)* appositamente creato in fase di sviluppo, in modo da facilitare la distribuzione dell'applicazione. Nella parte finale del capitolo sono riportati alcuni risultati ottenuti durante il collaudo (*debugging*) del software.

Capitolo 1

Interferometro ottico per idrogeli

1. Introduzione

Il sistema è costituito da un interferometro ottico tramite il quale studiare le variazioni nel tempo delle caratteristiche di un materiale trasparente. Il materiale è un polisaccaride (alginato, fig. 1.1) che si presenta sottoforma di idrogelo, cioè di una matrice polimerica reticolata che trattiene una certa quantità di acqua al suo interno e che mostra proprietà intermedie fra il solido ed il liquido (fig. 1.2).

L'interesse per gli idrogeli in generale, e per quelli di alginato in particolare, è notevole, soprattutto in campo biomedico, ed è legato alla possibilità di realizzare sistemi le cui applicazioni vanno dal rilascio controllato di farmaci all' ingegneria dei tessuti.

La caratteristica principale alla base dell'interesse verso l'alginato, oltre alle proprietà di biocompatibilità per le quali risulta un materiale molto ben tollerato dall'organismo umano, è il meccanismo attraverso il quale si ottiene la struttura dell' idrogelo a partire dal polisaccaride in soluzione acquosa. Il meccanismo di reticolazione, infatti, richiede semplicemente la presenza di ioni di calcio che, una volta aggiunti alla soluzione, si legano stabilmente a coppie di gruppi carbossilici posti in determinate posizioni lungo le molecole di polisaccaride (fig. 1.3). Si ha così la creazione di legami fra le catene polimeriche e la formazione di una rete tridimensionale le cui proprietà dipendono, oltre che dalla natura del polimero, dal numero dei legami, cioè dal cosiddetto grado di reticolazione.

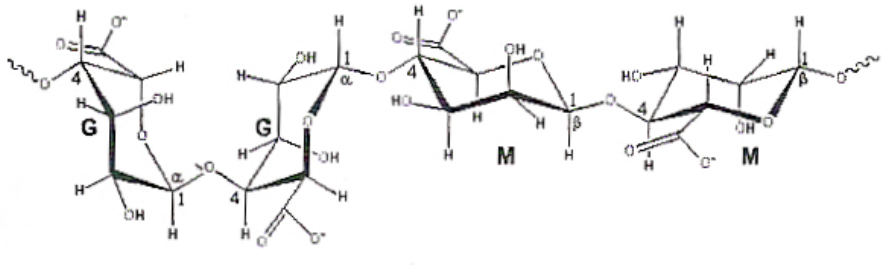


Figura 1.1: L'alginato è un polisaccaride costituito dalla ripetizione , più o meno casuale, di unità di acido D-guluronico (G) ed L-mannuronico (M). Le coppie di unità G lungo le catene del polimero rappresentano siti capaci di interagire con gli ioni Calcio.

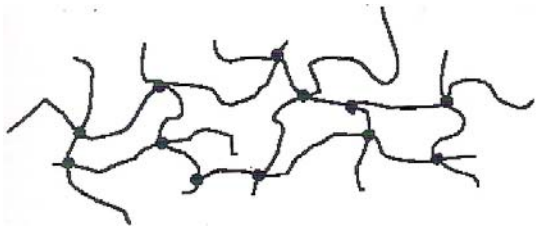


Figura 1.2: Esempio schematico della struttura di un idrogelo. Le catene polimeriche (linee nere) sono legate fra loro in punti di reticolazione che possono essere legami di vario tipo (covalenti, ionici, a idrogeno). Se le catene polimeriche hanno caratteristiche idrofiliche, gli spazi vuoti fra esse possono riempirsi di molecole d'acqua che rimangono intrappolate nella rete polimerica.

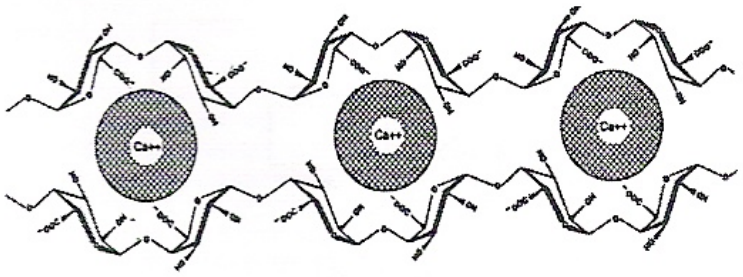


Figura 1.3: Esempio schematico della formazione di legami fra le catene attraverso ioni calcio. Gli ioni calcio riescono a coordinare coppie G-G fra catene diverse attraverso interazioni di tipo ionico. Il numero e la regolarità delle coppie G-G nonché la quantità di ioni calcio determina il grado di reticolazione dell'idrogelo.

1.1 La valutazione del grado di reticolazione di un idrogelo

Se da un lato la conoscenza del grado di reticolazione è indispensabile per progettare un idrogelo avente determinate proprietà , dall'altro la sua determinazione non è affatto semplice e si deve ricorrere a valutazioni indirette tramite la misura di altre proprietà ad esso correlate ad al supporto di modelli.

Tipicamente, allo scopo di valutare il grado di reticolazione degli idrogeli, viene utilizzata la misura di proprietà meccaniche, in condizioni sia statiche che dinamiche. Ma a fronte della loro relativa semplicità, i test meccanici non permettono di effettuare una valutazione cinetica delle caratteristiche del materiale, cioè non è possibile monitorare l'andamento nel tempo delle proprietà di un materiale che si modifica nel tempo. Nei pochi casi in cui ciò è possibile (ad esempio, al variare della temperatura), i risultati hanno necessariamente il carattere di dati medi senza alcuna indicazione sulla reale distribuzione all'interno del materiale.

1.2 Perché un interferometro ottico?

La maggior parte dei materiali a struttura idrogelica sono trasparenti alla luce. In particolare, gli idrogeli di alginato, dato l'elevato contenuto di acqua che li caratterizza, risultano trasparenti su quasi tutto lo spettro visibile, con effetti di assorbimento solo nella zona prossima alle lunghezze d'onda del violetto.

E' noto anche che una caratteristica macroscopica come l'indice di rifrazione dipende dalla natura microscopica di un materiale. Nel caso degli idrogeli di alginato, è possibile quindi studiare il grado di reticolazione correlando l'indice di rifrazione del materiale alla concentrazione di ioni calcio [1] [2]. A tale scopo, l'impiego di un interferometro ottico offre alcuni vantaggi:

- Le variazioni di indice di rifrazione sono in generale piccole quando la composizione del materiale varia entro un intervallo abbastanza limitato di valori. I metodi interferometrici, basati sulla rivelazione delle figure di

interferenza, hanno una sensibilità sufficientemente elevata per permettere la misura di tali piccole variazioni.

- La variazione di indice di rifrazione può essere facilmente monitorata nel corso del tempo.
- Utilizzando la radiazione coerente emessa da un laser He-Ne, per le piccole dimensioni trasversali (rispetto alla direzione di propagazione) del raggio laser permettono di effettuare misure in posizioni diverse all'interno del materiale e consentono quindi di ottenere un andamento anche spaziale, oltre che temporale, dell'indice di rifrazione.

1.3 La configurazione proposta

La configurazione che viene proposta è quella dell'**interferometro di Mach-Zehnder**, nel quale il raggio emesso da un laser He-Ne (potenza nominale 20mW) alla lunghezza d'onda di 632,8nm viene suddiviso in due parti attraverso un *beam splitter* . Uno dei due raggi risultanti costituisce il raggio di riferimento, mentre l'altro rappresenta il raggio di misura, che attraversa il materiale in studio contenuto in una opportuna cella portacampioni. Dopo aver attraversato il campione, il raggio di misura viene ricombinato, attraverso un secondo *beam splitter*, con il raggio di riferimento e la figura di interferenza che si crea viene rivelata da un fotodiodo.

1.4 Specifiche del sistema di misura

Il sistema di misura deve permettere l'acquisizione del segnale proveniente da un fotorivelatore e la sua elaborazione tramite PC (*client software*). A questo riguardo è opportuno prevedere che il sistema (*hardware* e *software*) possa essere espandibile per attuare anche funzioni di controllo in vista di futuri miglioramenti dell' interferometro quali, ad esempio, il controllo di un microposizionatore piezoelettrico, il controllo della temperatura ed altre grandezze fisiche di interesse. La presenza di un microposizionatore piezoelettrico è una possibile soluzione alla necessità di misurare le variazioni dell'indice di rifrazione in funzione della direzione di diffusione degli ioni di calcio. I costi di realizzazione del sistema devono essere ridotti. L' ordine di grandezza della frequenza del segnale sinusoidale proveniente dal fotorivelatore non è nota a priori, si può solo affermare con certezza che questa non supererà i 10-20KHz.

Il sistema deve poter prevedere la possibilità di facilitare l'allineamento dell'interferometro, ricercare cioè la massima potenza incidente sul fotorivelatore grazie ad una corretta orientazione degli specchi, dei beam-splitter e del puntatore laser stesso. Inoltre, tramite il Client Software, la cui interfaccia grafica deve essere semplice ed intuitiva, deve essere possibile visualizzare ed archiviare l'andamento della variazione dell'indice di rifrazione nel tempo.

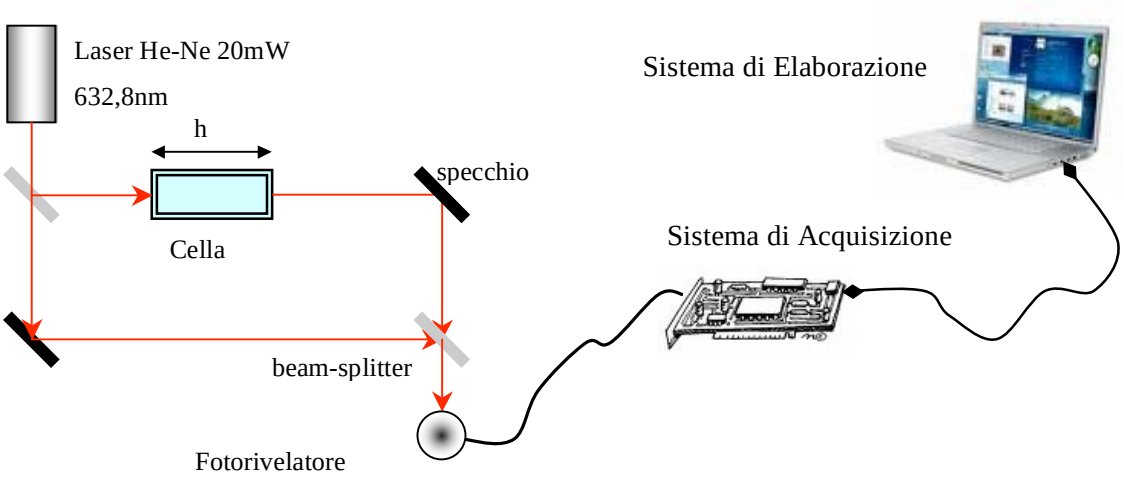


Figura 1.4: Interferometro di Mach-Zehnder nel sistema di misura.

1.5 Come calcolare la variazione dell'indice di rifrazione

Si supponga che l'interferometro sia montato e allineato come in figura 1.5 .

A tale proposito conviene fare alcune considerazioni. Se l'interferometro fosse perfettamente allineato, in modo che la sovrapposizione dei fronti d'onda fosse perfetta, e se i due bracci fossero esattamente di pari lunghezza ottica, sullo schermo non vedremmo frange di interferenza, piuttosto un'illuminazione con intensità uniforme, dovuta alla sovrapposizione dei due campi con le rispettive fasi (che non è la somma delle intensità come sarebbe se i campi non fossero coerenti). Vedremmo, in sostanza, un'unica frangia di interferenza . La figura di interferenza è dovuta quindi al fatto che i due fasci che si ottengono per divisione dal fascio principale non sono perfettamente paralleli. A seconda della regolazione della direzione di propagazione dei due fasci, tramite le regolazioni di specchi e beamsplitter, potremo osservare frange di interferenza con diverse interfrange.

La differenza di fase φ[3] che i due fasci hanno quando arrivano sul secondo beamsplitter , dipende dalla differenza di cammino ottico percorso e , supponendo l'indice di rifrazione del mezzo in cui è realizzato l'interferometro (aria) uniforme e pari a $\mathbf{n_i}$ tale sfasamento è dato da:

$$\varphi = \frac{2\pi}{\lambda} n_i (L_1 - L_2) \qquad (1.1)$$

dove $\mathbf{L_1}$ e $\mathbf{L_2}$ sono le lunghezze dei due bracci dell'interferometro e λ è la lunghezza d'onda del laser che è pari a 632,8nm.

Ponendo ad esempio una cella di precisione di vetro ottico: con indice di rifrazione $\mathbf{n_0}$, spessore **h**, spessore del vetro **s**, contente un idrogelo: con indice di rifrazione **n**; posto sul fondo della cella un cartoncino contenete ioni di calcio e posizionato il tutto all'interno dell'interferometro la nuova differenza di fase sarà:

$$\varphi = \frac{2\pi}{\lambda}\left[n_i (L_1 - L_2) + (n - n_i)h + (n_o - n_i)s \right] \qquad (1.2)$$

Col passare del tempo gli ioni di calcio diffonderanno all'interno della cella modificando così l'indice di rifrazione e di conseguenza la differenza di fase tra i due fasci.

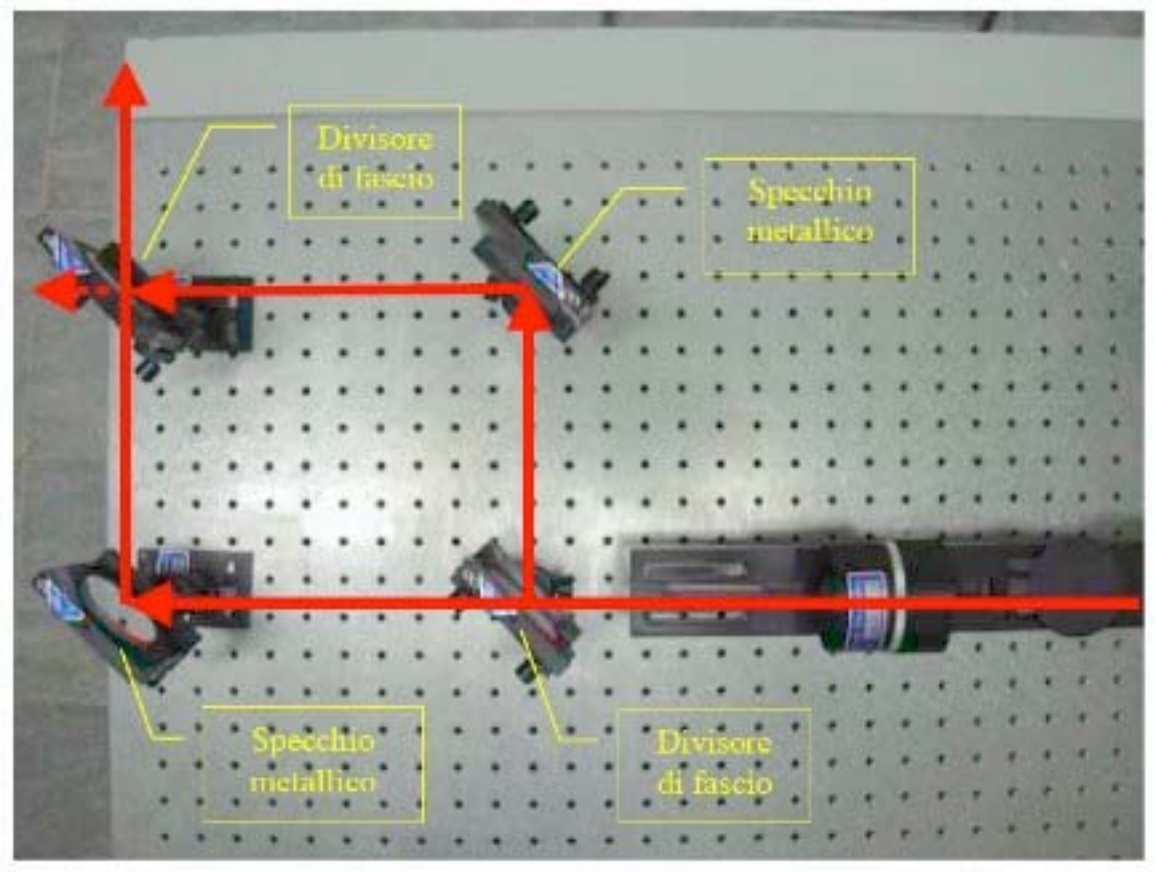


Figura 1.5: Esempio di un interferometro di Mach-Zehnder montato e allineato.

In realtà la variazione dell'indice di rifrazione dell'idrogelo non è l'unico effetto. Volendo essere precisi, bisogna considerare anche che l'indice di rifrazione del vetro ottico e dell'aria circostante la cella vengono modificati dalla temperatura. Lo stesso vale per l'idrogelo. Si considerano, però queste variazioni dell' indice di rifrazione dovute alla variazione di temperatura trascurabili rispetto alla variazione dell'indice di rifrazione dell'idrogelo indotto dalla diffusione di ioni calcio. Trascuriamo inoltre le variazioni anomale della differenza di fase dovute al fatto che, raffreddandosi, il vetro che costituisce la cella diminuisce il suo spessore e aumenta la sua densità. In definitiva la variazione di sfasamento tra i due fasci , nel tempo, è data da:

$$\Delta\varphi = \frac{2\pi}{\lambda} h\Delta n \qquad (1.3)$$

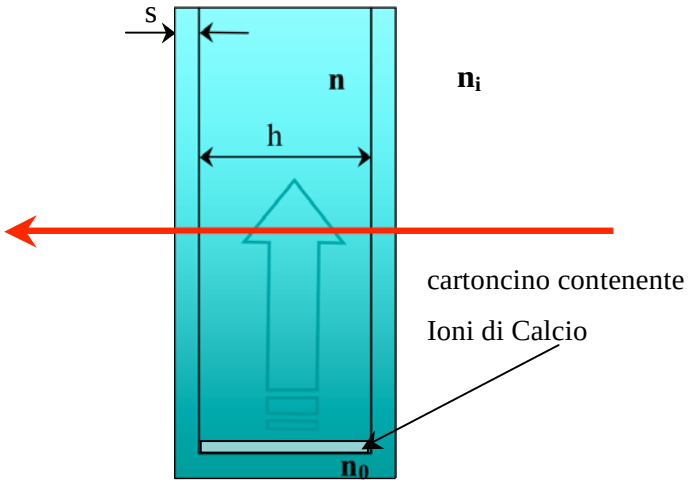


Figura 1.6: Struttura del porta campioni e posizionamento del cartoncino contenente ioni di calcio.

Se andassimo a monitorare la corrente che scorre all'interno del fotodiodo ci accorgeremmo che ha andamento sinusoidale. I picchi corrispondono agli instanti in cui l'indice di rifrazione è tale da creare interferenza costruttiva tra i due fasci, mentre i ventri corrispondono agli istanti in cui si ha interferenza distruttiva. Se i massimi si trovano a distanza $\mathbf{t_0}$ allora si può ricavare la variazione dell'indice di rifrazione nel tempo $\mathbf{t_0}$. Infatti dalla relazione (1.3) si ottiene:

$$\Delta n = \frac{\Delta\varphi\lambda}{2\pi h} = \frac{2\pi\lambda}{2\pi h} = \frac{\lambda}{h} \qquad (1.4)$$

Se ad esempio la cella è profonda h=10mm e λ=632,8nm si avrà una variazione dell'indice di rifrazione pari a $63{,}28*10^{-6}$,che è anche la minima variazione dell'indice di rifrazione che è possibile rilevare, nel tempo $\mathbf{t_o}$. Per aumentare la risoluzione della misura è necessario aumentare le dimensioni della cella.

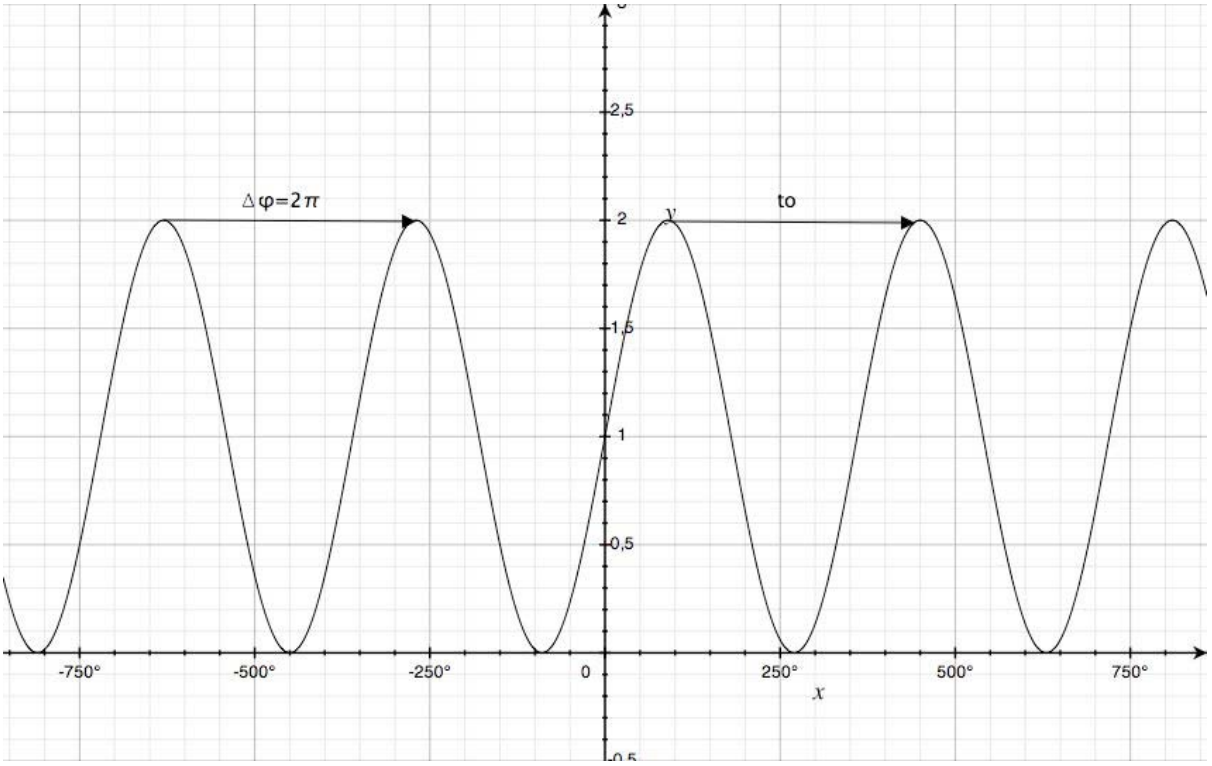


Figura7: Misurando il tempo t_o si ricava la minima variazione dell'indice di rifrazione

Se invece, si è interessati a misurare il tempo **t** impiegato per avere una variazione dell'indice di rifrazione pari a **Dn** si dovrà misurare il tempo tra **k** massimi ovvero k2π variazioni di fase Δφ.

$$Dn = k\Delta n = \frac{k\Delta\varphi\lambda}{2\pi h} = \frac{k2\pi\lambda}{2\pi h} = \frac{k\lambda}{h} \quad (1.5)$$

Se ad esempio si vuole sapere quanto tempo impiega la soluzione per variare l'indice di rifrazione di $10*10^{-3}$ (h=10mm) allora si dovranno contare 158 massimi.

$$k = Dn/\Delta n = 10\cdot 10^{-3}/63{,}28\cdot 10^{-6} = 158{,}0278 \cong 158$$

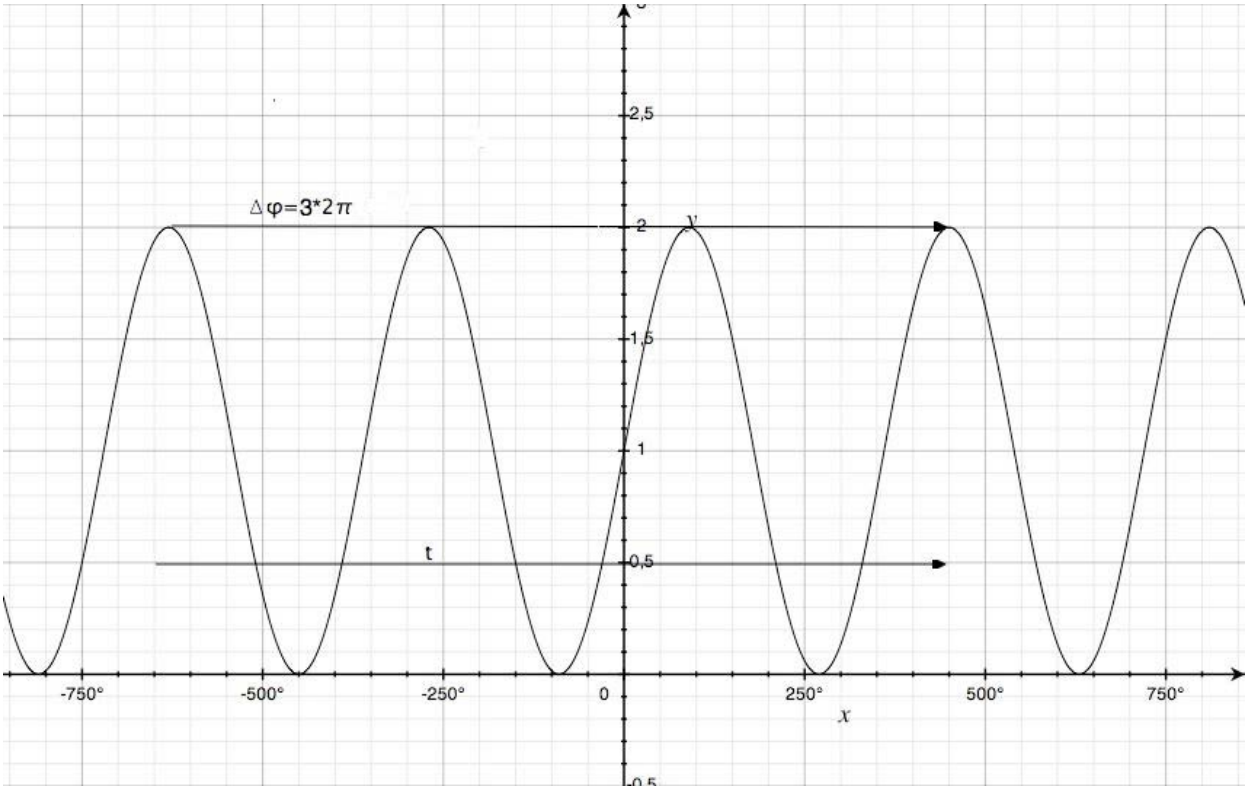


*Figura 1.8: Caso in cui Dn= 189,84*10^{-6} e Δn=63,28*10^{-6} (k=3)*

Il processo di diffusione degli ioni di calcio non è detto che avvenga sempre alla stessa velocità, quindi la corrente nel fotodiodo avrà andamento sinusoidale con frequenza variabile. Non ci è dato di sapere come varia la frequenza, essendo argomento di studio, ma è stata fatta una stima sul valore massimo. Una stima grossolana, nel caso peggiore, sul valore massimo della frequenza prevede che questa si aggiri intorno ai 10-20 KHz anche se si ritiene ragionevole che le frequenze in gioco siano di qualche centinaio di Hertz.

1.6 Metodi di valutazione della variazione dell' indice di rifrazione nel tempo.

Abbiamo visto, nel paragrafo precedente, che per valutare la variazione dell'indice di rifrazione nel tempo è necessario misurare il tempo che intercorre fra il massimo preso come riferimento e il massimo *k-esimo*. Si è pensato a due possibili soluzioni al problema, di seguito illustrate.

1.6.1 Il metodo del comparatore a soglia

E' il metodo più semplice. Consiste nel convertire la corrente proveniente da un fotodiodo in tensione e poi squadrare il segnale attraverso un comparatore con isteresi a soglia regolabile[1]. I fronti di salita di questo segnale vengono poi usati per incrementare un contatore che non appena raggiunge il valore impostato (k) stoppa un timer che fornisce così il tempo **t** (o $\mathbf{t_0}$) su 11 bit di cui: 10 per il valore della misura: tmax=1023ms ovvero (f_{min}=0,977Hz) ed 1 per indicare l'unità di misura: ms o μs. L'importante è scegliere la base dei tempi giusta per il timer e memorizzarne ogni *overflow*. Si tenga presente però che ad ogni *overflow* prima di poter ricominciare il conteggio, il timer, introduce un ritardo, pari a qualche ciclo di clock. Si deve prevedere di poter recuperare questo tempo in qualche modo. Altro aspetto importante è il ritardo introdotto dall'operazione di stop , reset e trasferimento del valore del timer sul bus dati. Un' ulteriore variante consiste nell'usare solo il comparatore (figura 1.9) e mandare così il segnale squadrato su un ingresso digitale del calcolatore, il quale attraverso una applicazione apposita sarà in grado di misurare il tempo intercorrente fra *k* massimi. Questa soluzione è sicuramente più semplice a livello circuitale e richiede un solo collegamento di interfaccia tra il trasduttore ed il calcolatore.

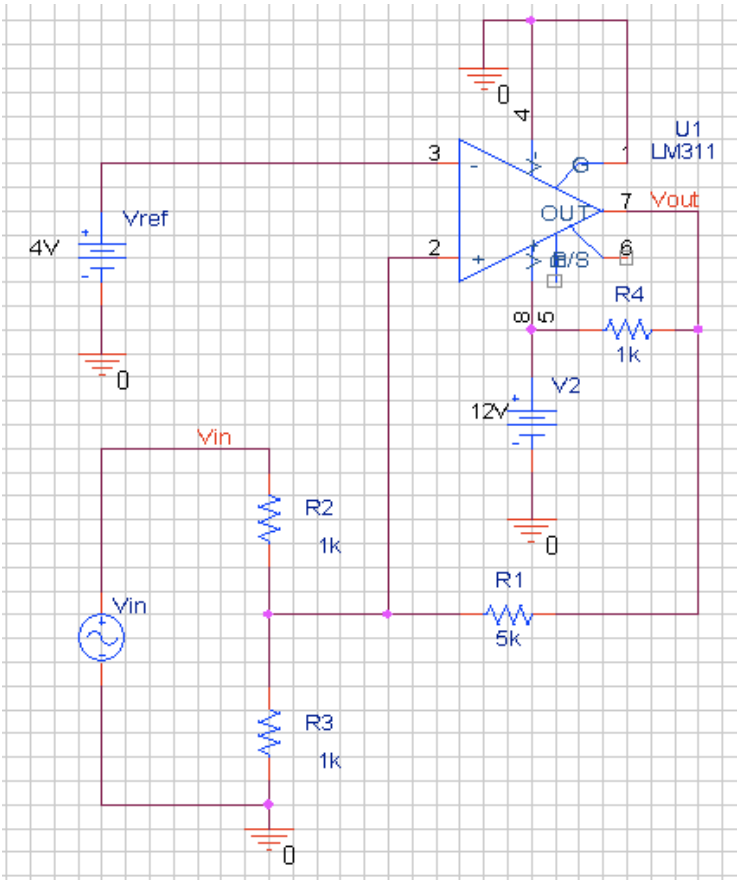


Figura 1.9: Esempio circuitale di comparatore a soglia

Le soglie di questo comparatore con isteresi ad alimentazione singola sono:

$$V_{LH}^{+} = Vref = Vin_{LH}\frac{R1//R3}{R2 + R1//R3}$$

$$V_{HL}^{+} = Vref = Vin_{HL}\frac{R1//R3}{R2 + R1//R3} + Vcc\frac{R2//R3}{R1 + R2//R3}$$

Dove:

Vin_{LH} è la tensione di ingresso alla quale la tensione di uscita del comparatore diventa 12V.

Vin_{HL} è la tensione di ingresso alla quale la tensione di uscita del comparatore diventa 0V.

Per il circuito di figura le soglie saranno:

$$\frac{Vref}{\frac{R1//R3}{R2+R1//R3}} = Vin_{LH} = \frac{4}{\frac{833}{833+1k}} \cong 8{,}8V$$

$$\frac{Vref - Vcc\frac{R2//R3}{R1+R2//R3}}{\frac{R1//R3}{R2+R1//R3}} = Vin_{HL} = \frac{4-12\frac{500}{500+5k}}{\frac{833}{833+1k}} \cong 6{,}4V$$

E' facile verificare la bontà del risultato ottenuto osservando la simulazione del circuito di figura 1.9 riportata in figura 1.10

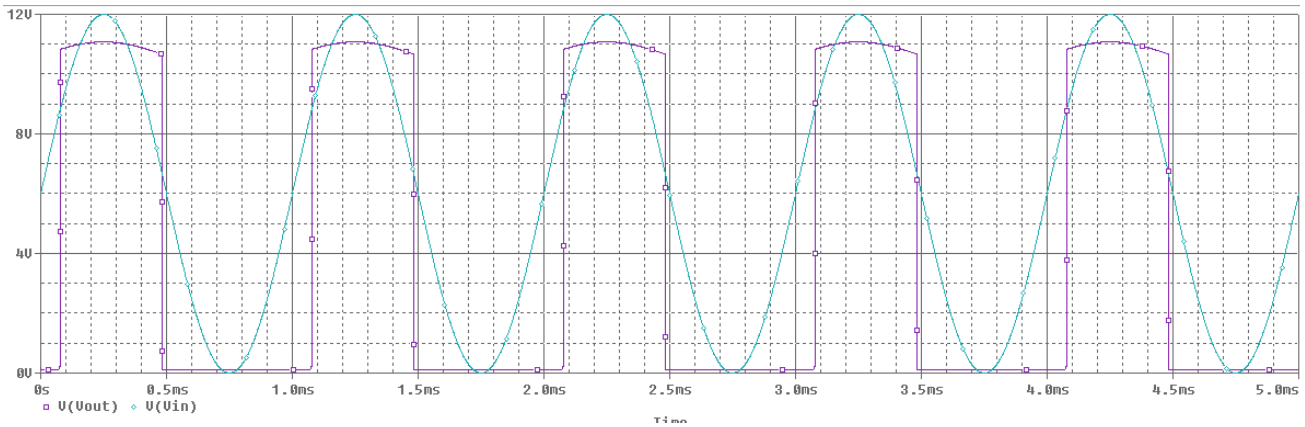


Figura 1.10: Risultato della simulazione del circuito di figura 1.9.

Per poter progettare adeguatamente il circuito è necessario tenere presente che l'intensità luminosa incidente sul fotorivelatore varia idealmente tra 0 e Pi_{max}, nel nostro caso tra 0 e 20mW, ma nella realtà a causa di perdite di potenza dovute a riflessioni ed assorbimenti da parte dei *beam splitter* e degli specchi ciò non avviene. Si deve quindi impostare la tensione di soglia Vin_{LH} al di sotto del massimo di tensione rivelabile, che non corrisponde per l'appunto ad una potenza incidente di 20mW. Per facilitare questa operazione è necessario rimpiazzare R1 con un trimmer e dimensionare adeguatamente R2 e R3 affinchè lo *sweep* di tensione ottenibile variando R1 sia il più ampio possibile , in modo da garantire la condizione sopra citata.

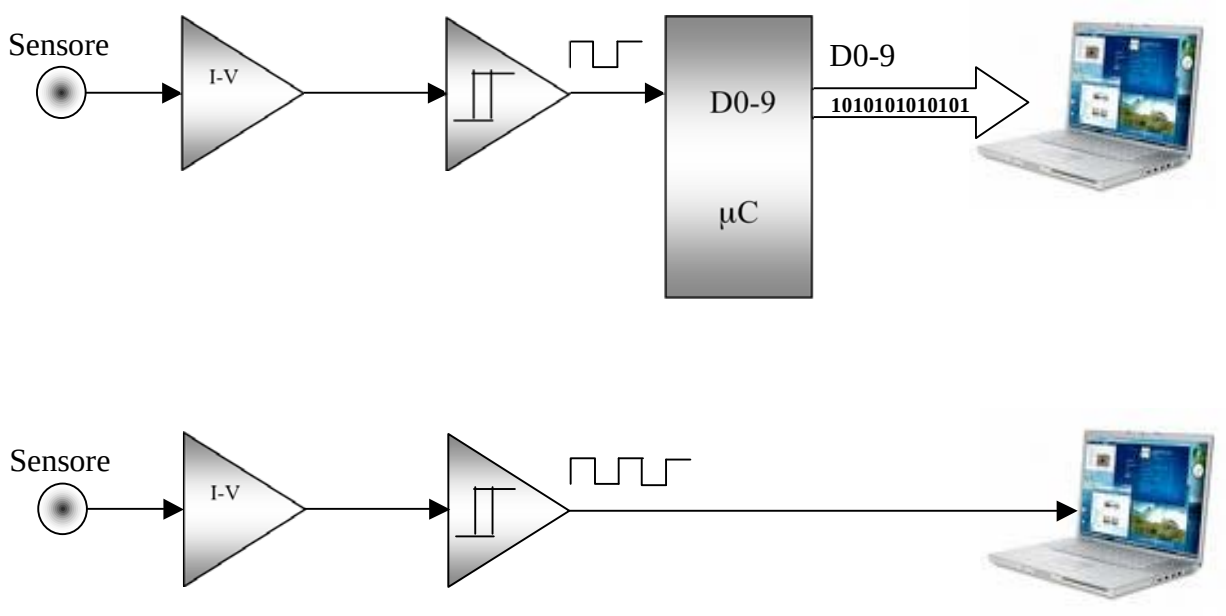


Figura 1.11: Schema a blocchi del metodo del comparatore con e senza microcontrollore.

Il pregio di questa soluzione sta sicuramente nella semplicità circuitale.
Lo svantaggio principale risiede nella perdita dell'informazione vera e propria, ovvero il segnale proveniente dal convertitore I-V. Questo, infatti, può essere molto utile, ad esempio, durante le operazioni di calibrazione e allineamento dell' interferometro nelle fasi iniziali della misura, nelle quali si ricerca un massimo di tensione (massimo di potenza incidente).

1.6.2 Il metodo con convertitore analogico-digitale

Questa è la classica soluzione adoperata nella configurazione di un sistema di misura e prevede l'utilizzo di un convertitore ADC veloce .
L'applicazione in esecuzione sul calcolatore provvederà ad analizzare tutti i dati ricevuti, individuare i massimi e contarli. Conoscendo poi il tempo di campionamento si risale all'istante temporale nel quale si è verificato il massimo attraverso questa semplice relazione:

$$t_{\max} = t_o + t_{sample} \cdot n°sample_{\max}$$

Se ad esempio si campiona a 100 KHz (t_{sample}= 10μs) e l'applicazione rileva la presenza di un massimo in corrispondenza del 1111 campione allora saranno passati 11,11 ms.

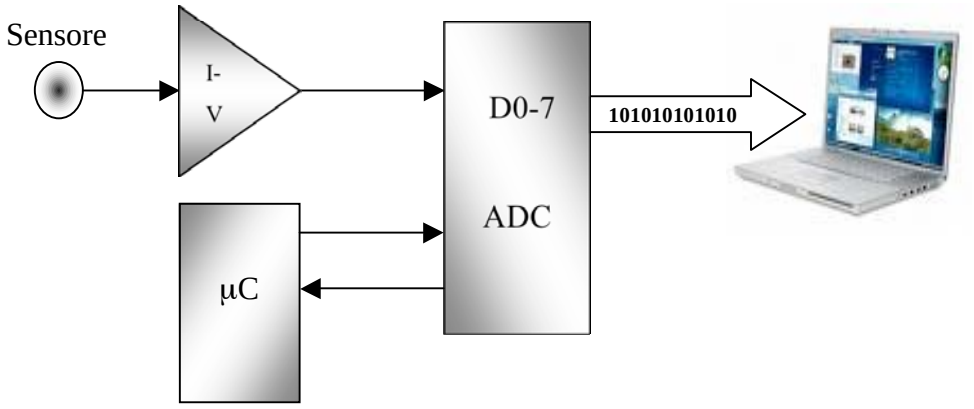


Figura 1.12: Schema a blocchi del metodo con convertitore 8 bit

Volendo invece rilevare il tempo che occorre per avere una variazione dell'indice di rifrazione pari a **Dn** allora l'applicazione dovrà contare **k** massimi pari a:

$$k = \frac{Dn \cdot h}{\lambda}$$

In corrispondenza del *k-esimo* massimo l'applicazione ricava l'istante temporale a cui questo si è verificato mediante la relazione:

$$t_{k\max\ j} = t_{sample} \cdot n°sample_{k\max}$$

Al termine della misura si avranno *j* istanti $\mathbf{t_{kmax}}$. Questi *j* eventi sono associati a *h* eventi di egual numero. Ogni *h-esimo* evento corrisponde ad una variazione dell'indice di rifrazione Dn secondo la seguente relazione:

$$Dn_h = Dn \cdot j$$

E' possibile così ricostruire l'andamento dell'indice di rifrazione nel tempo come mostrato in figura 1.13.

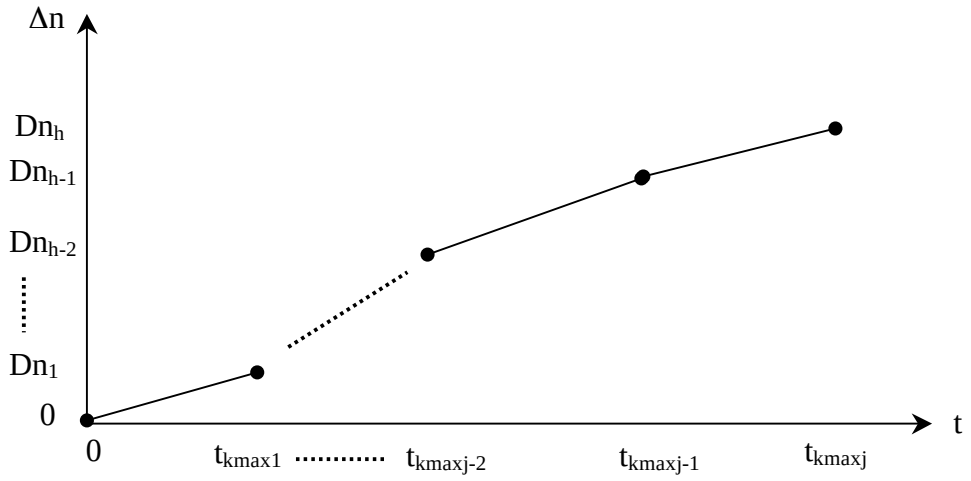


Figura 1.13: Ricostruzione dell'andamento nel tempo della variazione dell'indice di rifrazione

Va precisato che affinché ciò avvenga è necessario che non vi sia una perdita dei dati inviati dall' ADC al calcolatore. Smettendo infatti di campionare o bloccando la ricezione dei dati non è più possibile ricavare con esattezza l'istante temporale a cui si verificano i massimi. Questo problema viene aggirato adoperando un protocollo di comunicazione veloce fra ADC e calcolatore. Altro aspetto da valutare è il rumore che può "camuffare" un massimo di tensione. L'applicazione: **client software**, dovrà quindi essere in grado di interpolare i campioni del segnale proveniente dal trasduttore e individuare gli istanti esatti a cui si verificano i massimi, tenendo conto del rumore e che l'algoritmo deve garantire precisione e velocità. Queste ultime due caratteristiche sono poco compatibili, verrà adottato quindi un sistema (paragrafo 5.4.4) in grado di coniugare le due cose.

1.7 Struttura del sistema e metodo impiegato per la valutazione dell'indice di rifrazione.

La scelta del metodo e della struttura del sistema da utilizzare si basa sulle specifiche del sistema di misura. Abbiamo visto come, in buona sostanza, il *"metodo del comparatore a soglia"* abbia un'interfaccia molto semplice ma richieda particolari attenzioni nella gestione del timer interno al microcontrollore. Il tempo misurato ha poi una lunghezza pari a 10 bit, il che complica il trasferimento al calcolatore, perché un'operazione di scrittura richiede l'invio di 2 byte anziché 1, dimezzando così il *throughput* di dati effettivi trasmessi all'elaboratore. Quest' ultimo problema è aggirabile solo adoperando un canale di comunicazione che permette la trasmissione di dati di 10 bit alla volta. Nonostante l'elevato numero di bit impiegati non sarà mai possibile misurare tensioni continue e con frequenza minore di 0,977 Hz (vedi par. 1.6.1) Questi problemi non si presentano qualora si decida di non adoperare il microcontrollore esterno e delegando tutto il lavoro di elaborazione del dato all'applicazione un esecuzione sul calcolatore. L'uso di questo metodo però non permette di adoperare il segnale del trasduttore per poter effettuare operazioni di calibrazione ed allineamento dell' interferometro come richiesto nelle specifiche del sistema. Impiegando invece il *"metodo con ADC"* sarà possibile integrarlo all'interno della configurazione tipica di un sistema di misura e controllo a più canali, illustrato in figura 1.14.

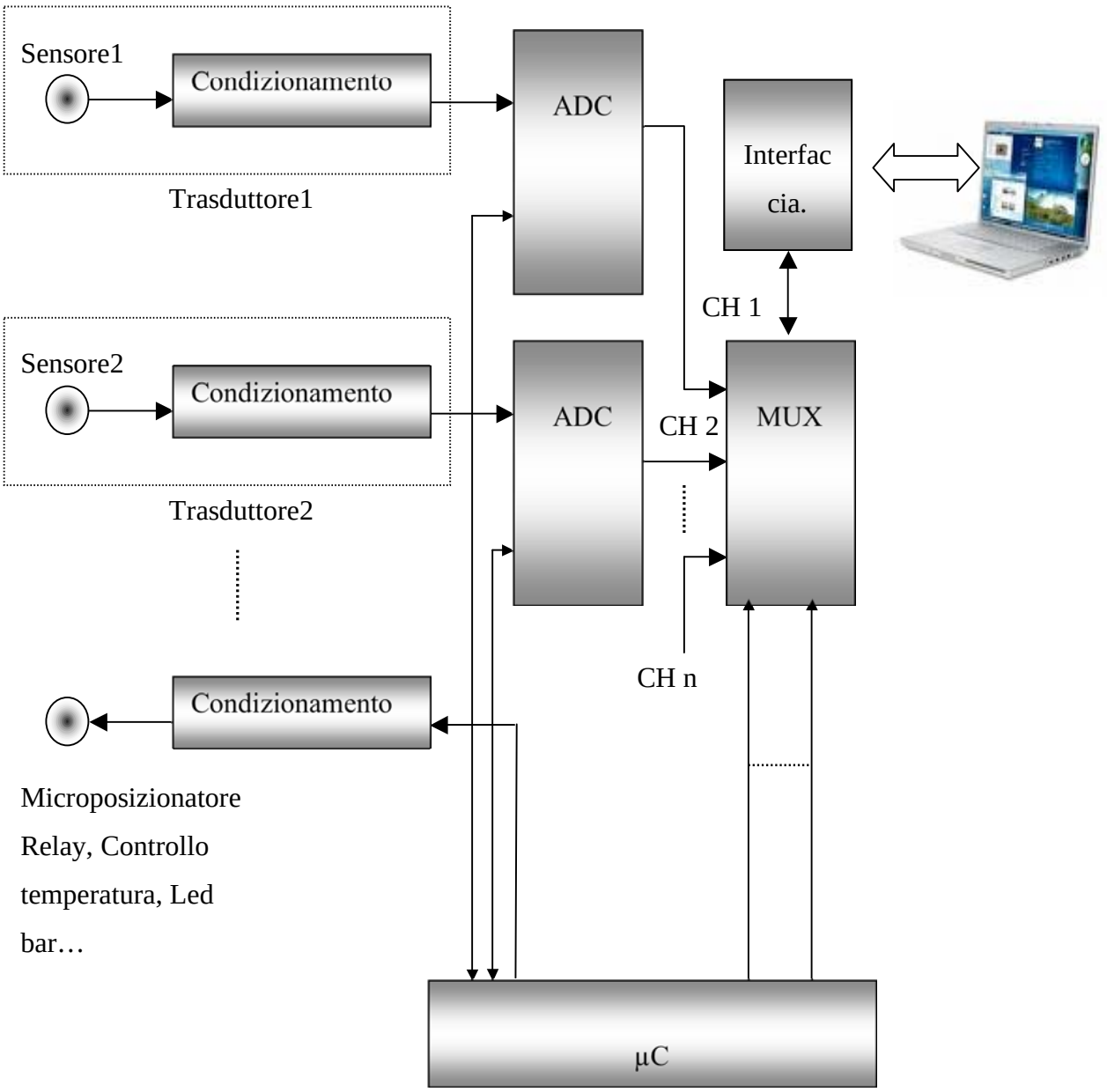


Figura 1.14: Possibile configurazione di un sistema di acquisizione e controllo a più canali.

Strutture simili e più complete di quella illustrata sono implementate nelle schede DAQ della National Instruments. Offrono un'ampia modalità di collegamenti possibili fra DAQ e PC, tra cui: USB, PCI, PMCI,PXI.. etc. Una vasta scelta viene offerta anche per quanto concerne la frequenza di campionamento (simultaneo e non), il numero di canali di acquisizione: analogici e digitali, il numero di I/O digitali e di uscite analogiche, timer interni etc. Tali schede di acquisizione vincolano all'utilizzo di LabVIEW anche

se al momento,nel nostro caso, sembra essere l'ambiente di sviluppo più adatto per la progettazione del Client Software. Tale vincolo,in questo ambiente di lavoro, diventa un vantaggio poiché esistono delle funzioni di libreria che permettono di velocizzare le operazioni di configurazione ed acquisizione dati dal DAQ. L'unico svantaggio risiede nell' elevato costo della scheda, indipendentemente dal tipo di comunicazione implementato. Ad incidere maggiormente sul prezzo è infatti proprio la frequenza di campionamento che in questo caso dovrà essere, nel caso peggiore, pari a 20KHz*4 =80Ks/s (su un canale). A seconda poi del DAQ scelto è possibile che sia necessario comprare dei moduli opzionali per la corretta messa in servizio dello stesso (cavo, blocco connettore…). Questo fa si che i DAQ in questione non siano idonei ad essere utilizzati in questo lavoro di tesi. Dopo una attenta ricerca non è stato possibile reperire sul mercato, concorrente alla National Instruments, un DAQ che offrisse le caratteristiche elettriche strettamente necessarie e una completa compatibilità con LabVIEW ad un prezzo ridotto. Ciò ci ha spinto nella direzione di una progettazione *ad hoc* dell'intero sistema finalizzata a ridurre al minimo i costi a parità di prestazioni funzionali necessarie.

1.8 Dimensionamento del sistema.

Arrivati a questo punto è necessario fare un dimensionamento di massima del sistema di acquisizione, strutturato come in figura 1.15, per poi andare a definire nei dettagli le caratteristiche di ogni singolo blocco funzionale.
E' necessario definire:

- ***Caratteristiche del sensore***
- ***Caratteristiche del circuito di condizionamento***
- ***Risoluzione del convertitore ADC***
- ***Frequenza di campionamento dell'ADC***
- ***Bus di comunicazione con l'elaboratore e relativa interfaccia***
- ***Modalità di alimentazione del circuito***
- ***Realizzazione e distribuzione del sistema su PCB***
- ***Ambiente di sviluppo per la realizzazione del Client Software***

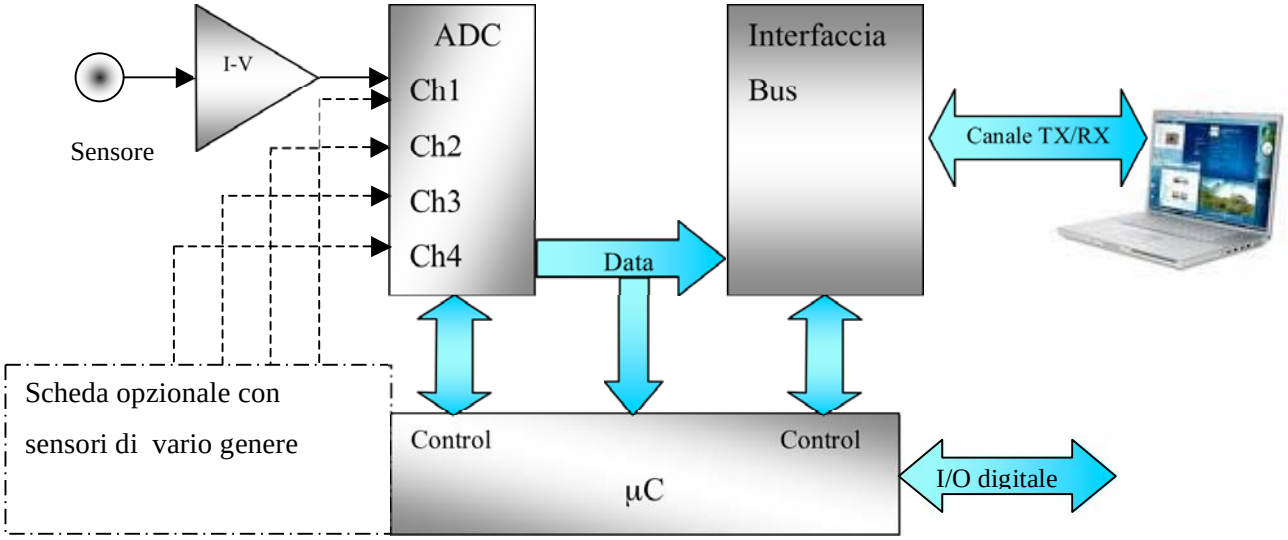


Figura 1.15: Struttura a blocchi dell'intero sistema di misura

Il sensore sarà un fotodiodo. Deve essere scelto in maniera che lo *spot*, cioè l'area sensibile del sensore, abbia dimensione superiore a quella del raggio laser,in modo da facilitare l'allineamento del sensore con il raggio ma non precluderne la velocità di misura. E' necessario inoltre, che presenti il massimo rapporto corrente generata/potenza incidente in corrispondenza della lunghezza d'onda del raggio laser, ovvero 632,8nm.

Il circuito di condizionamento sarà un semplice convertitore corrente-tensione, conoscendo la banda massima del segnale di ingresso (10-20KHz) sarà opportuno che la banda del convertitore sia B>50KHz, mentre il guadagno dipende dal sensore scelto e dal *range* di tensione ammesso sull'ingresso dell'ADC.

Supponiamo di aver già scelto un ADC a 8 bit e che la tensione di riferimento sia 2,5V. Scelto il guadagno del convertitore corrente- tensione tale che in corrispondenza di una potenza incidente pari a 20mW la tensione sia pari a 2,5V allora:

$$LSB = \frac{V_{REF}}{2^n} = \frac{2,5}{2^8} = 9,765mV$$

La risoluzione , cioè la minima parte del fondo scala che si riesce ad apprezzare, è pari a:

$$risoluzione = \frac{\frac{V_{REF}}{2^n}}{V_{REF}} = \frac{1}{2^n} = \frac{1}{2^8} \cong 4 \cdot 10^{-3} (0{,}4\%)$$

il che vuol dire che siamo in grado di leggere variazioni dell'intensità del raggio laser pari allo 0,4%, ovvero:

$$P_{\min} = P_{INCnom} \cdot risoluzione = 20mW \cdot 4 \cdot 10^{-3} = 80uW$$

Se si considera che la potenza emessa dal raggio laser in genere non è molto stabile ma oscilla tra ±1~5% che corrispondo ad oscillazioni, rispetto alla potenza nominale di 20mW, pari a 200μW~1mW allora possiamo concludere che un convertitore ad 8 bit è più che sufficiente poiché la risoluzione è maggiore di quella della fonte luminosa.

La frequenza di campionamento dell'ADC deve essere per il teorema di *Nyquist-Shannon:*

$$f_{sample} \geq 2 f_{\max} \quad (1.6)$$

di solito si ottengo buoni risultati già con

$$f_{sample} = 4 f_{\max} \ (1.6a)$$

si evince che :

$$f_{sample} = 4 \cdot 20KHz = 80KHz$$

Questo dimensionamento è stato fatto mettendoci nel *worst case* cioè : f_{max}= 20KHz (par. 1.4). Una stima ragionevole infatti ci lascia supporre che la frequenza del segnale sia di gran lunga inferiore, centinaia di Hertz. Trattandosi quindi di un sistema che non effettua un campionamento simultaneo su più canali (un solo ADC per più canali) la frequenza di campionamento per ogni canale sarà pari a $f_{sample-1channel} = f_{sample} / n°canali$. Se i canali sono 4 allora la frequenza di campionamento per canale è pari a 20KHz; è quindi possibile rispettare la condizione (1.6a) con segnali a 5KHz e quindi abbondantemente al di sopra della frequenza stimata.

Il sistema per evitare la perdita di dati in fase di trasmissione deve necessariamente adottare un protocollo di comunicazione in grado di trasmettere i dati alla velocità richiesta dal sistema. Supponendo una frequenza di campionamento di 100KHz allora la velocità del convertitore sarà di 100KSPS (Sample Per Second).La banda del canale di trasmissione dovrà essere quindi maggiore i 100KByte/s (1Byte=1Sample). Per scegliere il canale di trasmissione adatto si fa riferimento alla tabella che riporta la massima velocità di comunicazione di tutti i protocolli di comunicazione supportati da un calcolatore.

Porta	Tipo di comunicazione	Massima velocità di trasferimento	Massima lunghezza collegamento
RS232	Seriale	115,2Kb/s	25m
USB1.0	Seriale	1,5Mb/s (187,5KB/s)	7x5m
USB1.1-2.0	Seriale	12Mb/s (1,5MB/s)	7x5m
IEEE 1394	Seriale	400Mbps (50MB/s)	4,5m
High Speed USB2.0	Seriale	480Mps (60MB/s)	7x5m

Tabella 1.1: I protocolli di comunicazione più impiegati a confronto.

Il protocollo che adopera la porta parallela non è riportato nella tabella perché non è più usato, tanto è che sui calcolatori di recente fattura ormai è diventato impossibile avere a disposizione tale porta se non attraverso una scheda aggiuntiva da installare sulla scheda madre. La velocità di comunicazione di questo protocollo in ogni caso resta molto bassa perché pensata e sviluppata per interfacciare le stampanti e gli scanner di prima generazione. Va detto inoltre che i *laptop* d'ultima generazione, per ridurre i consumi e le dimensioni non integrano più nemmeno la porta seriale RS-232 ma solo connessioni di tipo USB e FireWire (IEEE 1394). A dispetto di ciò, però la porta seriale si è ritagliata un notevole spazio nel campo dell'automazione industriale per la programmazione ad esempio di PLC , pannelli *touch screen* e ogni sorta di inverter e servodrive, costringendo così l'utente finale ad usare adattatori USB-RS232.
Si evince che oggi giorno si è costretti, dal settore *consumer* a quello industriale ad interfacciarsi con la porta USB 1.1 e 2.0 (la USB 1.0 non è quasi più presente sul

mercato) mentre la porta High-speed USB 2.0 attualmente non si è ancora notevolmente diffusa anche perchè i costruttori,al momento, si limitano ad usarla nei collegamenti interni delle periferiche del calcolatore come ad esempio *webcam* e *fingerprint recognition technology* (riconoscimento di impronte digitali) integrati. Per tutta questa serie di ragioni il protocollo più adeguato è quello USB 2.0 che garantisce fra l'altro una velocità di comunicazione fra calcolatore e sistema di acquisizione dati pari a 1,5MB/s permettendo così, in linea teorica, di campionare a 1,5MHz e quindi di ricostruire correttamente segnali con banda pari a 375-750KHz.

La scelta delle tensioni di alimentazione è quasi obbligata, infatti, l'amplificatore operazionale impiegato per realizzare il convertitore corrente tensione, nella maggior parte dei casi necessita di una alimentazione duale compresa fra ±12V e ±15V. La stragrande maggioranza dei convertitori ADC invece hanno una tensione di alimentazione compresa tra i 3V e i 5V. Per semplificare la progettazione delle alimentazioni allora si pensa di adottare due alimentatori *switching* da 12V mentre la tensione di 5V necessaria al convertitore la si preleva dall' USB. Questo è in grado di erogare un massimo di 500mA, più che sufficienti per alimentare un ADC, un microcontrollore, un' interfaccia USB e qualche *led* per monitorare il funzionamento del dispositivo.

La separazioni delle tensioni di alimentazione e la necessità di svincolare la posizione del calcolatore da quella dell'interferometro e del fotodiodo ci porta a dividere il sistema in altri due sottoblocchi come i figura 1.16. Così è possibile porre il sensore e il circuito di condizionamento del segnale sulla piastra dove è montato l'interferometro e portare poi il segnale attraverso un cavo schermato al **DAQ** (Digital AcQuisition) collegato al calcolatore attraverso un cavo USB.

L'ambiente di sviluppo usato per la realizzazione del client software è **LabVIEW** (**Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench) della National Instrument che adopera un linguaggio di programmazione grafico detto anche *Linguaggio G*. Questo ambiente è il più idoneo a realizzare in maniera semplice, efficace e veloce uno strumento virtuale. Oltre a mettere a disposizione una libreria di controlli ed indicatori molto vasta, che permette di realizzare un'interfaccia grafica

accattivante ed intuitiva, fornisce anche un'ampia libreria di funzioni matematiche e dedicate allo studio di segnali.

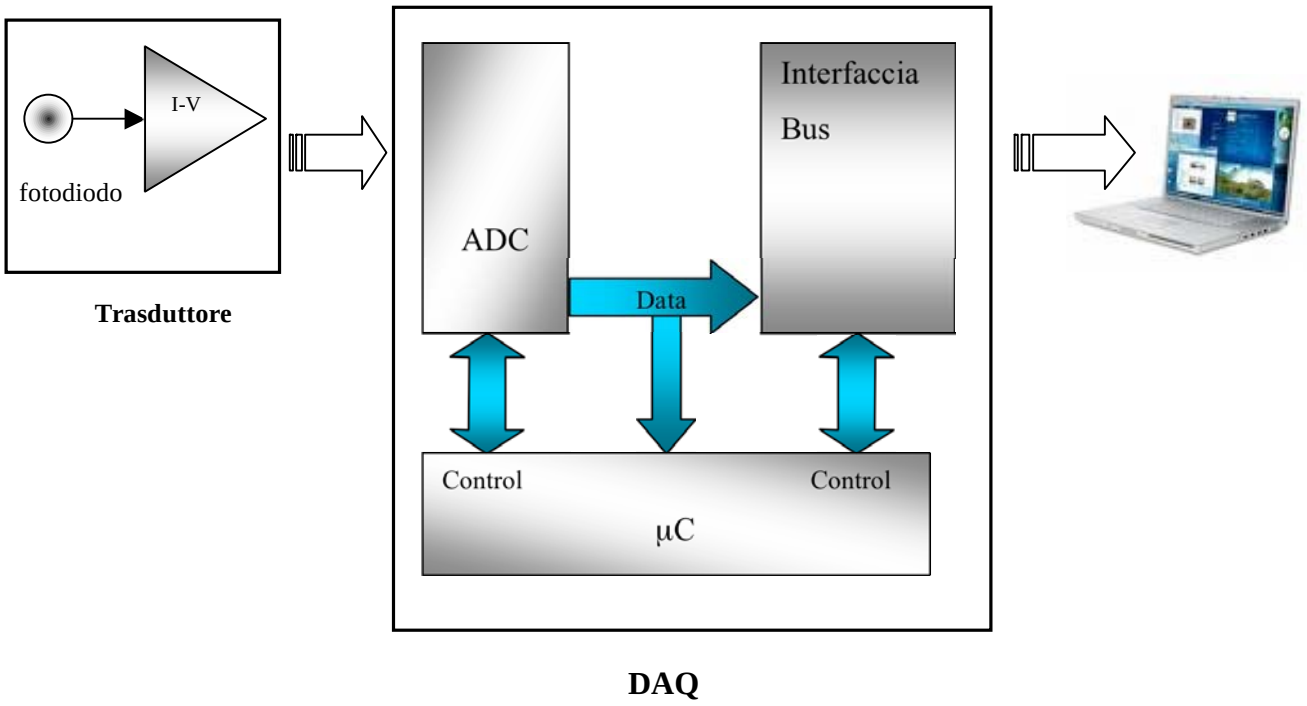


Figura 1.16: Struttura del sistema suddiviso in blocchi funzionali

Al termine del dimensionamento di massima del sistema abbiamo ricavato queste specifiche di progetto:

- ***Caratteristiche del sensore***: sensibilità a lunghezze d'onda λ=632,8nm ,elevato rapporto corrente/potenza incidente e *spot* di dimensioni opportune.
- ***Caratteristiche del circuito di condizionamento:*** convertitore corrente tensione con Banda≥50KHz e guadagno dipendente dalle caratteristiche del fotodiodo. Tensione di uscita 0-2,5V.
- ***Risoluzione del convertitore ADC:*** 8bit, tensione di ingresso 0-2,5V.
- ***Frequenza di campionamento dell'ADC:*** $f_{sample} \geq 80KHz$
- ***Bus di comunicazione e relativa interfaccia***: protocollo e interfaccia USB
- ***Modalità di alimentazione del circuito:*** doppia alimentazione ±12V e 5V ricavata rispettivamente da alimentatori switching e dalla porta USB.

- ***Realizzazione e distribuzione del sistema su PCB:*** suddivisione del sistema in due blocchi funzionali. Il primo costituito dal trasduttore ed il circuito di condizionamento mentre il secondo costituito dall' ADC dal microcontrollore e dall'interfaccia USB.
- ***Ambiente di sviluppo per il Client Software:*** *LabVIEW 8.0 professional*

In figura 1.17 è riportata anche la connessione delle alimentazioni e dei segnali tra i due blocchi funzionali ed il calcolatore.

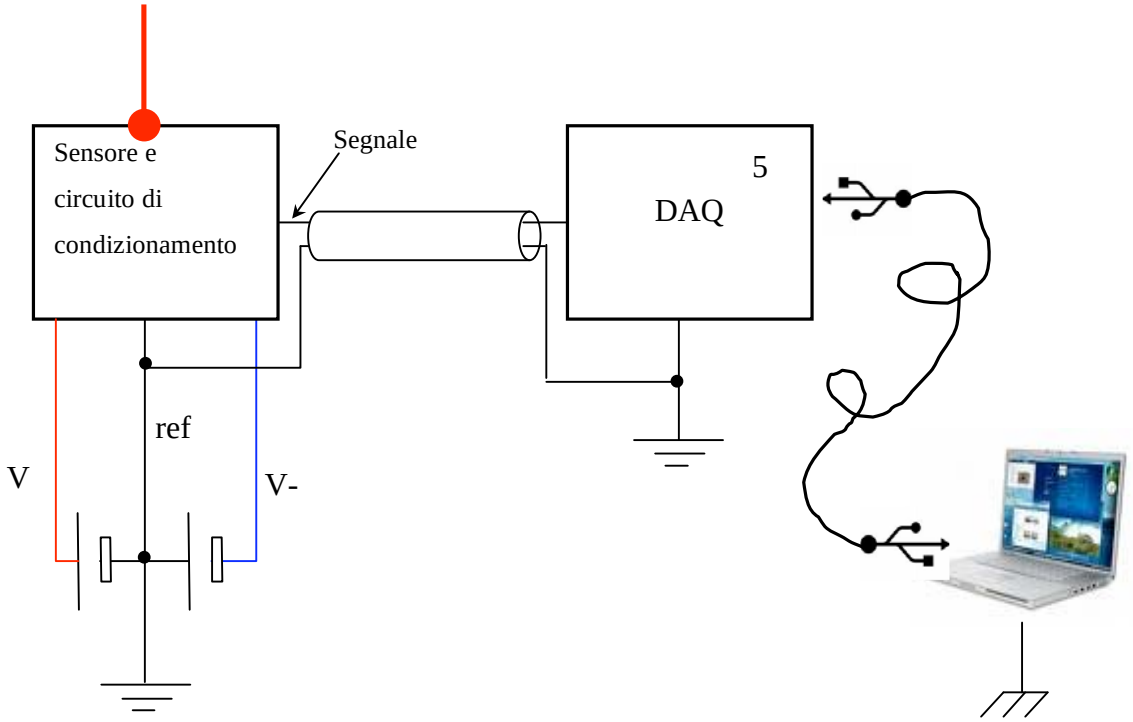


Figura 1.17 : Schema di collegamento fra i blocchi funzionali e il calcolatore. La tensione di 5V generata dall' bus USB è riferita a terra, potenziale a cui si trova il calcolatore stesso. Il collegamento fra la terra e il riferimento a cui è riferito il segnale è interno al DAQ, conseguentemente il terminale di riferimento è a potenziale nullo.

Capitolo 2

Il trasduttore

2. Introduzione

In questo capitolo ci occuperemo di progettare, dimensionare, realizzare e collaudare il trasduttore, costituito dal fotodiodo e dal circuito di condizionamento, ricordandoci le specifiche tecniche che devono avere.

Fotodiodo:

- sensibilità a lunghezze d'onda λ=632,8nm
- elevato rapporto corrente/potenza incidente
- spot di dimensione superiore alla superficie illuminata dal raggio laser

Circuito di condizionamento:

- amplificatore transresistivo
- banda≥50KHz
- guadagno che dipende dalle caratteristiche del fotodiodo
- tensione di uscita 0-2,5V

2.1 Principio di funzionamento del fotodiodo.

Occorre sapere che una giunzione *pn*, polarizzata inversamente ed illuminata , è attraversata da una corrente che varia linearmente al variare del flusso luminoso ed è praticamente indipendente dal valore della polarizzazione inversa, su queste proprietà è basato il funzionamento dei fotodiodo a semiconduttore. In assenza di luce, la corrente di buio di un fotodiodo corrisponde alla corrente di saturazione inversa dovuta ai portatori minoritari generati termicamente, se invece la luce colpisce la superficie del fotodiodo, in prossimità della giunzione (zona di svuotamento), con energia $E=hc/\lambda$ maggiore del *band-gap* del semiconduttore (Ec-Ev), si formano nuovi portatori minoritari, coppie elettrone - lacuna, il cui numero è proporzionale al numero di fotoni incidenti; tali coppie vengono separate dal campo elettrico (ε), dovuto alla polarizzazione inversa della giunzione, gli elettroni diffondono così verso la zona *n* mentre le lacune verso la zona *p* dando vita alla corrente di illuminamento $\mathbf{I_{ph}}$. Tutte le coppie elettrone - lacuna generate al di fuori della zona di svuotamento diffondono nella zona di competenza, gli elettroni verso la zona *n* e le lacune verso la zona *p*, dando vita ad una corrente di diffusione che limita la velocità di risposta del diodo. Questa corrente può essere diminuita aumentando la dimensione della zona di svuotamento in maniera da limitare il numero di coppie elettrone - lacuna che si formano al di fuori di tale questa. Questa soluzione viene adottata nei *fotodiodi p-i-n.*

Se la tensione di polarizzazione a cui è sottoposto il diodo è diretta ed è tale da ridurre il dislivello della barriera di potenziale in modo da permettere ad alcuni portatori maggioritari di attraversarla, allora si ha una corrente diretta il cui verso è tale da ridurre la corrente inversa.

La fotocorrente prodotta è proporzionale alla potenza incidente $\mathbf{P_i}$ con la seguente legge:

$$I_{ph} = R \cdot P_i$$

Dove R è la "***responsivity***" del fotodiodo, cioè il rapporto [A/W].

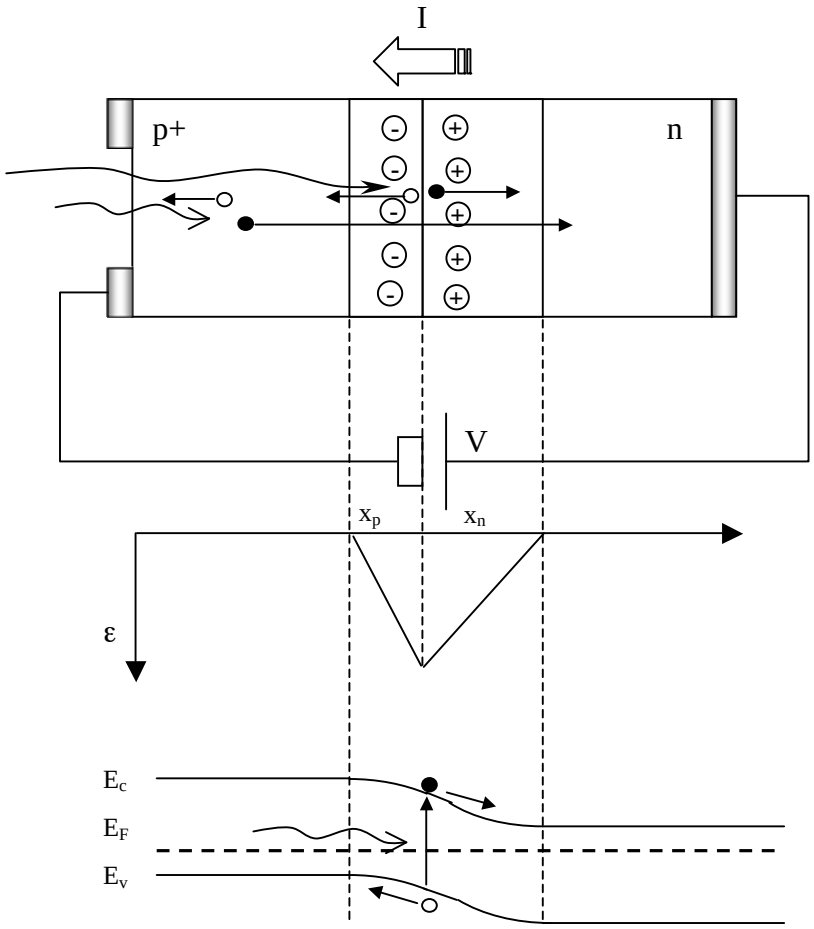


Figura 2.1: Struttura tipica di un fotodiodo polarizzato inversamente e relativo andamento del campo elettrico e dell' energia delle bande.

La ***responsivity*** può essere espressa in funzione di un altro importante parametro che è ***l'efficienza quantica***, definita come il rapporto tra il numero di elettroni prodotti (che contribuiscono alla corrente nel fotodiodo) ed il numero di fotoni incidenti:

$$\eta = \frac{I_{ph}/q}{P_i/h\nu} = \frac{h\nu}{q}R$$

è immediato ricavare R dalla relazione precedente:

$$R = \frac{\eta q}{h\nu} = \frac{\eta\lambda}{1{,}24}$$

con λ espressa in micron.

La corrente di illuminamento dipende dal punto della superficie del fotodiodo in cui viene focalizzata la radiazione luminosa. La diminuzione della corrente dovuta a una focalizzazione lontana dalla giunzione è dovuta alla ricombinazione dei portatori minoritari prima di diffondere verso la giunzione. Per questo motivo, i sensori in commercio sono dotati di un sistema ottico particolare, che in alcuni casi prevede anche l'uso di filtri ottici in modo da attenuare le lunghezze d'onda fuori dallo spettro. La risposta spettrale del fotodiodo dipende invece dalla natura del semiconduttore,da essa infatti dipende il valore del band-gap e quindi l'energia del fotone ($E=hc/\lambda$) necessaria a creare una coppia elettrone lacuna.

Materiale	λ[nm]
Silicio	190-1100
Germanio	800-1700
Arsenuro di Indio Gallio	800-2600

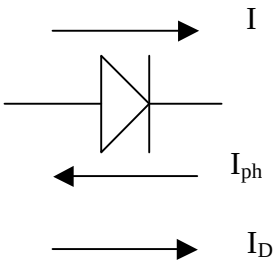
Tabella 2.1: Sensibilità alle varie lunghezze d'onda in funzione del materiale utilizzato.

Le variazioni di sensibilità sono funzione della temperatura ambiente, della focalizzazione della luce sulla superficie del diodo, della frequenza di taglio e dell'angolo di incidenza della radiazione luminosa sulla superficie del fotodiodo. La massima sensibilità è ottenuta per un'incidenza normale mentre la variazione della sensibilità è espressa o dall'angolo di ricezione, che è l'angolo compreso fra le due direzioni per le quali la sensibilità relativa riferita al valore massimo si riduce del 50% , o dal diagramma polare di sensibilità relativa.

2.2 Caratteristica elettrica e circuito equivalente del fotodiodo.

La corrente che attraversa il fotodiodo ad una data tensione V è data dal contributo della corrente I_{ph} fotogenerata e dalla corrente che scorre in una giunzione *pn*, I_D.

$$I = I_o\left(e^{\frac{V}{\eta V_T}} - 1\right) - I_{ph} = I_D - I_{ph}$$



Il fattore di idealità η dipende dal materiale usato ed è η=1 nel caso del Germanio e $\eta \cong 2$ nel caso del Silicio. La corrente I_o è la corrente di buio ed è proporzionale alla superficie della giunzione e alla temperatura; quindi dispositivi con un ampia superficie sensibile all'illuminazione avranno cuna corrente di buio maggiore. La caratteristica del dispositivo è riportata in figura 2.2.

Il circuito equivalente del dispositivo è approssimabile al circuito di figura 2.3, dove il diodo è considerato ideale, la R_d è la resistenza di giunzione del fotodiodo, C_d è la capacità di giunzione mentre R_s è la resistenza di contatto del dispositivo che essendo molto bassa può essere tranquillamente trascurata. La capacità di giunzione dipende dalla larghezza della zona di svuotamento e diminuisce all' aumentare di questa , quindi polarizzando in inversa la giunzione, C_d diminuisce e l'elevato campo elettrico fa si che diminuisca il tempo di transito delle cariche rendendo il dispositivo più veloce. C_d dipende infine linearmente dalla superficie sensibile all'illuminazione, quindi dall' area della giunzione. La resistenza Rd invece si dimezza ad ogni incremento di 10°C della temperatura.

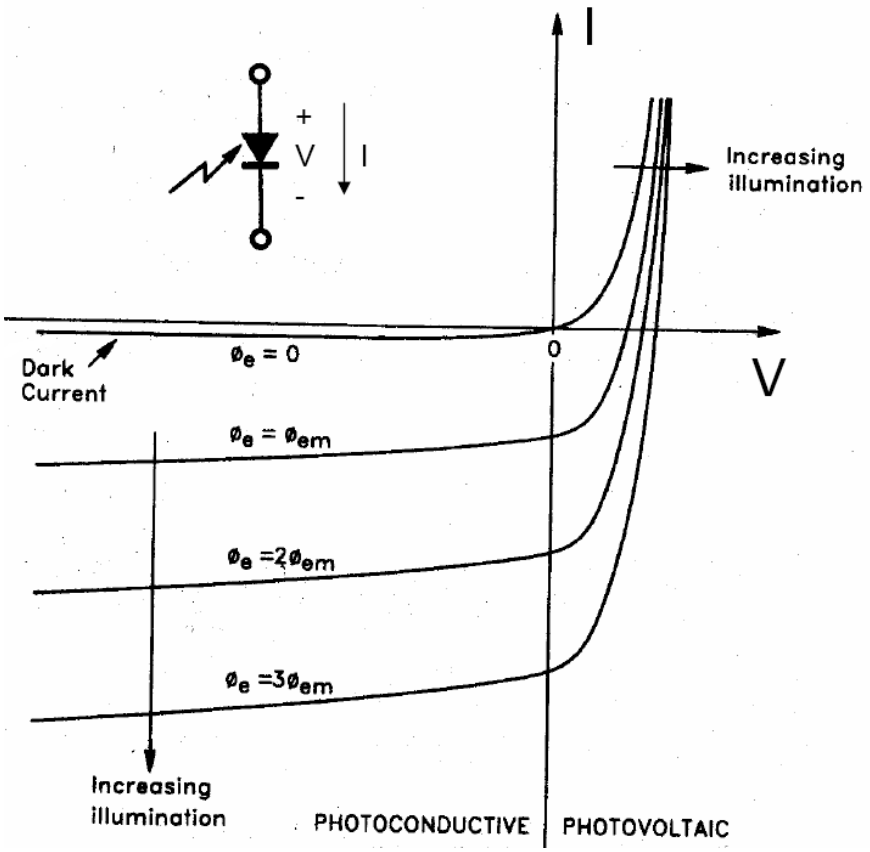


Figura 2.2: Caratteristica I-V del fotodiodo ed individuazione delle regioni di funzionamento

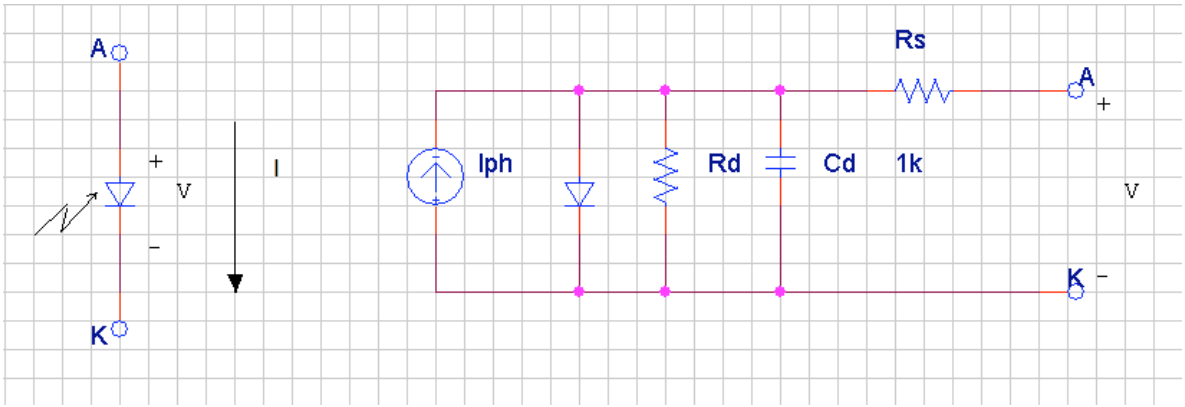


Figura 2.3: Circuito equivalente del fotodiodo

2.3 Modalità di funzionamento di un fotodiodo

Il fotodiodo può operare in modo fotovoltaico o fotoconduttivo. La regione di funzionamento della prima modalità di funzionamento è il 4° quadrante, mentre per la seconda è il 3° quadrante delle caratteristiche I-V(fig. 2.2). La modalità di funzionamento influisce sulle prestazioni del dispositivo in termini di velocità di risposta e linearità .

2.3.1 Funzionamento fotoconduttivo

Se il fotodiodo è **polarizzato con tensione inversa V<<0** :

$$I = I_o\left(e^{\frac{V}{\eta V_T}} - 1\right) - I_{ph}$$

allora la corrente che scorre sarà pari a:

$$I = -I_o - I_{ph}$$

e si dice che si ha un funzionamento **fotoconduttivo** del fotodiodo.

In questa configurazione il fotodiodo eroga una corrente proporzionale alla potenza ottica incidente a cui è "sovrapposta" la corrente di buio, la quale varia con la temperatura; la risposta non è quindi lineare. La velocità del dispositivo è massima poiché la tensione inversa riduce la capacità di giunzione e diminuisce il tempo di transito delle cariche. Le sorgenti di rumore per questo dispositivo possono essere individuate nel rumore shot, e in quello termico dovuto alla resistenza Rd.

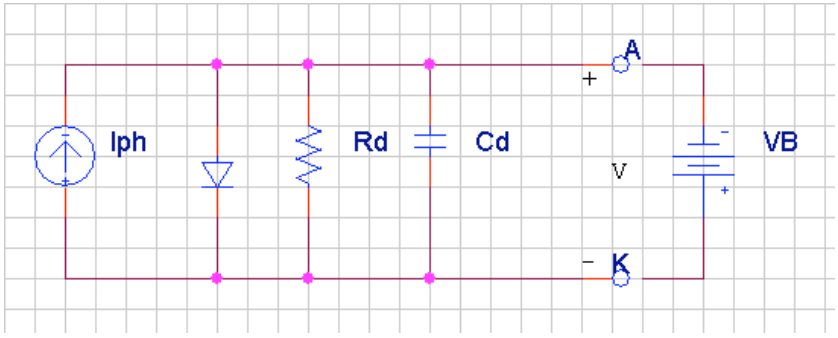


Figura 2.4: Circuito equivalente di un fotodiodo polarizzato inversamente(V_B)

Il circuito di figura 2.5 invece raffigura un' amplificatore di transresistenza. Il fotodiodo, in virtù del cortocircuito virtuale è polarizzato inversamente e la tensione di uscita è pari a :

$$V_0 = -Rf(I_{ph} + I_0)$$

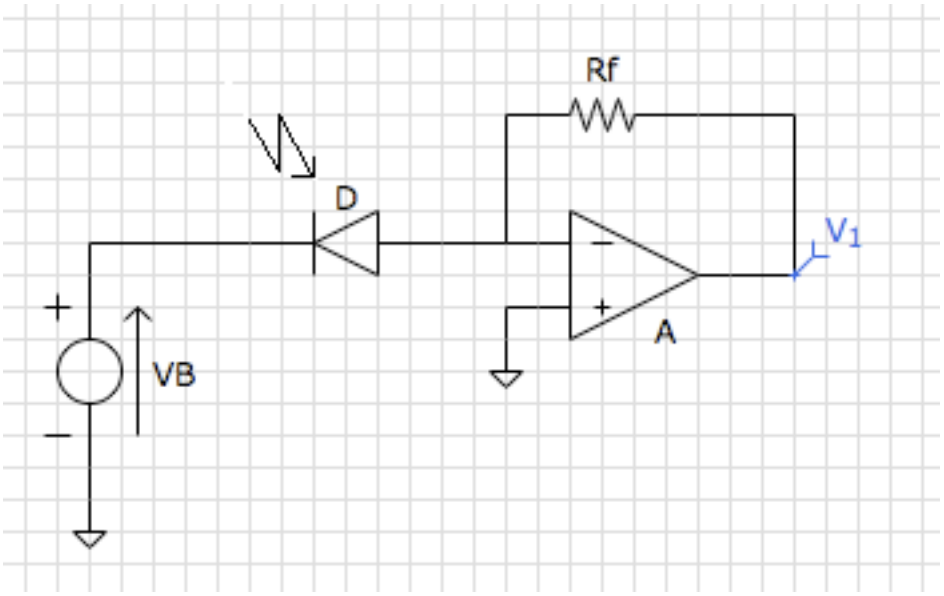


Figura 2.5 : La corrente I che scorre nel diodo è convertita in tensione dall'amplificatore di transresistenza.

2.3.2 Funzionamento fotovoltaico

Se il **fotodiodo non è polarizzato** allora si dice che si ha un funzionamento **fotovoltaico** del fotodiodo e il circuito equivalente del dispositivo si riduce a quello rappresentato in figura 2.6.

Se **RL>>Rd** allora la corrente I che scorre nel carico è trascurabile e si può approssimare che :

$$I = I_o\left(e^{\frac{V}{\eta V_T}} - 1\right) - I_{ph} = 0$$

da cui si ricava V, la tensione ai capi di R_L:

$$V = \eta V_T \log\left(\frac{i_{ph}}{i_0}\right)$$

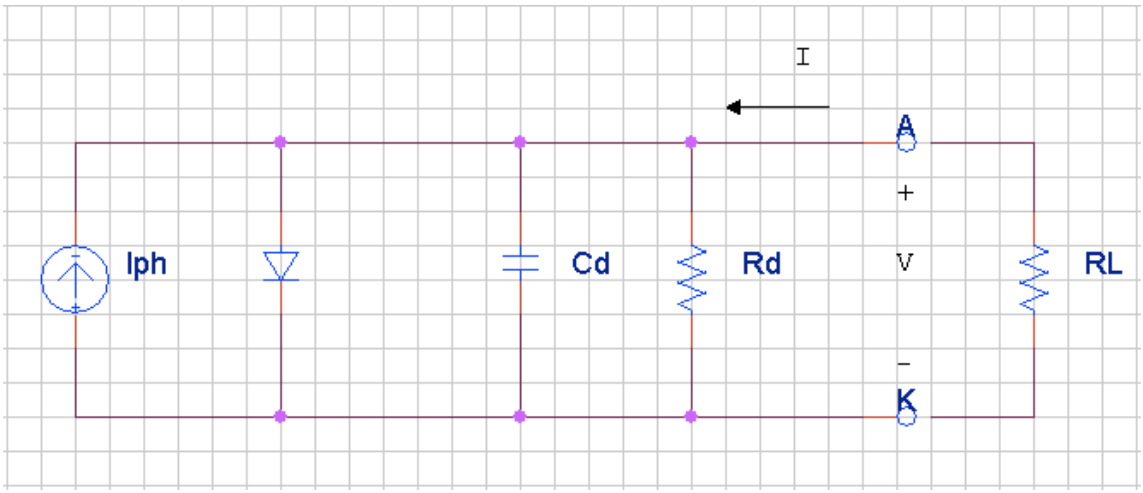


Figura 2.6: Circuito equivalente del fotodiodo funzionante in modo fotovoltaico.

La tensione ai capi del fotodiodo varia quindi in modo logaritmico con l'illuminazione. Un esempio di funzionamento fotovoltaico del fotodiodo è riportata nel circuito di figura 2.7.

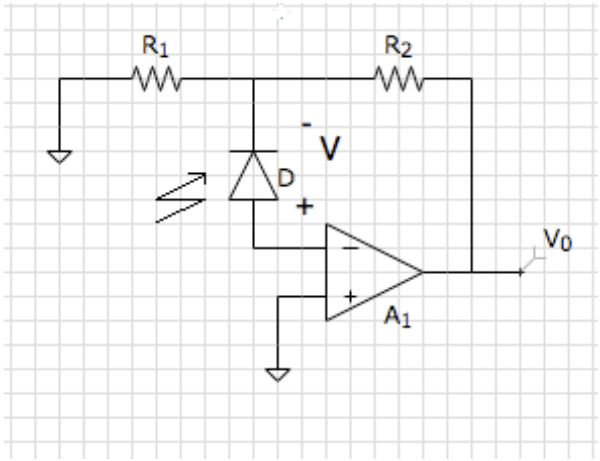


Figura 2.7 : In questo circuito il fotodiodo ha un funzionamento di tipo fotovoltaico e la tensione di uscita V_0 varia in modo logaritmico con la potenza incidente.

La corrente I che scorre all' interno del fotodiodo in direzione del terminale invertente è da considerarsi nulla in virtù dell'elevata impedenza di ingresso dell' amplificatore. La tensione V ai capi del fotodiodo invece è pari a:

$$V = -V_0 \frac{R_2}{R_1 + R_2}$$

che è anche uguale a

$$-V_0 \frac{R_2}{R_1 + R_2} = \eta V_T \log\left(\frac{I_{ph}}{I_o}\right)$$

da cui si ricava la tensione di uscita:

$$V_0 = -(1 + \frac{R_2}{R_1}) V_T \log\left(\frac{i_{ph}}{i_0}\right)$$

Se invece $\mathbf{R_L << R_d}$ allora tutta la corrente generata dal fotodiodo fluisce in R_L e vale, con **V=0** :

$$I = -I_{ph}$$

Il fotodiodo eroga una corrente che è proporzionale esclusivamente alla potenza incidente dato che il contributo dovuto alla corrente di buio è nullo.
Il contributo del rumore introdotto si riduce a quello termico generato dalla resistenza Rd, il che rende questa configurazione particolarmente adatta per applicazioni di precisione. Un esempio di funzionamento fotovoltaico con uscita del fotodiodo su impedenza praticamente nulla, grazie alla reazione (tensione-serie), è riportato in figura 2.8. L'amplificatore usato in questa applicazione è un amplificatore di transimpedenza la cui tensione di uscita è:

$$V_0 = Rf \cdot I_{ph}$$

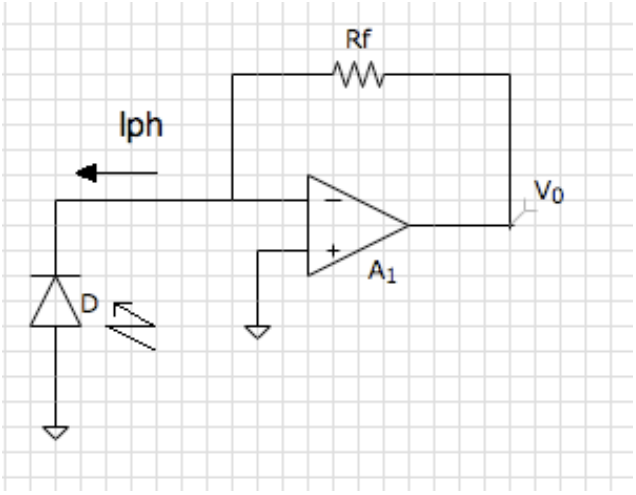


Figura 2.8: Il fotodiodo funziona in modo fotovoltaico ma la tensione di uscita varia linearmente con la potenza incidente.

A titolo riassuntivo le principali differenze tra le due configurazioni sono riportate nella tabella seguente:

Fotovoltaico	Fotoconduttivo
Polarizzazione nulla	Polarizzazione inversa
Contributo nullo della corrente di buio	Contributo della corrente di buio
Risposta lineare	Risposta non lineare (funzione della T°)
Basso rumore (termico)	Rumore elevato (termico+shot)
Applicazioni di precisione	Applicazioni ad alta velocità

Tabella 2.2: Tabella riassuntiva delle caratteristiche delle due configurazioni

2.4 Scelta del fotodiodo

La specifica più restrittiva nella scelta del fotodiodo, oltre alla sensibilità a lunghezze d'onda pari a 632,8nm, è la dimensione dello *spot*, ovvero dell'area sensibile alla radiazione luminosa.

Una delle richieste fatte in fase di studio del sistema è stata proprio quella di avere un fotodiodo il cui *spot* fosse di dimensioni elevate in modo da facilitare l' operazione di allineamento del sensore con il raggio laser. Abbiamo visto però come la dimensione

della zona sensibile, ovvero la superficie della giunzione *pn* influenzi notevolmente la corrente di buio ,la capacità e la resistenza di giunzione. Uno spot di dimensioni elevate comporta quindi un'elevata corrente di buio , capacità e resistenza di giunzione; penalizzando così le prestazioni dinamiche del sensore e l'accuratezza della misura. Si è cercato quindi di limitare le dimensioni dello spot allo stretto necessario senza però penalizzare la semplicità di allineamento del raggio.

Considerando che il raggio laser adottato genera un fascio il cui diametro è pari a 3 mm; sarà pertanto necessario che la dimensione dello spot sia:

$$S \geq \pi \cdot r^2 \geq 7mm^2$$

in modo che l'intero fascio illumini la superficie sensibile del sensore.

Va considerato però che la superficie sensibile non è circolare ma quadrata(vedi figura 2.9); la dimensione minima dovrà pertanto essere 3x3 mm pari a $9mm^2$.

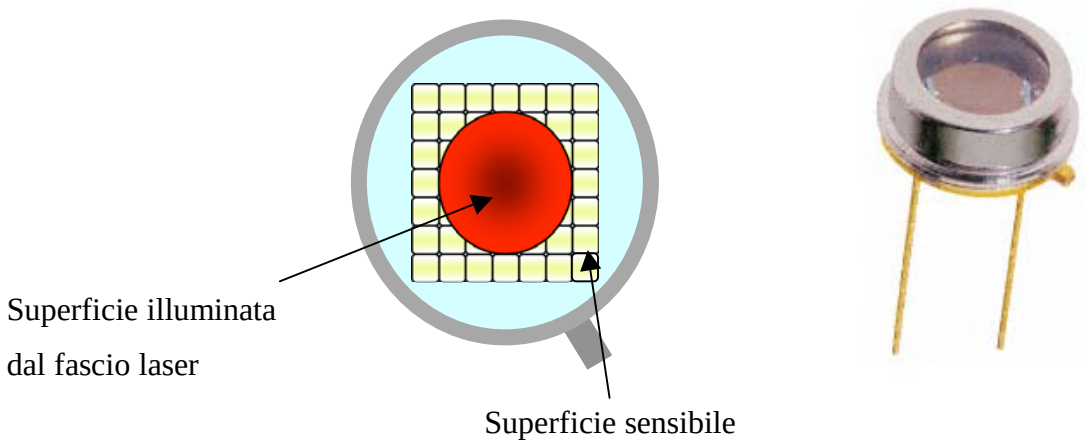


Figura 2.9: In figura viene individuata la superficie sensibile del fotodiodo e quella illuminata dal fascio laser. E' riportato inoltre l'aspetto del fotodiodo OSD15-5T della Centronic.

Fra i vari fotodiodi presenti sul mercato è stato scelto **l'OSD15-5T** prodotto da **Centronic**[4].
L'OSD15-5T è stato scelto per i seguenti motivi:

- sensibilità alle lunghezze d'onda comprese tra 430 e 900nm
- responsivity compresa fra 0,2 e 0,45 A/W
- elevata resistenza di giunzione Rd
- ridotta corrente di buio
- superficie sensibile di 15mm^2 (3,8x3,8 mm)
- possibilità di funzionamento fotovoltaico e fotoconduttivo
- basso costo

Va aggiunto inoltre che non è stato possibile reperire in commercio, fotodiodi con superficie sensibile compresa fra 10 e 15mm^2 e/o provvisti di circuito di condizionamento integrato.

Tipo	Area attiva		Responsivity (A/W) @ 436nm		Dark current (nA)		Cd (pF)		Rd (MΩ)	
							Vf=0	Vf=12V		
	mm^2	mm	Min	Typ	Max	Typ	Max	Max	Min	Typ
OSD15-5T	15	3,8x3,8	0.15	0.21	50	3	390	80	5	80
OSD7.5-5T	7,5	2.75x2.75	0.15	0.21	25	2	150	40	30	300

Tabella 2.3: In questa tabella sono elencate le caratteristiche del sensore OSD15-5T e del suo equivalente con superficie sensibile pari a 7,5mm^2 OSD7.5-5T.

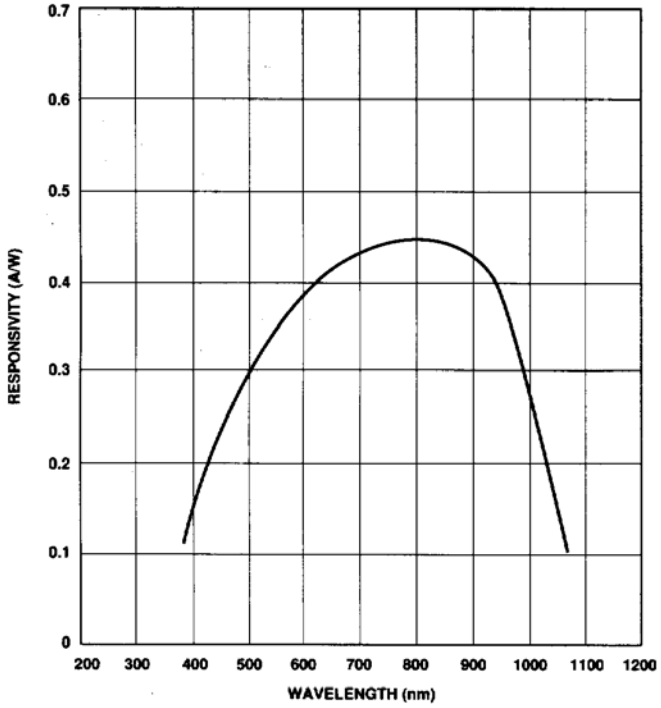


Figura 2.10 : Andamento della responsivity in funzione della lunghezza d'onda del raggio incidente.

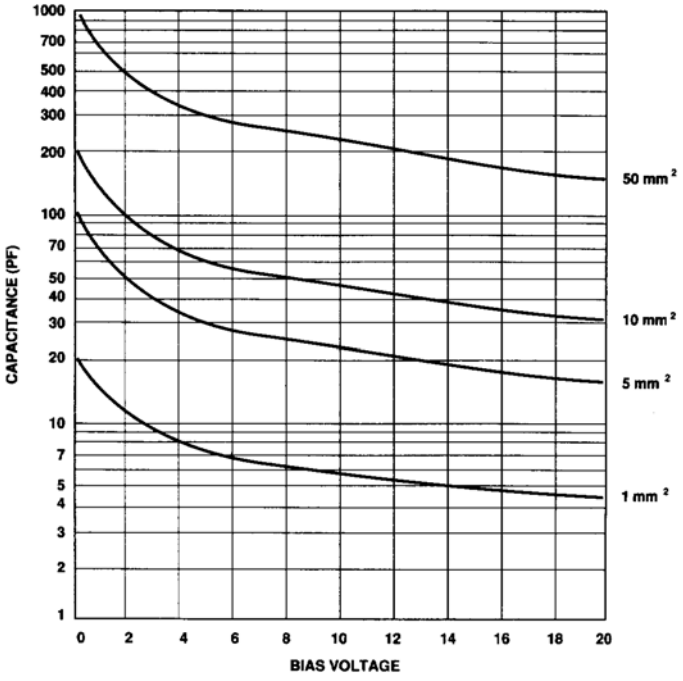


Figura 2.11: Variazione della capacità di giunzione Cd in funzione della tensione inversa applicata al fotodiodo al variare della superficie attiva.

2.5 Il circuito di condizionamento

Come abbiamo in precedenza visto il tipo di funzionamento del fotodiodo influisce sul tipo di risposta: lineare o meno; sull' accuratezza della misura; sulla velocità di risposta del fotodiodo : a banda larga se polarizzato in inversa o a banda stretta se non è polarizzato.

In questo caso il circuito di condizionamento deve essere in grado di garantire una risposta lineare,indipendente dalla corrente di buio ed una banda B≥50KHz. La configurazione più adatta è quella che prevede l'impiego del fotodiodo in funzionamento fotovoltaico ed è riportata nel circuito di figura 2.12 dove quest'ultimo è sostituito dal suo circuito equivalente (R1,C1, Iph).

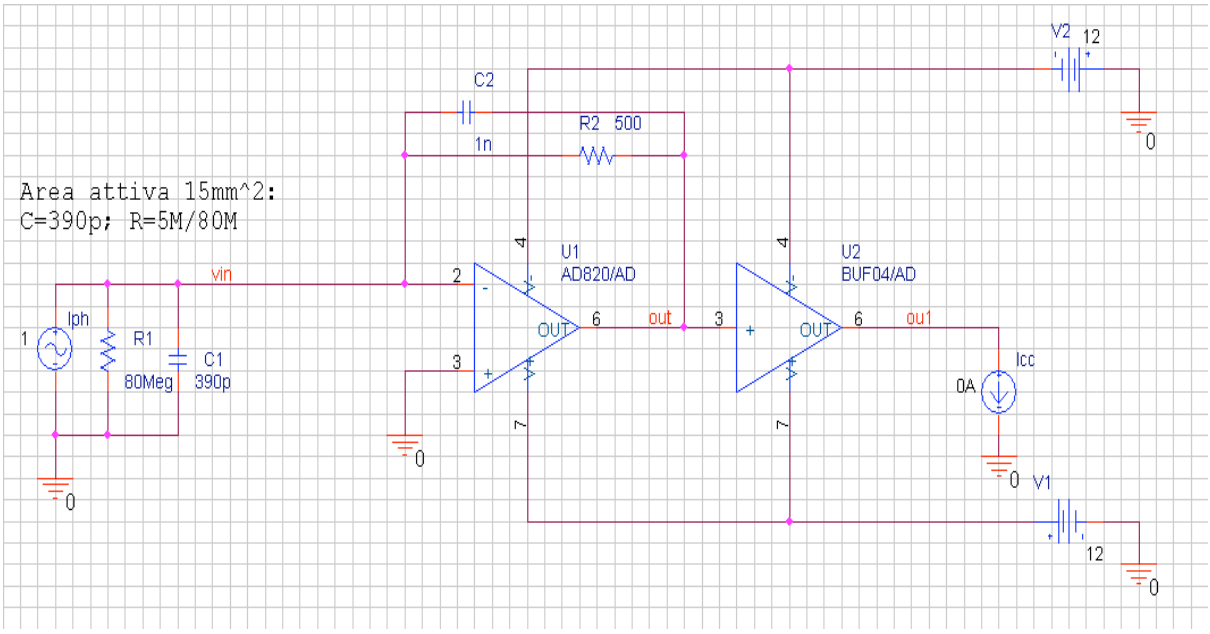


Figura 2.12: Schema elettrico del circuito di condizionamento usato per le simulazioni.

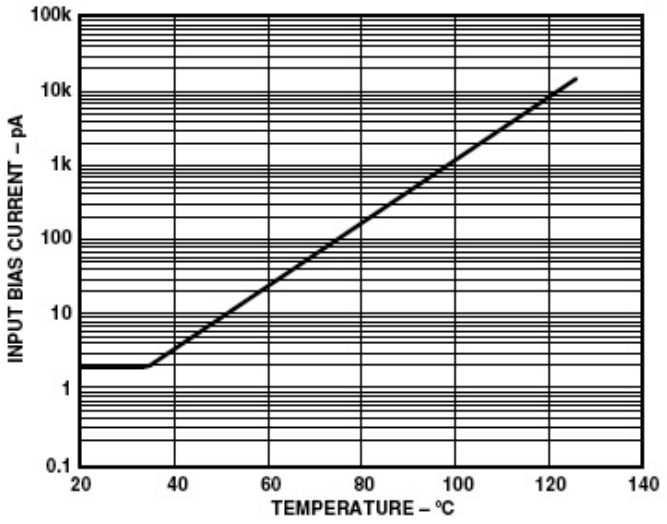
In questo tipo di configurazione, soprattutto per il valore esiguo delle correnti in gioco, assume una notevole importanza la corrente di polarizzazione I_B e la tensione di offset dell'amplificatore V_{OS}. Se si considera per comodità la corrente di sbilanciamento I_o =0 allora si ha che la tensione di uscita all'amplificatore di transresistenza sarà pari a :

$$V_{0UT} = R2 \cdot (I_{ph} + I_B) + Vos \cdot \left(1 + \frac{R_2}{R_1}\right)$$

anziché:

$$V_{0UT} = R2 \cdot I_{ph}$$

Per ridurre al minimo il termine di errore introdotto sarà pertanto necessario scegliere un amplificatore la cui corrente di polarizzazione e tensione di offset siano ridotte. La scelta quindi ricadrà su un amplificatore a FET: l'**AD820** prodotto e consigliato dall' ***Analog Device*** [5][6] per questo genere di applicazioni. L' AD820 ha una I_B il cui valore è compreso tra 2 e 25 pA il cui andamento al variare della temperatura è riportato in figura 2.13 mentre la tensione di offset Vos_{max}=800μV e la sua deriva termica è pari a 2 μV/°C. Se per ipotesi pensassimo di inserire una resistenza di valore pari a R2//R1 tra il terminale non invertente e massa per annullare la corrente I_B otterremmo lo spiacevole effetto di applicare la caduta di tensione su questa resistenza, pari a I_B*(R2//R1), direttamente sulla giunzione del diodo polarizzandolo inversamente. E' sconsigliabile per tanto questo tipo di annullamento della corrente di polarizzazione. E' infine impensabile applicare questo metodo poiché R1, la resistenza di giunzione del fotodiodo, varia con la temperatura e dimezza il suo valore ad ogni incremento della temperatura corrispondente a 10°C.



TPC 6. Input Bias Current vs. Temperature;
V_S = 5 V, V_{CM} = 0

Figura 2.13: Variazione della corrente di bias in funzione della temperatura

Per capire quanto e qual è il contributo di errore che incide maggiormente sulla tensione di uscita è necessario tener presente che il convertitore ADC è a 8 bit e la risoluzione è 10mV, se si considera la V_{REF}=2,5V, che corrisponde ad una potenza del raggio incidente pari a circa 80μW. Supponiamo che il sensore OSD15-5T alla lunghezza d'onda di 632,8nm abbia una *responsivity* R=0,4 [A/W] allora la corrente minima rilevabile, associata alla più piccola variazione di potenza incidente rilevabile, è pari a:

$$Iph_{MIN} = R \cdot Pi_{MIN} = 0,4 \cdot 80\mu W = 32\mu A$$

Il guadagno G dell' amplificatore di transimpedenza, se R=0,4 , si ricava associando alla massima potenza incidente la tensione V_{REF} :

$$\frac{V_{REF}}{G} = R \cdot Pi_{MAX}$$

da cui:

$$G = \frac{V_{REF}}{R \cdot Pi_{MAX}}$$

quindi essendo il guadagno pari a R2:

$$R_2 = \frac{2,5}{0,4 \cdot 20mW} = 312,5\Omega$$

Proviamo adesso a fare un calcolo della tensione errore presente in uscita al variare della temperatura con il guadagno appena calcolato.

Se T= 25°C , I_B=2pA e Vos=800μV allora la tensione di errore corrisponde a :

$$V\varepsilon = R2 \cdot I_B + Vos\left(1 + \frac{R_2}{R_1}\right) = 312,5 \cdot 2pA + 800\mu V\left(1 + \frac{312,5}{80M}\right) = 625pV + 800uV \cong Vos$$

ed è approssimabile al valore della tensione di offset.

Se T= 50°C , I_B=10pA e Vos=850μV allora la tensione di errore corrisponde a :

$$V\varepsilon = R2 \cdot I_B + Vos\left(1 + \frac{R_2}{R_1}\right) = 312{,}5 \cdot 10\,pA + 850\mu V\left(1 + \frac{312{,}5}{15M}\right) = 3{,}125nV + 850uV \cong Vos$$

che è nuovamente approssimabile con il valore della tensione di offset ed è inferiore alla risoluzione del convertitore ADC che è di 10 mV. E' immediato osservare come una bassa corrente di polarizzazione ed un basso guadagno (R_2) fanno si che il termine di errore dovuto a questo contribuito sia ridotto al minimo e possa essere trascurato rispetto alla contributo dovuto a Vos. La tensione di offset può essere considerata con buona approssimazione indipendente dalla temperatura, data la bassa deriva termica (2 μV/°C) ed è inoltre cancellabile attraverso un partitore resistivo collegato al terminale non invertente. Si conclude quindi che il contributo di errore sulla tensione in uscita dovuto alla I_B e alla Vos è da considerarsi trascurabile ed eliminabile.
La cancellazione dell'offset può essere fatta, come già detto, attraverso un circuito esterno o attraverso i pin di offset come nel circuito di figura 2.14

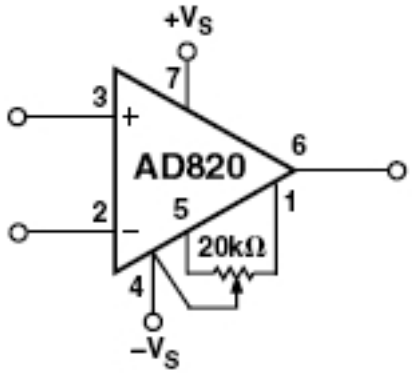


Figura 2.14 Pinout dell' AD820 (DIP package) e circuito per l'annullamento della tensione di offset.

Questo tipo di soluzione non è sempre attuabile. I pin 1 e 5 dell' AD820 non sono disponibile nel package SO-8 (risultano non connessi) ma solo in quello DIP. Inoltre questa configurazione comporta una deriva termica di 4 μV/°C sulla tensione di annullamento dell'offset il che peggiora le prestazioni del sistema. E' quindi più opportuno, anche per motivi di *layout* che andremo a studiare più avanti, adoperare un circuito di regolazione dell'offset esterno.

Andiamo ora ad analizzare quanto influisca, al variare della frequenza, l'errore di tensione dovuto alla Vos supponendo che C2 =0 (circuito aperto) e che l'amplificatore sia ideale (Zin=∝, Zout=0, Avol=∝). La funzione di trasferimento è quella di un amplificatore non invertente quindi:

$$H(s)=\frac{Vout}{Vos}=1+\frac{R_2}{\frac{R_1}{1+sC_1R_1}}=1+\frac{(1+sC_1R_1)R_2}{R_1}=\frac{R_1+R_2+sC_1R_1R_2}{R_1}=\frac{C_1R_1R_2\left(s+\frac{R_1+R_2}{C_1R_1R_2}\right)}{R_1}$$

posto: $\omega_0=\frac{1}{C_1(R_1//R_2)}$ allora la funzione di trasferimento diventa:

$$H(s)=\frac{Vout}{Vos}=\left(1+\frac{R_2}{R_1}\right)\left(\frac{s}{\omega_0}+1\right)$$

Fino alla frequenza di zero: f_0 , il guadagno del contributo dovuto alla tensione di offset è costante e pari a $1+R_2/R_1$ dopodichè aumenta all'aumentare della frequenza come illustrato nel diagramma di Bode di figura 2.14.

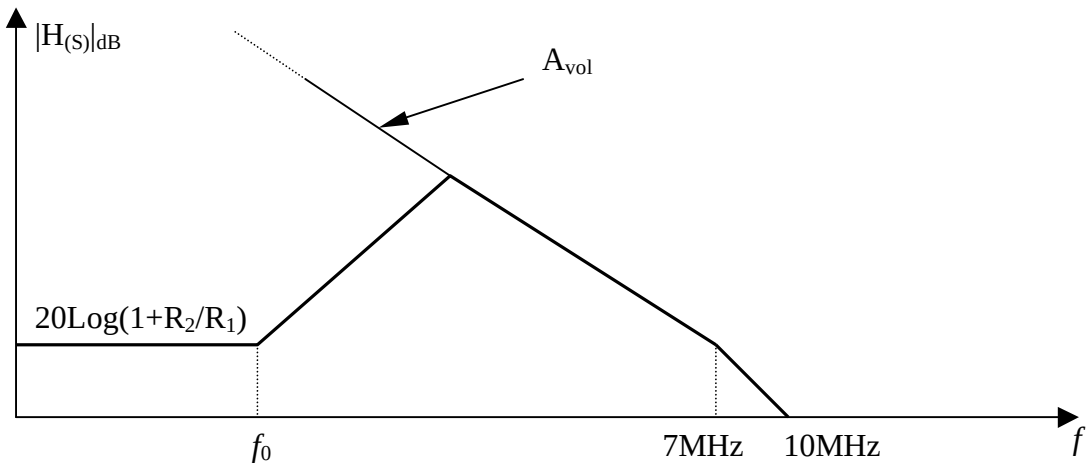


Figura 2.14: Diagramma qualitativo di Bode del modulo della funzione di trasferimento V_{out}/V_{os}

Va considerato che la frequenza di zero f_0 varia con la temperatura, poiché R1 si dimezza ad ogni incremento di temperatura pari a 10°C e varia in maniera proporzionale. Questo effetto però è trascurabile poiché $R_2 << R_1$.

L'introduzione della capacità C_2 modifica la funzione di trasferimento introducendo un polo:

$$H(s) = \frac{Vout}{Vos} = 1 + \frac{\frac{R_2}{1 + sC_2R_2}}{\frac{R_1}{1 + sC_1R_1}} = 1 + \frac{(1 + sC_1R_1)R_2}{(1 + sC_2R_2)R_1} = \frac{R_1 + R_2 + sC_2R_1R_2 + sC_1R_1R_2}{R_1 + sC_2R_1R_2} =$$

$$= \frac{(C_1 + C_2)R_1R_2\left(s + \frac{R_1 + R_2}{(C_1 + C_2)R_1R_2}\right)}{C_2R_1R_2\left(s + \frac{1}{C_2R_2}\right)}$$

posto: $\omega_0 = \frac{1}{(C_1 + C_2)(R_1 // R_2)}$ e $\omega_p = \frac{1}{C_2R_2}$ allora la funzione di trasferimento diventa:

$$H(s) = \frac{Vout}{Vos} = \left(1 + \frac{R_2}{R_1}\right)\frac{\left(\frac{s}{\omega_0} + 1\right)}{\left(\frac{s}{\omega_p} + 1\right)}$$

il guadagno a centro banda è :

$$H_{CB} = \frac{\left(1 + \frac{R_2}{R_1}\right)\omega_p}{\omega_0} = 1 + \frac{C_1}{C_2}$$

mentre quello a frequenze inferiori allo zero:

$$H_0 = \left(1 + \frac{R_2}{R_1}\right)$$

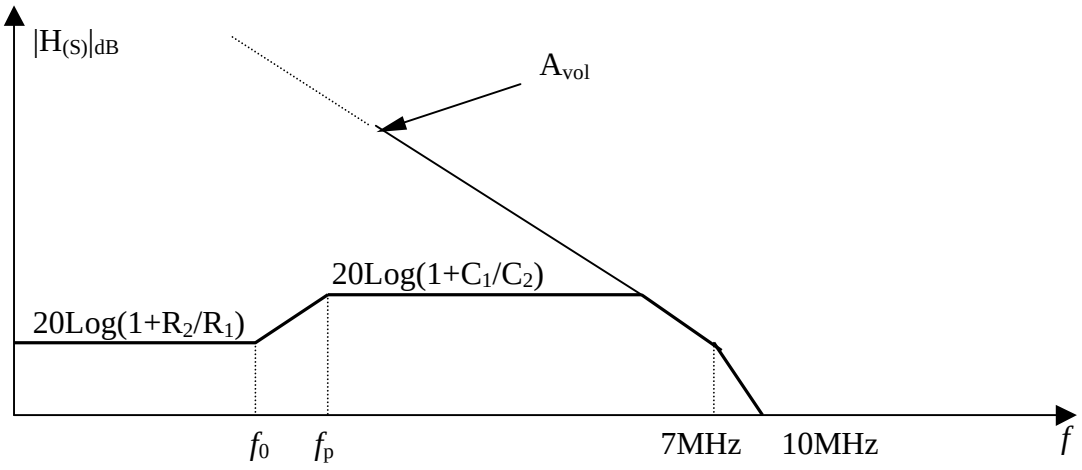


Figura 2.15: Diagramma qualitativo di Bode del modulo della funzione di trasferimento V_{out}/V_{os} dopo aver inserito il condensatore C_2.

La capacità C_2 permette quindi di limitare ad un guadagno pari a $1+C_1/C_2$ il termine di errore introdotto dalla tensione di offset.
Nel nostro caso essendo $R_2=312\Omega$ e $C_2=1nF$

$$f_0 = \frac{1}{2\pi(C_1 + C_2)(R_1 // R_2)} \cong 364KHz$$

$$f_p = \frac{1}{2\pi C_2 R_2} \cong 510KHz$$

i guadagni invece saranno:

$$H_0 = 1 + \frac{R_2}{R_1} \cong 1$$

$$H_{CB} = 1 + \frac{C_1}{C_2} \cong 1,4$$

a centro banda il contributo di tensione dovuto alla tensione di offset è pari a:

$$V\varepsilon_{MAX} = H_{CB} \cdot Vos = 1,4 \cdot 850\mu V \cong 1,2mV$$

Si può concludere quindi che la capacità C_2=1nF limita l'errore di conversione corrente tensione molto al di sotto della risoluzione dell'ADC (10mV).

In cascata all'amplificatore di transimpedenza c'è un buffer in modo da minimizzare l'impedenza di uscita del trasduttore e poter pilotare adeguatamente carichi capacitivi elevati. E' stato scelto il **BUF04** [8] della Analog Device particolarmente adatto in applicazioni ad alta velocità e basso consumo energetico; applicazioni audio ad alta fedeltà e interfaccia con convertitori AD e DA ad alta velocità. Il buffer in questione infatti consuma meno di 8,5mA, può essere alimentato con tensione compresa fra ±5V e ±15V , ha uno slew rate tipico di 3000V/μs ed una banda di 100MHz.

Nonostante la bassa tensione di offset del dispositivo, circa 0.6mV, è opportuno che questo sia provvisto di un circuito esterno di annullamento dell'offset e riportato nello schema completo del trasduttore.

I relativi diagrammi di Bode della funzione di trasferimento *Vout/Iph* sono riportati in figura 2.16. Si nota come il sistema sia caratterizzato da una banda piatta per frequenze comprese tra 0 e 100KHz ed il guadagno sia pari a circa 50db . Le stesse caratteristiche si hanno all'uscita del buffer BUF04 (fig. 2.17).

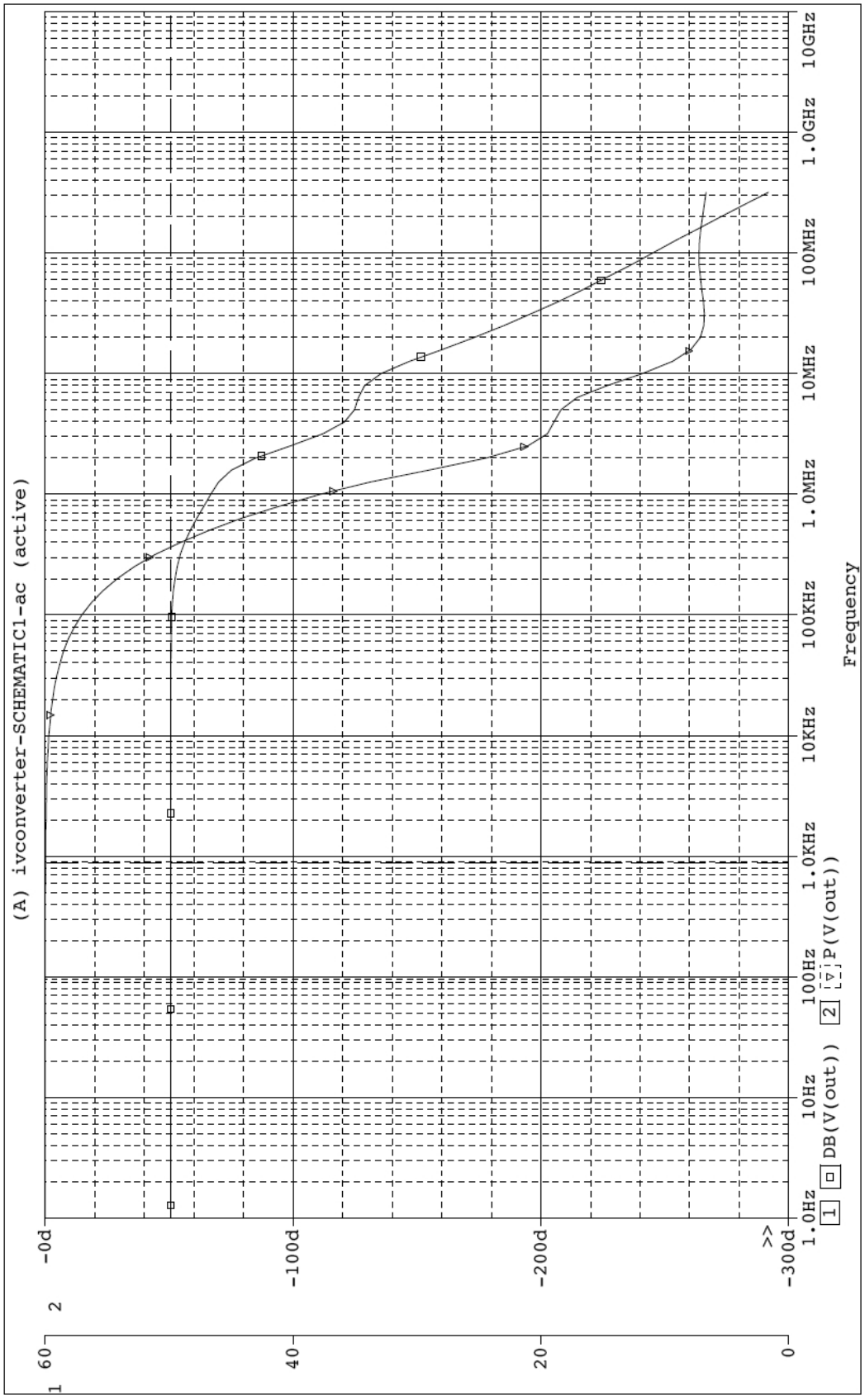


Figura 2.16 : Risposta del sistema con guadagno pari a 50dB : 20Log(312)

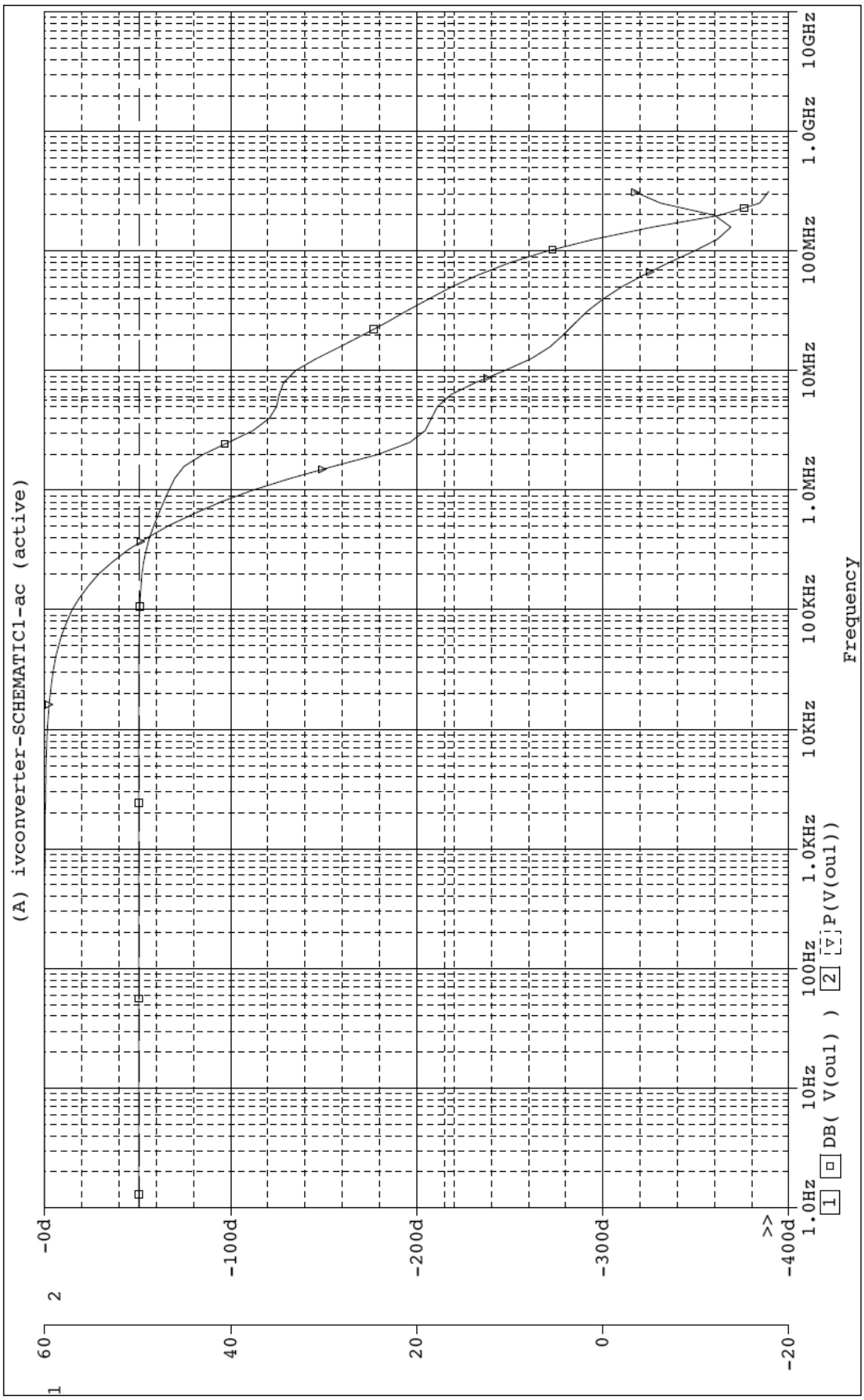


Figura 2.17 : Risposta del sistema con il BUF04 e guadagno pari a 50dB : 20Log(312)

E’ opportuno osservare che poiché non si ha la certezza assoluta sul valore della *responsivity* sarò necessario variare il guadagno dell’amplificatore di transimpedenza. Supponiamo che il valore della responsivity non sia quello stimato (0,4 A/W) ma sia di 0,25 A/W allora il guadagno dovrà essere pari a 500. Un aumento del guadagno, come si può vedere dalle simulazioni effettuate, influisce poco sulla banda dell’amplificatore riducendola da 100KHz a 70KHz. Le simulazioni in questione sono riportate in figura 2.18 e figura 2.19. Nonostante le specifiche relative alla banda siano ampiamente rispettate, diminuendo il valore di C2, portandolo da 1nF a 220pF , aumenta la banda a scapito della tensione di errore introdotta da Vos che comunque rimane abbondantemente al di sotto della risoluzone dell’ ADC poiché è pari a:

$$V\varepsilon_{MAX} = H_{CB} \cdot Vos = \left(1 + \frac{C_1}{C_2}\right) \cdot Vos = \left(1 + \frac{220p}{400p}\right) \cdot 850\mu V \cong 2{,}4mV$$

La risposta del sistema se il condensatore C2=220pF ed R2=500 è riportata in figura 2.20.

Durante la fase di colluado del trasduttore ci siamo accorti che la responsivity dichiarata per il fotodiodo non è 0,4 [A/W] ma circa 0,25 [A/W]. E’ stato quindi modificato il gaudagno portandolo da 312 a 500 ma lasciando la capacità C2=1nF.

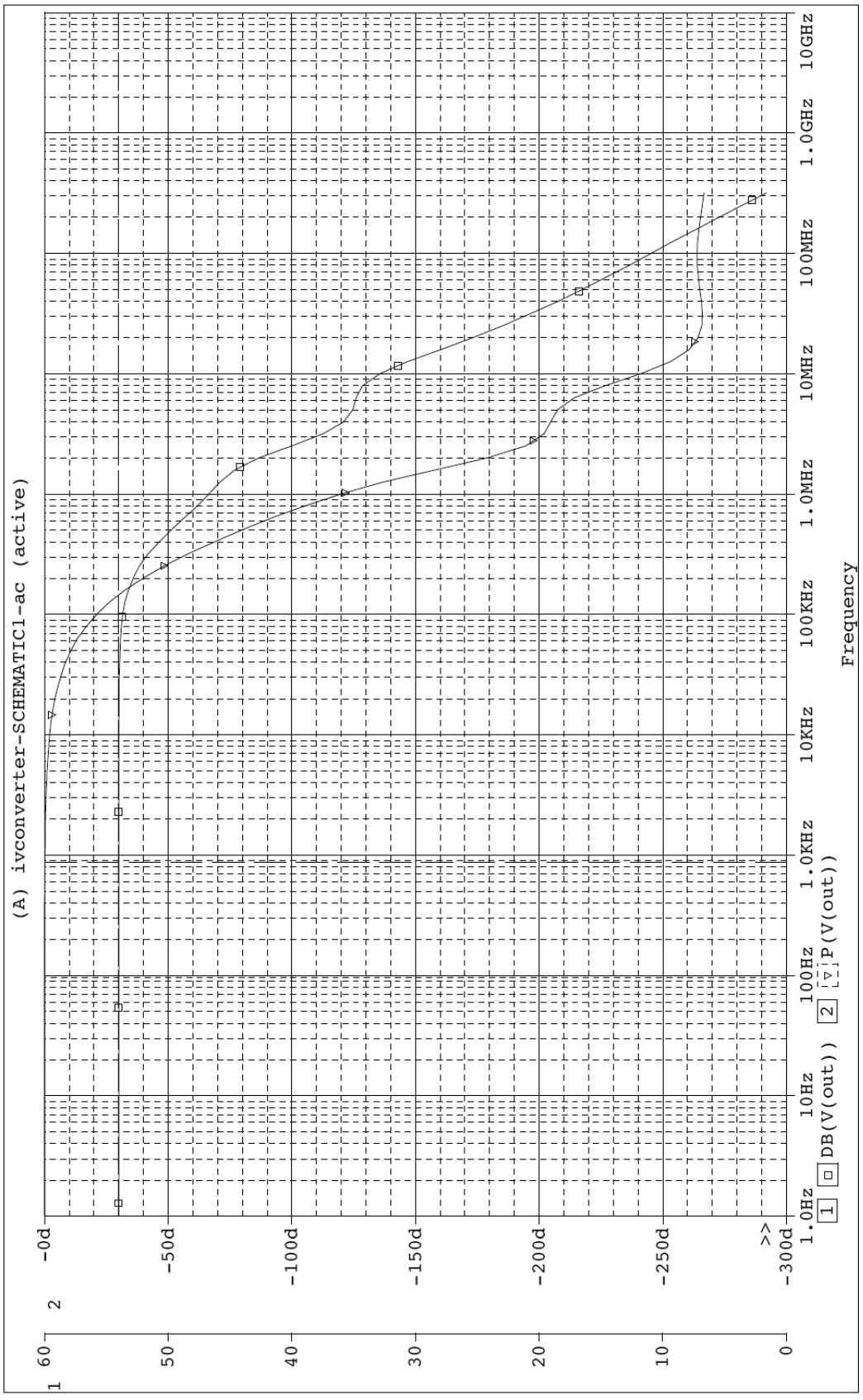


Figura 2.18 : Risposta del sistema con guadagno pari a 54dB : 20Log(500)

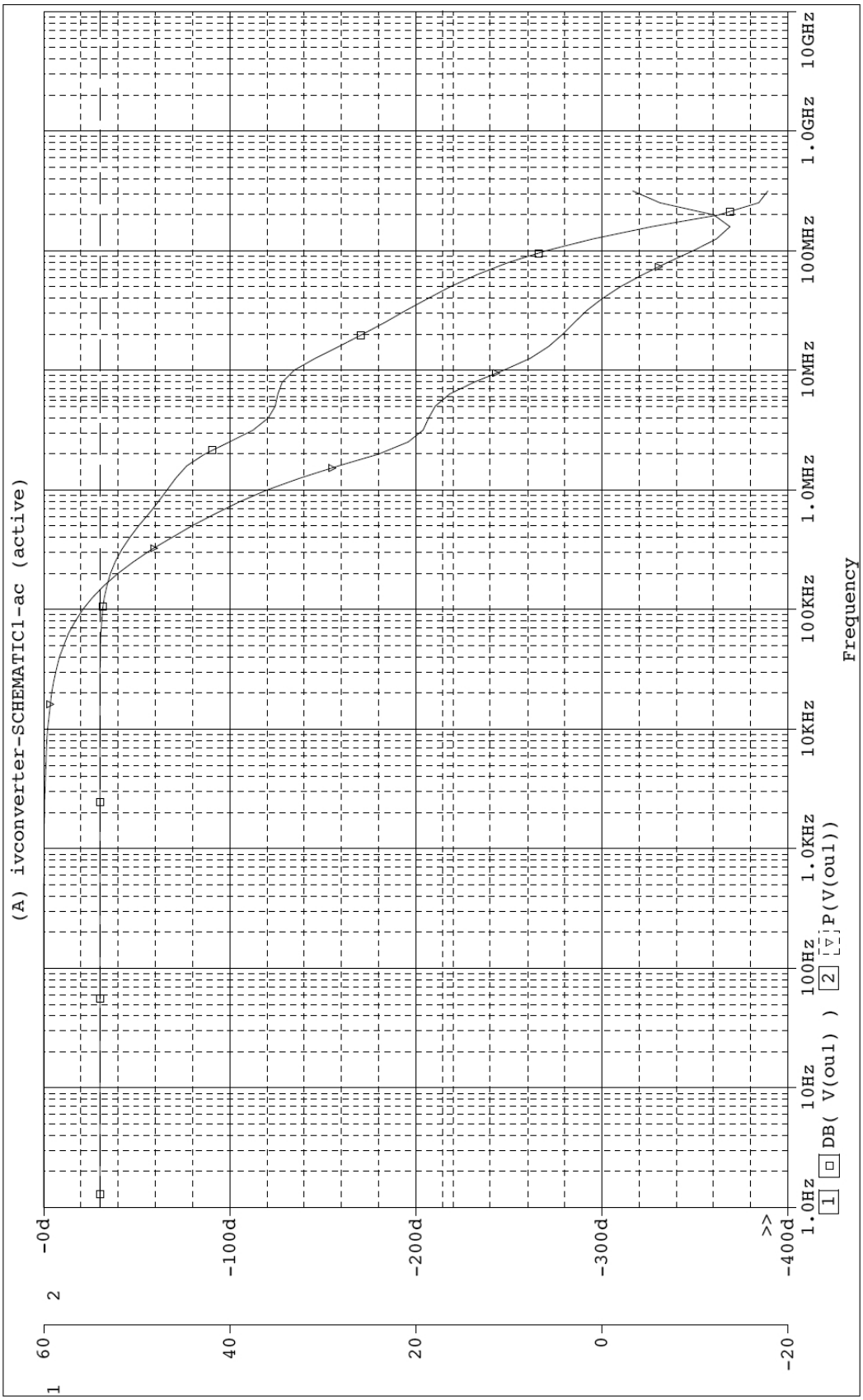


Figura 2.19 : Risposta del sistema con il BUF04 e guadagno pari a 54dB : 20Log(500)

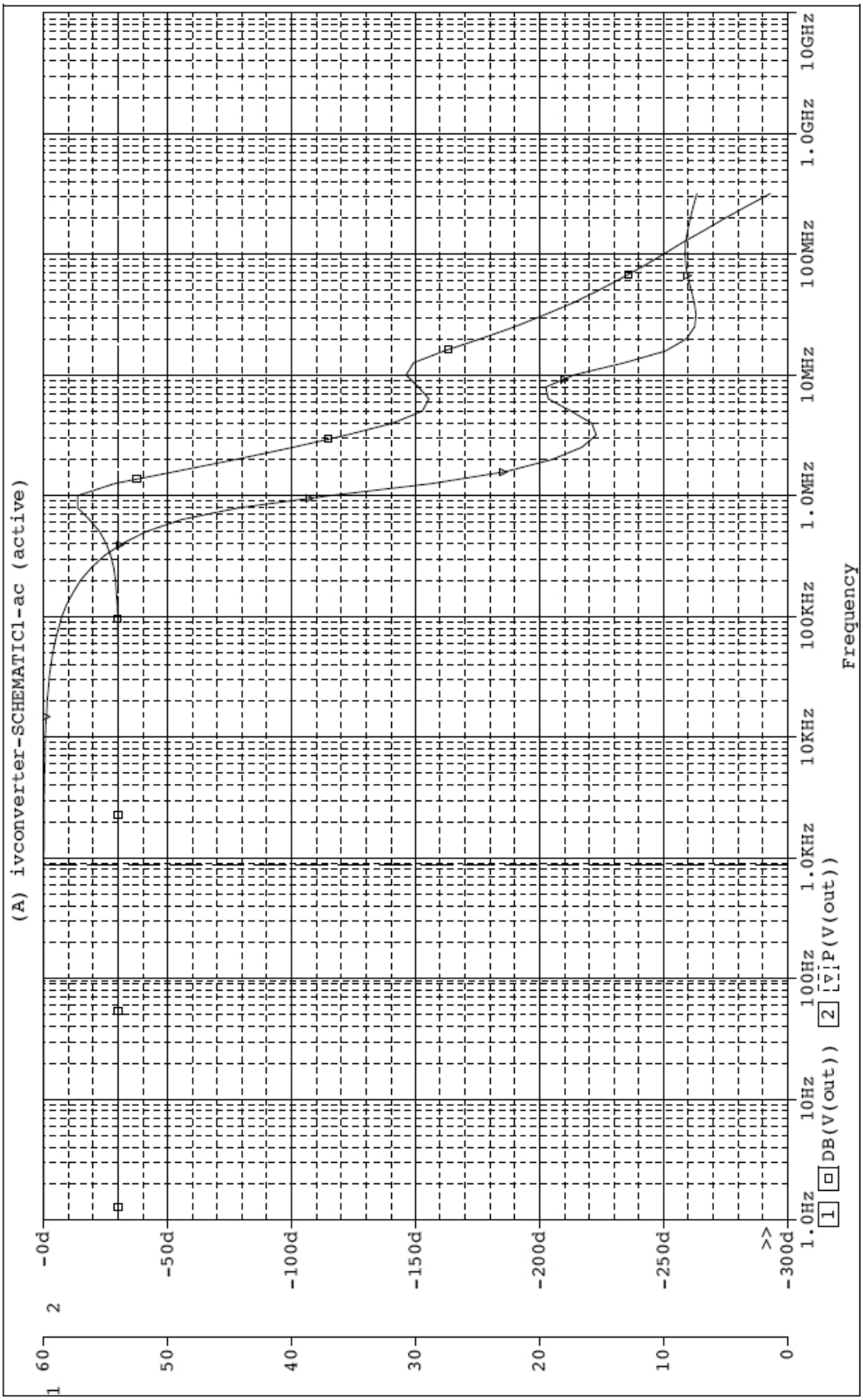


Figura 2.20 : Risposta del sistema con guadagno pari a 54dB e C2=220pF. Si noti come la banda del sistema è aumentata, passando da 70KHz a 100KHz.

2.6 Layout del trasduttore

Lo schema elettrico usato per lo sbroglio della scheda è riportato in figura 2.24.
Tutti i componenti eccetto i circuiti integrati U1 e U2 sono *smd (sourface mount device)* in modo da facilitare lo sbroglio delle connessioni ed ottimizzare le dimensioni della scheda. Per semplificare inoltre la saldatura dei componenti sulla scheda si è deciso di usare componenti non troppo piccoli e quindi **smd-1206** (12milsx06mils); ricordarsi che **1inch=25,4mm** mentre **10mils=0,10inch=2,54mm.** Durante la fase di layout del PCB (Printed Circuit Board) è necessario pensarlo come un componente stesso del progetto e prototipizzarlo il più possibile attraverso *test points, jumper* ,connettori di espansione, resistenze e capacità aggiuntive. C1_1 e R2 sono componenti che non è necessario montare sulla scheda ma che potrebbero essere necessari in futuro per modificare il guadagno e/o la banda dell'amplificatore.
Ogni ramo di alimentazione ,Vs+ e Vs-, viene filtrato dal ripple con un condensatore al tantalio da 10uF e bassissimo ESR=0,5Ω (*Equivalent Series Resistance*). In parallelo sono posti 2 condensatori ceramici da100nF ,1 da 10nF ed uno da 1uF in modo da eliminare il rumore ad alta frequenza ed diminuire l'ESR totale.
Le resistenze R3,R4,R5 e la capacità C4 costituiscono il circuito per l'annullamento dell'offset dell'amplificatore transresistivo. Il partitore di tensione così dimensionato permette di generare una tensione di offset compresa tra ±1,2mV. La resistenza R6 invece serve per annullare l'offset di tensione introdotto dal buffer BUF04.
Particolare attenzione va posta alla corrente indotta da accoppiamenti capacitivi ed induttivi sul collegamento tra il catodo e il terminale invertente dell'amplificatore. Tale corrente infatti diventerebbe fonte di errore poiché si sommerebbe alla corrente generata dal fotodiodo. E' opportuno quindi in fase di layout ridurre al minimo tale contributo.
Vedremo la causa di questi fenomeni e come è possibile ridurli.
I disturbi che nascono per accoppiamento induttivo sono dovuti alla circolazione di correnti. Si considerino pertanto due conduttori come rappresentato nello schema di figura 2.21. Il primo, detto circuito disturbante, è percorso dalla corrente I_1; il secondo, detto circuito disturbato, trasferisce il segnale di misura V_s allo stadio di ingresso R_{in} di uno strumento. La corrente I_1 circolante nel conduttore "1" determina un flusso Φ che si

concatena anche con il conduttore "2". La mutua induttanza M_{12} presente fra i due conduttori è responsabile dell'accoppiamento induttivo.

La tensione di disturbo V_n che risulta in serie con il segnale utile V_s è data da:

$Vn = j\omega M_{12} I_1$

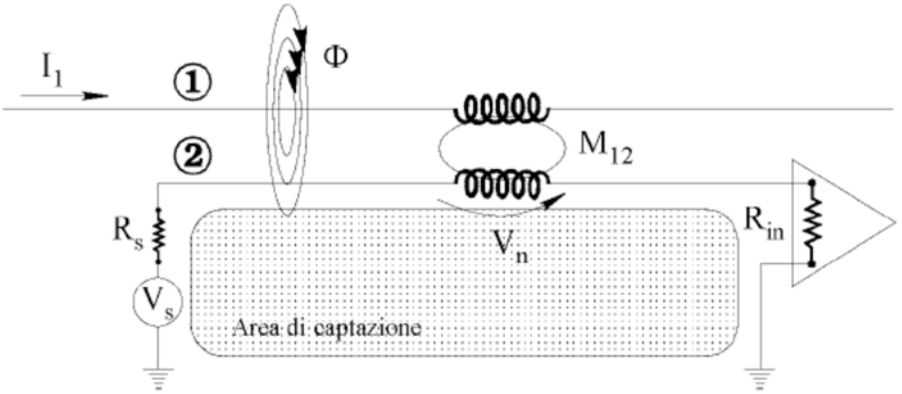


Figura 2.21: Esempio di accoppiamento induttivo fra due conduttori (1) e (2)

Per ridurre l'entità del disturbo dovuto all'accoppiamento induttivo, bisogna ridurre la mutua induzione. Questo si può ottenere ponendo i due circuiti (1 e 2) il più possibile distanti e perpendicolari. Inoltre è opportuno disporre il conduttore attivo del circuito disturbato il più possibile vicino al suo ritorno (sia esso il piano di massa, oppure la pista di un circuito stampato o un semplice cavo, purchè siano uguali le correnti di andata e ritorno): in tal modo si riduce l'area della spira con la quale si concatena il flusso magnetico prodotto dalla corrente I_1.

Consideriamo ora l'accoppiamento di tipo capacitivo (figura 2.22.a) Il circuito equivalente è quello mostrato in figura 2.22b, dove i valori degli elementi sono stati ottenuti moltiplicando le capacità per unità di lunghezza per la lunghezza della linea: $C_{RS}=C_{RS}L$ e $C_{GS}=C_{GS}L$.

L'accoppiamento capacitivo è

$$V_{NE}^{cap} = V_{FE}^{cap} = V_G \frac{j\omega RC_{RS} \parallel C_{GS}}{1 + j\omega RC_{RS} \parallel C_{GS}}$$

dove

$$R = \frac{R_{NE}R_{FE}}{R_{NE} + R_{FE}} \qquad C_{RS} \| C_{GS} = \frac{C_{RS}C_{GS}}{C_{RS} + C_{GS}}$$

A frequenza sufficientemente bassa, nell'espressione precedente si considera solo il termine a numeratore per cui

$$V_{NE}^{cap} = V_{FE}^{cap} \cong V_G j\omega \frac{R_{NE}R_{FE}}{R_{NE} + R_{FE}} \frac{C_{RS}C_{GS}}{C_{RS} + C_{GS}}$$

con

$$V_G = \frac{R_L V_S}{R_S + R_L}$$

Il contributo di accoppiamento capacitivo aumenta di 20 dB/dec come nel caso non schermato. Quindi lo schermo praticamente non ha effetto. Per rendere attivo l'effetto schermante occorre connettere a massa lo schermo. In questo caso la tensione dello schermo viene annullata per cui:

$$V_{NE}^{cap} = V_{FE}^{cap} = 0$$

Da un punto di vista fisico questo vuol dire che le linee del campo elettrico prodotto dal filo generatore terminano sullo schermo e non sul filo del ricevitore. Per far avvenire ciò occorre che la tensione dello schermo V_{Sh} sia nulla. Nel modello che abbiamo sviluppato è sufficiente connettere a terra lo schermo in un punto qualsiasi. Quando tuttavia la linea non è elettricamente corta, occorre assicurare che la tensione dello schermo sia nulla per tutta la sua lunghezza, per cui è buona norma connettere a terra lo schermo in più punti distanziati di circa $\lambda/10$.

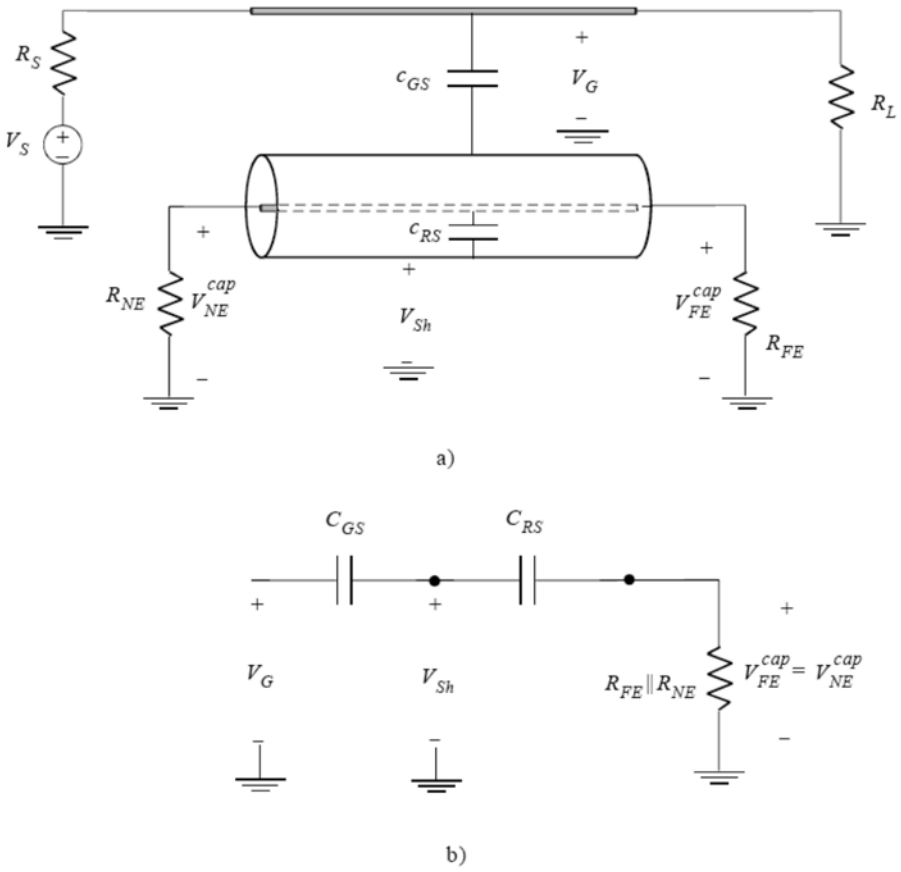


Figura 2.22 : Accoppiamento capacitivo fra due conduttori (a), schema equivalente del problema(b).

Sarà quindi opportuno schermare il piedino invertente (pin2) da eventuali accoppiamenti capacitivi ed induttivi procurati ad esempio dal pin1. Il pin1 se si adotta la versione smd dell' AD820 non è collegato quindi l'accoppiamento capacitivo tra il pin1 e il pin2 sarebbe ridotto al minimo. Per collaudare, cablare e sostituire l'amplificatore in maniera semplice però è opportuno adoperare la versione con package DIP dell'AD820. In questa versione il pin1 viene adoperato assieme al pin5 per l'annullamento dell'offset. Il pin1 però rimane flottante, dato che viene usato un circuito di annullamento dell'offset esterno. Risulterà quindi opportuno schermare il pin2 da eventuali accoppiamenti capacitivi attraverso la cosiddetta *guard techniques* [6] che prevede di interporre fra il potenziale che genera accoppiamento capacitivo e la pista soggetta al disturbo uno schermo che annulla l'accoppiamento capacitivo (figura 2.22b). Dal punto di vista pratico si tratta di circondare completamente il pin 2 con una pista di massa. Tale "anello di guardia" deve essere presente sia sul *Top* che sul *Bottom layer*. I

due anelli devono essere poi collegati tra loro attraverso più *vias* in modo da rendere più efficace lo schermo.

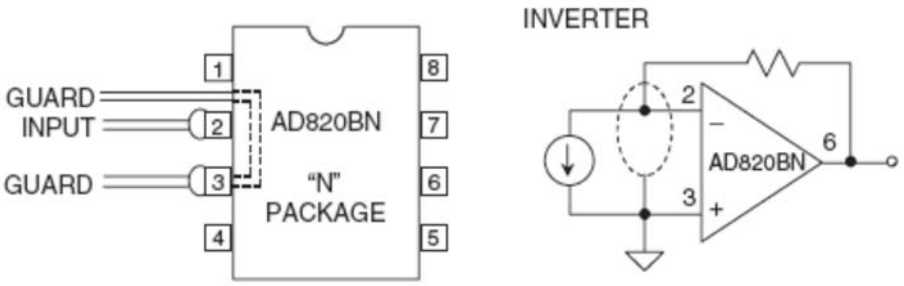


Figura 2.23 : Posizionamento dell' "anello di guardia" attorno al pin 2 dell'amplificatore invertente.

Le piste che realizzano i due anelli di guardia devono avere uno spessore sufficiente per poter inserire le *vias*. Queste non sono riportate nel layout finale del dispositivo ma sono state realizzate manualmente facendo piccoli fori ed usando del filo da *wire wrap* saldato sul *top* e sul *bottom layer*. Si è usata questa tecnica per realizzare tutte le *vias* presenti nel circuito stampato poiché questo è stato realizzato tramite la tecnica della fotoincisione e non permette quindi di realizzare *vias* metallizzate. L'anello di guardia sul *bottom layer* non è completamente chiuso ma l'efficacia dello schermo rimane pressoché invariata dato che il collegamento da schermare è sul *top layer* e qui l'anello di guardia è completamente chiuso. L'unico punto in cui i due anelli non sono collegati tra loro tramite *vias* è limitato alla zona tra il pin 1 ed il pin 2 dove l'anello di guardia inferiore rimane aperto e dove frall'altro non c'è lo spazio per effettuare questo tipo di operazione manualmente. Sarà possibile chiudere l'anello solo realizzando il PCB attraverso un processo industriale.

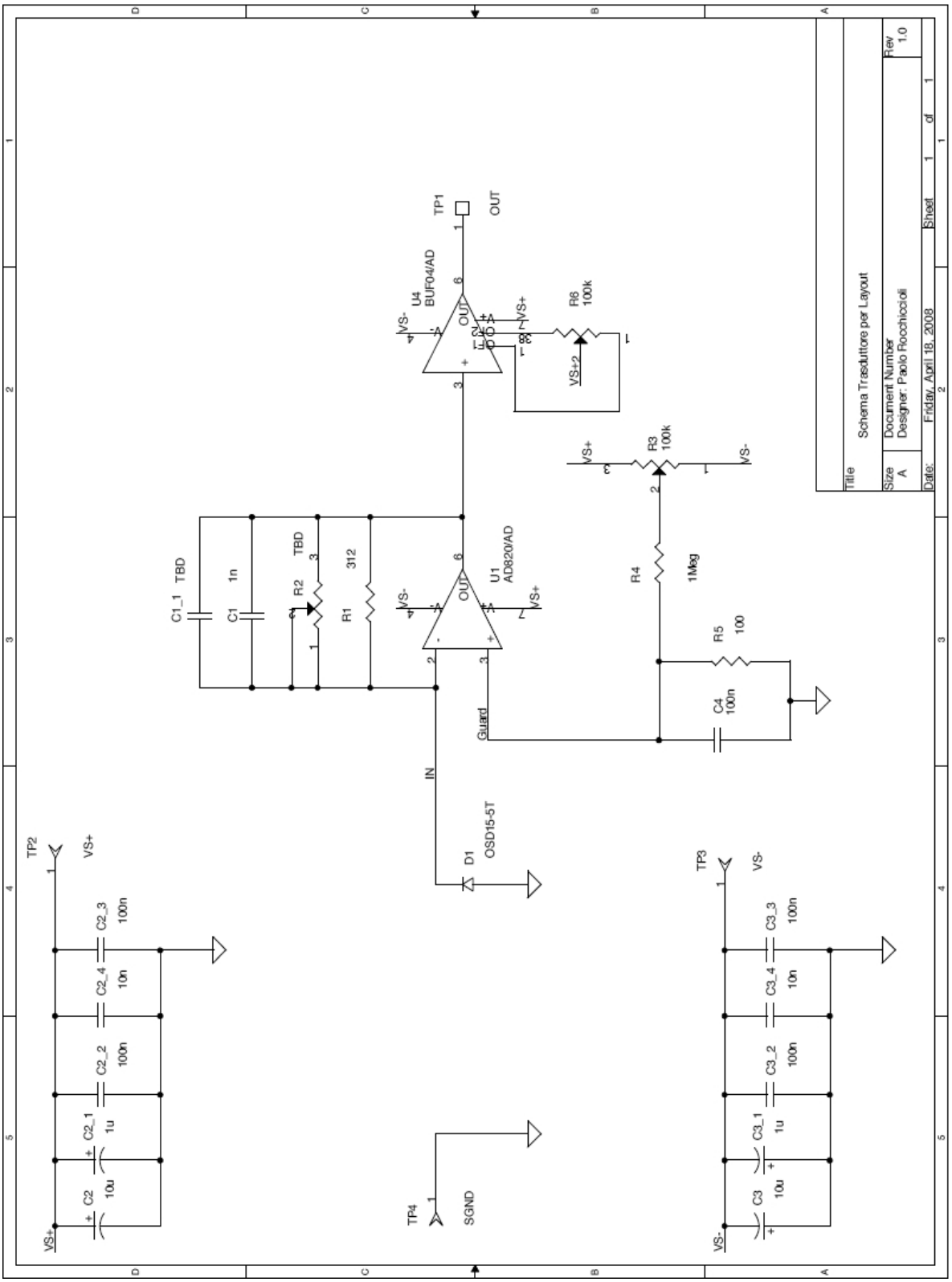


*Figura 2.24 Schema elettrico del trasdutt*ore

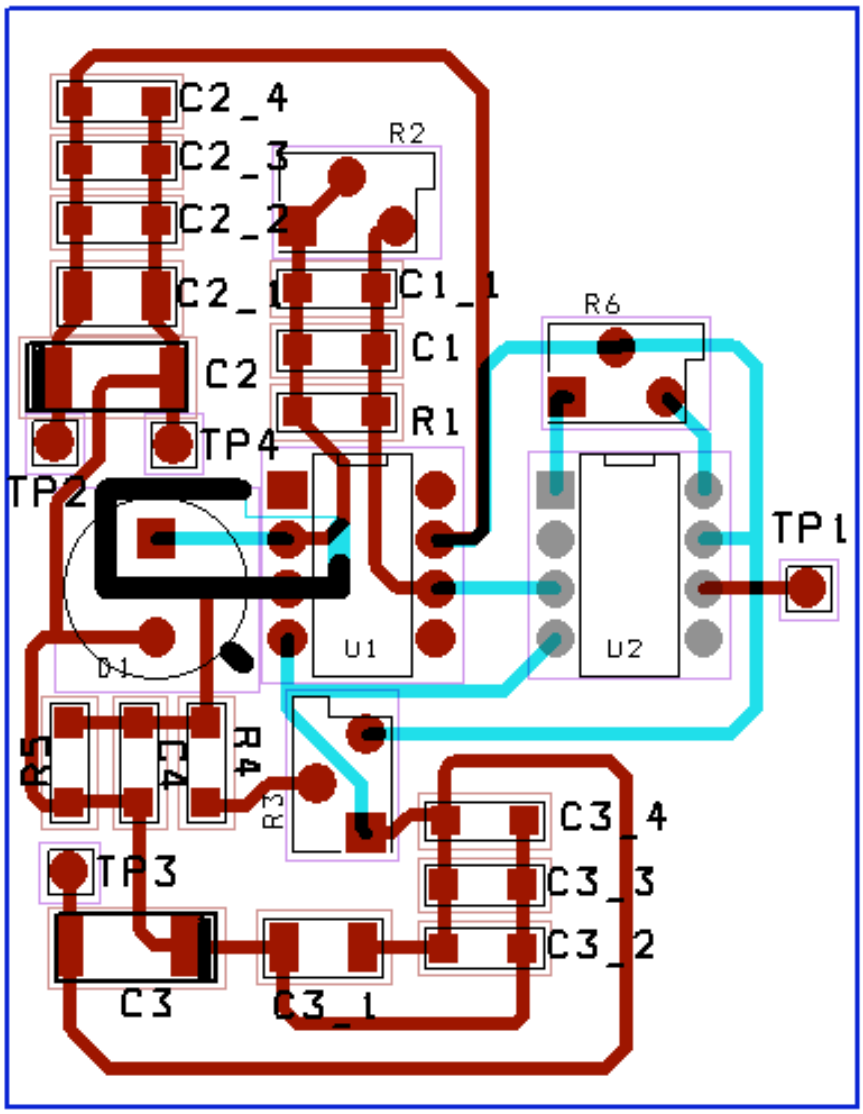


Figura 2.25: Layout del circuito stampato. In celeste sono riportate le connessioni sul Top Layer mentre in rosso quelle sul Bottom Layer

Sullo schema elettrico non sono riportate le connessioni delle alimentazioni e quella del segnale di uscita. Queste sono state cablate direttamente sulla scatola che racchiude il trasduttore. Tali connessioni sono riportate nella Foto1. In serie all'alimentazione è posto un interruttore tripolare che permette di accendere e spegnere il dispositivo. Su ogni alimentazione è posto poi un led di stato, con relativa resistenza di polarizzazione, che indica il corretto funzionamento delle alimentazioni. I due alimentatori da 12V sono infine collegati in serie ed il comune fra i due alimentatori è usato come massa per il dispositivo, si genera così un'alimentazione ±12V. Tale massa sarà presente anche sul

connettore di uscita e dovrà essere collegata alla massa del DAQ che è la terra del calcolatore.

Per completare non rimane che compensare l'offset. E' necessario alimentare il circuito, coprire il fotodiodo con un panno scuro, in modo da annullare la corrente fotogenerata e ruotare R3 finché la tensione sul pin 6 di U1 non si annulla. A questo punto è necessario compiere la stessa operazione su R6 in modo che la tensione di uscita, prelevata dal pin 6 di U2, si annulli.

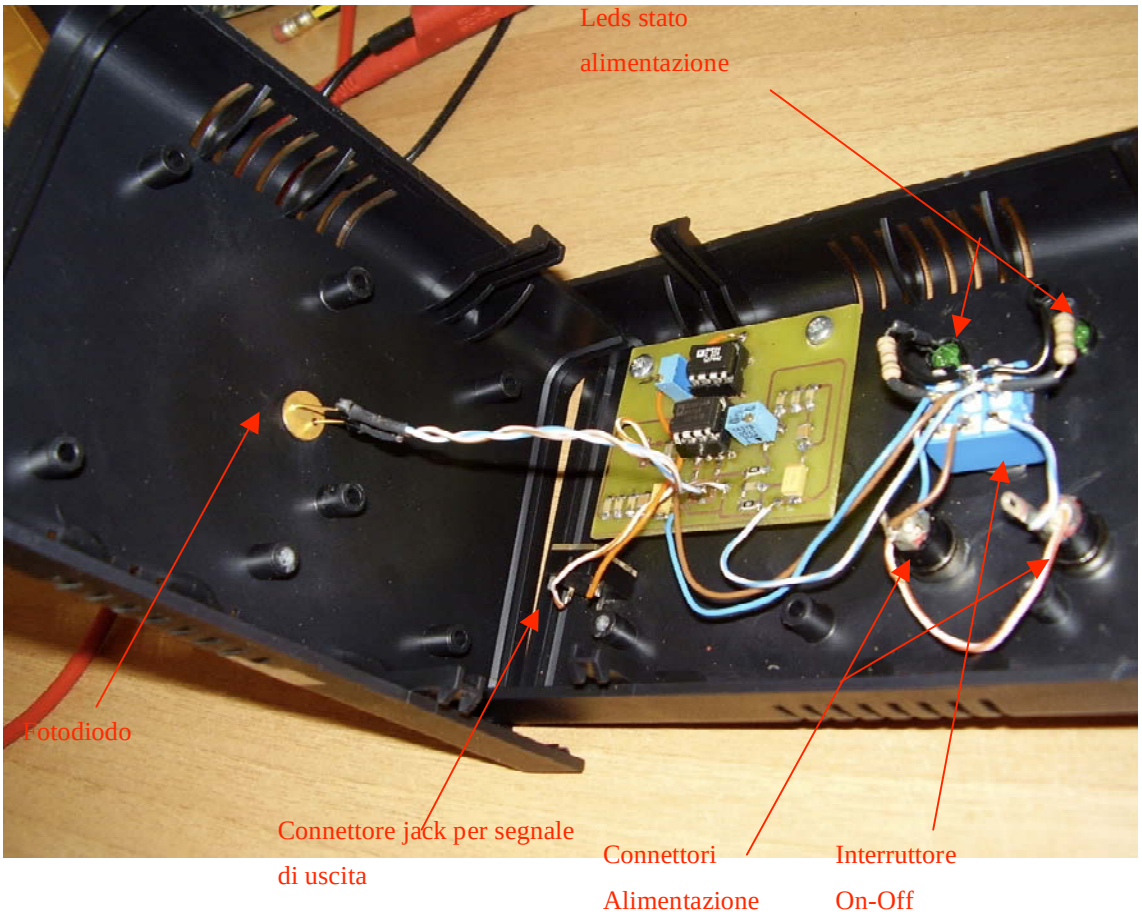


Foto1: Vista dei collegamenti interni al trasduttottore

Foto2: Vista della parte posteriore del trasduttore inscatolato.

Foto3: Vista frontale del trasduttore inscatolato

Capitolo 3

Il protocollo USB

3.1 Introduzione

Quanto di seguito non ha le pretese di esporre nei minimi particolari il protocollo USB ma di fornire i requisiti necessari alla comprensione e alla realizzazione del dispositivo in questione, che per le scelte progettuali effettuate, non richiede, per l'appunto, uno conoscenza nel dettaglio di tale protocollo. La porta **Universal Serial Bus (USB)** si trova ormai su tutti i calcolatori. Sono disponibili due versioni di USB**: USB 1.0/1.1 e USB 2.0**. [9]. Le periferiche si possono collegare in una configurazione a catena, ad albero o a stella, che ha origine da una o più porte di I/O su computer. Le periferiche vengono rilevate e configurate al momento del loro collegamento (Plug and Play) e possono essere aggiunte o rimosse senza che sia necessario spegnere e riavviare il computer.

I dispositivi possono essere collegati in qualsiasi ordine, è possibile ad esempio collegare la tastiera e la stampante al computer, il mouse e il joystick alla tastiera e collegare il modem e lo scanner attraverso la stampante. Lo standard USB 2.0 è quello più aggiornato e ha iniziato ad apparire nei nuovi prodotti nel 2001. Per usare questo standard, i computer e le periferiche devono essere compatibili USB 2.0.

Specifiche tecniche standard **USB1.0/1.1:**

- Velocità di trasmissione dati: 1.5Mbps USB 1.0
- Velocità di trasmissione dati: 12Mbps USB 1.1
- 127 dispositivi collegabili
- Capacità di collegamento a “caldo”
- Trasferimento dati isocrono e asincrono
- Lunghezza massima del collegamento: 5mt

Specifiche tecniche standard **USB2.0:**

- Tutte le funzionalità comprese nello standard USB 1.0 e 1.1
- Completamente compatibile all'indietro con USB 1.0 e 1.1
- Disponibile in due versioni: USB e Hi-Speed USB
- Velocità di trasmissione pari a 12-Mbps per USB e 480 Mbps per Hi-Speed USB
- Tutte le periferiche funzionano alla loro velocità massima, invece che alla velocità della periferica più lenta
- Collaudato a un livello più alto rispetto allo standard USB 1.0/1.1 in modo da risultare più affidabile

L’elevata velocità di comunicazione, il basso costo di realizzazione, la semplicità d’uso e la versatilità rende la porta USB molto usata, tant’è che oggi giorno i computer desktop ne montano 5-6 mentre i laptop fino ad un massimo di 4. Dalla tabella 3.1 si può notare come, in assenza di una porta USB 2.0 Hi-Speed, solo il protocollo Firewire(IEEE 1394), garantisca una velocità di comunicazione superiore allo standard USB 2.0. E’ riportata anche la porta parallela oggi ormai assente su tutti i laptop, per motivi di spazio e consumo, e molto raramente presente su i dekstop.

Collegamento	Massima Velocità di trasferimento	Logo
Porta seriale	920 kbps	
Porta parallela (standard)	920 kbps	
USB 1.0	1.5 Mbps	
USB 1.1	12 Mbps	
USB2.0	12 Mbps	
Porta Parallela (ECP)	24 Mbps	
Firewire(IEEE 1394)	400 Mbps	
Hi-speed USB 2.0	480 Mbps	

Tabella 3.1: Tabella riasuntiva delle interfacce di un calcolatore

Tenendo conto che il numero massimo di dispositivi collegabili sono 127 e che il collegamento non deve superare i 5 metri, si possono collegare all'Host più dispositivi attraverso degli Hub su più livelli : vedi figura 3.1. A causa però dei ritardi introdotti dagli Hub e dal collegamento stesso il numero massimo di livelli è 7, incluso il livello host. Il settimo livello supporta un massimo di 5 hub (hub1,2,4,5,7 di fig. 3.1) nel percorso fra host e ogni dispositivo del livello 7. In questo livello possono essere abilitati solo i dispositivi (function) e non gli Hub. Notare che l'ultimo dispositivo collegato occupa 2 livelli, quindi non può essere collegato al 7 livello, eventualmente al 6 in modo da soddisfare quanto detto sopra. Quando un singolo dispositivo implementa più function e la funzione di hub questo viene chiamato *compound device*. Attraverso un'opportuna interfaccia è possibile selezionare, attraverso l'hub, quale function selezionare.

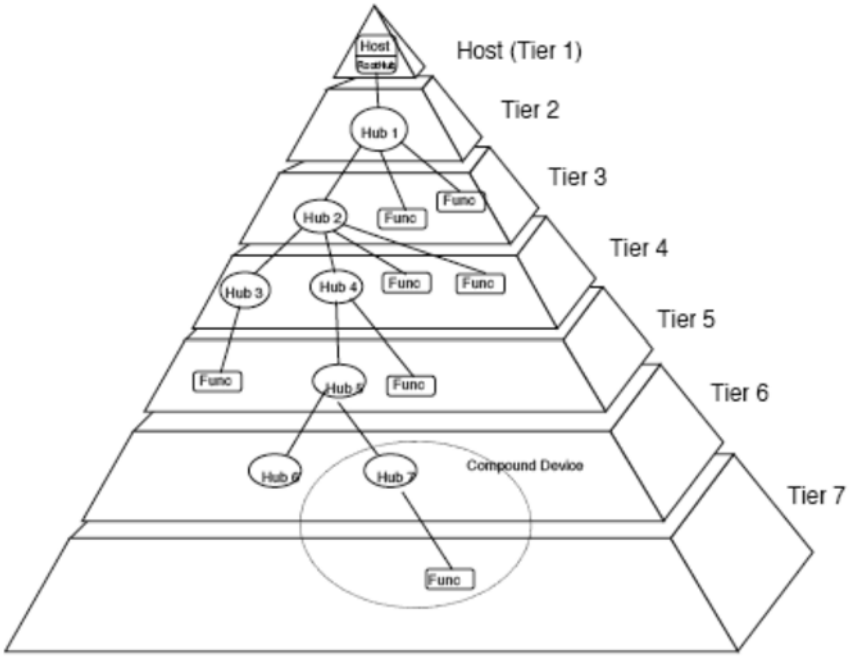


Figura 3.1: Struttura a livelli dei collegamenti tra host e function

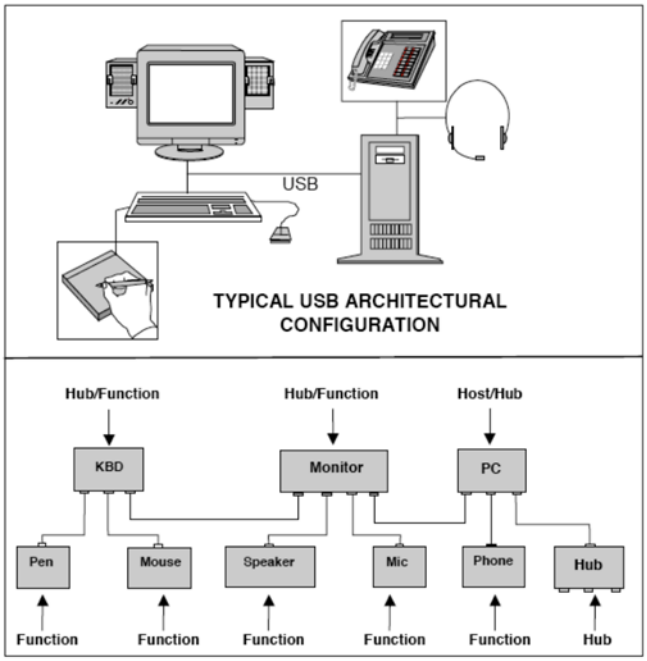


Figura 3.2: Esempio di una possibile configurazione dei collegamenti fra il PC e le varie periferiche.

3.2 Connettore e collegamento USB

Esistono due tipi di connettori : A e B usati secondo un preciso criterio.
Il ***connettore A*** di ***upstream*** (dal device) viene collegato sempre all'host mentre il ***connettore B*** di ***downstream*** (dall'host) viene collegato sempre al device. Il collegamento tra host e device varia a seconda della velocità di funzionamento della porta USB: Full-speed (12Mbps) o Low-speed (1,5Mbps) . Nel caso di Full-speed il collegamento richiede un cavo schermato 4 poli : 2 poli per l'alimentazione e massa (VBuse, GND) e un doppino intrecciato per i segnali D+ e D-. Nel caso di Low-speed può essere usato un semplice cavo a 4 poli non schermato.

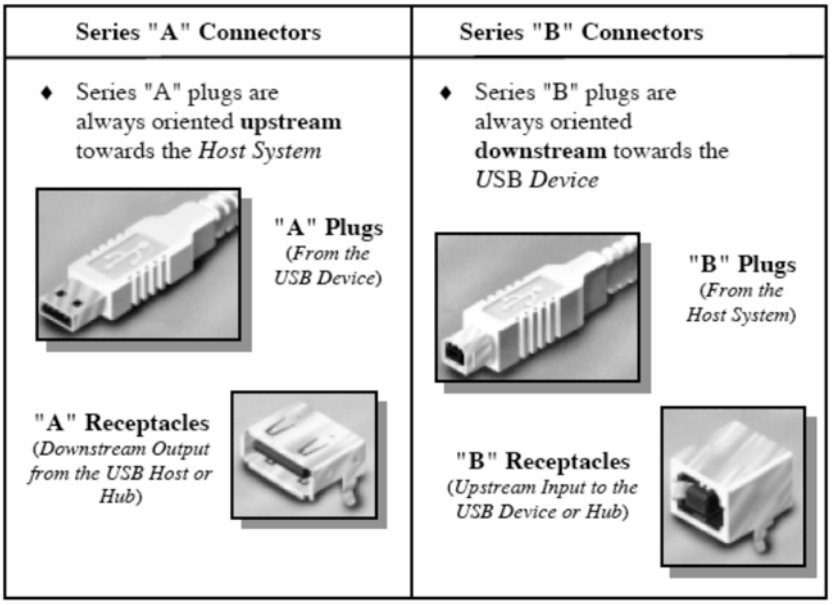
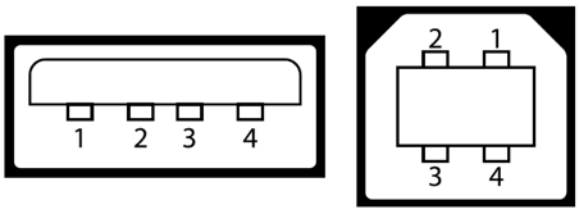


Figura 3.3 : Raffigurazione dei connettori (plugs e recptacles) A e B

I cavi pre-assemblati che si trovano in commercio , con connettore A e B sono cavi predisposti per il funzionamento full-speed. Possono essere quindi adoperati con qualsiasi porta USB. I collegamenti sono standardizzati secondo un particolare codice dei colori e piedinatura come riportato in figura 3.4



Pin	Nome	Colore
1	VBus	Rosso
2	D-	Bianco
3	D+	Verde
4	GND	Nero

Figura 3.4: Pinout del connettore A e B e codice a colori dei collegamenti

3.3 Classi dei dispositivi

Per evitare di usare un driver specifico per ogni dispositivo è stato deciso di definire delle classi di dispositivi che definiscono il comportamento del dispositivo e permettono attraverso un unico driver di pilotare più dispositivi. Il sistema operativo riconosce le classi di dispositivi e provvede a fornire drive generici per ogni classe di dispositivo.

Classe	Uso	Descrittore	Esempio
0x0	Device	Non Specificato	
0x1	Interface	Audio	Speaker, microphone, sound card
0x2	Both	Communication e CDC Control (Communication Device Class)	Ethernet adapter, modem, serial port adapter
0x3	Interface	Human Interface Device (HID)	Keyboard, mouse
0x5	Interface	Physical Interface Device (PID)	Joystick
0x6	Interface	Image	Digital camera
0x7	Interface	Printer	Laser printer, Inkjet printer
0x8	Interface	Mass Storage	USB flash drive, memory carc reader, digital audio player
0x9	Device	USB Hub	Full-speed Hi-speed Hub
0xA	Interface	CDC-Data	Come classe 0x2
0xB	Interface	Smart Card	USB smart card reader

0xD	Interface	Content Security	-
0xE	Interface	Video	Webcam
0xF	Interface	Personal Healthcare	-
0xDC	Both	Diagnostic Device	USB Compliance testing device
0xE0	Interface	Wireless Controller	Wi-Fi adapter, Bluetooth adapter
0xEF	Both	Miscellaneous	Active Sync device
0xFE	Interface	Application Specific	IrDA
0xFF	Both	Vendor Specific	

Tabella 3.2 : Tabella riassuntiva della classe di dispositivi USB[9]

3.4 Il protocollo di comunicazione

La porta USB è come noto una comunicazione seriale di tipo asincrono basata su due segnali differenziali D+ e D- che variano tra 0 e 3.3V con duty-cycle pari al 50% , frequenza di 12MHz (eccetto USB1.0) e codifica NRZI. Ogni dispositivo per ricevere i dati, in maniera corretta, dovrà campionare il segnale a frequenza superiore, pari a 48MHz. La comunicazione è basata sull'invio/ricezione di pacchetti dati. Esistono tre tipi di pacchetti dati**: SYNC, INFORMAZIONE,STOP**.

SYNC:

Quando il bus è inattivo D+ è alto mentre D- è basso. Questo stato è detto anche **stato *J***. Appena vi è una richiesta di scambio dati si ha la transizione del segnale D+ da alto a basso e D- da basso ad alto, portando così il bus nello **stato *K*** . Il passaggio dallo stato *J* allo stato *K* determina lo start della cominucazione: ***SOP (start-of-packet).*** Successivamente vengono inviati 8 bit (KJKJKJKK) che permettono al ricevente di sincronizzare la comunicazione con il suo clock interno. Gli ultimi 2 bit (KK) inviati dal nel frame di SYNC servono ad identificare la fine del frame stesso e l'inizio del frame di PID.

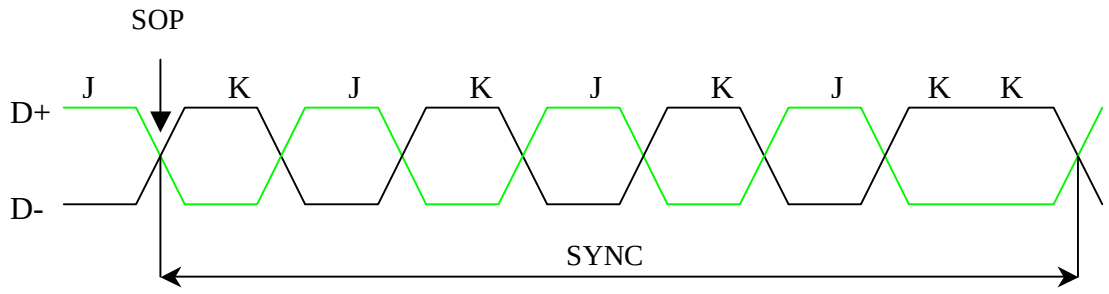


Figura 3.5: Invio di un pacchetto SYNC

Figura 3.6: Andamento dei segnali D+, D- nel tempo durante l'invio del pacchetto SYNC. Da questa immagine è possibile verificare i livelli di tensione (circa 3,3V) e la durata di ogni bit trasmesso, circa 83ns (1/12MHz).

STOP:

Lo stop della comunicazione : ***EOP (End-Of-Packet)*** si ha quando D+ e D- rimangono bassi per una durata pari a due volte il tempo di trasmissione e successivamente si ritorna nello stato J.

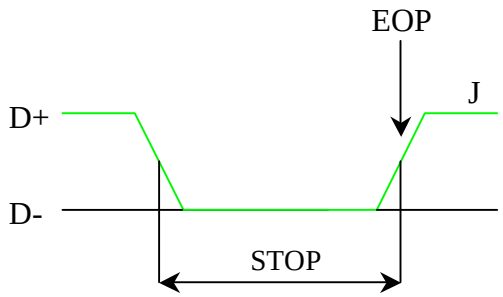


Figura 3.7 Invio del pacchetto di STOP

INFORMAZIONE:

L'informazione ha lunghezza variabile da 1 a 1025 bytes ed il primo bytes è sempre il Packet Identifier : ***PID*** e definisce come interpretare l'informazione. Il byte PID è composto da 4 bit ed il complemento di questi. Esistono 16 possibili PID di cui 6 riservati. I PID possono essere suddivisi in quattro categorie di pacchetti: ***Token***, ***Data***, ***Handshake*** e ***Special***

PID Type	PID Name	PID<3:0>*	Description
Token	OUT	0001B	Address + endpoint number in host-to-function transaction
	IN	1001B	Address + endpoint number in function-to-host transaction
	SOF	0101B	Start-of-Frame marker and frame number
	SETUP	1101B	Address + endpoint number in host-to-function transaction for SETUP to a control pipe
Data	DATA0	0011B	Data packet PID even
	DATA1	1011B	Data packet PID odd
	DATA2	0111B	Data packet PID high-speed, high bandwidth isochronous transaction in a microframe (see Section 5.9.2 for more information)
	MDATA	1111B	Data packet PID high-speed for split and high bandwidth isochronous transactions (see Sections 5.9.2, 11.20, and 11.21 for more information)
Handshake	ACK	0010B	Receiver accepts error-free data packet
	NAK	1010B	Receiving device cannot accept data or transmitting device cannot send data
	STALL	1110B	Endpoint is halted or a control pipe request is not supported
	NYET	0110B	No response yet from receiver (see Sections 8.5.1 and 11.17-11.21)
Special	PRE	1100B	(Token) Host-issued preamble. Enables downstream bus traffic to low-speed devices.
	ERR	1100B	(Handshake) Split Transaction Error Handshake (reuses PRE value)
	SPLIT	1000B	(Token) High-speed Split Transaction Token (see Section 8.4.2)
	PING	0100B	(Token) High-speed flow control probe for a bulk/control endpoint (see Section 8.5.1)
	Reserved	0000B	Reserved PID

*Note: PID bits are shown in MSb order. When sent on the USB, the rightmost bit (bit 0) will be sent first.

Tabella 3.3: Descrizione dei vari tipi di PID

(Per endpoint s' intende quella parte specifica del dispositivo che realizza lo scambio dati)

Ogni function endpoint è indirizzato usando i campi function address e endpoint.

Il campo ***function address (ADDR)*** specifica la function, attraverso l'indirizzo, con la quale vengono scambiati pacchetti dati. Tale campo è costituito da 7 bit permettendo di specificare così 127 function, che è proprio il numero massimo di function collegabili.

Il campo ***endpoint (ENDP)*** permette un indirizzamento più flessibile delle function qualora vi siano più endpoint possibili. Eccetto che per gli endpoint address zero, il numero dell'endpoint dipende dalla function. L' endpoint address è costituito dall'endpoint number (0x0-0xF)+ l'endpoint direction (in/out). Una function Full-speed può supportare un massimo di 16 endpoint.

L'affidabilità della comunicazione è gestita dal metodo **CRCs** (***Cyclic Redundancy Cheks***). Il CRC prevede la generazione di una stringa di bit di controllo che viene normalmente trasmessa assieme ai dati e il calcolo è basato sull'aritmetica modulare. Un codice CRC è definito dal suo polinomio generatore $g(x)$.

Il calcolo del CRC inizia con il messaggio che viene associato a un polinomio $b(x)$ di grado pari al numero di bit del messaggio, grado che indicheremo con n.

Per esempio, il messaggio 10011010 ($n = 8$) produce il polinomio

$$b(x) = x^7 + x^4 + x^3 + x^1$$

Supponendo che il CRC abbia un polinomio di grado g, si "trasla" a sinistra il polinomio $b(x)$ di g posizioni, ottenendo il polinomio $p(x)$ di grado ($h = n + g$).

Nell'esempio, supponendo un grado $g = 4$ si ottiene:

$$p(x) = x^{11} + x^8 + x^7 + x^5$$

Il calcolo algebrico del polinomio corrisponde alla seguente formula:

$$p(x) = xg * b(x)$$

Si esegue ora la divisione $p(x) / g(x)$ ottenendo un quoziente $q(x)$ e un resto $r(x)$. Tale divisione viene svolta con la lunga divisione dell'aritmetica modulo 2, ovvero non si fanno riporti: ricordiamo che nell'aritmetica modulo 2 la somma e la sottrazione sono eseguibili utilizzando la funzione logica XOR. Il messaggio inviato lungo il canale è formato dall'unione del messaggio $p(x)$ più il resto della divisione $r(x)$.

$m(x) = p(x) + r(x)$

Visto che $r(x)$ è il resto di una divisione che usa come divisore $g(x)$, questo resto può avere al massimo un grado pari a quello di $g(x)$; ricordando inoltre che $p(x)$ è ottenuto prendendo il messaggio da inviare $b(x)$ e traslandolo di g posizioni cioè del grado del polinomio $g(x)$, si ottiene che i polinomi $p(x)$ e $r(x)$, quando vengono sommati, non si sovrappongono nei termini di ugual grado e quindi il messaggio $b(x)$ viene inviato inalterato.

$p(x)$ è definibile anche come

$p(x) = q(x) * g(x) + r(x)$

Cioè il polinomio è uguale al quoziente per il divisore più il resto. Sostituendo questa descrizione nella formula precedente otteniamo che $m(x)$ diventa:

$m(x) = q(x) * g(x) + r(x) + r(x)$

Per le leggi dell'algebra sui cambi finiti sommando un polinomio con se stesso si ottiene un polinomio nullo e quindi $m(x)$ diviene:

$m(x) = q(x) * g(x)$

Si nota che il messaggio è quindi divisibile per il polinomio generatore con resto nullo. La rilevazione degli errori usa questa proprietà, difatti il ricevitore riceve $m(x)$ e lo divide per il polinomio $g(x)$ che è definito precedentemente. Se il messaggio è stato ricevuto inalterato la divisione non produce resto mentre se si sono verificati errori di trasmissione la divisione produrrà un resto. La presenza di errori multipli potrebbe produrre comunque una divisione senza resto in un messaggio errato. La probabilità che questo accada dipende dal polinomio e dal suo grado e statisticamente si dimostra che questo accade raramente. Nel caso il messaggio dia resto nullo basta traslare $m(x)$ a destra di un grado g per ottenere il messaggio $b(x)$ di partenza.
Il polinomio generatore per PID di tipo Token è $g(x) = x^5 + x^2 + 1$ mentre il messaggio associato al polinomio $b(x)$ è costituito da ADDR+ENDP.
Il polinomio generatore per PID di tipo Data è $g(x) = x^{16} + x^{15} + x^2 + 1$ mentre il messaggio associato al polinomio $b(x)$ è costituito dal pacchetto dati.

I ***Token packet***: IN, OUT,SETUP avranno la seguente forma:

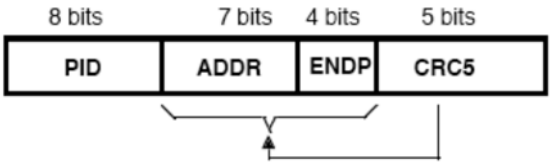


Figura 3.8: Formato Token packet tipo IN,OUT,SETUP

Da notare che il controllo CRCs non gestisce gli 8 bit che identificano il PID.
SOF viene trasmesso dall' host ogni 1ms (125us se hi-speed) ed il tempo intercorrente tra due SOF viene chiamato ***Frame (microframe)*** ed avrà la forma:

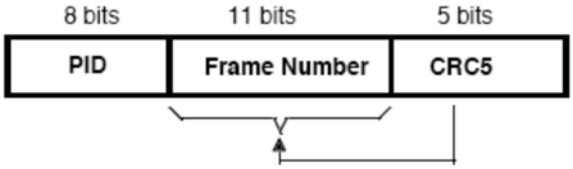


Figura 3.9: Formato Token packet di tipo SOF

Il pacchetto SOF è l'unico a non avere bisogno di essere indirizzato ed a non aver bisogno di alcun segnale di acknowledgment. Il ***Frame Number*** non è altro che un contatore a 11 Bit (valore massimo 2047) che si incrementa ogni 1ms e si resetta ogni 2048ms.

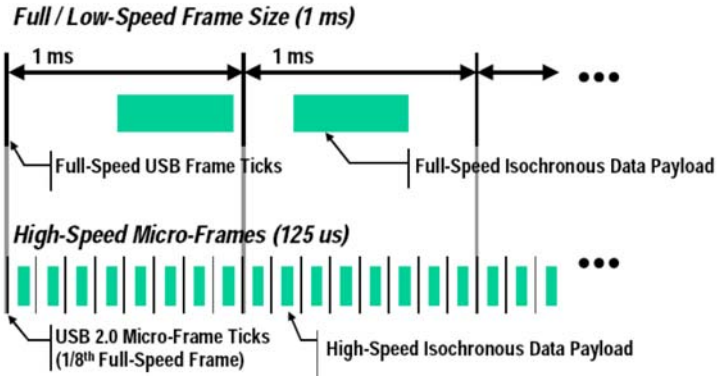


Figura 3.10: Differenza fra un frame (Full/Low Speed) ed un microframe (High-Speed)

Per quanto riguarda i ***Data packet*** questi avranno la seguente forma:

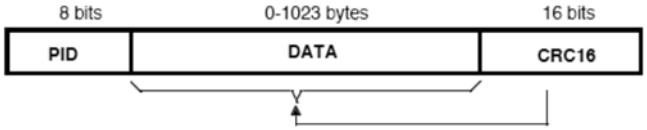


Figura 3.11 : Formato Data packet

Esistono 4 tipi di pacchetti dati (vedi tab. 3.3). ***DATA0*** e ***DATA1*** vengono trasmessi alternativamente in modo da rendere più facile la rilevazione di eventuli errori nella trasmissione. Nelle comunicazioni hi-speed vengono invece adoperati tutti i tipi di pacchetti dati.

Gli ***handshake packet*** sono usati per rilevare lo stato dell'esito della comunicazione tra host e endpoint, sia quando si scambiano token packet che data packet. Hanno dimensione pari a 8bit ,pari alla dimensione del PID, poiché non viene eseguito controllo CRC.
L'Hanshake ***ACK*** indica una corretta ricezione mentre ***NACK*** indica che il ricevitore è attualmente occupato in altra operazione o non ha le risorse per portare a termine il trasferimento del dato (p.e. manca il dato). Non funziona con *SETUP*. ***STALL*** viene inviata dalla function in risposta ad una *IN token* o dopo una operazione di *OUT* o dopo una *PING* ed indica che la function non è abilitata a trasmettere o ricevere dati o che la control pipe non è supportata. La function entra così in uno stato indefinito da cui l'host non può farla uscire. Questo tipo di hanshake viene usato solo quando si vogliono realizzare la funzioni di ***Halt*** (capitolo 9 spec usb2.0) e il ***Protocoll Stall*** (Sezione 8.5.3 spec usb2.0)
NYET (No response yet from receiver) è un'handshake usato nelle comunicazioni hi-speed ed inviato da un endpoint in seguito ad un ***PING*** e può essere inviato da un hub dopo una *split-transaction* (capitolo 11 spec USB2.0).

Gli ***Special packet*** non verranno analizzati in questa sede. Sono descritti ampiamente nelle specifiche della porta USb 2.0 (capitolo 8).

3.5 Implementazione della comunicazione fra Host e device

Lo scambio di dati attraverso la porta USB non è così semplice ed immediato poiché richiede una astrazione a più livelli del sistema stesso sia dal lato host che dal lato device. In particolare ci sono 4 aree di implementazione su cui focalizzare l'attenzione[9].

1. ***USB Physical Device:*** dispositivo hardware collegato che realizza la funzionalità preposta. Ad esempio un HD esterno, uno strumento di misura, un mouse.

2. ***Client Software***: Software eseguito dall'host che permette di comunicare con il dispositivo stesso. Ad esempio un' interfaccia grafica che permette di gestire e copiare i file all'interno di un HD, quella di un sistema di misura realizzato dal dispositivo collegato o di gestione di una stampante.

3. ***USB System Software***: software che permette di gestire la comunicazione sulla porta USB attraverso *l'Host Controller*. E' cosituito da*:* ***Host Controller Driver, USB Driver e Host Software.***

4. ***USB Host Controller*** (Host Side Bus Interface): Insieme dell' hardware e del software che permette di collegare i dispositivi all'host. E' costituito dall' **SIE *(Serial Interface Engine*)** e dall' ***Host Controller***

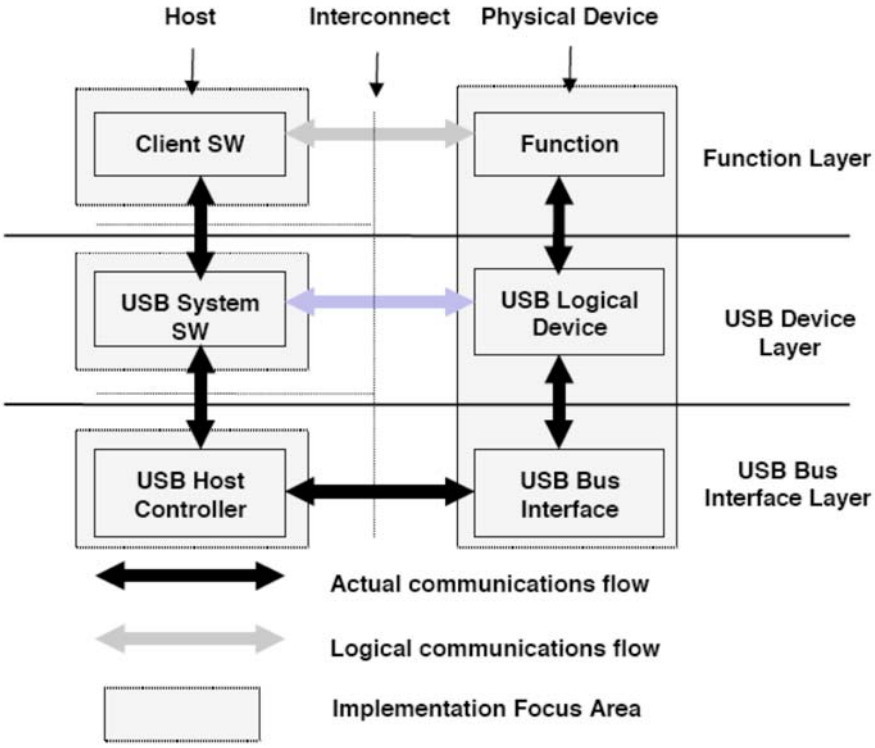


Figura 3.12: Schema a blocchi dell'implementazione della comunicazione

Come si nota, il sistema è organizzato su 3 livelli: l'***USB Bus Interface Layer*** che realizza fisicamente lo scambio dati e il ***Function Layer*** più vicino all'utente che utilizza il *Client Software* e che rende invisibili all'utente finali gli altri due livelli, compreso ***l'USB Device Layer*** che si occupa della comunicazione a livello software tra i due sistemi. L'end-user si interfaccia solo con il *Function Layer* che rende invisibili i livelli sottostanti, aumentando così la facilità d'uso dell'interfaccia.

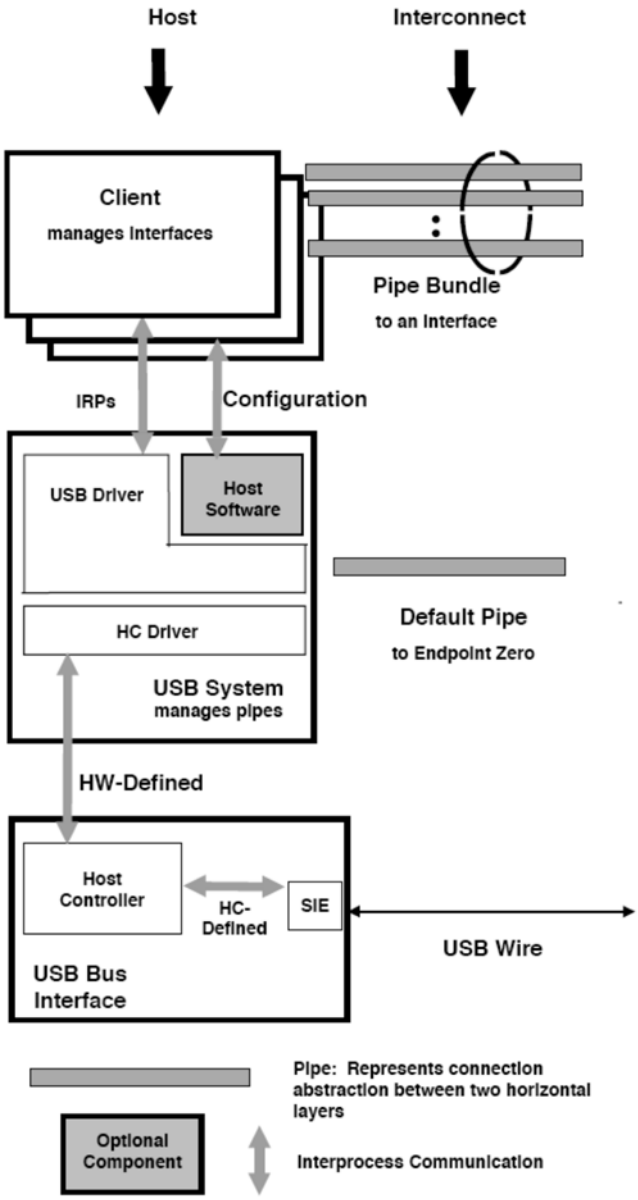


Figura 3.13: Schema a blocchi particolareggiato della comunicazione dal lato Host

L'USB System Software gestisce i dispositivi attraverso la *Default Control Pipe* mentre il *Client Software* gestisce le interfacce attraverso una serie di Pipe, associate ognuna ad ogni endpoint. La Pipe è un' astrazione logica che rappresenta l'associazione fra l' endpoint, cioè quella parte specifica del dispositivo che realizza lo scambio dati e il software sull'host. Una pipe ha degli attributi, per esempio: una pipe può trasferire dati come streams (stream pipe), ovvero senza una struttura precisa o messaggi (message pipe) che hanno struttura ben precisa. Più in generale si può dire che una Pipe è una astrazione logica che permette la comunicazione tra soggetti diversi appartenenti allo

stesso livello. L' USB System Software è costituito da più componenti, come si può notare da figura e che di seguito analizzeremo nel dettaglio.

L' ***USB Driver (USBD)*** fornisce al client software il meccanismo di trasferimento di dati sotto forma di *I/O Request Packets (IRPs)*, che consiste in una richiesta di trasporto dati attraverso la pipe. Fornisce inoltre una descrizione generale del dispositivo consentendo al *Client Software* di gestire e controllare lo stato del dispositivo.

L'***Host Controller Drive (HCD***) è l'interfaccia dell' Host Controller .

L'interfaccia fra HCD e USBD è chiamata ***Host Controller Driver Interface (HCDI)*** e non è direttamente accessibile dal client software, non è quindi discussa in questa sede.

L'Host Controller è il cuore hardware della comunicazione e provvede ad eseguire le seguenti funzionalità:

- ***State Handling***: L' host controller può assumere una serie di stati che l' USB System Software è in grado di gestire.
- ***Serializer/Deserializer***: convertire i dati e le informazioni sottoforma di uno stream di bit da trasmettere attraverso la porta USB. Questo compito è svolto dal ***Serial Interface Engine (SIE) .***
- ***(micro)frame Generation:*** l'host controller produce il SOF ogni 125μs quando ha a che fare con dispositivi Hi-speed (microframe) e ogni 1ms quando ha a che fare con dispositivi Full-spedd (frame).
- ***Data Processing:*** processa tutte le richieste di trasmissione dato da e per l'host.
- ***Protocollo Engine:*** gestisce la comunicazione secondo le regole del protocollo.
- ***Trasmission Error Handling:*** provvedere a gestire i seguenti errori di trasmissione:
  1. *Tiemout conditions*: trascorso un certo tempo dall'invio del pacchetto se non si ha alcuna risposta. Condizione che si verifica nel caso in cui l'endpoint indirizzato è insensibile o quando la struttura della trasmissione è dannegiata a tal punto che l'endpoint designato non la riconosce.

2. *Protocol errors*: Ad esempio un un handshake packet corrotto o inappropriato o un falso EOP.
3. *Missig or Invalid trasmission:* l'host controller non può portare a termine una trasmissione/ricezione perché ad esempio mancano le risorse o il controllo CRCs ha rivelato un errore.

- ***Remote Wakeup:*** l'USB system Software ha la possibilità di mettere l'Host Controller nello stato di sospensione mettendo così in stop il traffico sul bus. Quando si è in questo stato l'USB System Software può abilitare l'host controller ad uscire da questo stato all'arrivo di un segnale di wake-up.
- ***Root Hub:*** realizza la funzione standard di un hub, in modo da poter collegare l'Host controller a una o più porte USB.
- ***Host System Interface:*** Interfaccia ad alta velocità tra HC e Host System. Rende possibile lo scambio di dati, attraverso un buffer, tra la Memoria del sistema e l'Host Controller che gestisce automaticamente il trasferimento, informando l'USB System Software sullo stato del buffer (pieno o vuoto)

Della versione USB 1.0-1.1 esistono due implementazioni di Host Controller: l' ***Open Host Controller Interface (OHCI)*** realizzata da Compaq, Microsoft e National Semiconductor e l' ***Universal Host Controller Interface (UHCI)*** sviluppato da Intel che ha autorizzato la produzione e la distribuzione da parte di VIA Technologies.
UHCI è più *software-driven* rispetto all'OHCI, cioè è gestibile maggiormente da software, costringendo così il processore a dedicargli più risorse. Il vantaggio sta nel minor costo di realizzazione. L' elevata competizione tra questi due Host Controller ha costretto i realizzatori di sistemi operativi e i fornitori di hardware a testare entrambe le versioni, aumentando il costo finale del prodotto.
Nella versione USB2.0 è stato deciso quindi avere una sola implementazione , chiamata ***Enhanced Host Controller Interface (EHCI)*** in grado di realizzare una comunicazione Hi-speed. Molti degli attuali PC che hanno un EHCI montano ulteriori Host Controller chiamati "*companion host controller*" in modo da poter supportare comunicazioni sia Full-speed che Low-speed.

Tutte le caratteristiche e le funzionalità dell'*USB System Software* sono realizzate a livello di sistema operativo(OS). Uno dei compiti dell' OS [10], infatti, è proprio quello di mascherare il substrato hardware di I/O alle applicazioni, in modo che queste lo vedano sempre allo stesso modo, nascondendo le caratteristiche specifiche dei dispositivi fisici. E' facile individuare le funzionalità dell' *USB System Software* all'interno del software di I/O di un generico dispositivo come da figura 3.14

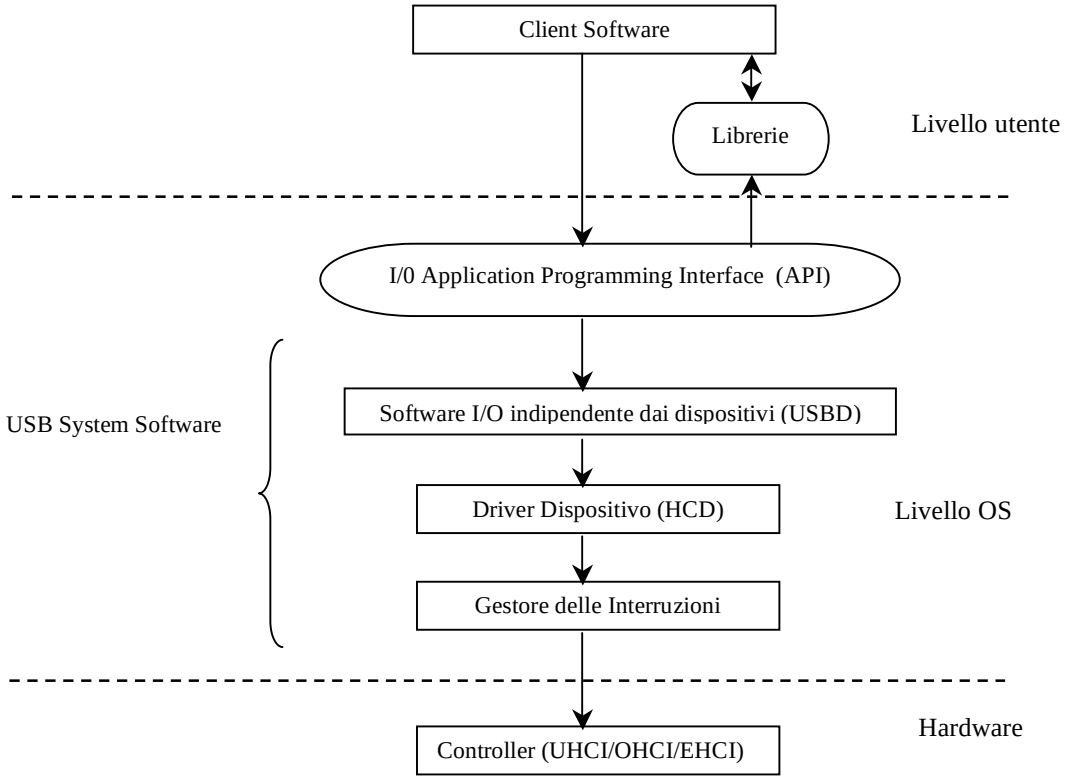


Figura 3.14 :Organizzazione del Software di I/O

Il software indipendente dal dispositivo pensa a:

- Associare un nome simbolico del dispositivo e driver relativo
- Scollegare la velocità del dispositivo dalla velocità della CPU attraverso un Buffer
- Gestire gli errori e propagare quelli ingestibili al livello utente.
- Allocazione e rilascio dei dispositivi dedicati.

Il driver del dispositivo:

- Fornisce una interfaccia astratta del dispositivo indipendente dai dettagli hardware al resto del sistema operativo
- Accetta e traduce le richieste di lettura/scrittura dal software indipendente in una sequenza di comandi comprensibili per il controllore del dispositivo.
- Scrive i comandi tradotti nei registri del controllore
- Si blocca in attesa che il controllore finisca il suo lavoro
- Si sblocca in caso di una interruzione
- Esegue controlli per rilevare eventuali errori
- In caso di assenza di errori passa i dati letti dal software indipendente dal dispositivo
- E' fornito dal costruttore del dispositivo.

Tipicamente i driver sono bloccati in attesa che arrivi una interruzione dalla periferica che stanno controllando. Quando arriva un' interruzione il gestore delle interruzioni pensa a:

- Salvare i registri del processo interrotto
- Segnalare al controllore delle interruzioni quando può inviare una nuova interruzione
- "svegliare" il driver opportuno incaricato di svolgere l'operazione richiesta

In questa sede non ci addentreremo in questo meccanismo troppo nel dettaglio, cosa assai complicata e lunga, ma ci limiteremo a capire a grandi linee il funzionamento. Lo scopo di questo lavoro, infatti, non è quello di analizzare nel dettaglio tutte le dinamiche e i meccanismi hardware e software dell'interfaccia USB ma di avere le nozioni di base per poter realizzare uno scambio dati tra una function e il PC.

Nel lato host ci preoccuperemo di realizzare il *Client Software*. La gestione dell' *USB Host Controller* e la realizzazione dell' *USB System Software* viene delegato ai realizzatori di driver e viene tipicamente fornito dal costruttore.

La complessità di progettazione e sviluppo dell' ***USB Physical Device*** dipende in buona parte dal tipo di controllore scelto. Ne esistono alcuni con CPU integrata (Microchip PIC18F4550 , Cypress EZ-USB Cypress, enCoRe II , Freescale MC68HC908JB16 ,Freescale MCF5482 ColdFire) e altri che ne necessitano di una esterna (National Semiconductor USBN9603, Philips Semiconductors ISP1181B, Philips Semiconductors ISP1581, PLX Technology NET2272, FTDI Chip FT232BM and FT245BM)

3.6 Modalità di trasferimento dei dati

Esistono quattro possibili modalità di trasferimento dati, ognuna di queste rispetta una particolare sequenza di invio di pacchetti e devono essere portate a termine all'interno dello stesso frame. Ovvero, una volta che è stata avviata una sequenza in una certa modalità, non sarà possibili modificarla, se non una volta terminato il trasferimento.
Le modalità di trasferimento si differenziano, principalmente, per accuratezza e velocità di trasmissione:

Control Mode: usato nelle trasmissioni brevi, non periodiche, in operazioni di comando/stato e inizializzazione del software sull'host.
Isochronous Mode: usato nelle trasmissioni periodiche, la dove c'è una comunicazione continua e costante tra l'host e l'endpoint e si può tollerare qualche errore di comunicazione come nell' audio e video streaming.
Interrupt Mode: usato nelle comunicazioni a bassa velocità e limitata latenza come i mouse e le tastiere.
Bulk Mode: usato in comunicazioni non periodiche e che richiedono un trasferimento veloce di un'elevata quantità di dati. Impiegato per interfacciare stampanti e plotter.

Tipo	Caratteristica	Data-rate	
		Low-speed	Full-speed
Interrupt	Accuratezza	48KB	1216KB
Bulk	Accuratezza	N.A.	1216KB
Isochronous	Velocità	N.A.	1023KB
Control	Velocità e Accuratezza	24KB	832KB

Tabella 3.4: Tabella riassuntiva che mette a confronto le velocità di trasferimento nelle varie modalità di funzionamento

Di seguito saranno spiegati, senza entrare troppo nel dettaglio i vari modi di trasferimento, ampiamente descritti nelle specifiche USB 2.0 al capitolo 8 [9].

Interrupt Mode:

Tipicamente quando si pensa ad una interrupt, si pensa ad un evento esterno sul programma principale generato da una periferica. Ciò provoca una immediata interruzione del programma stesso e l'esecuzione di una subroutine, detta routine di interrupt. Questo però non accade nella comunicazione USB poiché l'hub principale esegue il polling su tutti i dispositivi di I/O collegati per testare se richiedono attenzione una volta ogni 1ms. Questo tempo di scansione è sufficiente nella maggior parte dei dispostivi HMI (*Human Machine Interface*) che richiedono tempi di aggiornamento di circa 10ms. Qualora il dispositivo necessitasse di scansioni con tempistiche molto ridotte, sarà il suo microcontroller a processare e gestire tali richieste, riproponendo tale richiesta a tempo debito all'host.

L'interrupt mode opera attraverso delle IN e OUT. Al seguito di una IN token, la function può fornire all'host il dato oppure due handshake packet: NAK e STALL. Se l'endpoint non ha nessuna nuova richiesta di interrupt la function invierà NAK. Se invece c'è una richiesta di interrupt sospesa allora la function invierà il dato. Se il dato viene ricevuto in maniera corretta dall' host allora questo provvederà a rispondere con

un ACK. STALL invece viene generato solo quando la funzione di Halt è attiva (indica quando l'endpoint è "fermo").

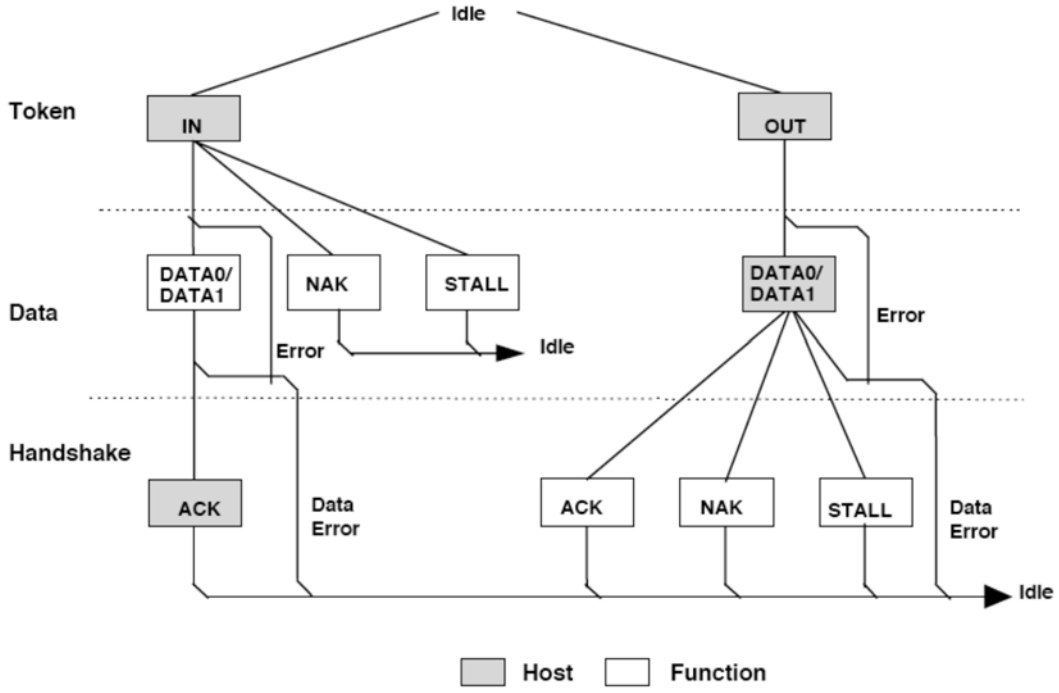


Figura 3.15: Rappresentazione grafica dell' Interrupt Mode

Bulk Mode:

I pacchetti usati per implementare il trasferimento sono identici a quelli usati all' inetrrupt mode e come quest'ultima è molto indicata per limitare al massimo gli errori di trasmissione ma più indicata pr le trasmissioni High-speed. E' previsto, infatti anche un handshake di tipo NYET.

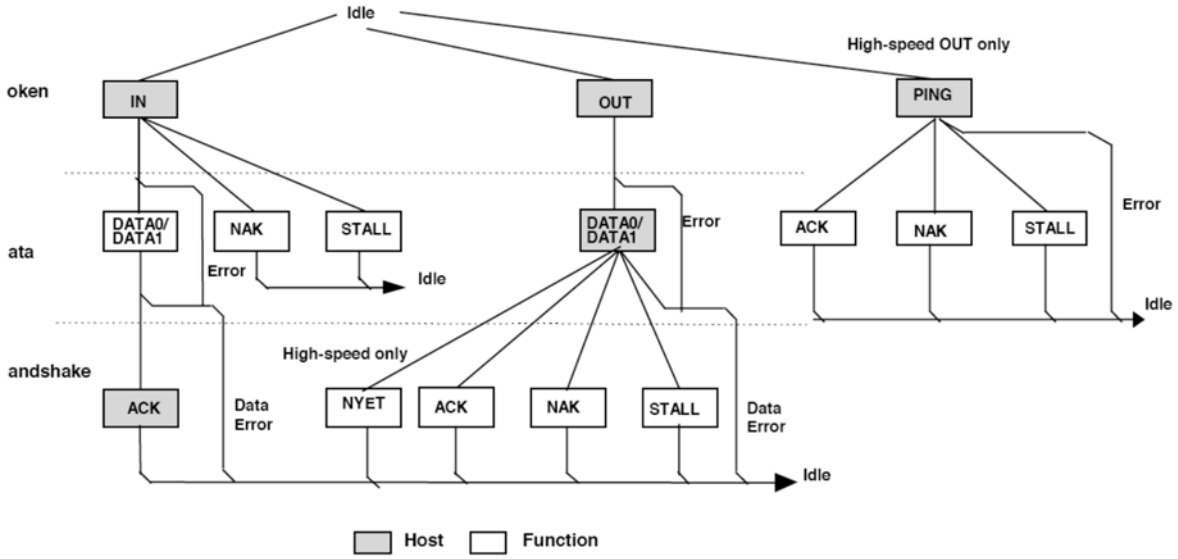


Figura 3.16: Rappresentazione grafica del Bulk Mode

Isochronous Mode:

In questa modalità di trasferimento non è previsto alcun segnale di handshake, la comunicazione è quindi più veloce, rispetto a quelle viste fino ad ara anche se non c'è alcuna garanzia sull' accuratezza del dato . *L'USB System Software* usa la dimensione massima del dato da trasferire/ricevere (data payload) e la usa per assicurarsi di avere sufficiente tempo per completare il trasferimento del dato all'interno di ogni frame. Se la verifica va a buon fine allora viene stabilita la connessione con il dispositivo.
Questa modalità si presta quindi per essere usata in comunicazioni periodiche e continue nel tempo, quando i tempi di trasferimento sono particolarmente critici.

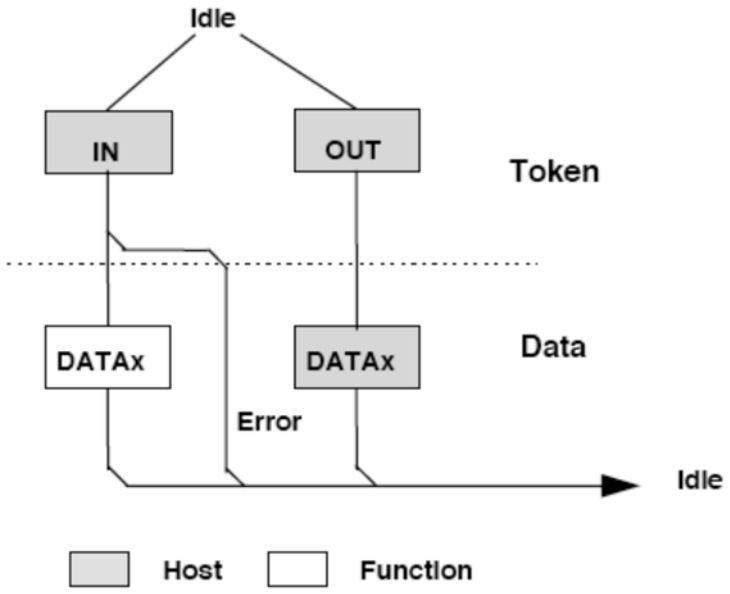


Figura 3.17 : Rappresentazione grafica dell' Isochronous Mode

Control Mode:

Come noto, ogni dispositivo di I/O ha un proprio ***descrittore di dispositivo***. Questo rappresenta un' astrazione del controllore , nel nostro caso l 'USB host controller, ne riassume tutti i dettagli e permette la comunicazione tra il processo applicativo e il controllore. In generale, ogni descrittore riporta l'indirizzo del registro di controllo, di stato e registro dati del controllore, il semaforo di sincronizzazione, il numero dei dati da trasferire, l'indirizzo del buffer e l'esito dell'operazione eseguita. Il descrittore del dispositivo assieme alle funzioni del controllore costituisce il driver del dispositivo.
Questa modalità è stata appositamente studiata per poter accedere al descrittore del dispositivo. Si ha infatti la possibilità di comandare, configurare e interrogare lo stato della periferica attraverso il *client software*. Questa modalità di comunicazione prevede 3 fasi: ***Setup Phase, Data Phase, Status Phase.*** La *Setup Phase* attraverso il PID di SETUP invia la richiesta di configurazione dall'host alla function e durante la *Data Phase* il dato viene inviato alla function che risponde attraverso un Handshake packet (*Status Phase*) indicando all'host l'esito dell'operazione.

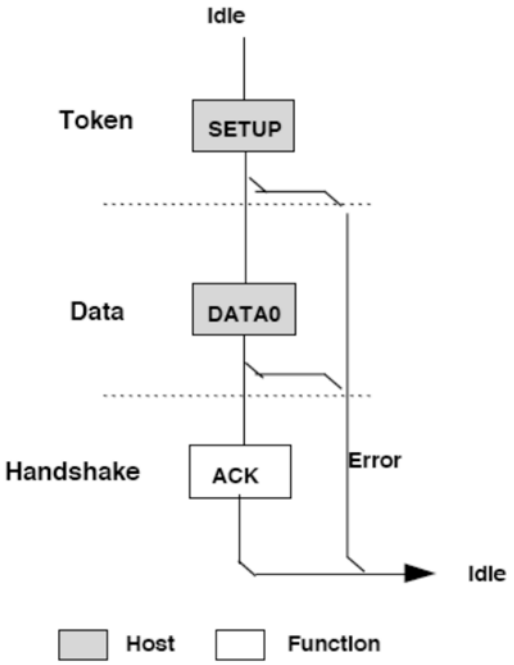


Figura 3.18: Rappresentazione grafica del Control Mode

Capitolo 4

Il DAQ

4.1 Struttura e caratteristiche del DAQ

La struttura del DAQ (Digital AcQuisition) già illustrata nel paragrato 1.7 e riportata in figura 4.1 può essere ulteriormente semplificata , riducendo il numero dei componenti necessari. Scegliendo infatti un ADC a più canali è possibile diminuire il numero dei convertitori, abbattendo così i costi di produzione e di progettazione del firmware del microcontrollore. Se il numero di canali dell' ADC è sufficiente il multiplexer diventa superfluo poiché si usa il multiplexer interno per commutare da un canale all'altro. Dopo un' attenta valutazione sulle possibili grandezze da acquisire è stato stabilito che **4 canali** di acquisizione sono più che sufficienti. In questo DAQ si sfrutta un solo canale (CH:1), quello collegato al trasduttore. Il controllo dell' eventuale microposizionatore viene affidato direttamente al firmware dell' MCU il quale dedica, a tale scopo, una porta di I/O. La struttura del sistema è rappresentata in figura 4.2 , nella quale è riportato anche l'USB controller. Quest'ultimo può essere parte integrante del microcontrollore.

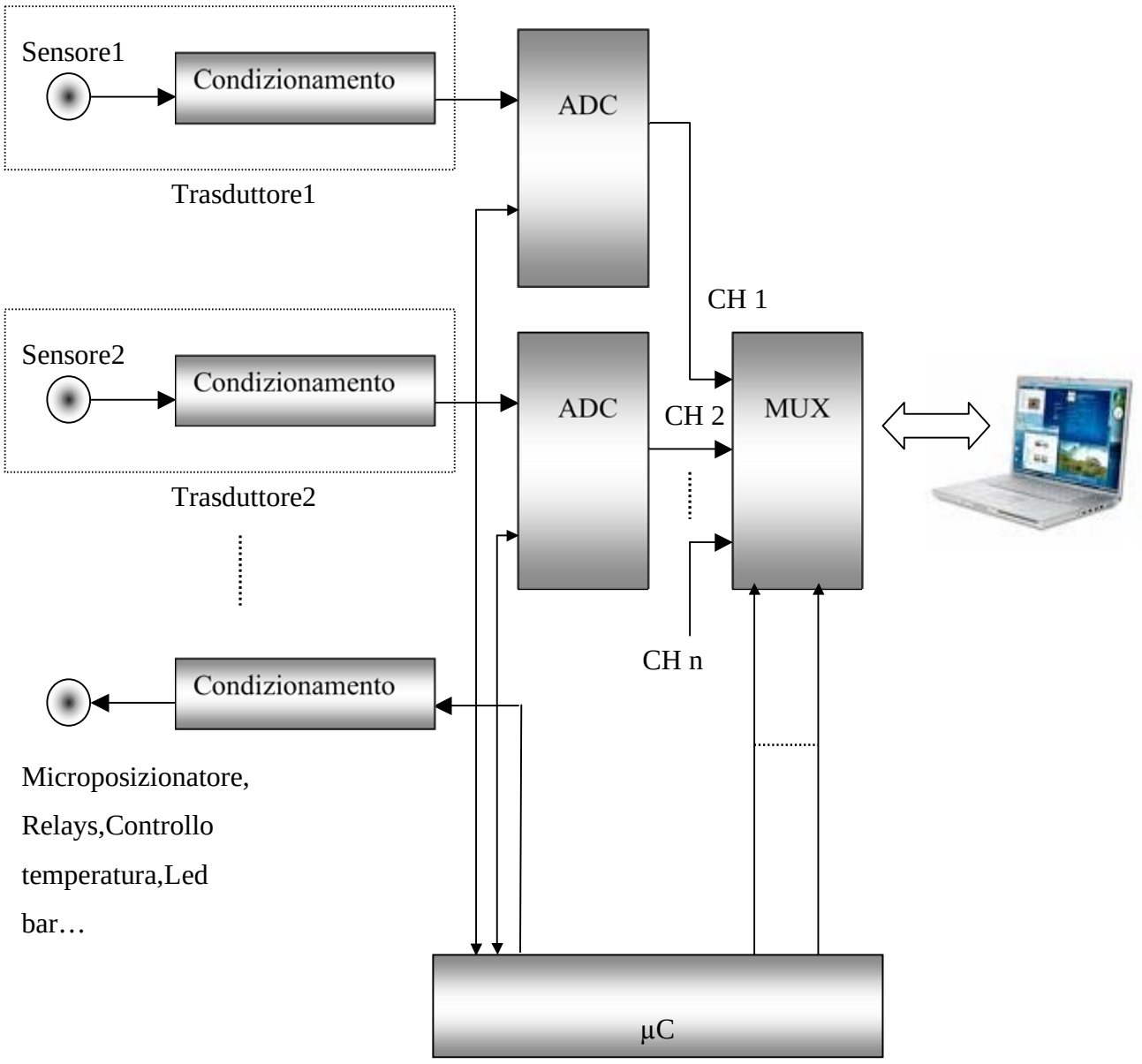


Figura 4.1 : Architettura del DAQ.

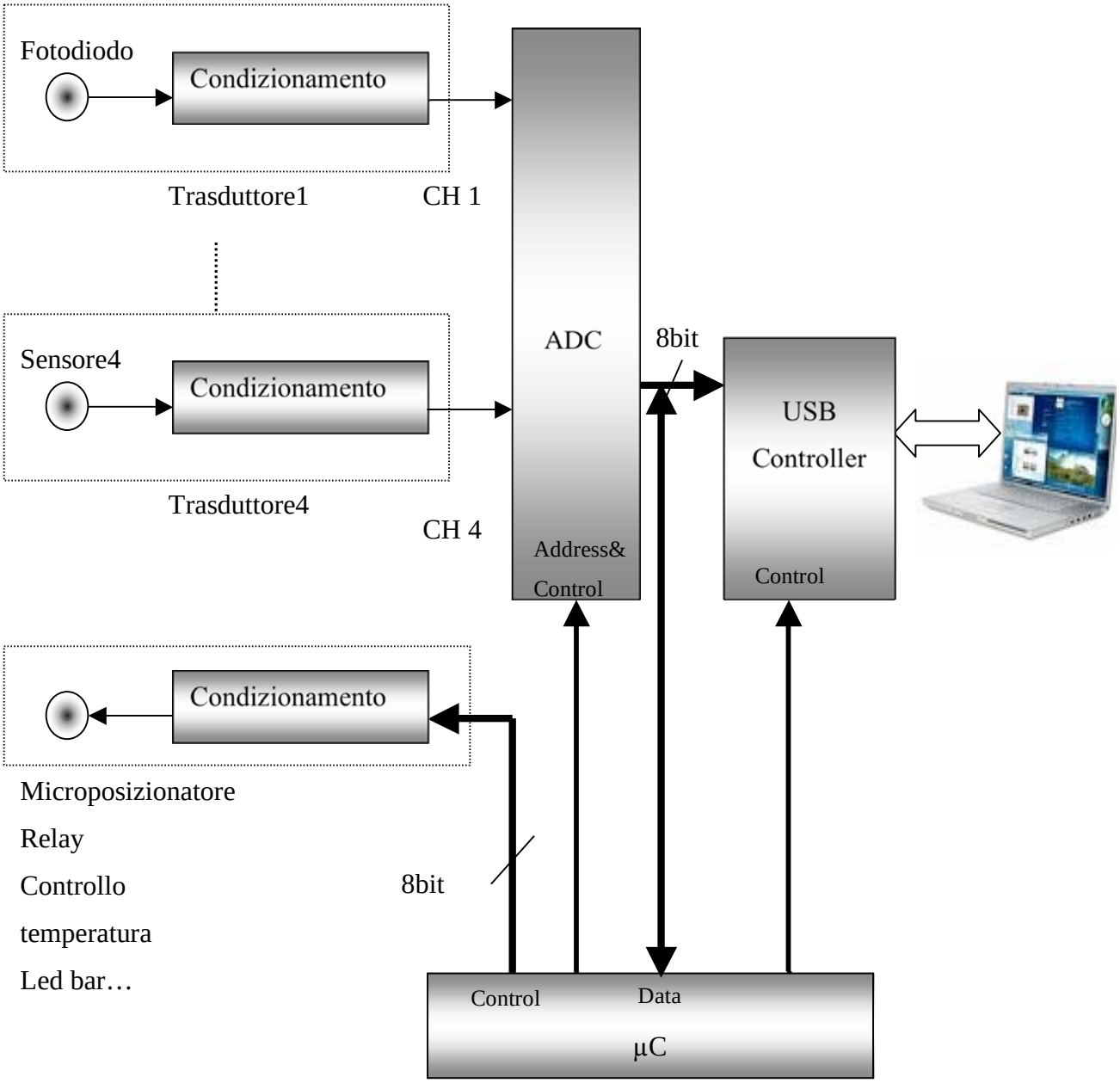


Figura 4.2: Architettura del DAQ con ADC a 4 canali e controllore USB non integrato nell' MCU.

Il DAQ dovrà essere in grado di acquisire dati, nel caso peggiore, ad una **velocità di 80KSPS** ($f_{sample} \geq 80KHz$) e trasmetterli alla stessa velocità al calcolatore attraverso l'USB, anche se, come abbiamo già detto (par.1.8) è ragionevole ipotizzare che la frequenza del segnale non sia così elevata. L'alimentazione del DAQ è prelevata direttamente dal bus in modo da rendere la scheda indipendente dal trasduttore. L' MCU avrà un ruolo fondamentale all'interno della sistema poiché avrà il compito di permettere una corretta comunicazione fra USB controller e ADC. Dovrà inoltre selezionare il canale di acquisizione ed eventualmente gestire un **microposizionatore**, un **servomotore** o un **motore passo-passo** tramite la **porta di I/O digitale.** Queste ultime due funzioni non verranno realizzate in questo lavoro di tesi poiché è richiesta la gestione di un solo canale , mentre il microposizionatore è un'applicazione futura. Attraverso il microcontrollore sarà poi possibile acquisire ad esempio la temperatura del portacampioni utilizzando un sensore con interfaccia digitale (attraverso gli I/O digitali liberi) o meno (attraverso un canale dell' ADC). Sfruttando poi gli I/O digitali liberi è possibile implementare, attraverso del firmware opportuno, un controllo della temperatura, per esempio gestendo una resistenza corazzata ed un ventilatore in modo da termostatare la cella portacampioni.

4.2 Il controller USB

Come già accennato in precedenza, esistono diverse interfacce USB, con CPU integrata: Microchip PIC18F4550 , Cypress EZ-USB , Cypress enCoRe II , Freescale MC68HC908JB16 ,Freescale MCF5482 ColdFire e non : National Semiconductor USBN9603, Philips Semiconductors ISP1181B, Philips Semiconductors ISP1581, PLX Technology NET2272, FTDI Chip FT232BM e FT245BM. Questa fase progettuale è delicata, in quanto, una scelta poco felice del controller può allungare molto i tempi di progettazione ,realizzazione e collaudo del sistema. Per scegliere il controller adatto ci siamo serviti dei seguenti criteri :

1. Il controller deve essere in grado di soddisfare la velocità di comunicazione, secondo le specifiche di sistema (almeno 80-100KB/s)
2. Lo studio del dispositivo deve essere ridotto al minimo, in modo da poter dedicare più tempo alla fase di progettazione, la messa a punto ed il collaudo dell'intero sistema.
3. Disponibilità dei driver.
4. *Tools* di sviluppo gratuiti e possibilmente già noti al progettista.
5. Semplice interfaccia hardware per l'accesso ai dati e per il controllo del dispositivo.
6. Basso costo e facilità di acquisto attraverso i più conosciuti distributori di materiale.
7. Facilità di cablaggio su PCB o possibilità di avere una board pre-assemblata, in modo da ridurre i costi tecnologici nella realizzazione del prototipo.

I controllori con CPU integrata necessitano di un *tool* di sviluppo per la stesura del firmware che varia a seconda della casa costruttrice del chip. Tipicamente consiste in un editor ed un compilatore C che traduce il firmware in linguaggio macchina. Le istruzioni e la sintassi poi, generalmente, cambiano a seconda del compilatore e del microcontrollore adoperato, anche se, per esempio, alcune *software-house* come la IAR System le riducono al minimo. I controller USB sono inseriti all'interno dell'architettura del microcontrollore e richiedono quindi lo studio di quei pochi registri

necessari al funzionamento. Qualora invece, il progettista non conosca già i registri del microcontrollore, si dovrà armare di pazienza e del manuale, con notevole dilatazione dei tempi di progettazione. Non dobbiamo scordare di mettere in preventivo anche il tempo per lo studio delle librerie e delle funzioni dedicate al funzionamento del controller USB interno al uC. C'è da considerare poi che una volta scritto il firmware questo deve essere scaricato sul μC attraverso un programmatore. Quest'ultimo, ovviamente, cambia a seconda del μC . Per ridurre quindi i tempi di progettazione l'unica soluzione sensata è fare affidamento sul *background* e sul *know-how* del progettista cercando poi un compromesso con le specifiche di progetto. Ad esempio, Il PIC18F4450 è largamente impiegato, poiché appartiene ad una casa costruttrice, Microchip,che ha una buona fetta del mercato dei MCU e quindi può attrarre la preferenza di tutti quei progettisti che si avvalgono quotidianamente di questo tipo di controllori. Un progettista abituato ad usare i μC della famiglia Atmel invece preferirà usare, ad esempio, l'AT90USB128. C'è comunque da tener conto che di solito la stesura e la fase di debug del firmware necessario per un uC complesso, come possono essere questi, aumenta. Tenendo conto di quanto detto, dei mezzi (HW e SW) e delle conoscenze a disposizione si opta per l'uso di un normale uC ATMEL: **ATMEGA16L** [11] ed un controller USB esterno soddisfacendo così i criteri di scelta (2),(4),(6), e (7). Le specifiche riguardanti l' MCU sono riportate nel paragrafo 4.6

Tra i vari controller USB senza CPU il più performante è sicuramente l' **FT245BM** della *FTDi chip* [12]. Il protocollo USB è gestito interamente dal chip e il dato viene poi messo a disposizione su 8 bit paralleli grazie alla conversione interna USB-FIFO. Il controllore rende al minimo i segnali esterni di controllo(2) ed è in grado di assicurare una comunicazione bi-direzionale sufficientemente veloce(1), anche se questa dipende in buona misura dalla bontà dei driver, dalla velocità del calcolatore e dal tempo di latenza introdotto dal processo che si occupa di elaborare ed acquisire i dati. I driver messi a disposizione per questo IC sono disponibili per piattaforme Windows ,Mac e Linux e creano direttamente una COM virtuale o permettono l'accesso al dispositivo attraverso una libreria API(3). In seguito vedremo quale di questi driver è il più adeguato. Il chip è acquistabile dal sito della FTDi chip ed è disponibile anche in versione chip-board(DPL-USB245M) a prezzi ridotti rispetto ad una realizzazione diretta(7),(6). Altro punto a favore di questo controller è l'ampia documentazione tecnica.

4.2.1 Caratteristiche dell' FT245BM

Di seguito sono riportate le principali caratteristiche[12]:

- Velocità di trasferimento dati: 1MB/s (D2xx drivers) o 300KB/s (VCP drivers)
- Trasferimento dati bi-direzionale.
- Alimentazione singola: 4,35 - 5,25V
- USB 1.1 e 2.0 compatibile
- UHCI/OHCI/EHCI host controller compatibile
- Interfaccia di controllo del chip ridotta a 4 segnali di handshake controllabili tramite MCU, PLD o FPGA.
- FIFO di trasmissione di 384 Byte
- FIFO di ricezione di 128 Byte
- Funzione di Suspende/Resume
- Clock integrato
- Regolatore da 3.3V integrato
- Collegamento opzionale di EEPROM in cui memorizzare il VID, PID, Serial Number e la Descrizione del Prodotto.(VID = Vendor ID, PID = Product ID)
- EEPROM programmabile on-board tramite USB
- Trasferimento dati in Bulk e Isochronous Mode

Per quanto riguarda i driver, messi a disposizione sul sito www.ftdichip.com , ne esistono di due tipi: **VCP** e **D2XX**. I Driver **VCP:*(Virtual Com Port)*** sono disponibili per Windows 98/2000/ME/Server 2003/XP/XP64bit/XP embedded/CE4.2 , Mac OSX8 e OSX9 e Linux 2.40 in poi. I driver **D2XX *(USB direct drivers+dll S/W interface)*** sono disponibili, al momento, per Windows e Linux. Permettono di accedere "direttamente" al dispositivo attraverso una libreria **DLL *(Dynamic Lynk Library)* API *(Application Programming Interface)*** incrementando così la velocità di comunicazione.

(a) (b)

Figura 4.3: (a) Chip-board DPL-USB245M (b)FT245BM

4.2.2 Descrizione funzionale a blocchi dell' FT245BM

Il convertitore internamente si presenta come in figura 4.4

3.3 LDO regulator:

Regolatore di tensione Low Drop Out (LDO) per alimentazione dell' *USB Transceiver* ed il *Reset Generator*. Per il corretto funzionamento necessita di un condensatore da 33 μF da collegare al pin *3V3OUT*. Da questo pin è possibile prelevare una corrente massima di 5mA.

USB Transceiver:

Interfaccia fisica con il sistema di trasmissione che provvede a generare i segnali differenziali USBDP (D+) e USBDM (D-) con livelli di tensioni massimi pari a 3.3V

Reset generator:

Il Reset generator provvedere a resettare il circuito all'accensione. L'ingresso RESET# e l'uscita RSTOUT# permettono, ad altri dispositivi di resettare il controller e di resettare altri dispostivi con il controller. RSTOUT# all'accensione , quando VCC>3.5V , rimane basso per circa 5ms dopo di che il clock interno si avvia e RSTOUT#=3.3V. Quando RESET#=0 allora RSTOUT#=0. Se il segnale di RESET# non viene usato va collegato a VCC.

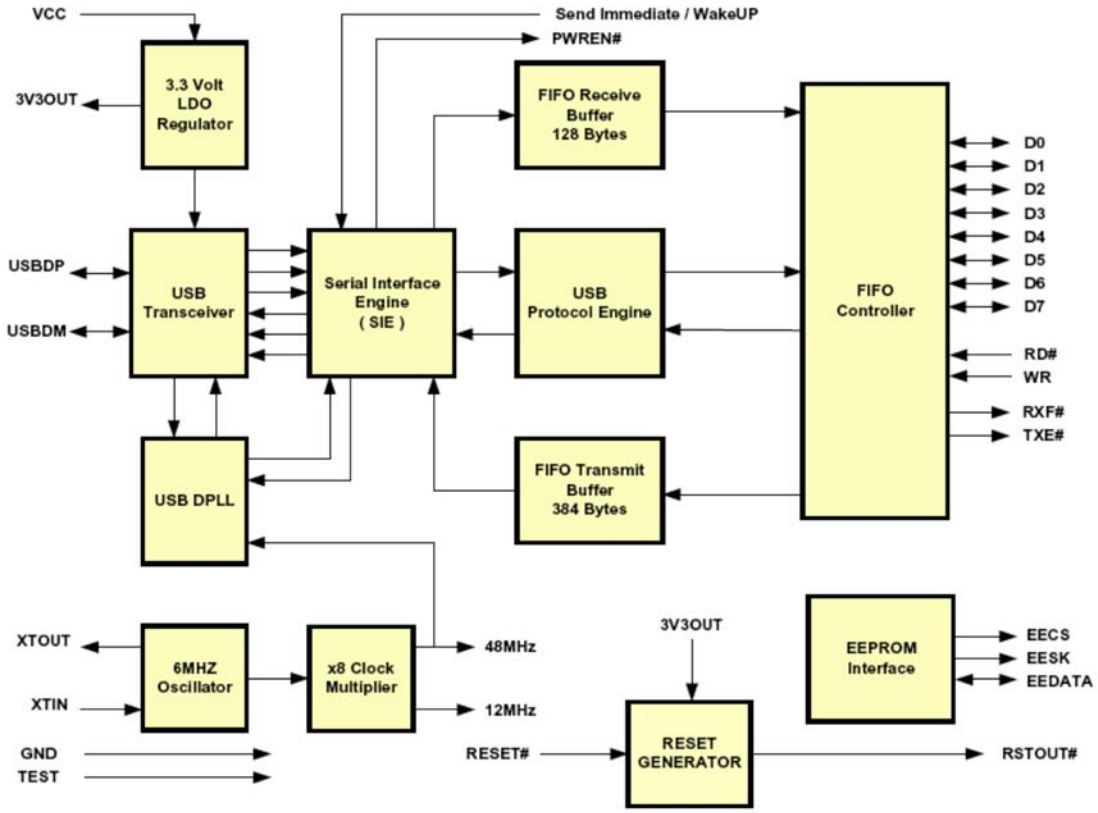


Figura 4.4: Schema a blocchi del dispositivo

USB DPLL (Digital PLL):

Il DPLL , Digital PLL (Phase-Locked-Loop) è un sistema di controllo il cui obiettivo è la sincronizzazione della fase e della frequenza, di un segnale generato localmente a quello di uno posto come ingresso. In questo sistema ha la funzione di *Clock-data recovery*, ovvero deve sincronizzare il segnale USB, di ingresso, con il segnale a frequenza di 48MHz generato localmente in modo da recuperare l'informazione sul clock e il dato e fornirli al SIE.

Engine (SIE):

Esegue la conversione seriale-parallelo del dato durante la fase di ricezione e parallelo-seriale in quella di trasmissione. Codifica i dati in uscita secondo la logica NRZI (Not Return to Zero Inverted) e decodifica i dati in ingresso. Genera il codice di controllo CRCs per i dati in uscita e verifica quelli in ingresso. Quando il segnale di ingresso (Send Immediate/ WakeUp) SI/WU=0, se PWREN#=1, cioè se l'USB è in suspend mode, e se la funzione di WakeUp all' interno della EEPROM è abilitata, genera una

richiesta di WakeUp sul bus USB. Durante le normali operazioni (PWREN#=0) se SI/WU=0 ogni volta che si scrive un dato nel buffer di trasmissione, allora si genera una richiesta di Bulk-IN dai driver senza che questi si curino della lunghezza del pacchetto. Questo sistema viene utilizzato per ottimizzare la velocità di trasferimento in molte applicazioni. Se questo pin non è utilizzato va collegato a VCCIO

USB Protocollo Engine:

Gestisce i dati da e per il dispositivo, tutte le richieste generate dall' Host controller, secondo il protocollo USB e i segnali di controllo del FIFO Controller.

FIFO Receive Buffer (128 Bytes):

I dati spediti dall'host al dispositivo vengono memorizzati in questo buffer. Per rimuovere i dati è necessaria un'operazione di lettura attraverso il segnale RD#

FIFO Transmit Buffer (384 Bytes):

I dati da spedire all' host vengono scritti in questo buffer , tramite WR#. E' poi l'host stesso che pensa a rimuoverli inviando una richiesta dati al dispositivo.

FIFO Controller:

Attraverso i segnali RD# e WR# gestisce il trasferimento dei dati nei buffer di ricezione e trasmissione , mentre con TXE# e RXF# segnala la possibilità scrivere e leggere dati nei buffer.

EEPROM Interface:

Il chip può interfacciarsi con una EEPROM (93C46) esterna, permettendoci di personalizzare:

- ***VID(Vendor ID)***
- ***PID(Product ID)***
- ***Serial Number***

- ***Manufacturer***
- ***Product Descriptor***
- ***Power Descriptor***
- ***Remote Wake up***
- ***USB version***

La EEPROM è programmabile *on board* attraverso l' applicazione ***MProg3.0*** distribuita da FTDi chip o attraverso le funzioni della libreria *dll* fornita con i driver D2XX . MProg3.0 per funzionare necessita l'installazione dei driver D2XX e permette di leggere, scrivere e salvare le configurazione su un ***file template*** rendendo più semplice la programmazione in serie di molte EEPROM.

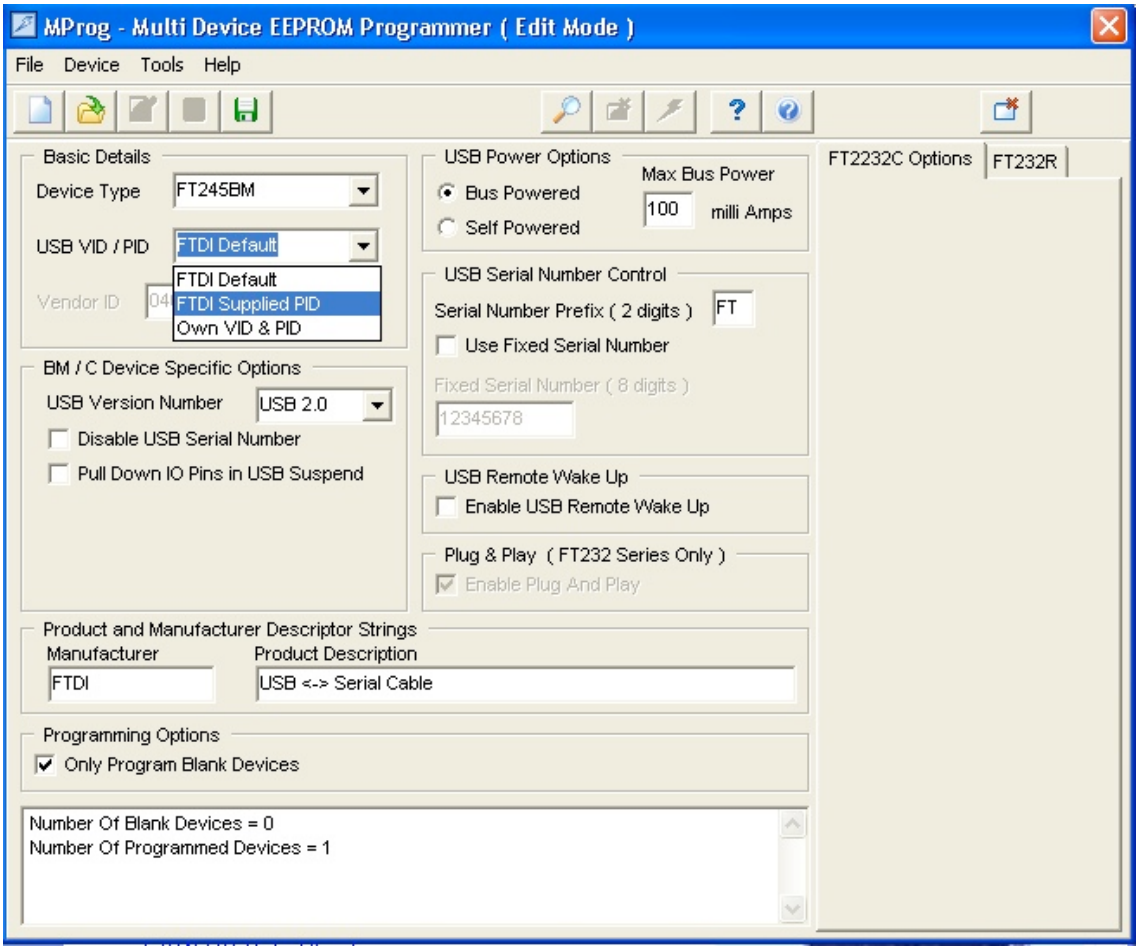


Figura 4.5: Come si presenta Mprog3.0. EEPROM in configurazione di default

4.2.3 Lettura e scrittura dati sul bus USB

I dati inviati dall' host all' FT245BM vengono memorizzati nel *FIFO Receive buffer*.
Se il buffer non è vuoto (RXF#=0) il dato è disponibile sul bus dati (D0-D7)in seguito ad un fronte di discesa ed uno di salita, istante di campionamento, del segnale RD#. Successivamente RXF# è ianttivo (H) per 80 ns. Il tempo impiegato a fare due operazioni di lettura successive è pari a T1+T5+T6=50+25+80=155ns.

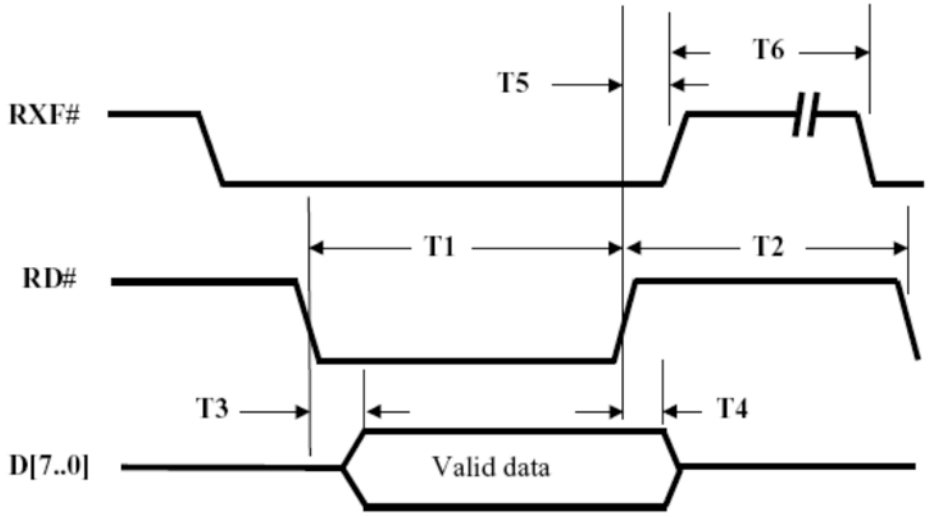


Figura 4.6: Ciclo di lettura dal buffer di ricezione(invio dati dal PC)

Tempo	Descrizione	Min	Max	unit
T1	T_{WL}	50	-	ns
T2	T_{WH}	50	-	ns
T3	T_{ACC}	20	50	ns
T4	T_{H}	0	-	ns
T5	T_{LH}	0	25	ns
T6	$T_{INACTIVE}$	80	-	ns

I dati inviati dall' FT245BM all' host vengono prima memorizzati nel *FIFO Transmit buffer*. Se TXE#=1 non è possibile scrivere dati nel buffer, altrimenti sarà necessario un fronte di salita ed uno di discesa ,istante di campionamento del dato,del segnale WR. Al

termine della scrittura TXE# è inattivo (H) per 80ns. Il tempo impiegato a fare due operazioni di scrittura successive è pari a T7+T11+T12= 50+25+80=155ns .

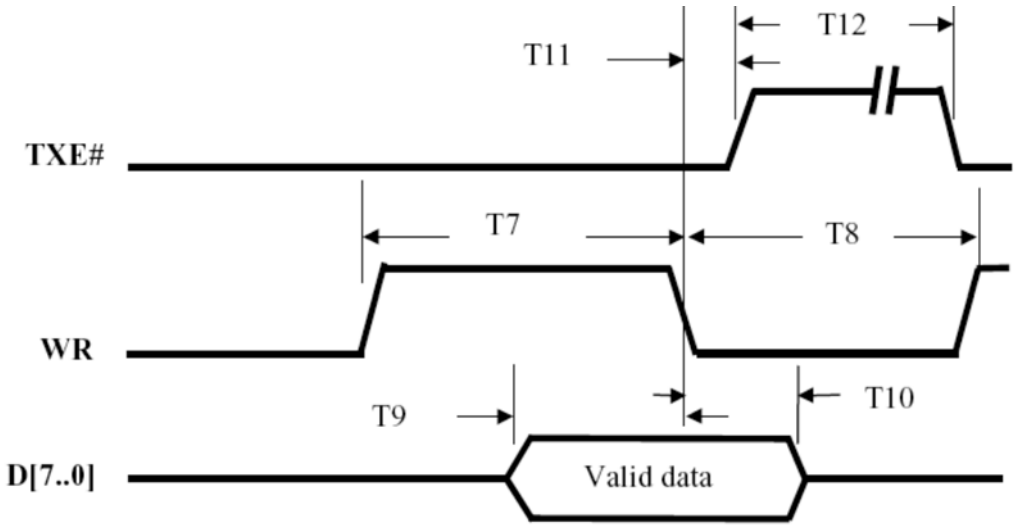


Figura 4.7: Ciclo di scrittura del buffer di trasmissione(invio dati al PC)

Tempo	Descrizione	Min	Max	unit
T7	T_{WH}	50	-	ns
T8	T_{WL}	50	-	ns
T9	T_{SU}	20	-	ns
T10	T_{H}	0	-	ns
T11	T_{LH}	5	25	ns
T12	$T_{INACTIVE}$	80	-	ns

In linea puramente teorica,se il buffer non è vuoto e se il dispositivo genera dati alla stessa velocità con cui l'host li acquisisce sarebbe possibile scambiare dati ad una velocità di 1/155n = 6,451MB/s.

Ovviamente non è possibile perché la comunicazione USB full-speed ha una velocità massima di 12Mbit/s(1,5MByte/s) e perché tali velocità difficilmente saranno uguali. Nel caso in cui la velocità di generazione dei dati è superiore a quella di acquisizione si avrà il cosiddetto "*problema del produttore e del consumatore*". La differenza di velocità tra produttore e consumatore viene "ammortizzata" dal buffer di

trasmissione/ricezione e non appena questo si riempie costringe il produttore ad aspettare nuovamente il consumatore. Può decidere di farlo finché quest'ultimo non ha svuotato il buffer o fino a che non ha acquisito un byte con conseguente rallentamento, in ogni caso, della comunicazione tra i due. Nel caso specifico, questo problema viene esasperato, perché si vuole realizzare un sistema in grado di misurare una grandezza fisica che varia nel tempo per un periodo prolungato nel tempo senza che vi sia una perdita di dati (il produttore aspetta il consumatore). Quindi, o si ha un produttore molto lento, cioè un sistema di acquisizione a bassa frequenza di campionamento, con conseguente riduzione della banda del segnale analogico, o si ha un consumatore molto veloce, cioè un sistema di trasferimento ed elaborazione dati molto veloce. La prima soluzione non ci permette di rispettare le specifiche di progetto. La seconda ci consente di arrivare ad ottenere il risultato prefissato, stando però ben attenti ad adottare tutta una serie di accorgimenti durante la fase di realizzazione del Client Software(par. 5.1)

4.2.4 Come ottenere la massima velocità di comunicazione.

Come abbiamo detto, la velocità di comunicazione dipende:
- dal driver
- dal client software incaricato di acquisire ed elaborare i dati
- dalla velocità del calcolare.

Il driver più performante in questo senso è il D2XX poiché permette una velocità di comunicazione pari a 1MByte/s. Non basta adoperare questo driver per velocizzare la comunicazione ma è necessario ottimizzare il trasferimento tra host e FT245BM in modo da ridurre il *"problema del produttore e del consumatore"*. Ogni volta che un dato, sottoforma di pacchetti, viene inviato dal chip all' host e viceversa, viene momentaneamente bufferizzato in un'area di memoria messa a disposizione dal driver. A gestire le richieste di invio e ricezione dati ci pensa ***l'USB scheduler*** inserendole nella lista dei task che l'host controller deve effettuare. La lista dei task si aggiorna ad ogni frame , cioè ogni 1ms (vedi par. 3.4), ad ogni invio o ricezione di un dato è quindi sempre associato un certo *overhead*. Dopo quanto detto, risulta evidente che l'invio di 1 byte alla volta limiterebbe notevolmente la velocità di comunicazione. La dimensione del dato, in byte, da inviare assume quindi un ruolo importante e varia a seconda

dell'applicazione. Nelle applicazioni ad alta velocità viene massimizzato, in quelle real-time, come il trasferimento audio a 115200 Baud, viene minimizzato per limitare l'effetto "*jerky"*(tremolio). Nel driver D2XX questa dimensione massima del dato è di 64KByte ed è programmabile attraverso la funzione ***FT_SetUSBParameters*** presente nella libreria *dll*. Non appena si accede al dispositivo, il driver in questione, alloca due buffer, uno di lettura e uno di trasmissione. Il buffer di trasmissione ha dimensione pari alla dimensione massima del dato mentre il buffer di lettura ha una dimensione che è basata su quest'ultima ma non è uguale. Il driver esegue le richieste di trasferimento dati, di dimensione qualsiasi, fino a che queste non raggiungono la dimensione massima impostata dalla funzione *FT_SetUSBParameters*. In seguito copia i dati dal buffer di trasmissione in quello di lettura. Se il buffer di lettura è pieno il driver blocca qualsiasi attività di trasferimento dati fino a che il buffer non viene svuotato. I dati vengono rimossi dal buffer di lettura attraverso un'operazione di lettura ***FT_Read*** eseguita dal *client software.*

I dati sono trasferiti attraverso l'USB come una serie di pacchetti da 64byte. Ogni pacchetto contiene 2 byte riservati all' FT245BM e 62 byte sono dati. Se quindi la dimensione massima del dato è 4 KB allora i dati trasferiti saranno 3968 byte mentre se invece è 64KB i dati trasferiti saranno 63488 byte.

Più in generale i dati effettivamente trasferiti sono:

$$ByteTrasferiti = D\max - \left(2 \cdot \frac{D\max}{64}\right)$$

Dmax= dimensione massima del dato trasferito in byte

Il buffer di lettura avrà dimensione pari a quella degli effettivi byte trasmessi .

Il massimo rendimento della comunicazione si ha quando la massima quantità di dati è trasferita con il minor numero possibile di transazioni sull'USB. Tale condizione è verificata quando il client software preleva i dati dal buffer di lettura quando sono un multiplo degli effettivi byte trasferiti. Ad esempio, se il buffer di trasmissione è 64KB (dimensione massima del dato), allora il client software dovrà prelevare dati a multipli di 63488 byte .

Il ***latency timer*** è un timer interno all' FT245BM che misura il tempo intercorso tra due istanti successivi all' invio di dati verso il PC. La condizione di *time out* di questo timer fa si che il dato venga mantenuto nel buffer e possa essere letto dal *client software*. Il valore di default di questo timer è 16 ms ed è impostabile attraverso la funzione ***FT_Set LatencyTimer***. Se ad esempio si trasferiscono 5 pacchetti, come in figura 4.8, il latency timer si resetta ad ogni invio di un pacchetto. Dopo l'invio dell'ultimo pacchetto è necessario aspettare(16ms) che il *latency timer* vada in *time out* prima di poter accedere al dato attraverso il buffer di lettura. Ogni trasferimento è quindi soggetto ad un ritardo pari a 16 ms .

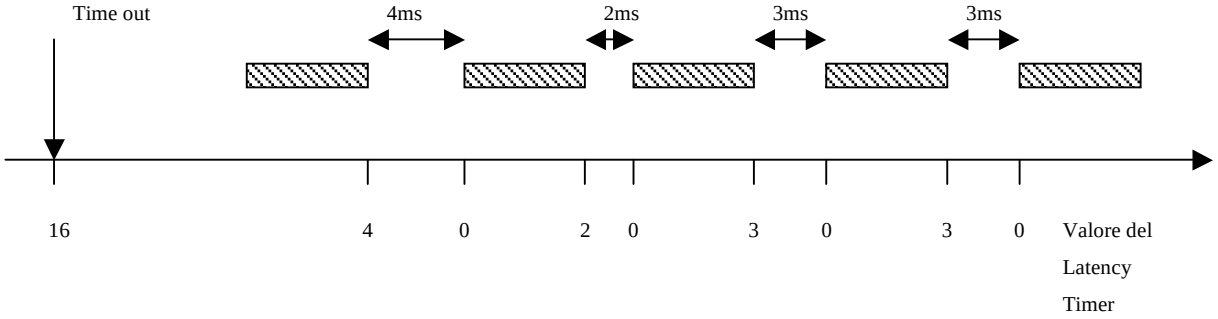


Figura 4.8: Il latency timer misura l'istante intercorso tra l'invio di due dati. Solo dopo l'invio dell'ultimo dato si verifica la condizione di time out del timer. Solo allora è possibile accedere al buffer di lettura.

Questo metodo di trasferimento del dato risulta poco efficiente e non viene usato in quei sistemi che trasferiscono grandi quantità di dati ad alta velocità.
Esiste quindi una relazione tra il valore del *latency timer*, la velocità di trasmissione dei dati e la l'istante in cui questi vengono resi disponibili al *client software*.
Se ad esempio il latency timer =16ms allora la velocità di trasmissione dati dall' FT245BM al PC è:

$$Data-rate = \frac{62}{LatencyTimer} = \frac{62}{16^{-3}} = 3875\left[\frac{Byte}{s}\right]$$

Se l'FT 245BM trasmette ad una velocità superiore a 3,875KB/s allora si evita, sempre, che il latency timer vada in time out, avendo così immediatamente a disposizione il dato

nel buffer di lettura. Al contrario si ha l'introduzione di un ritardo di acquisizione del dato pari a 16ms.
Nel caso specifico, l'FT245BM trasmette dati ad una velocità di 100KB/s, sarà quindi sufficiente impostare il timer al valore minimo impostabile : 1 o 2ms poiché

$$LatencyTimer = \frac{62}{Data - rate} = \frac{62}{100^3} = 620[\mu s]$$

Se il *latency timer* =2 ms la velocità di comunicazione dovrà essere superiore a :

$$Data - rate = \frac{62}{LatencyTimer} = \frac{62}{2^{-3}} = 31000\left[\frac{Byte}{s}\right]$$

Condizione verificata poiché il DAQ acquisisce e trasmette ad una velocità di almeno 80KB/s.

Infine, per completare il quadro della situazione, è necessario tenere presente che anche il protocollo di comunicazione tra FT245BM e PC influisce sulla velocità di scambio di dati e deve essere scelto adeguatamente in funzione dell'applicazione.
Nel caso specifico, l'FT245BM trasferisce periodicamente e in maniera continua al PC una grossa mole di dati a velocità elevata. Dato che il sistema poi misura il variare di una grandezza fisica nel tempo e ne trasmette il valore al PC, la mancanza di controlli sull'integrità del dato trasferito risulterebbe superflua se non inutile, vista l'impossibilità di ritrasmettere il dato. L' isochronous mode è quindi la modalità di trasferimento più adeguata (vedi anche par. 3.6) ed è possibile selezionarla programmando adeguatamente la EEPROM attraverso MProg3.0 . Il file *template* si trova assieme alla documentazione completa in formato elettronico. Va comunque precisato che questo tipo di comunicazione viene implementata di default , ovvero la EEPROM montata sulla chip board è già stata programmata per far funzionare l'FT245BM in questa modalità.

4.3 Il driver D2XX

L'architettura del driver D2xx è basata su quella dei WDM (Windows Drive Model) la cui struttura è rappresentata in figura.

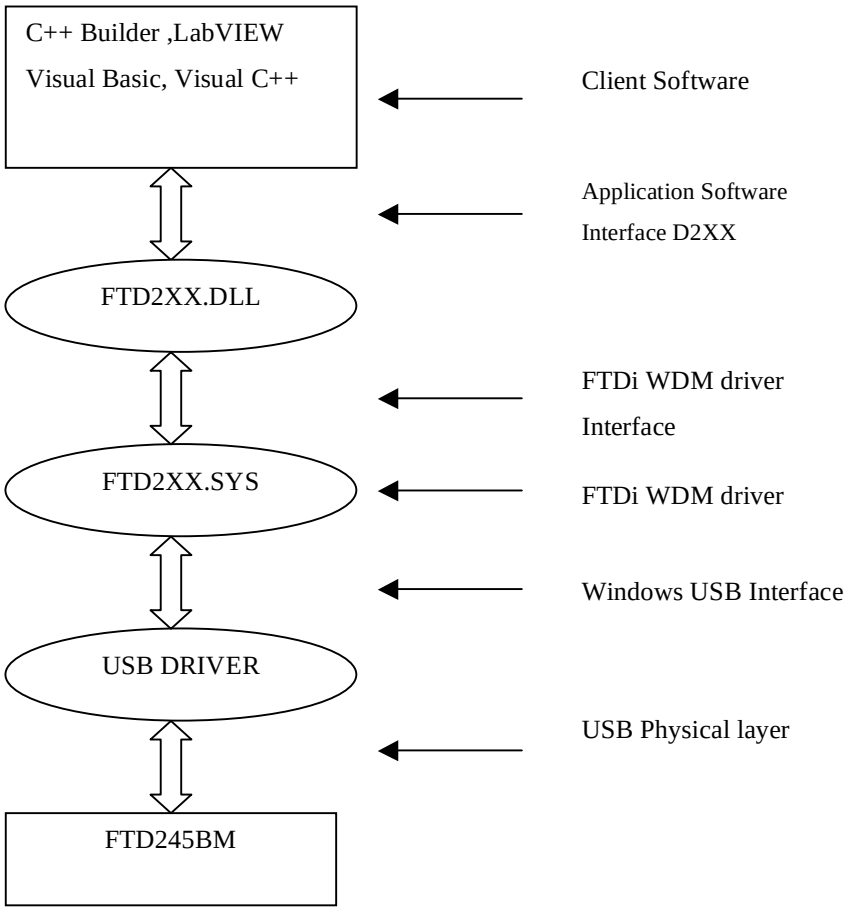


Figura 4.9: Architettura del driver D2XX

Andremo ad analizzare nel dettaglio alcune funzioni messe a disposizione dal driver D2XX nel file **FTD2XX.DLL** [13] che ci ritorneranno utili nella realizzazione dello strumento con LabVIEW.

4.3.1 FT_ListDevices:

Fornisce informazioni sui dispositivi connessi come il numero di dispositivi connessi, il loro serial number, il descrittore del dispositivo e l'IDs.
La sintassi è la seguente:

FT_STATUS **FT_ListDevices** (PVOID *pvArg1*, PVOID *pvArg2*, DWORD *dwFlags*)

Dove:
FT_STATUS: Stato di esecuzione dell' operazione. Se uguale a FT_OK allora l'operazione è stata eseguita con successo, altrimenti viene generato il codice di errore.

FT_STATUS (DWORD)
FT_OK = 0
FT_INVALID_HANDLE = 1
FT_DEVICE_NOT_FOUND = 2
FT_DEVICE_NOT_OPENED = 3
FT_IO_ERROR = 4
FT_INSUFFICIENT_RESOURCES = 5
FT_INVALID_PARAMETER = 6
FT_INVALID_BAUD_RATE = 7
FT_DEVICE_NOT_OPENED_FOR_ERASE = 8
FT_DEVICE_NOT_OPENED_FOR_WRITE = 9
FT_FAILED_TO_WRITE_DEVICE = 10
FT_EEPROM_READ_FAILED = 11
FT_EEPROM_WRITE_FAILED = 12
FT_EEPROM_ERASE_FAILED = 13
FT_EEPROM_NOT_PRESENT = 14
FT_EEPROM_NOT_PROGRAMMED = 15
FT_INVALID_ARGS = 16
FT_NOT_SUPPORTED = 17
FT_OTHER_ERROR = 18

pvArg1, pvArg2: argomenti, il cui significato varia a seconda di dwFlags.
dwFlags: decide il formato del valore di ritorno e può essere come di seguito:

FT_LIST_NUMBER_ONLY:pvarg1 è interpretato come a un puntatore ad una locazione DWORD(4byte) nella quale è memorizzato il numero di dispositivi collegati. Ad esempio: (&indirizzo puntatore *contenuto puntatore)

```
FT_STATUS ftStatus;
DWORD numDevs;
ftStatus = FT_ListDevices(&numDevs,NULL,FT_LIST_NUMBER_ONLY);
if (ftStatus == FT_OK) {
// FT_ListDevices OK, number of devices connected is in numDevs
}
else {
// FT_ListDevices failed
}
```

FT_OPEN_BY_SERIAL_NUMBER: fornisce la stringa con il serial number del dispositivo.

FT_OPEN_BY_DESCRIPTOR: fornisce la descrizione del dispositivo sottoforma di stringa.

FT_OPEN_BY_LOCATION: fornisce la locazione ID del dispositivo. Questo flag non è valido sotto Windows CE e Linux.

FT_LIST_BY_INDEX | FT_OPEN_BY_DESCRIPTOR: in questo caso pvArg1 è l'indice del dispositivo e pvArg2 è un puntatore ad un buffer che contiene la stringa del descrittore. L'indice base è lo zero, nel caso indice errato la funzione ritorna FT_DEVICE_NOT_FOUND.

Esempio in LabVIEW:

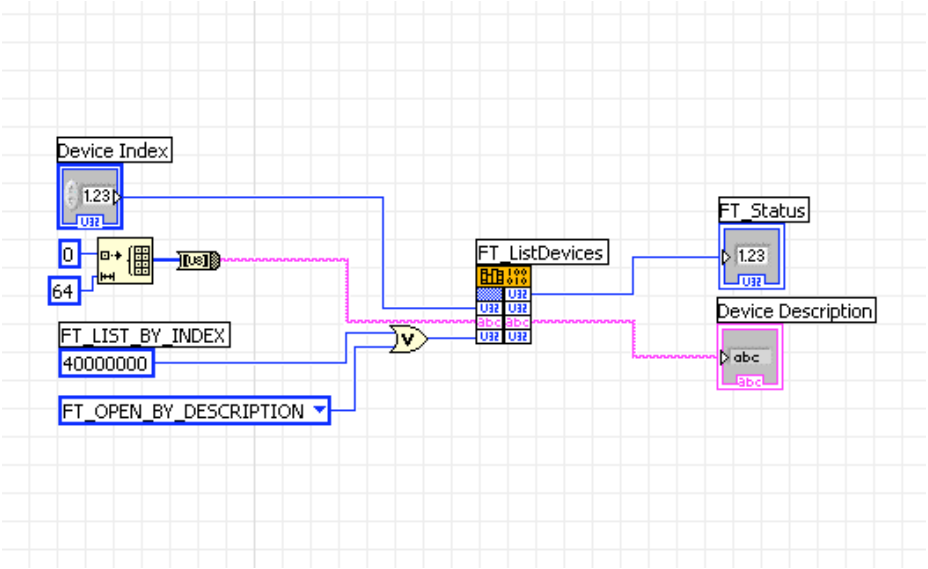


Figura 4.10 : Esempio in LabVIEW che cerca il dispositivo di indice =#Device Index e fornisce il descrittore del dispositivo stesso sottoforma di stringa.

FT_LIST_BY_INDEX | FT_OPEN_BY_SERIAL_NUMBER: in questo caso pvArg1 è l'indice del dispositivo e pvArg2 è un puntatore ad un buffer che contiene il serial numeber. L'indice base è lo zero, nel caso indice errato la funzione ritorna FT_DEVICE_NOT_FOUND.

Esempio in C:

```
DWORD devIndex = 0; // Primo dispositivo
char Buffer[64]; // Spazio più che sufficiente.
ftStatus=FT_ListDevices((PVOID)devIndex,Buffer,FT_LIST_BY_INDEX|FT_OPEN_
BY_SERIAL_NUMBER);
if (ftStatus == FT_OK) {
// FT_ListDevices OK, serial number is in Buffer
}
else {
// FT_ListDevices failed
}
```

FT_LIST_ALL | FT_OPEN_BY_DESCRIPTOR: in questo caso pvAgr1 è un puntatore ad un array di puntatori ai buffer che contengono la stringa del descrittore,

mentre pvArg2 è un puntatore ad una locazione DWORD che contiene il numero di dispositivi connessi. Notare che l'ultimo dei dispositivi

I flags sono così definiti:

FT_LIST_NUMBER_ONLY = 0x80000000
FT_LIST_BY_INDEX = 0x40000000
FT_LIST_ALL = 0x20000000
FT_OPEN_BY_SERIAL_NUMBER = 1
FT_OPEN_BY_DESCRIPTION = 2
FT_OPEN_BY_LOCATION = 4

Essendo questa la prima funzione dei driver si ritiene opportuno fare qualche esempio al riguardo.
Se per esempio vogliamo acquisire il descrittore di tutti i dispostivi collegati sarà necessario scrivere :

```
char *BufPtrs[3];   // pointer to array di 3 pointers
char Buffer1[64];   // buffer per description del primo device
char Buffer2[64];   // buffer per description del secondo device
// initialize the array of pointers
BufPtrs[0] = Buffer1;
BufPtrs[1] = Buffer2;
BufPtrs[2] = NULL; // l'ultima  entry può essere NULL
ftStatus =
FT_ListDevices(BufPtrs,&numDevs,FT_LIST_ALL|FT_OPEN_BY_DESCRIPTION);
if (ftStatus == FT_OK) {
// FT_ListDevices OK, product descriptions are in Buffer1 and Buffer2, and
// numDevs contains the number of devices connected
}
else {
// FT_ListDevices failed
}
```

4.3.2 FT_OpenEx:

Apre il dispositivo specificato e fornisce l'handle del dispositivo. Il dispositivo viene specificato attraverso il serial numer, il device descriptor o la locazione. Questa funzione permette di aprire simultaneamente più dispositivi poiché questi vengono distinti attraverso il serial number o il device descriptor o la locazione. La sintassi è la seguente:

FT_STATUS **FT_OpenEx** (PVOID *pvArg1*, DWORD *dwFlags*, FT_HANDLE **ftHandle*)

Dove:

pvArg1: il significato dipende da dwFlags anche se in generale può essere interpretato come un puntatore ad una stringa.

ftHandle: è un puntatore ad una variabile dove viene memorizzato il tipo dell'Handle ed utile per poter eseguire tutte le altre funzioni di seguito elencate.

dwFlags: possono essere di tre tipi; FT_OPEN_BY_DESCRIPTOR , FT_OPEN_BY_SERIAL_NUMBER , FT_OPEN_BY LOCATION. Nel primo caso pvArg1 è una stringa di caratteri, nel secondo una stringa di numeri ed infine una numero che rappresenta la location ID del dispositivo.

Se ad esempio si vuole aprire un dispositivo il cui device descriptor è: "USB Serial Converter"

```
ftStatus = FT_OpenEx("USB Serial
Converter",FT_OPEN_BY_DESCRIPTION,&ftHandle1);
if (ftStatus == FT_OK) {
// success - device with device description "USB Serial Converter" is open
}
else {
// failure
}
```

Nel linguaggio LabVIEW sarà:

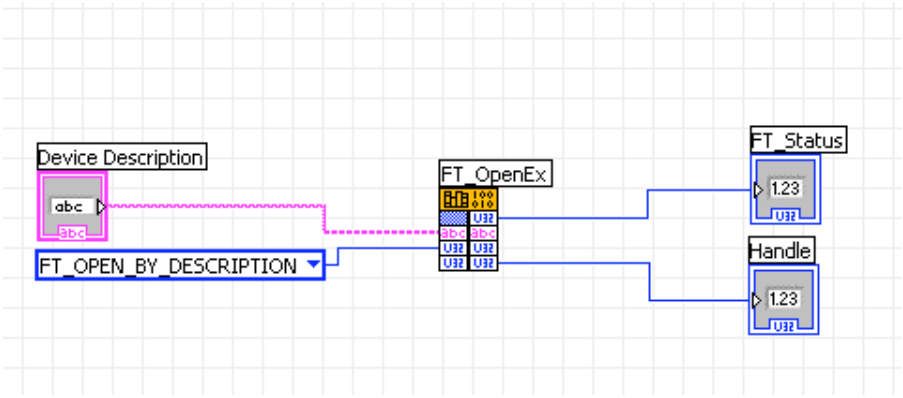


Figura 4.11: Esempio di apertura del dispositivo attraverso il descrittore dello stesso. Aperto il dispositiv, questi genera l'handle utile per tutte le funzioni elencate di seguito

4.3.3 FT_ResetDevice:

Permette di resettare il dispositivo. Non è specificato altro su questo tipo di funzione, ma è da considerarsi molto utile nel caso di mal funzionamenti del dispositivo.
La sintassi della funzione è:

FT_STATUS **FT_ResetDevice** (FT_HANDLE *ftHandle*)

ftHandle: Handle del dispositivo, generato precedentemente dalla funziona FT_OpenEx.

4.3.4 FT_SetFlowControl:

Questa funzione viene impiegata nel convertitore FT232 , un convertitore USB-RS232, per impostare il flusso di controllo dei dati.
Di per se' questa funzione non è necessaria, viene spontaneo domandarsi perché allora questa funzione viene menzionata. Durante la fase di sviluppo del client software ci siamo resi conto che è presente un bug nel driver che ci costringe a impostare il controllo RTS/CTS per evitare di perdere i dati trasmessi dall'FT245 al PC.

FT_STATUS **FT_SetFlowControl** (FT_HANDLE *ftHandle,* USHORT usFlowControl, UCHAR u*Xon,*UCHAR u*Xoff*)

Dove:

ftHandle: handle del dispositivo precedentemente generato dall'FT_OpenEx.

uXon e uXoff: caratteri usati nel controllo di flusso Xon/Xoff

usFlowControl: permette di impostare i vari controlli sul flusso dati tra DTE(Data terminale Equipment) e DCE (Data Control Equipment).

Può essere impostato come **FT_FLOW_NONE(**0x00): nessun controllo.

FT_FLOW_RTS_CTS (0x100): controllo RTS/CTS

FT_FLOW_XON_XOFF(0x400): controllo Xon/Xoff

FT_FLOW_DTR_DSR(0x200): controolo DTR/DSR:

4.3.5 FT_SetLatencyTimer:

Come già accennato in precedenza il latency timer deve essere impostato ad un valore opportuno. Per farlo ci si serve di questa funzione la cui sintassi è:

FT_STATUS **FT_SetLatencyTimer** (FT_HANDLE *ftHandle*, UCHAR *ucTimer*)

Dove:

ftHandle: handle del dispositivo precedentemente generato dall'FT_OpenEx.

ucTimer: valore in millisecondi a cui viene impostato il latency timer. Il valore ammesso è compreso tra 2 e 255.

Esempio:

```
HANDLE ftHandle;
FT_STATUS ftStatus;
UCHAR LatencyTimer = 10;

ftStatus = FT_Open OpenEx("USB
SerialConverter",FT_OPEN_BY_DESCRIPTION,&ftHandle);
```

```
if(ftStatus != FT_OK) {
// FT_Open failed
return;
}
ftStatus = FT_SetLatencyTimer(ftHandle, LatencyTimer);
if (ftStatus == FT_OK) {
// LatencyTimer set to 10 milliseconds
}
else {
// FT_SetLatencyTimer FAILED!
}
FT_Close(ftHandle);
```

4.3.6 FT_SetUSBParameters:

Abbiamo già visto l'importanza di questa funzione quando abbiamo parlato di come si può ottenere la massima velocità di trasmissione tra il dispositivo e il PC. Questo perché ci permette di dimensionare il buffer allocato dal driver in fase di scrittura (input) e di lettura(out) da parte del dispositivo.

FT_STATUS **FT_SetUSBParameters** (FT_HANDLE *ftHandle*, DWORD *dwInTransferSize*,DWORD *dwOutTransferSize*)

Dove:
ftHandle: handle del dispositivo precedentemente generato dall'FT_OpenEx
dwInTransferSize: dimensione del dato durante le richieste di scrittura del dispositivo sull' USB
dwOutTransferSize: dimensione del dato durante le richieste di lettura del dispositivo.

La dimensione del dato può essere compresa tra 64byte e 64Kb , a multiplid i 64byte, e di default è pari a 4096 bytes. L'esecuzione di questa funzione ha effetto immediato e

provoca la perdita di tutti i dati presenti nel buffer allocato dal driver. Attualmente è supportato solo dwInTrasferSize.

4.3.7 FT_Purge:

Questa funzione permette di cancellare il contenuto dei buffer di ricezione e trasmissione dell' FT245. E' conveniente usarla prima di qualsiasi trasmissione dati dal dispositivo al PC per evitare che nella successiva trasmissione dati vi sia qualche dato spurio.

FT_STATUS **FT_Purge** (FT_HANDLE *ftHandle,* DWORD *dwMask*)

ftHandle: handle del dispositivo precedentemente generato dall'FT_OpenEx
dwmask: permette di scegliere quale buffer cancellare. Può essere impostato come
FT_PURGE_RX (0x1): si cancella il buffer di ricezione dell' FT245BM
FT_PURGE_TX(0x02): si cancella il buffer di trasmissione dell' FT245BM.
FT_PURGE_RX | FT_PURGE_TX: si cancella il buffer di trasmissione e ricezione dell' FT245BM.

4.3.8 FT_GetQueueStatus:

Questa funzione permette di monitorare quanti byte effettivi sono stati trasmessi dal dispositivo al PC. In pratica consente di monitorare lo stato del buffer di lettura allocato dal driver in modo da poterlo svuotare, attraverso un'operazione di lettura, non appena si riempie ed evitare così che la trasmissione di dati dal dispositivo si blocchi.

FT_STATUS **FT_GetQueueStatus** (FT_HANDLE *ftHandle,* LPDWORD *lpdwAmountInRxQueue*)

Dove:

ftHandle: handle del dispositivo precedentemente generato dall'FT_OpenEx

lpdwAmountInRxQueue: puntatore ad una variabile DWORD che contiene il numero di byte presenti nel buffer di lettura allocato dal driver.

4.3.9 FT_Read:

Questa funzione permette di leggere i dati trasmessi dal dispositivo al PC e memorizzati all'interno del buffer di lettura allocato dal driver.La sintassi della funzione è:

FT_STATUS **FT_Read** (FT_HANDLE *ftHandle*, LPVOID *lpBuffer*, DWORD *dwBytesToRead*, LPDWORD *lpdwBytesReturned*)

Dove:

ftHandle: handle del dispositivo precedentemente generato dall'FT_OpenEx

lpBuffer: puntatore al buffer che riceve i dati dal dispositivo.

dwBytesToRead: numero di bytes che devono essere letti dal dispositivo. Questi vengono determinato dalla funzione **FT_GetQueueStatus**

lpdwBytesReturned: puntatore ad una variabile DWORD che riceve il numero di bytes letti dal dispositivo.

Esempio in LabVIEW:

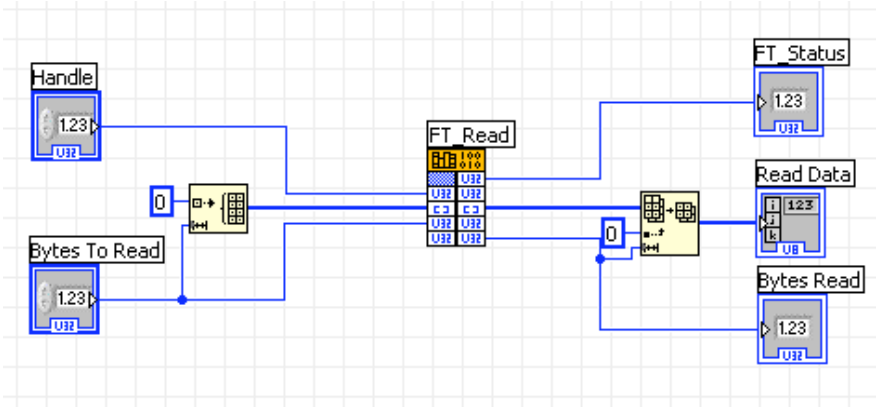


Figura 4.12: Esempio in LabVIEW che utilizza la funzione FT_Read per leggere dal dispositivo di cui si conosce l'handle e i byte da leggere determinati con FT_GetQueueStatus

4.3.10 FT_Close:

Questa funzione permette di chiudere del dispositivo e rilasciarlo ; la sintassi è:

FT_STATUS **FT_Close** (FT_HANDLE *ftHandle*)

Con questa funzione si chiude la parte che tratta delle funzioni del driver D2XX. Tale sezione non ha la pretesa di sostituire il manuale ma di illustrare le funzioni significative e necessarie alla stesura del client software con LabVIEW.

4.4 Collegamento elettrico dell' FT245BM

L' FT245BM può essere alimentato sia direttamente dalla porta USB che da circuiti di alimentazione esterna ,che provvedono a fornire i 5 V necessari al funzionamento. Per rendere il sistema di misura più semplice all'utente finale , meno complesso, costoso ed indipendente dal trasduttore, si è deciso di adottare la ***configurazione Bus-Powered*** [14]il cui schema elettrico è riportato in figura 4.14 è il medesimo della chip-board DLP-USB245M. Per completare la configurazione *Bus-Powered* è necessario collegare i pin della chip-board : 3,10,11 e 12 tra di loro. Il partitore resistivo R10-R11 serve per distinguere le chip-board che montano l'FT232M da quelle che montano l'FT245BM.In questo caso R11 non è presente mentre R10 vale 100K(boardID=0).
La chip-board è stata comprata e poi montata su un comune zoccolo DIL 12x2 presente sulla scheda DAQ (Digital Acquisition).

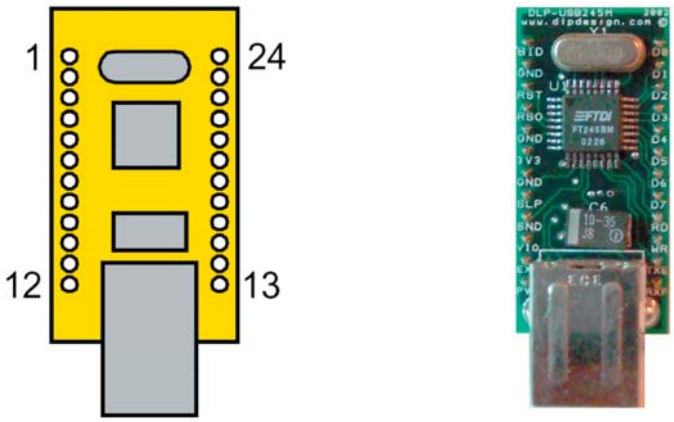


Figura 4.13: Pinout della chip-board DLP-USB245M

E' necessario tenere presente che vanno inserite delle resistenze (27-47Ω) in serie a RD# e WR per minimizzare il ringing ed evitare che i segnali no vengano male interpretati con conseguente perdita di dati.

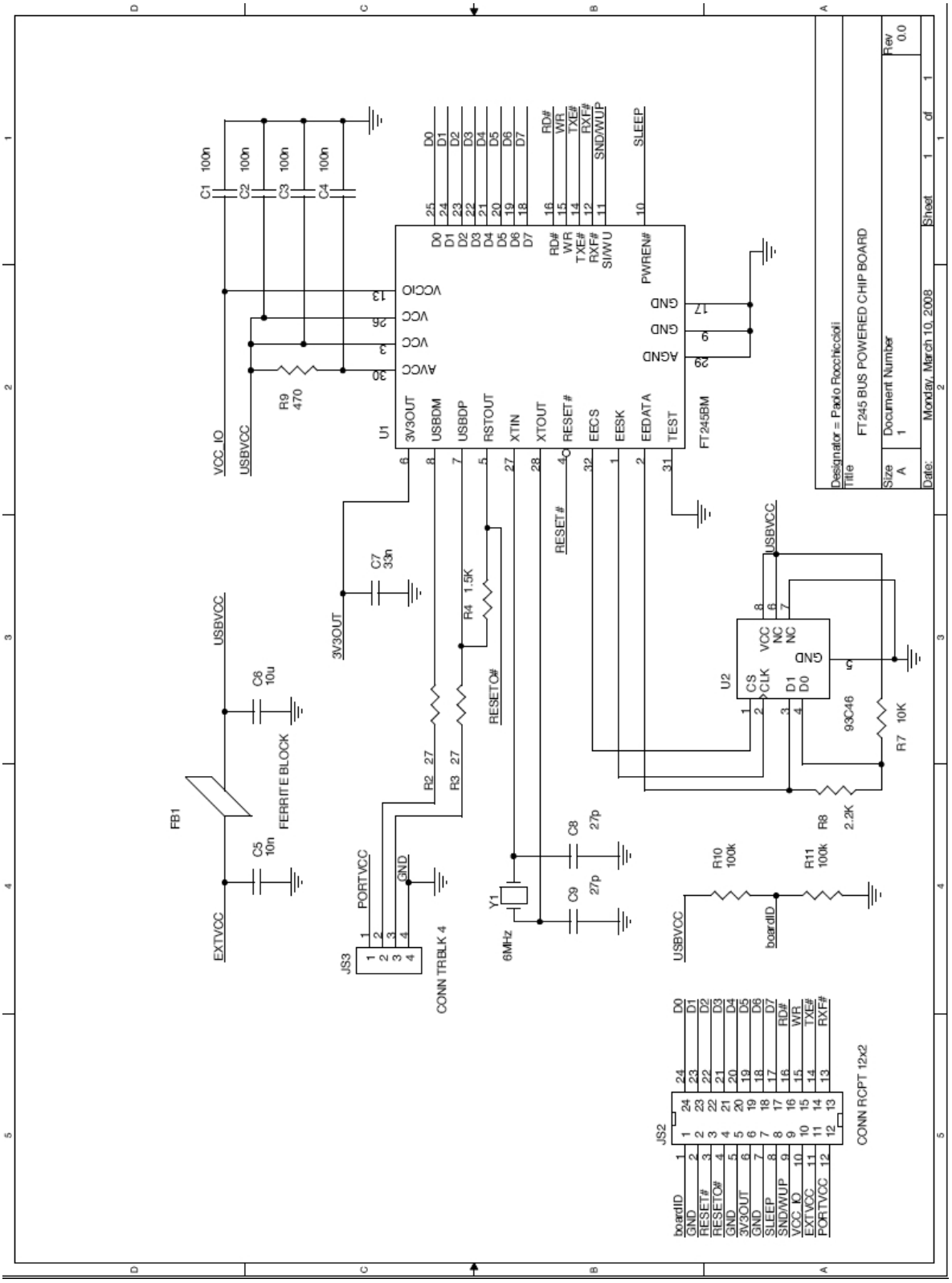


Figura 4.14: Schema elettrico della chip board DLP-USB245M

4.5 Il convertitore analogico digitale

Il convertitore AD in questione deve avere una $f_{sample} \geq 80KHz$ e risoluzione su 8bit paralleli, dato che l'FT245BM ha un'interfaccia di tipo parallelo. Esistono molto convertitori con queste caratteristiche; si è optato per il convertitore **AD7825BRZ** prodotto dalla **Analog Device** perché è possibile averne dei campioni gratuiti in modo così da ridurre i costi di *prototyping* e realizzazione.
Queste le principali caratteristiche dell' AD7825BRZ [15]:

- alimentazione 3÷5V
- 8 bit (paralleli)
- 4 canali
- tempo di conversione di 420ns (2MSPS)
- range della tensione di ingresso compreso tra 0-2V (V_{DD}=3V) e 0-2,5V (V_{DD}=5V)
- semplice interfaccia di controllo
- package SMD di tipo SOIC_W (passo 50mils=1,27mm)

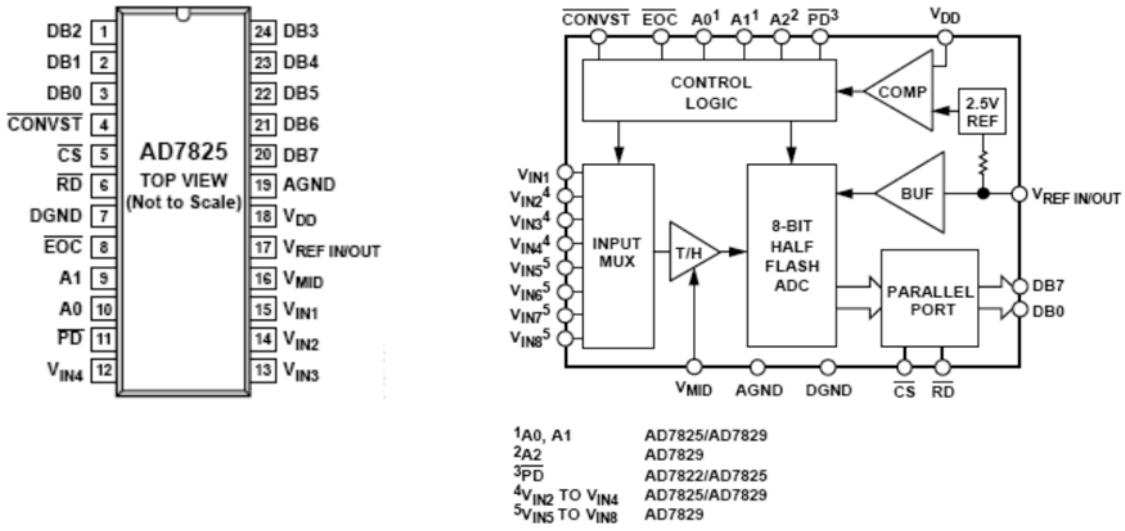


Figura 4.15: Struttura a blocchi funzionali dell' AD7825 e relativo pinout.

Il convertitore dispone di 4 canali: $V_{IN1} \div V_{IN4}$, ciò ci permette di acquisire ulteriori grandezze analogiche come da specifiche di progetto (par. 4.1). Il canale è selezionabile attraverso i *channel address pin*: A0-A1. V_{DD} è il piedino di alimentazione ed in questo caso la tensione è pari a 5V. AGND e DGND sono i pin di massa analogica e digitale e devono essere collegati tra loro il più vicino possibile al dispositivo in modo da ridurre il rumore. V_{REF} è il pin al quale applicare la tensione di riferimento attraverso la quale si determina la quantizzazione del segnale:

$$Q = \frac{V_{REF}}{2^n} = \frac{2,5}{2^8} = 9,765mV$$

Se si usa il riferimento interno di 2,5V basta adoperare un capacità di 100nF collegata tra il pin e massa. Il range della tensione di ingresso applicabile agli ingressi analogici dell' ADC è variabile tra 2 e 2,5V e dipende dal tipo di alimentazione usata. Nel nostro caso, essendo V_{DD}=5V , il range è di 2,5V ed è centrato intorno al valore di tensione impostato attraverso il pin V_{MID}. V_{MID} si usa in sostanza per togliere un eventuale offset presente sul segnale di ingresso. Se tale pin non viene adoperato allora lo si collega ad una capacità di 100nF , così V_{MID} =1,25V .

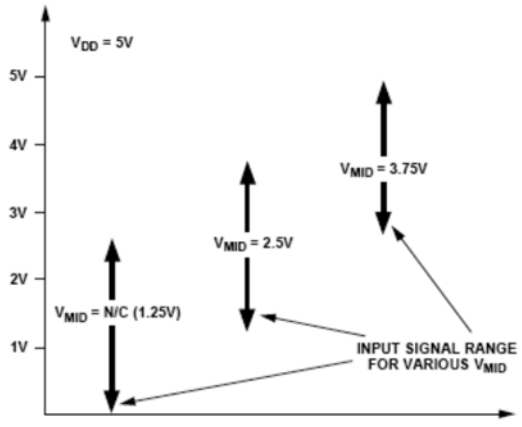


Figura 4.16: Range di tensione ammesso sugli ingressi analogici al variare della tensione sul pin V_{MID}

Quando sul pin di ingresso **/CONVST** (Conversion Start) si ha un fronte di discesa, allora la conversione ha inizio e il circuito di track&hold va in stato di hold per 120ns e quando ritorna nello stato di track il segnale viene campionato dalla capacità di

campionamento. Quando la conversione è terminata il segnale di uscita **/EOC** (End Of Conversion) ha un fronte di discesa; t1 rappresentato in figura 4.17 è il tempo impiegato per eseguire una conversione ed è pari a 420ns. Un successivo fronte di discesa del segnale di ingresso **/CS** (Chip Select)abilita la porta parallela del convertitore. Solo quando si ha un fronte di discesa del segnale **/RD** (Read) allora il dato viene trasferito sul bus dati DB0-DB7. Il pin **/PD** (Power Down) invece serve per adoperare l'ADC in modalità power-down. Infatti se /PD=0 , al termine di ogni conversione l'ADC entra in uno stato di basso consumo energetico. Questa modalità di funzionamento non interessa la nostra applicazione, pertanto /PD verrà collegato direttamente a V_{DD}. La potenza assorbita dall'ADC dipende dalla velocità , in termini di KSPS, del convertitore. Nella nostra applicazione (100-200KSPS) il consumo è limitato ad un massimo di 10mW (2mA@V_{DD}=5V).

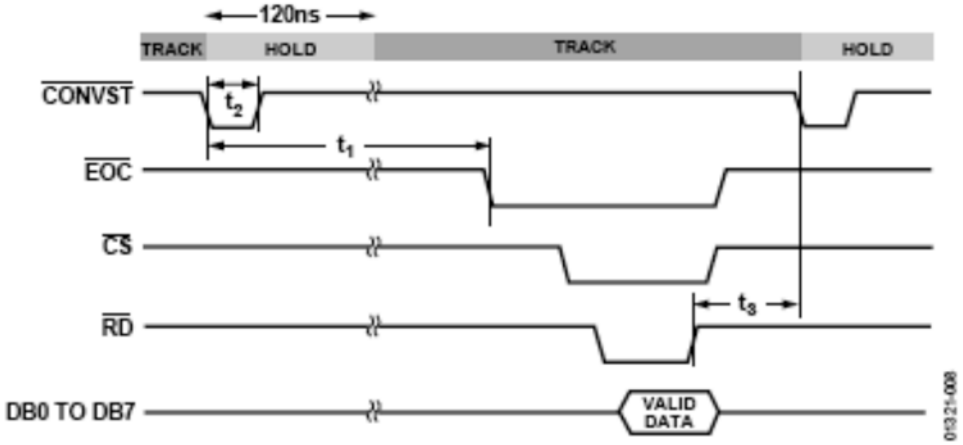


Figura 4.17: Funzionamento in "High-speed conversion mode"

Nel caso di acquisizione da più canali, l'indirizzo del canale del quale si deve effettuare la conversione viene campionato nell'istante in cui si ha il fronte di discesa di /CS e /RD. Il tempo, t13, che è necessario aspettare prima di effettuare una nuova conversione è pari a 200ns (fig. 4.18).

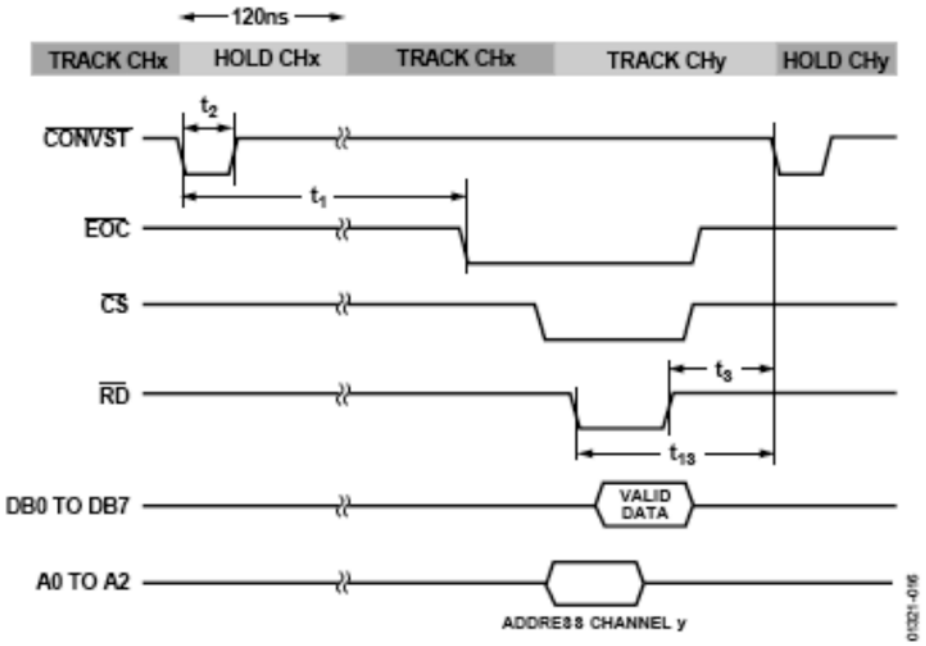


Figura 4.18: Funzionamento in "High-speed conversion mode" nel caso di conversione a più canali.

4.6 Il microcontrollore

Il microcontrollore permette il corretto funzionamento del DAQ in quanto gestisce la comunicazione fra l'FT245BM e l'AD245BRZ e fra l'FT245BM e il calcolatore. Determina inoltre, attraverso il fronte di discesa del segnale /CONVST la frequenza di campionamento del dato ed il canale da cui prelevare la grandezza analogica da convertire. Il microcontrollore in questione dovrà, come da specifica, permettere all' FT245BM di trasferire i dati provenienti dall' ADC al calcolatore attraverso l'USB. Il canale dal quale prelevare il segnale è sempre il canale 1 (V_{IN1}). Tenuto conto del funzionamento in scrittura dell' FT245BM e del ciclo di funzionamento dell' AD7825BRZ sarà necessario che il microcontrollore piloti i due dispositivi come riportato in figura 4.19 (in rosso i segnali di ingresso al controllore e in blu quelli di uscita)

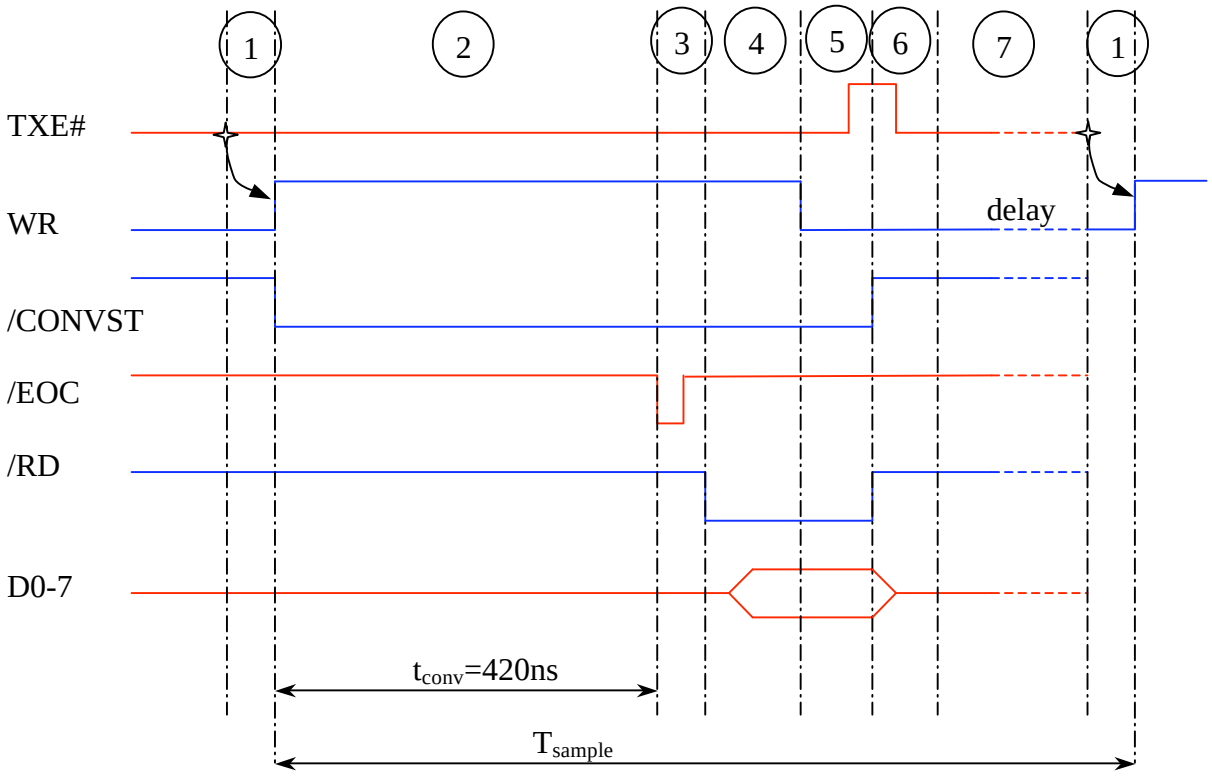


Figura 4.19: Diagramma temporale dei segnali di I/O del MCU; in rosso gli ingressi, in blu le uscite. Si consideri inoltre che i segnali non riportati hanno il seguente stato : /CS=0; SI/WU=1; PWREN=0 ; A0=0; A1=0.

Non appena TXE#=0 **(stato 1)**l' MCU provvede a mettere WR=1 e /CONVST=0 dando così il via alla conversione del segnale presente sull'ingresso V_{IN1}. Durante lo **stato 2** si ha la conversione del dato che termina non appena /EOC=0. A questo punto il controllore provvede ad abilitare la porta parallela del convertitore **(stato 3)** mettendo /RD=0. Il dato presente sul bus dati dell' ADC viene campionato dall' FT245BM tramite il fronte di discesa di WR **(stato 4)** . Successivamente si disabilita la porta parallela del convertitore: /RD=1 e si ripristina lo stato di /CONVST=1 **(stato 5).** Nello **stato 6** TXE# rimane alto per un tempo che dipende dallo stato del buffer interno all' FT245BM. Il successivo ritardo **(stato 7)** determina la velocità di acquisizione del DAQ. Il suo valore dipende dal tempo impiegato dall' MCU nel passare dallo stato1 allo stato 7 ed è strettamente dipendente dalla velocità con cui questo riesce ad eseguire le operazioni di I/O e dalla struttura del firmware. Tale ritardo va ricavato empiricamente, osservano la frequenza del segnale WR attraverso l'oscilloscopio. Il

microcontrollore adottato (leggi criterio par. 4.2) è l' ATmega16L della Atmel. E' un microcontroller a 8 bit a basso consumo energetico (≈1mW) e struttura RISC. E' dotato di 16Kbytes memoria flash per la programmazione, 1024 bytes di SRAM interna e EEPROM interna da 512 bytes. Le linee di I/O sono 32 suddivise in 4 porte da 8 bit. La tensione di alimentazione è compresa fra 2,7-5,5V e la frequenza di lavoro fra 8MHZ e 16 MHZ. A tale proposito è importante sottolineare che l' ATmega16L lavora con tensioni di alimentazione compresa fra 2,5 e 5,5V e frequenza di lavoro di 8MHz mentre l' ATmega16 può lavorare fino a 16MHz ma la tensione di alimentazione deve essere compresa fra 4,5 e 5,5 V. Se in futuro si volessero aumentare le prestazioni del sistema sarà necessario rimpiazzare il microcontrollore con un ATmega16 dato che il *pin-out* è il medesimo. In figura 4.20 è rappresentata la "mappatura" delle risorse di I/O del microcontrollore.

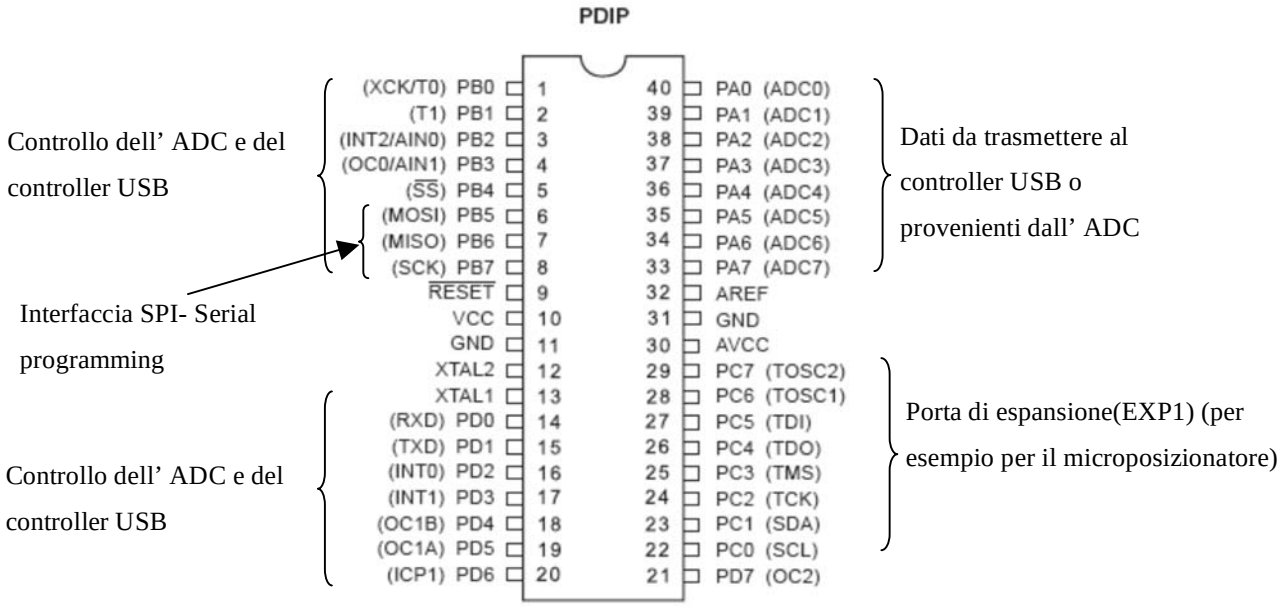


Figura 4.20: I/O Mapping dell' ATmega16L.

Il progetto del firmware è stato realizzato in ambiente **WinAVR**, un *tool* di sviluppo *open source* che comprende anche il compilatore (C e C++) GNU GCC. La versione finale del firmware è riportata di seguito ed implementa il diagramma temporale dei segnali di I/O, riportato in figura 4.20

4.6.1 Come impostare la velocità di campionamento tramite il firmware.

Abbiamo visto che lo stato 7 (par. 4.6) prevede l'introduzione di un ritardo che determina la velocità di campionamento del DAQ ed è realizzato tramite una serie di operazioni di scrittura fittizie del solito dato sulla porta PORTB. Il firmware riportato di seguito permette una acquisizione pari a 105.3KSPS. Questa velocità di acquisizione permette di rispettare le specifiche di sistema (80KSPS) e limita l'utilizzo della CPU da parte del *Client Software* nell'acquisizione dei dati dall' USB. Come abbiamo visto infatti, per risolvere il "*problema del produttore e del consumatore*" è necessario che il consumatore, cioè il programma di acquisizione , sia molto veloce. Ciò implica un maggior utilizzo della CPU e quindi un calcolatore con elevate prestazioni ed un sistema operativo che renda il processo di acquisizione dati dalla porta USB il più veloce possibile. Se la CPU è troppo lenta e/o la parte del *Client Software* che realizza l'acquisizione dei dati dall' USB (par. 4.2.4), non introduce un tempo di latenza ridotto, può succedere che TXE#=1 per un tempo troppo lungo ,maggiore del ritardo introdotto dallo stato 7, causando un campionamento non uniforme del segnale o un mancato campionamento.
Dalle prove effettuate su un calcolatore avente:

- Processore Intel 2Duo
- Windows XP Professional
- 1 GB di RAM
- 512 MB di scheda video
- Task Antivirus bloccato
- Nessun altro processo in esecuzione a parte il Client Software

si è visto che è possibile acquisire dati ad una velocità massima di 200KB/s – 250KB/s senza che vi sia perdita di dati o un campionamento non uniforme del dato. Qualora si intenda acquisire dati ancora più velocemente sarà necessario passare ad un calcolatore con diverso sistema operativo (Mac OSX o Linux). Nel nostro caso, non avendo un calcolatore eccessivamente potente e dotato di un sistema operativo efficiente si è deciso di limitare la velocità di acquisizione a 105.3 KB/s.

```
/* =================FIRMWARE V1.0 ========================*/
/*****************************************/
/* CKOPT=1(unprogrammed);                */
/* CKSEL3=1;CKSEL2=1;CKSEL1=1 (3-8MHz)  */
/* CKSEL0=1                              */
/* SUT=10 (FAST RISING POWER)            */
/* Quarzo da 8MHz                        */
/* AMBIENTE SVILUPPO: WinAVR 2.0.7       */
/*****************************************/


#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdint.h>
#include <avr/boot.h>


/*********************************************/
/*    TABELLA INGRESSI/USCITE                */
/*                                           */
/*  D0-D7:      PAO-PA7          (I/O)       */
/*  A0-A1:      PBO-PB1           (O)        */
/*  /CONVST:  PB2                 (O)        */
/*  /CS:          PB3             (O)        */
/*  /RD:          PB4             (O)        */
/*   RD:          PB5             (O)        */
/*   WR:          PB6             (O)        */
/*  SI/WU:       PB7              (O)        */
/*  EXP1:        PC0-PC7          (I/O)      */
/*  PWREN:    PD5                 (I)        */
/*  /EOC:        PD4              (I)        */
/*  TXE:         PD3              (I)        */
/*  RXF:         PD2              (I)        */
/*                                           */
/*********************************************/


int
main ()

{

  DDRA= 0B00000000; // PORTA di ingresso
  PORTA=0B00000000; // PORTA in tri-state
  DDRD= 0B00000000; // PORTD di ingresso
  PORTD=0B00000000; // PORTD in tri-state
  DDRB= 0B11111111; // PORTB di uscita
  PORTB=0B10110100; //
SI/WU=1;WR=0;RD=1;/RD=1;/CS=0;/CONVST=1;A1=0;A0=0;
```

```
  // configurare porta EXP1 a seconda dell' uso//

        // if TXE=0
while(1) { if (!(PIND&8)) {  PORTB=0B11110000;     //WR=1;/CONVST=0;
        STATO (1)

                                 while (!(PIND&16)) ;  // Esco se /EOC=0 STATO (2)

                                 PORTB=0B11100000;      // /RD=0;         STATO (3)
                                 PORTB=0B10100000;      // WR=0;          STATO (4)
                                 PORTB=0B10110100;      // WR=0;/RD=1;/CONVST=1;
STATO (6)

                                // delay  STATO (7)
                                 PORTB=0B10110100;
                                 PORTB=0B10110100;
                                 PORTB=0B10110100;   //1

                                 PORTB=0B10110100;
                                 PORTB=0B10110100;
                                 PORTB=0B10110100;
                                 PORTB=0B10110100;   //2

                                 ……..

                                 PORTB=0B10110100;
                                 PORTB=0B10110100;
                                 PORTB=0B10110100;
                                 PORTB=0B10110100;   //16

                           }

else  PORTB=0B10110100;
//SI/wu=1;WR=0;RD=1;/RD=1;/CS=0;/CONVST=1;A1=0;A0=0

        }
}
/* ======================================================*/
```

Una volta compilato il programma in C questo verrà scaricato nella memoria flash dell' MCU o tramite una **programmazione *In-System***, attraverso l'interfaccia SPI (connettore PROG sullo schematico) o attraverso lo *starter kit* **ATSTK500**. Il primo metodo è molto utile i fase di aggiornamento del firmware , poiché non necessita la rimozione del microcontrollore dallo zoccolo. Basterà connettersi con il programmatore seriale rappresentato in figura 4.21 e caricare il **file .hex** generato in fase di compilazione con il software gratuito **PonyProg**, reperibile su www.lancos.com.

E’ necessario far corrispondere i pin del connettore J2 del programmatore con i pin del connettore PROG presente sul DAQ. A tal fine è utile tenere presente il pinout di J2. La piedinatura dei connettori è riportata di seguito:

PROG	
1	VCC
2	RESET
3	SCK
4	MISO
5	MOSI
6	GND

J2	
1	VCC
2	RESET
3	SCK
4	MOSI
5	MISO
6	-
7	-
8	-
9	-
10	GND

Nel caso invece si utilizzi la scheda ATSTK500 è necessario togliere l’ MCU dallo zoccolo e caricare il programma tramite **AVR STUDIO4**. In questo lavoro sono state adottate entrambe le soluzioni.

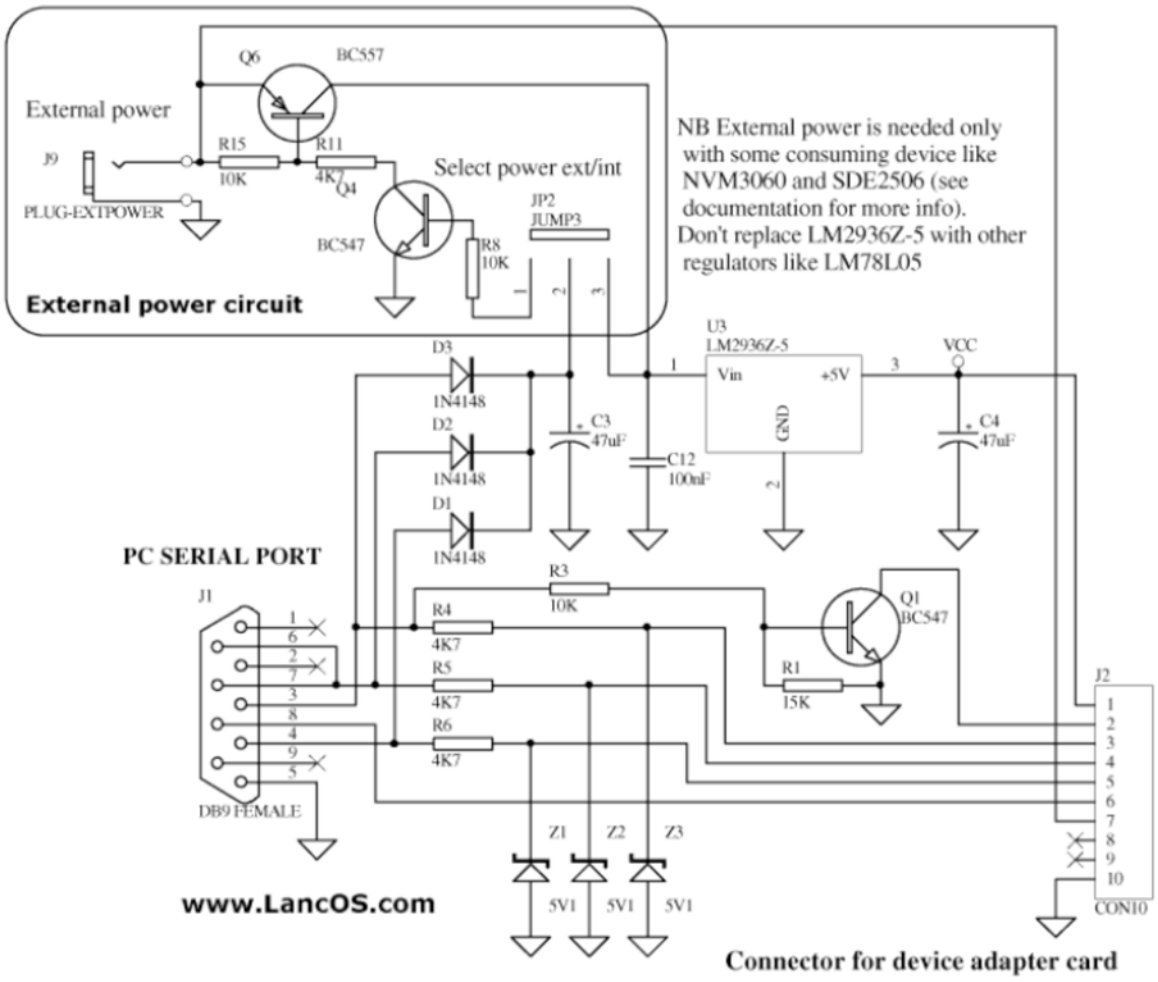
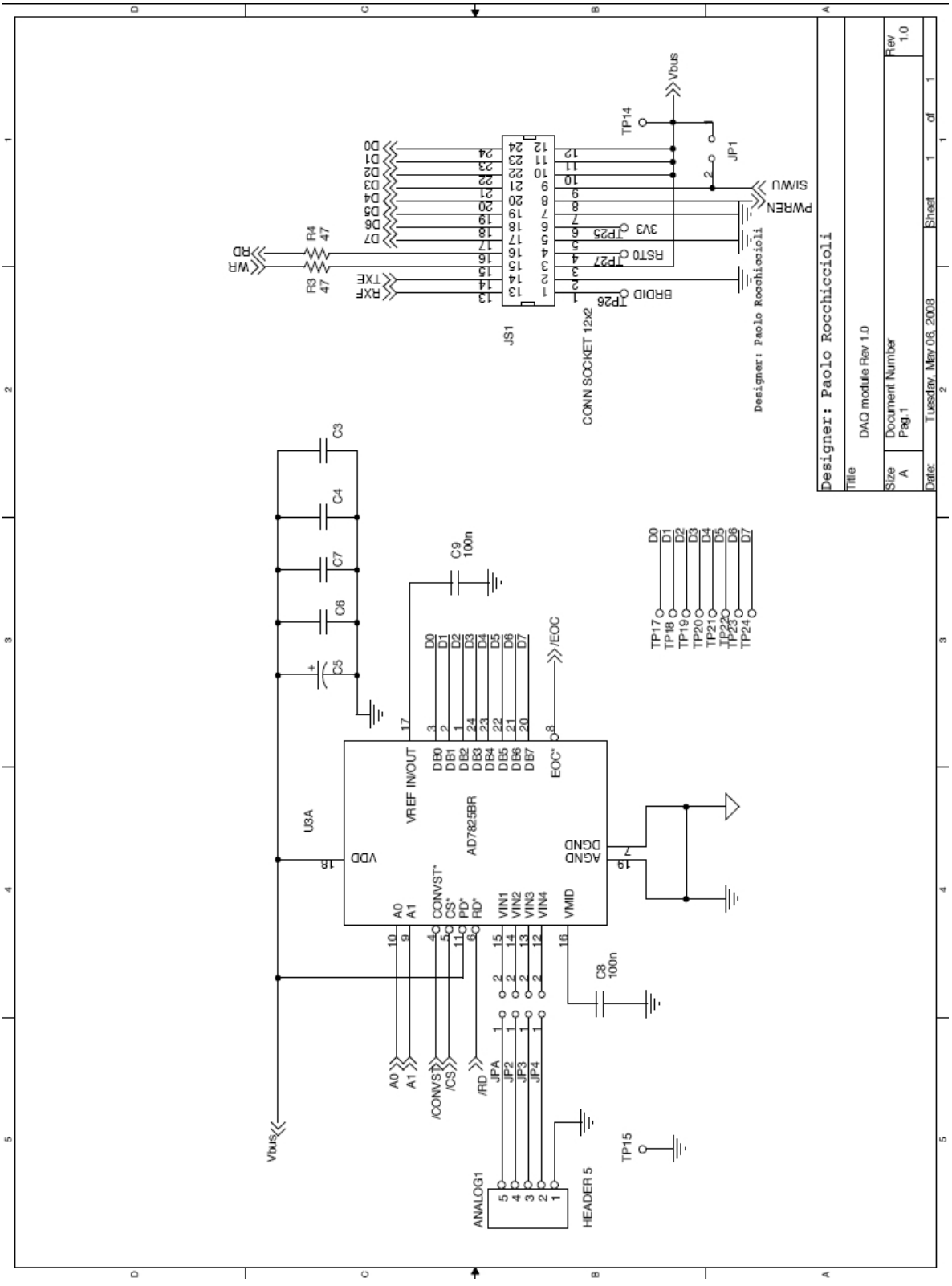


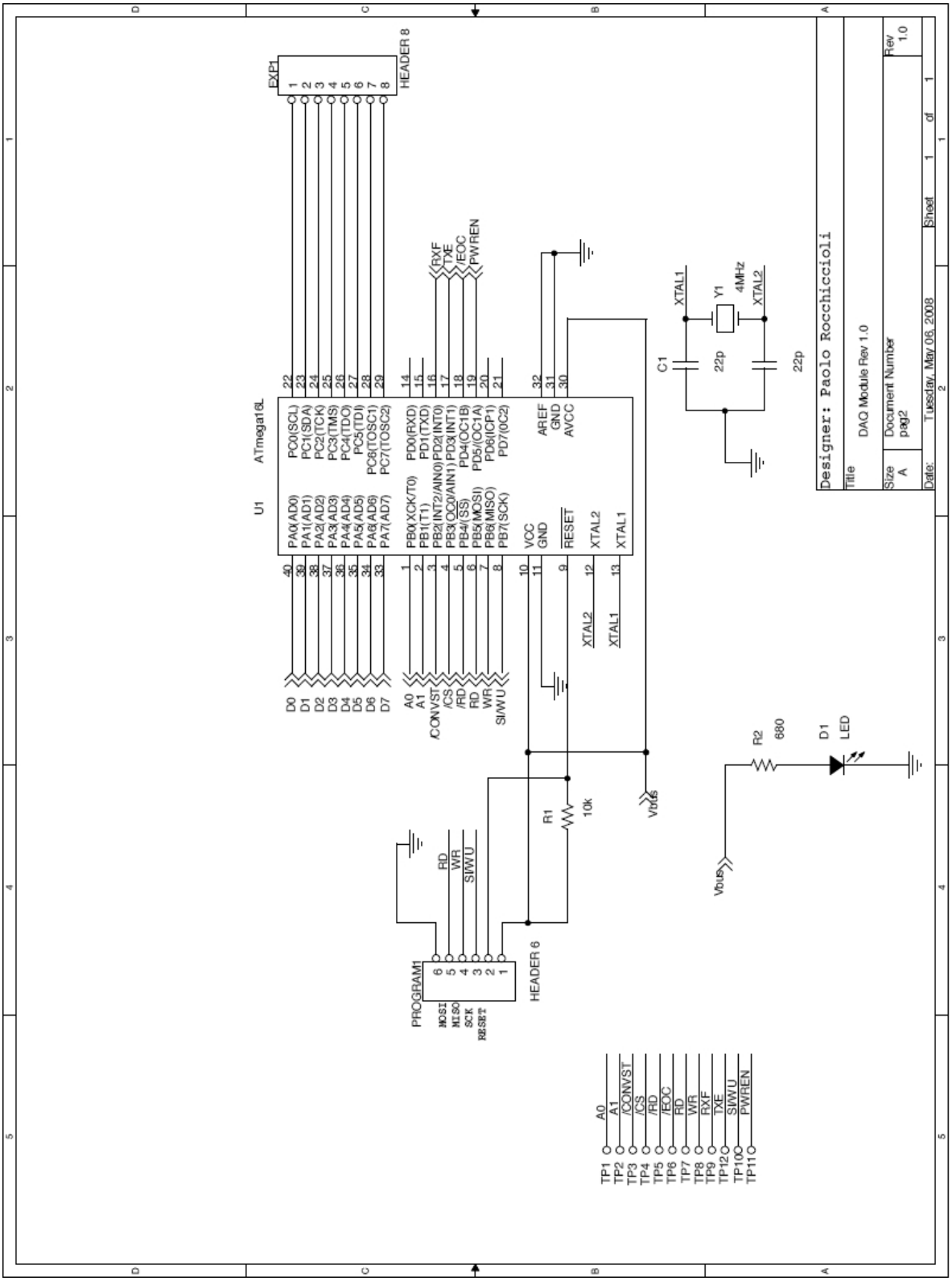
Figura 4.21: Schema elettrico del programmatore seriale da utilizzare con il software PonyProg e da collegare al connettore PROG sul DAQ.

4.7 Schema elettrico e layout del DAQ

Lo schema elettrico del DAQ, riportato di seguito, è stato realizzato tenendo presente quanto detto nel paragrafo precedente. E' opportuno, visto che il DAQ è ad uno stato iniziale, **prototipizzare** il più possibile il PCB. A Tale fine è stato inserito un ***test point*** (TP) su ogni modo di segnale in modo da facilitare la fase di collaudo della scheda. I ***jumper*** JP2,JP3,JP4 e JPA su tutti gli ingressi analogici dell' ADC in modo da poterli isolare senza dover scollegare il cavo dalla morsettiera. Il *jumper* JPA, come si vede anche dalla foto, deve essere chiuso in modo da connettere il segnale analogico proveniente dal trasduttore al canale 1 dell' ADC. Questo segnale è portato dalla trasduttore (presa jack) al DAQ tramite un cavo schermato. Lo schermo deve essere collegato al morsetto di massa presente dal lato del DAQ in modo da limitare gli accoppiamenti capacitivi sul segnale trasmesso. Il *jumper* vicino al controllore USB invece permette di forzare a 1 in maniera hardware lo stato del pin SI/WU. Nel firmware interno al microcontrollore è prevista una "forzatura software" del segnale, quindi questo può essere lasciato aperto.

Sempre al fine di semplificare il collaudo e la progettazione del *firmware* della scheda è stato ritenuto opportuno inserire anche la serigrafia sulla scheda. Visto però che il **PCB** è stato realizzato con la **tecnica della fotoincisione** su una basetta di FR4 a doppia faccia questo non è possibile. Il problema è stato aggirato inserendo il nome del nodo attraverso del **testo in rame** in corrispondenza di ogni *test point* e delle morsettiere e dei connettori presenti sulla scheda.


Designer: Paolo Rocchicciolì
Title
DAQ module Rev 1.0
Size A
Document Number
Pag.1
Rev 1.0
Date: Tuesday, May 06, 2008
Sheet 1 of 1
Designer: Paolo Rocchicciolì
CONN SOCKET 12x2
JS1
Vbus
TP14
JP1
SI/WU
PWREN
3V3
RST0
BRDID
TP25
TP27
TP26
R4 47
R3 47
WR
RD
RXF
TXE
D0
D1
D2
D3
D4
D5
D6
D7
C3
C4
C7
C6
C5
C9 100n
C8 100n
U3A
AD7825BR
VREF IN/OUT
VDD
AGND
DGND
EOC*
/EOC
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
A0
A1
CONVST*
CS*
RD*
VIN1
VIN2
VIN3
VIN4
VMID
/CONVST
/CS
/RD
JPA
JP2
JP3
JP4
ANALOG1
HEADER 5
TP15
TP17
TP18
TP19
TP20
TP21
TP22
TP23
TP24


EXP1
HEADER 8
U1
ATmega16L
PA0(AD0)
PA1(AD1)
PA2(AD2)
PA3(AD3)
PA4(AD4)
PA5(AD5)
PA6(AD6)
PA7(AD7)
PC0(SCL)
PC1(SDA)
PC2(TCK)
PC3(TMS)
PC4(TDO)
PC5(TDI)
PC6(TOSC1)
PC7(TOSC2)
PB0(XCK/T0)
PB1(T1)
PB2(INT2/AIN0)
PB3(OC0/AIN1)
PB4(/SS)
PB5(MOSI)
PB6(MISO)
PB7(SCK)
PD0(RXD)
PD1(TXD)
PD2(INT0)
PD3(INT1)
PD4(OC1B)
PD5/(OC1A)
PD6(ICP1)
PD7(0C2)
AREF
GND
AVCC
VCC
GND
RESET
XTAL2
XTAL1
RXF
TXE
/EOC
PWREN
D0
D1
D2
D3
D4
D5
D6
D7
A0
A1
/CONVST
/CS
/RD
RD
WR
SIWU
XTAL1
Y1
4MHz
XTAL2
C1
22p
22p
Designer: Paolo Rocchiccioli
Title
DAQ Module Rev 1.0
Size
A
Document Number
pag2
Rev
1.0
Date:
Tuesday, May 06, 2008
Sheet
1
of
1
R1
10k
R2
680
D1
LED
Vbus
PROGRAM1
MOSI
MISO
SCK
RESET
HEADER 6
TP1
TP2
TP3
TP4
TP5
TP6
TP7
TP8
TP9
TP10
TP11
TP12

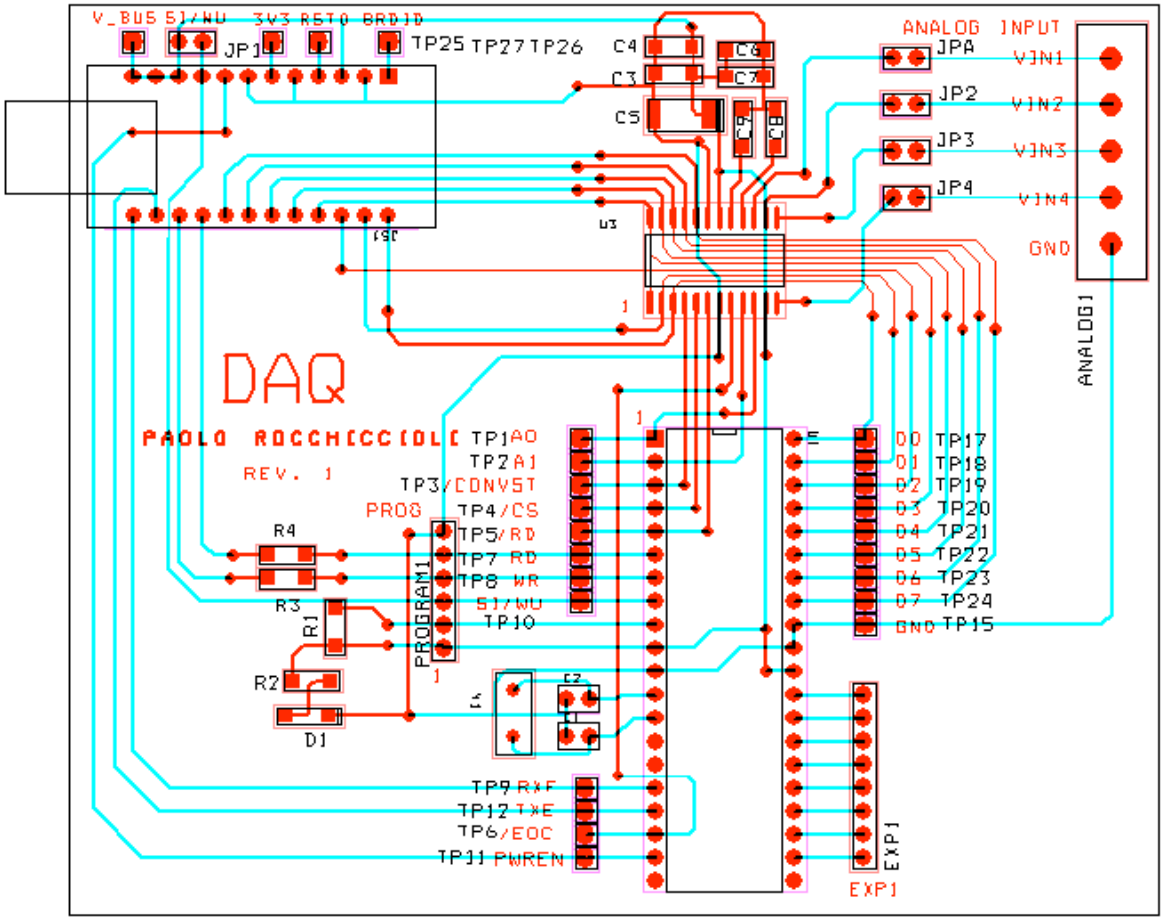


Figura 4.22: Layout del DAQ. In celeste il Bottom Layer e in rosso il Top Layer . Il testo in rosso è la "serigrafia" in rame sul Top Layer.

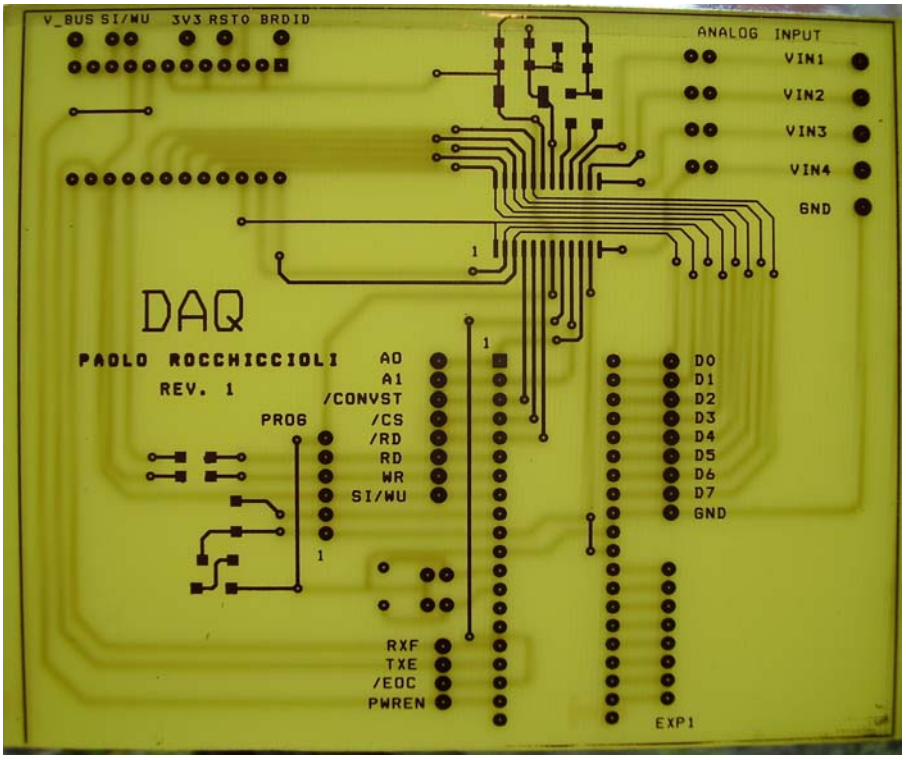


Figura 4.23: Top Layer del circuito stampato ottenuto per fotoincisione

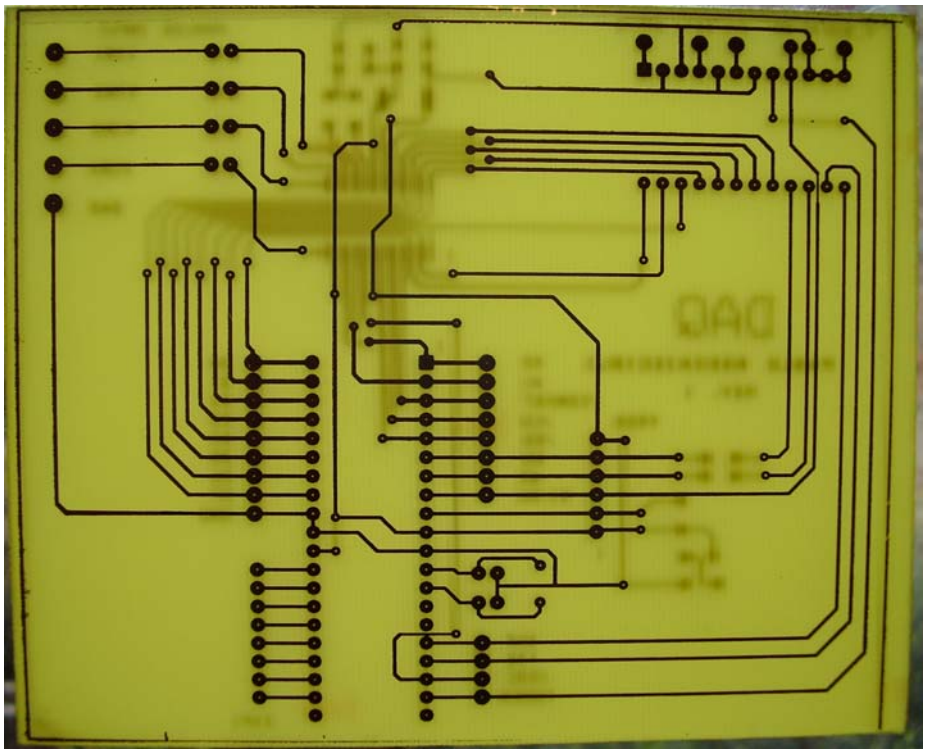


Figura 4.24: Bottom Layer del circuito stampato ottenuto per fotoincisione

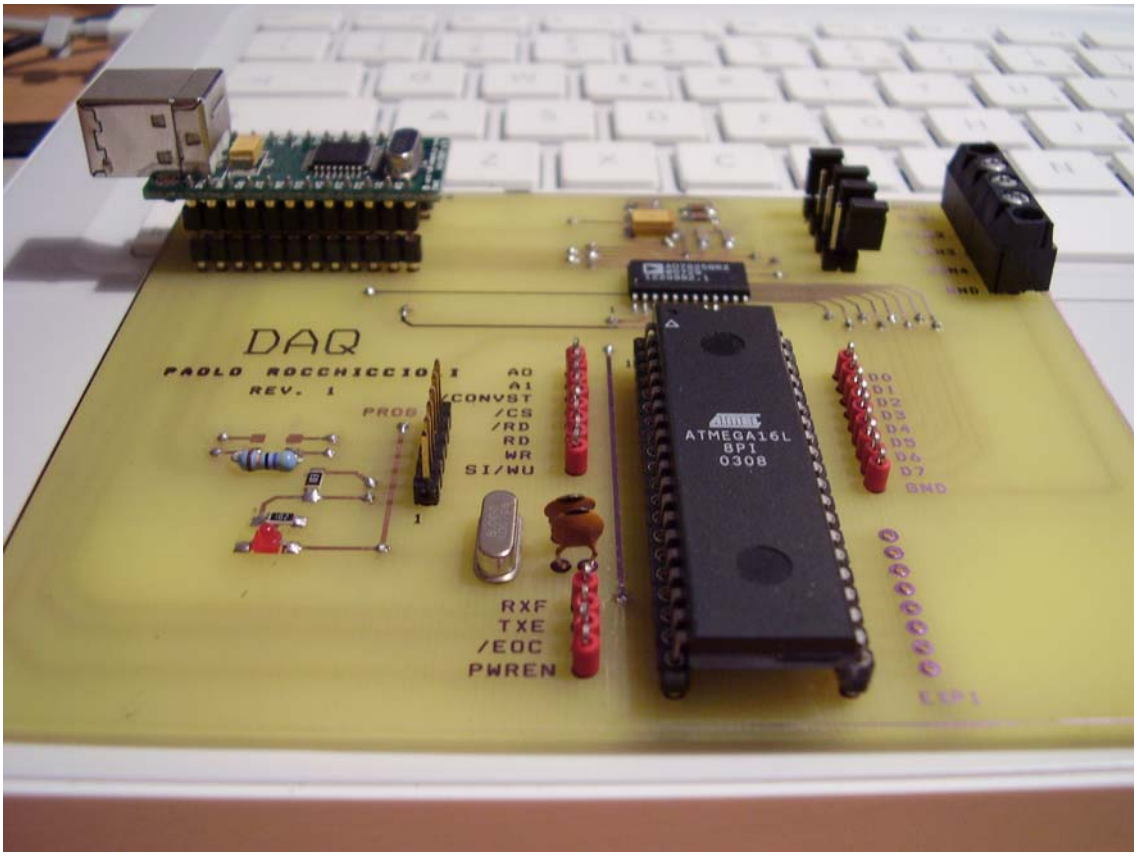


Figura 4.25: Vista della scheda realizzata e montata

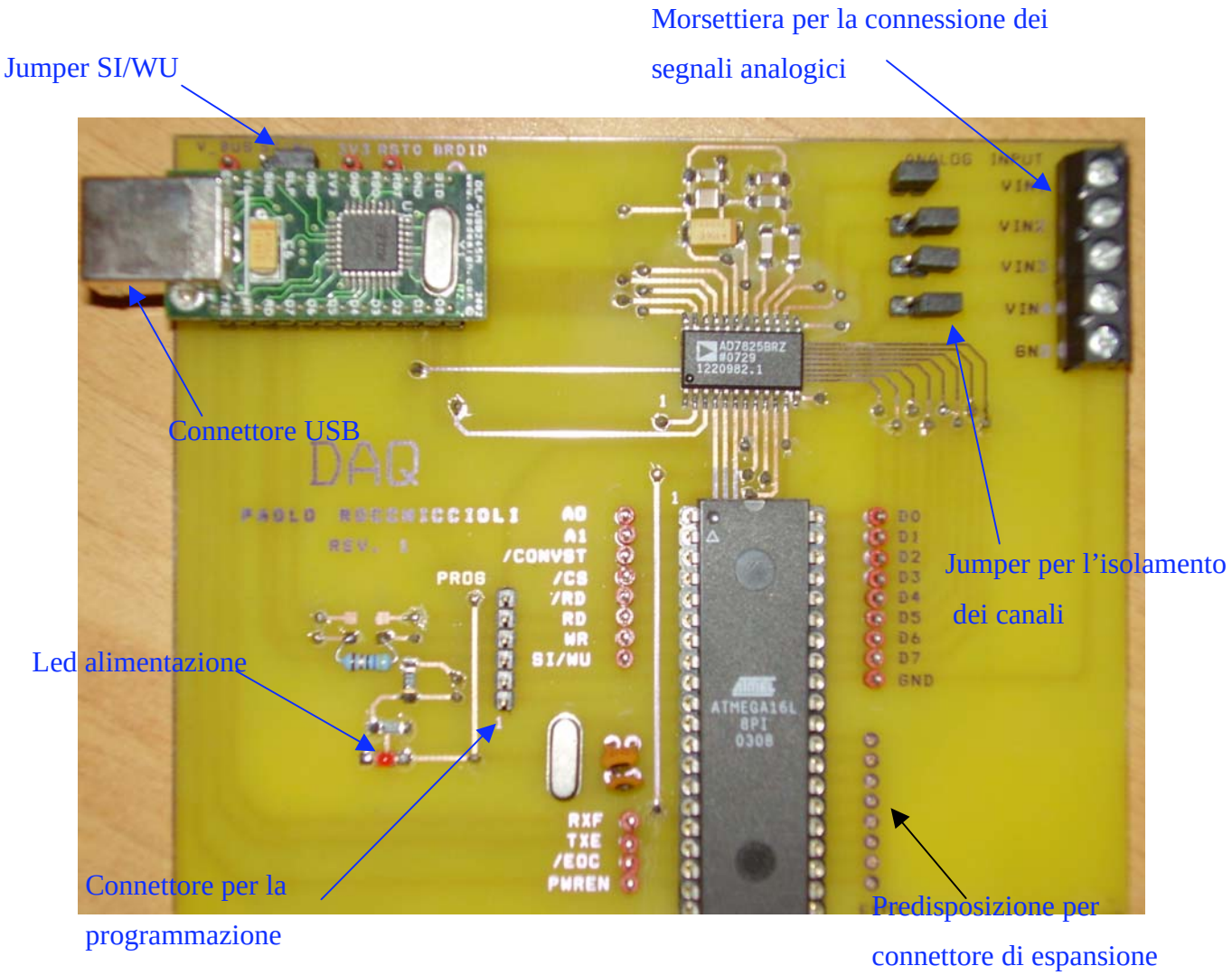
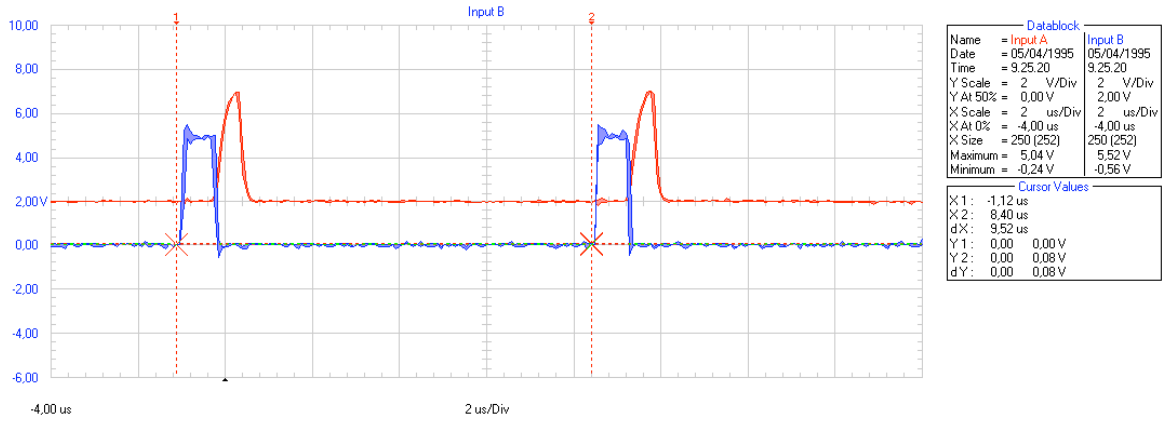


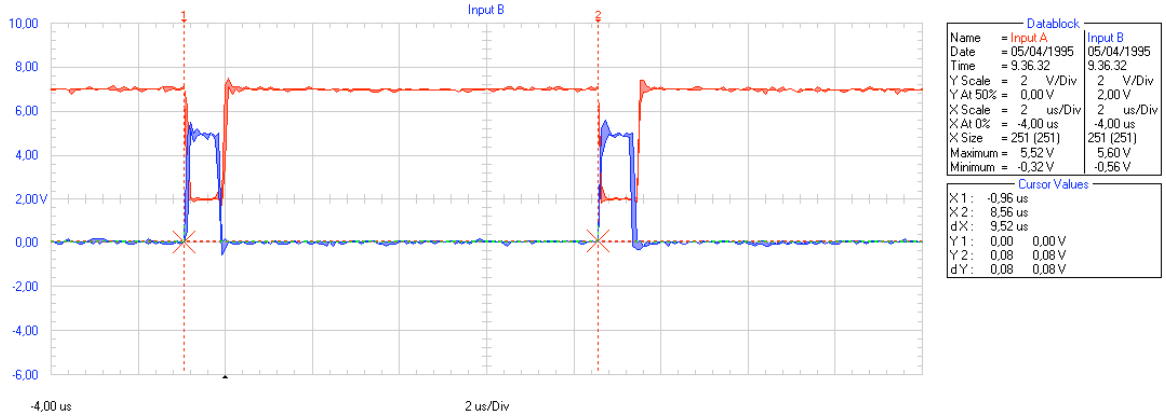
Figura 4.26: Vista dei connettori, delle morsettiere, dei jumpers e test points presenti sulla scheda.

4.8 Debugging del firmware

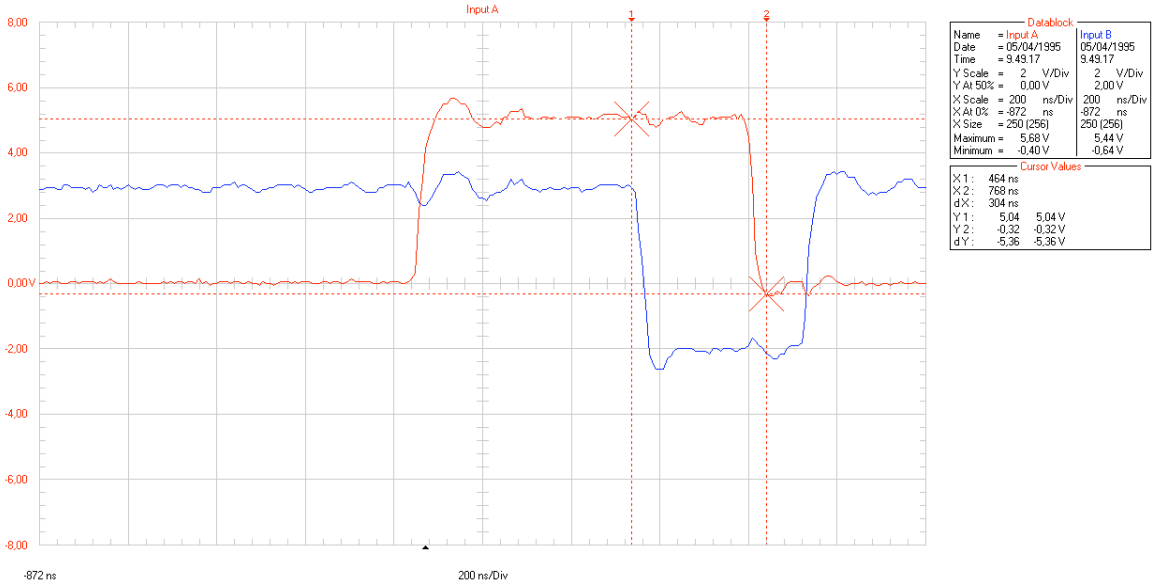
Di seguito sono riportati i segnali acquisiti durante la fase di debugging con l' oscilloscopio (**Fluke 123**) collegato alla seriale del PC utilizzando **FlukeView®**.



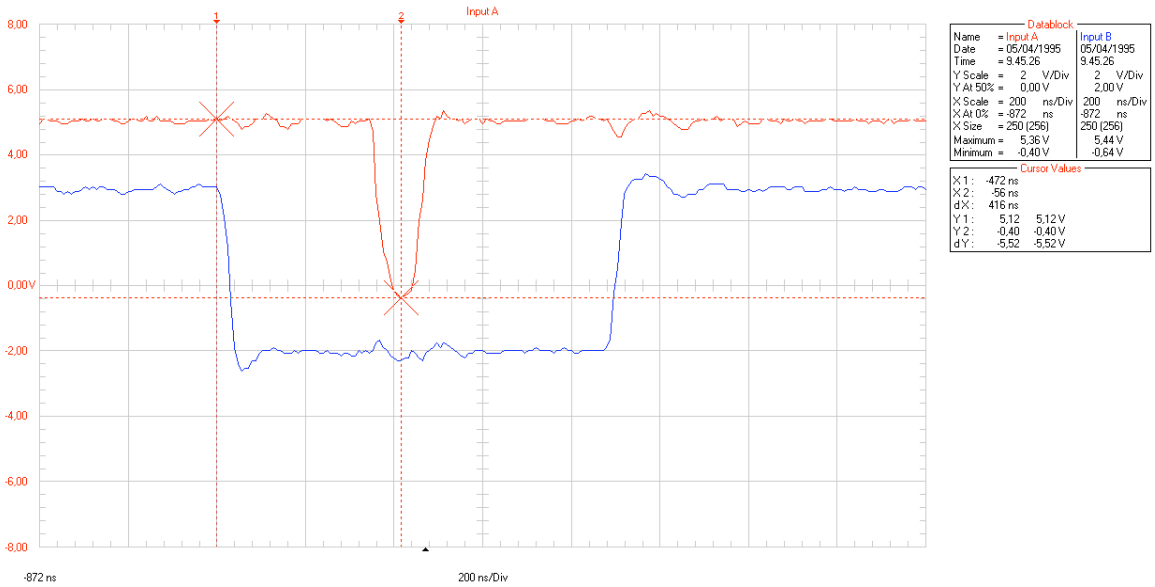
Canale A:***TXE#*** **Canale B**: ***WR*** *: La frequenza di campionamento è 105,3KHz. TXE diventa inattivo(H) sul fronte di discesa di WR come previsto. La frequenza è ritoccabile togliendo qualche istruzione fittizia di I/O (par.4.6.1). Questa frequenza dovrà essere poi impostata nel client software per valutare correttamente la variazione dell' indice di rifrazione nel tempo (par. 1.6.2 e 5.5.2).*



Canale A: ***/CONVST***, **Canale B**: ***WR***: *Sincronismo di inizio conversione del dato verificato e WR verificato (stato1).*



Canale A: ***WR*** , **Canale B**: ***/RD***: *Il segnale da trasmettere sul bus USB viene campionato dall' FT245BM sul fronte di discesa di WR solo dopo che il segnale è stato messo sull'uscita del convertitore (fronte di discesa di /RD). Tra i due istanti intercorre un tempo di circa 304ns. Ciò permette di rispettare il tempo di setup T9=20 ns di WR (par. 4.2.3 fig. 4.8) garantendo così un corretto campionamento del dato.*



Canale A: *EOC* **,** **Canale B**:*/CONVST : Verifica del tempo di conversione del dato. Come da specifiche dell' AD7825 risulta essere 420ns.*

CARATTERISTICHE DEL DAQ @25°C	
Numero canali Analogici	*4*
Risoluzione ADC	*8 bit*
Bus comunicazione	*USB 1.0-2.0*
Frequenza campionamento	*105.3KS/s: 1canale*[1]
	26KS/s: 4 canali [2]
Campionamento simultaneo	*No*
Tipo di misura	*Tensione*
Input range [V]	*0-2,5V*
Corrente di ingresso ADC	*1μA max*
Capacità di ingresso ADC	*15pF max*
Isolamento tra canali ADC	*-70dB typ*
I/O digitali configurabili	*8 (connettore EXP1)*
Aggiornamento del firmware	*Programmazione In-System* [3]
Memoria	*512Byte (MCU-eeprom)*
MCU	*ATMEGA16 o ATMEGA16L*
Alimentazione	*5V (Bus Powered)*
Riferimento di massa	*Massa del PC (Ground)*
Massima corrente assorbita	*25mA*
Sistema Operativo	*Windows XP*

(1)La frequenza di campionamento può essere aumentata in funzione delle caratteristiche del calcolatore (par 4.6.) fino ad un massimo di 200-250KS/s modificando opportunamente il firmware del microcontrollore.

(2) La frequenza di campionamento può essere aumentata in funzione delle caratteristiche del calcolatore (par 4.6.) fino ad un massimo di 50-75KS/s per canale modificando opportunamente il firmware del microcontrollore. La versione del firmware attualmente presente nel DAQ supporta l'acquisizione del dato da un solo canale a 105.3KS/s, sarà pertanto necessario aggiornare il firmware in modo che abbia le specifiche illustrate nel par. 4.6

(3) La programmazione In-System avviene secondo le specifiche illustrate al par. 4.6.1

Capitolo 5

Il client software

5.1 Introduzione

L'ambiente di sviluppo usato per la realizzazione del client software è **LabVIEW** 8.0 *Professional* (**Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench) della National Instrument che adopera un linguaggio di programmazione grafico detto anche *Linguaggio G*. Questo ambiente è il più idoneo a realizzare in maniera semplice, efficace e veloce (da 4 a 10 volte rispetto ai linguaggi testuali) uno strumento virtuale. Oltre a mettere a disposizione una libreria di controlli ed indicatori molto vasta, che permette di realizzare un'interfaccia grafica accattivante ed intuitiva, fornisce anche un'ampia libreria di funzioni matematiche e per lo studio dei segnali. Nei paragrafi seguenti verranno illustrate le funzionalità dello strumento virtuale (VI), per maggiori dettagli si rimanda al programma allegato nella documentazione in formato elettronico. Il client software è stato progettato in modo che abbia le seguenti caratteristiche:

- **interfaccia semplice**: deve poter essere usato anche da utenti non esperti.
- **tempo di latenza ridotto**: la fase di acquisizione dei dati deve essere molto veloce in modo da essere sempre pronta a ricevere i dati dal DAQ.
- **richiedere poche risorse CPU:** la fase di elaborazione deve essere indipendente dalla fase di acquisizione dei dati poiché richiede maggior tempo. Una cattiva progettazione del VI oltre ad aumentare il tempo di latenza può impegnare

eccessivamente la CPU aumentando il rischio che il calcolatore ignori la presenza di dati sul bus USB provocandone di fatto la perdita.

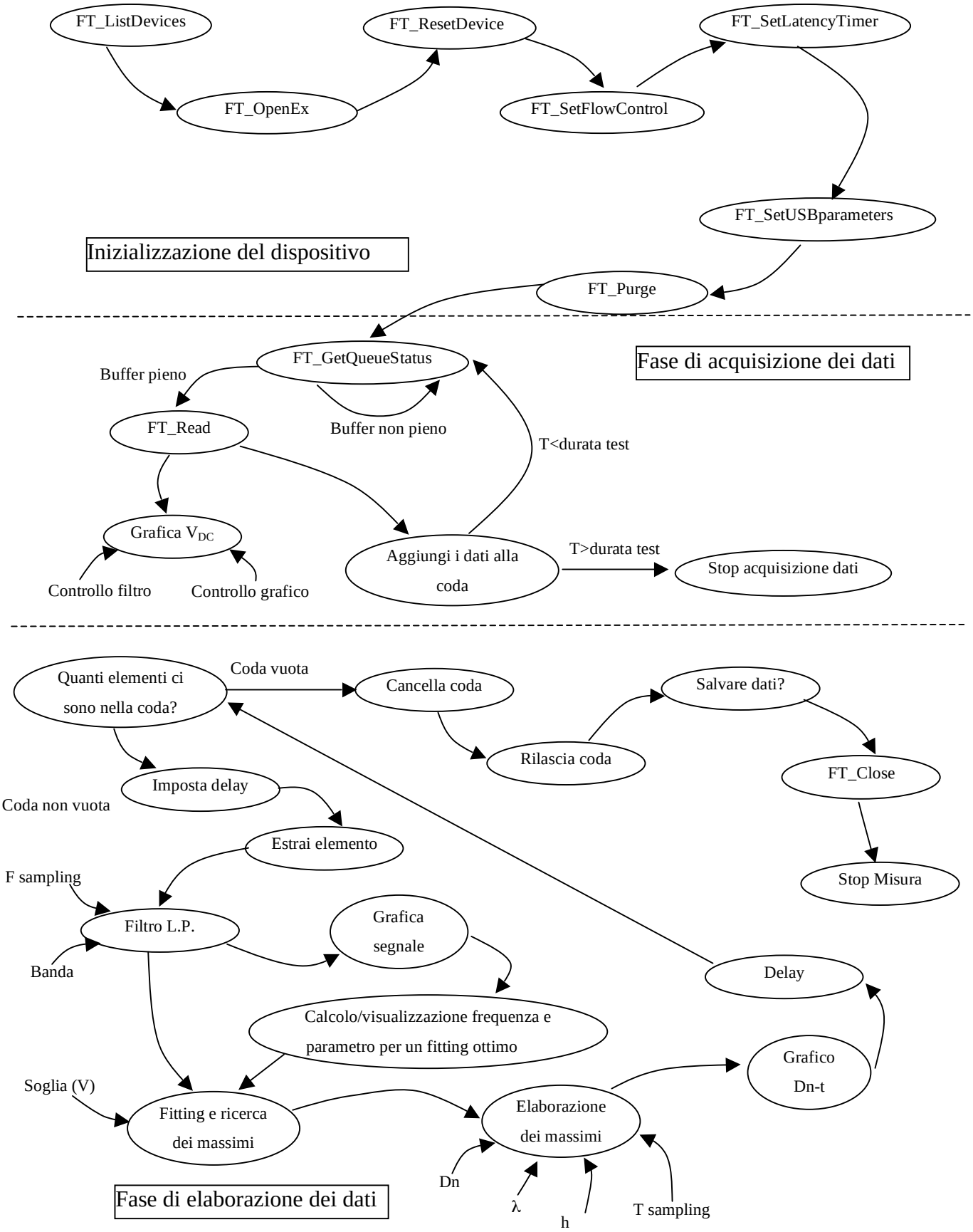
- **processi paralleli comunicanti tramite una "coda"**: per le due ragioni sopra citate la fase di acquisizione deve passare i dati da elaborare alla parte del VI incaricato tramite una coda (*queue*). In tal maniera i due processi risultano indipendenti.
- **deve essere possibile "ottimizzare" la coda**: avendo la coda lo scopo di passare i dati da un processo all'altro deve essere possibile "ottimizzarla", deve cioè essere possibile impostare la velocità con cui il processo di elaborazione attinge dati da quest' ultima in modo che non si svuoti mai fino a che la misura non è termina e che le sue dimensioni non accrescano troppo. La velocità con cui viene svuotata la coda dipende dalla velocità del processo di elaborazione che come abbiamo detto, dipende dalle prestazioni del calcolatore.
- **deve essere possibile visualizzare un segnale continuo (tensione) proveniente dal trasduttore:** questa funzione si rende molto utile durante la fase di allineamento dell'interferometro. Se infatti gli specchi e i beamsplitter sono adeguatamente orientati in modo da avere i due raggi allineati è possibile visualizzare il valore massimo di tensione (massima potenza incidente sul fotodiodo). Dovrà inoltre essere possibile filtrare il rumore introdotto a frequenze superiori a 50 Hz e calcolare il valore efficace e continuo della tensione monitorata in modo da facilitare ulteriormente l'allineamento dei raggi.
- **deve essere possibile visualizzare, qualitativamente, il segnale analogico proveniente dal trasduttore:** se la frequenza del segnale non è elevata basterà riportare i campioni (punti) ed effettuare un'interpolazione lineare. Se la frequenza del segnale è massima (10-20 KHz) il numero di campioni per ogni periodo del segnale diminuisce(10-5) e l'interpolazione lineare non è più sufficiente. Trattandosi però di un segnale sinusoidale questa soluzione è più che sufficiente per valutare l'andamento del segnale. D'altronde l'uso di un algoritmo dedicato al *fitting* richiederebbe risorse di calcolo non trascurabili che risultano sprecate considerando che lo scopo del VI non è quello di graficare correttamente il segnale acquisito ma individuarne i massimi in modo a calcolare le variazioni dell'indice di rifrazione.

- **deve prevedere un filtraggio numerico del segnale acquisito:** nel caso di segnali a bassa frequenza (centinaia di Hz) il contributo del rumore sulla forma d'onda è visivamente più accentuato. Per poter aumentarne la leggibilità ed evitare che vi siano degli errori in fase di valutazione dei massimi si introduce un filtro passa basso a banda variabile in modo da "spianare" (*smoothing*) la forma d'onda.
- **deve implementare un algoritmo ottimizzato per l'interpolazione dei campioni e la ricerca degli istanti temporali a cui si verificano i massimi del segnale:** un' interpolazione lineare dei campioni del segnale ed una semplice ricerca del massimo non è sufficiente quando il segnale è ad alta frequenza (pochi campioni per periodo) ed è affetto da rumore; la valutazione sull'istante temporale nel quale si ha un massimo risulterebbe grossolana. Se la scelta di interpolazione lineare risulta poco efficiente nel valutare correttamente i massimi, un *fitting* maggiormente performante che utilizza una funzione interpolante più complessa comporta tempi di calcolo più lunghi impegnando per lungo tempo la CPU del calcolatore. Vedremo in seguito come è stato risolto questo problema.
- **graficare le variazioni dell' indice di rifrazione in funzione del tempo:** dopo aver elaborato il segnale secondo il criterio adottato nel paragrafo 1.6.2 si riportano su un grafico gli istanti in cui si è verificata la variazione dell' indice di rifrazione (Dn) che si è deciso di osservare. I punti poi saranno interpolati linearmente.
- **salvare su un file la misura:** tutte le coppie di punti che costituiscono il grafico Dn-t dovranno essere salvati su un file di testo su due colonne distinte in modo da facilitare l'importazione del file in Excel per graficare ed analizzare nuovamente i dati della misura.
- **modularizzare tramite subVI:** per facilitare la "lettura" ed il collaudo del sistema si preferisce progettare VI con funzioni dedicate(subVI) poi richiamate nel progetto principale (VI).

5.2 Inizializzazione del dispositivo

Il VI (**V**irtual **I**nstrument) una volta stabilita la comunicazione con il DAQ provvede ad acquisire ed in seguito elaborare i dati nella maniera più efficiente possibile, ovvero quella che richiede minor tempo e minori risorse CPU per i calcoli. La comunicazione con il DAQ viene stabilita adoperando le funzioni presenti nella libreria *D2XX.dll* del driver del controllore USB della FTDi. Queste funzioni sono già state ampiamente illustrate nei paragrafi da 4.3.1 a 4.3.7. La comunicazione verrà stabilita solo dopo che è stata fatta una lista dei dispositivi collegati, una volta riconosciuta la presenza del DAQ è possibile accedere al controllore USB che verrà resettato in modo da cancellare qualsiasi impostazione precedente. Successivamente verrà impostato il flusso di controllo dei dati RTS/CTS. Questa operazione è necessaria a causa della presenza di un *bug* nel driver fornito dalla FTDi che comporterebbe la perdita di una parte dei dati acquisiti. Successivamente si impostano tutti quei parametri che permettono di massimizzare la velocità di comunicazione tra DAQ e PC (105.3KB/s). Si imposta quindi il latency timer a 2 ms e la dimensione del dato da trasferire pari a 64KB (par. 4.2.4). L'operazione finale consiste nel cancellare il contenuto dei buffer interni al controllore in modo da evitare l'invio di dati della misura precedente.

Se l'inizializzazione del dispositivo , per qualsiasi motivo, non va a buon fine, la misura viene interrotta e un messaggio di errore notifica il tipo di errore e suggerisce come risolvere il problema. Questo è possibile grazie alla creazione e all'interrogazione di un apposito vettore di diagnostica, reso poi disponibile in una *tab* apposita, all' interno della quale sarà possibile monitorare i dati acquisiti dal DAQ , modificare il latency timer e la dimensione massima del dato trasferito.


FT_ListDevices
FT_ResetDevice
FT_SetLatencyTimer
FT_OpenEx
FT_SetFlowControl
FT_SetUSBparameters
Inizializzazione del dispositivo
FT_Purge
FT_GetQueueStatus
Fase di acquisizione dei dati
Buffer pieno
Buffer non pieno
FT_Read
T<durata test
Grafica V_{DC}
Aggiungi i dati alla coda
T>durata test
Stop acquisizione dati
Controllo filtro
Controllo grafico
Quanti elementi ci sono nella coda?
Coda vuota
Cancella coda
Salvare dati?
Rilascia coda
FT_Close
Imposta delay
Coda non vuota
Estrai elemento
Stop Misura
F sampling
Filtro L.P.
Grafica segnale
Banda
Delay
Calcolo/visualizzazione frequenza e parametro per un fitting ottimo
Grafico Dn-t
Soglia (V)
Fitting e ricerca dei massimi
Elaborazione dei massimi
Dn
λ
h
T sampling
Fase di elaborazione dei dati

5.3 Acquisizione dei dati

Se il dispositivo è stato inizializzato correttamente allora è possibile cominciare ad acquisire i dati inviati dal DAQ sul bus ad una velocità di 105.3KB/s e allocati nel buffer di trasmissione messo a disposizione dal driver, che in questo caso ha dimensione pari a 64KB. Il buffer di lettura (par. 2.4.2) invece è di 63488 Byte. Tramite la funzione *FT_GetQueueStatus* è possibile monitorarne lo stato in modo da poterlo svuotare, attraverso un'operazione di lettura (*FT_Read*), non appena si riempie ed evitare così che la trasmissione di dati dal dispositivo si blocchi. All'incirca si effettua un'operazione di lettura ogni:

$$t_{read} = \frac{63488}{f_{sample}} = \frac{63488}{105,3K} = 602,92ms$$

Una volta acquisiti i dati, se si sta allineando l'interferometro, ovvero si sta leggendo una tensione continua, verrà graficato l'andamento della tensione, come su un comune oscilloscopio e ne verrà indicato il valore continuo ed efficace. Il grafico della tensione continua è ottenuto riportando tutti i campioni ed interpolandoli linearmente. Nonostante questa operazione risulti molto veloce è consigliabile disattivare questa funzione durante la fase di misura vera e propria in modo da non sprecare inutilmente le risorse di calcolo del PC. Si consideri a tale proposito che comunque è possibile valutare qualitativamente il segnale acquisito attraverso un' altro grafico ottimizzato a tale scopo (par 5.4.3). Per facilitare la lettura del grafico della tensione sono riportati i valori della tensione continua ed efficace attraverso due indicatori ed è possibile abilitare un filtro passa basso di *Butterworth* del 2° ordine e frequenza di taglio di 30 Hz, in modo da attenuare il rumore a 50Hz introdotto dalla rete di alimentazione. Sfruttando il parallelismo del linguaggio di programmazione è possibile aggiungere i dati appena acquisiti nella coda senza il bisogno di aspettare che questi vengano filtrati e graficati. Concludendo, possiamo affermare che la parte del VI che si occupa di acquisire i dati dal DAQ è indipendente dal tempo impiegato a svolgere le operazioni di filtraggio e visualizzazione e grazie alla coda, anche dal tempo impiegato dal VI per elaborare i campioni alla ricerca dei massimi di tensione.

5.4 Elaborazione dei dati

5.4.1 Gestione della coda

La parte del VI che si occupa di elaborare i campioni del segnale provenienti dal trasduttore risulta indipendente dal processo di acquisizione del dato attraverso una coda. La prima cosa che in questa parte del VI è necessario fare è monitorare lo stato della coda e gestirne l'accrescimento, in modo da evitare che cresca a dismisura ed occupi così troppa RAM. La soluzione adottata è la seguente: si impostano tre soglie, se il numero di elementi (1 elemento = 63488Byte) della coda è inferiore alla prima, anche se i dati sono stati elaborati, si introduce un ritardo in modo che il numero di elementi nella coda aumenti. Quando il numero degli elementi è superiore alla prima soglia ma è inferiore alla seconda il ritardo introdotto diminuisce. Mano a mano che si superano le soglie il ritardo introdotto decresce, fino a che , una volta superata l'ultima soglia, il ritardo si è minimizzato e gli elementi vengono continuamente prelevati dalla coda. Quando poi la misura è terminata, questa verrà svuotata nel minor tempo possibile, il ritardo prima citato viene quindi azzerato. Dalla coda verrà estratto un elemento alla volta che verrà poi passato al filtro ed in seguito elaborato. Il numero di elementi da estrarre dalla coda dipende dalla velocità di variazione dell' indice di rifrazione , quindi dalla frequenza del segnale, e dalle dimensioni del porta campioni, questo tipo di considerazioni vengono illustrate nel paragrafo 5.4.4.

5.4.2 Il filtro passa basso

I 63488 byte (campioni) prelevati dalla coda (1 elemento) vengono filtrati in modo da attenuare il rumore , visibile sui segnali a bassa frequenza. Questa funzione è implementata attraverso un filtro passa basso del 2° ordine di Butterworth la cui banda è impostabile attraverso un apposito controllo. Come è possibile notare dalla figura 5.2 le componenti armoniche del rumore rispetto al segnale rappresentato in figura 5.1 sono state notevolmente attenuate. Per segnali a bassa frequenza , quindi per variazioni lente dell' indice di rifrazione, è opportuno diminuire la banda del filtro passa basso in modo da ridurre il rumore e "spianare" (smoothing) il segnale.

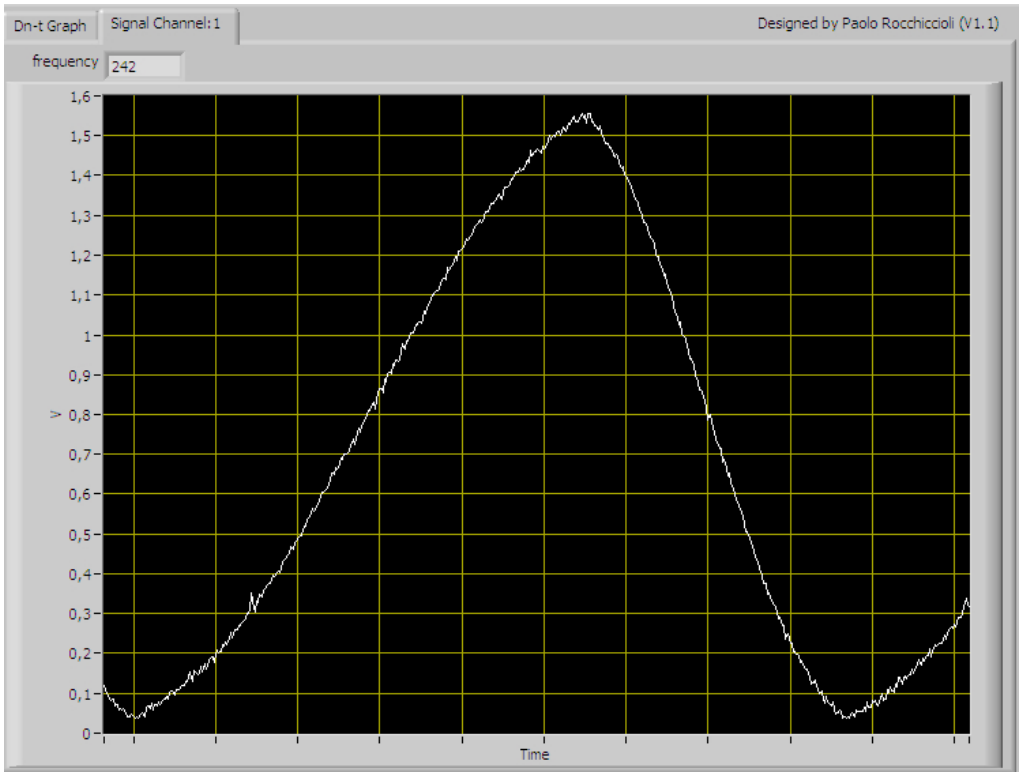


Figura 5.1: Segnale a 242Hz filtrato dal LPF con banda 56KHz

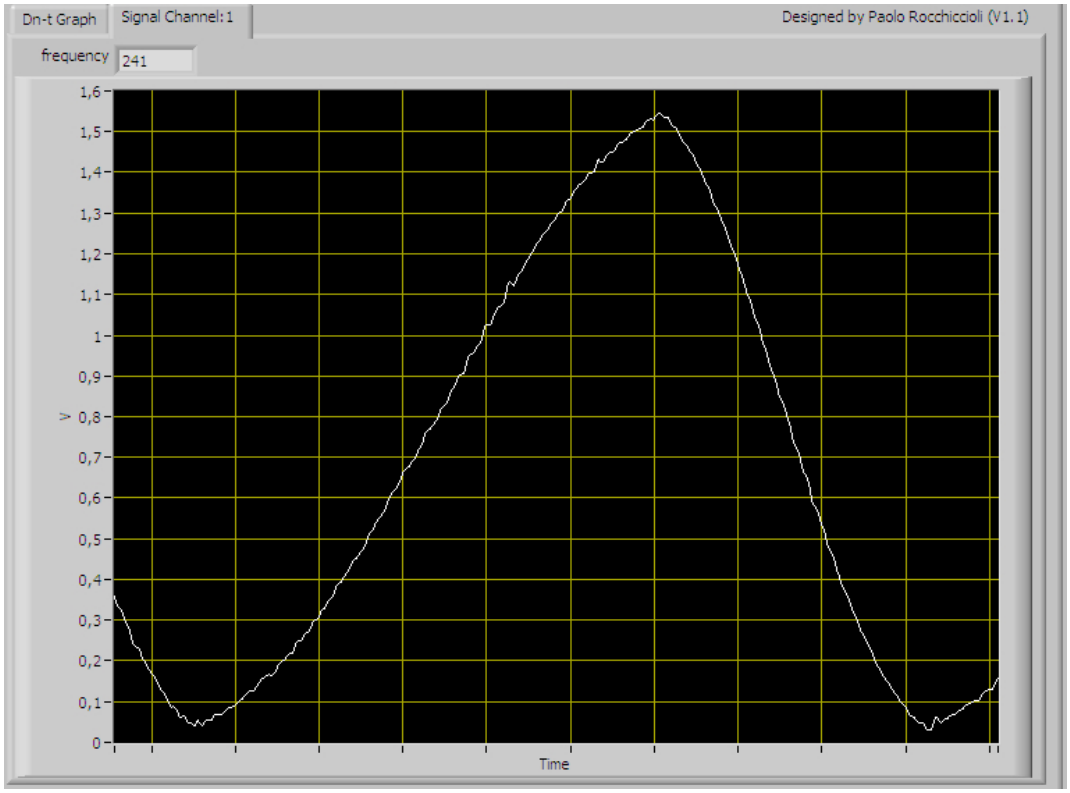


Figura 5.2: Segnale a 242Hz filtrato dal LPF con banda 20KHz.

5.4.3 Rappresentazione del segnale

Il segnale filtrato viene graficato attraverso un'interpolazione lineare dei punti (figure 5.1 e 5.2) . Questo tipo di soluzione è la più semplice è garantisce una buona lettura del segnale fino a 5-10KHz, dopo di che si ha solo una rappresentazione qualitativa del segnale, che comunque rimane più che sufficiente. Un algoritmo di interpolazione più complesso porterebbe via delle risorse di calcolo non trascurabili rendendo più "pesante" il VI in fase di esecuzione. E' opportuno sottolineare che questo tipo di scelta è giustificata anche dalla finalità che si propone di realizzare il VI, che non è quella di visualizzare fedelmente il segnale proveniente dal trasduttore ma di graficare le variazioni dell'indice di rifrazione in funzione del tempo attraverso lo studio dei massimi.

5.4.4 Ricerca dei massimi

Per valutare le variazioni dell'indice di rifrazione nel tempo è necessario individuare con esattezza i campioni c_{max} in cui si verificano i massimi del segnale (par. 1.6) vedi figura 5.3. La difficoltà principale risiede nel fatto che il segnale è discretizzato nel dominio del tempo a causa della conversione A/D. Per valutare correttamente i campioni c_{max} è pertanto necessario che la frequenza di campionamento sia elevata (minore distanza tra un campione e l'altro) oppure che la frequenza del segnale sia sufficientemente bassa (più campioni per periodo), inoltre va tenuto presente che il rumore può "camuffare" tali istanti. Per risolvere il problema è stata adoperata una funzione di libreria presente in LabVIEW 8.0 professional chiamata *Peak detector VI* [16] che ricerca il campione c_{max} attraverso lo studio del polinomio interpolante o della funzione approssimante ai minimi quadrati. E' possibile scegliere quale metodo usare impostando il numero di campioni (punti) da analizzare. Se sono 3 il VI provvede ad effettuare un' interpolazione quadratica (parabola passante per 3 punti) e attraverso lo studio del polinomio di Lagrange (2° grado) ricava con buona approssimazione il campione c_{max}. Questo tipo di analisi è consigliata quando la frequenza del segnale è elevata ed il numero dei campioni per periodo è ridotto.

Se invece i punti da interpolare sono più di 3 il VI effettua una approssimazione ai minimi quadrati, determina cioè una semplice funzione analitica che approssima l' insieme dei campioni (punti) senza necessariamente passare per essi. Questo tipo di soluzione implicitamente riduce il rumore e facilita così la ricerca dei picchi, risultando così particolarmente adatta per i segnali a bassa frequenza .

I punti che vengono interpolati/approssimati devono però avere una certa caratteristica, devono cioè avere un valore superiore ad una soglia impostata altrimenti non vengono analizzati. Tale soglia influisce quindi sul numero di massimi individuabili. Supponiamo ad esempio che il segnale sinusoidale che si sta acquisendo abbia una frequenza di 4KHz ed una ampiezza che varia tra 0,2 e 1,7V. Il numero di campioni (punti) per periodo, se la frequenza di campionamento è di 105,3KHz, utilizzando la (5.1) sarà circa 26.

$$sample = \frac{f_{sample}}{f_{signal}} \qquad (5.1)$$

Se impostiamo la soglia ad un valore superiore al valor medio del segnale (0,95 V) e pretendiamo di interpolare/approssimare un numero superiore o uguale alla metà dei campioni per periodo (13) il VI in questione non riuscirà ad individuare correttamente tutti i massimi del segnale. Se però la frequenza diminuisce,aumenta il numero di campioni per periodo, sarà possibile individuarli tutti.
Al fine di rendere lo strumento più semplice e intuitivo, si è deciso di trovare una regola empirica che permetta di stabilire quali sono i parametri che permettono di rilevare tutti i picchi del segnale indipendentemente dalla frequenza del segnale.

Dopo una serie di prove effettuate con un generatore di onde sinusoidali con frequenza compresa fra 20Hz e 20KHz siamo arrivati alla conclusione che la soglia che si deve avere:

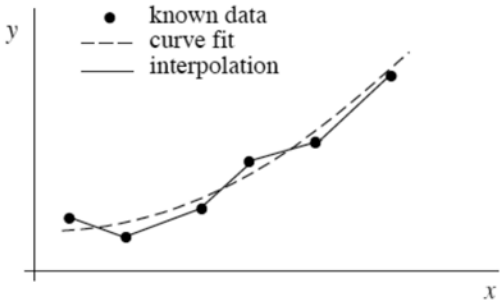
$$V_{soglia} \cong \frac{V_{MAX}}{3,5} \qquad (5.2)$$

mentre il numero di punti da interpolare/approssimare deve essere:

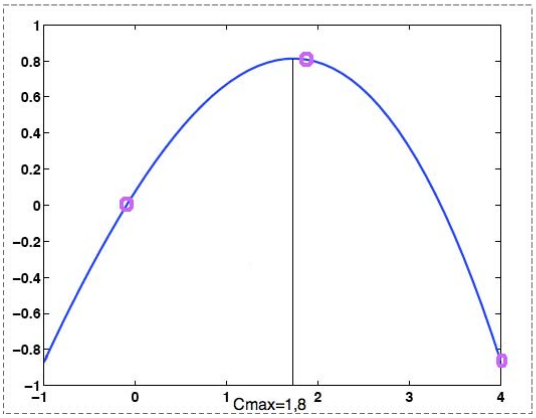
$$punti \cong \frac{sample}{3,5} = \frac{f_{sample}}{f_{signal} \cdot 3,5} \qquad (5.3)$$

Si è arrivati a questa conclusione confrontato il numero dei massimi individuati dal VI, con il numero di quelli reali del segnale, dato dalla relazione (5.4) e riportato anche su pannello dello strumento.

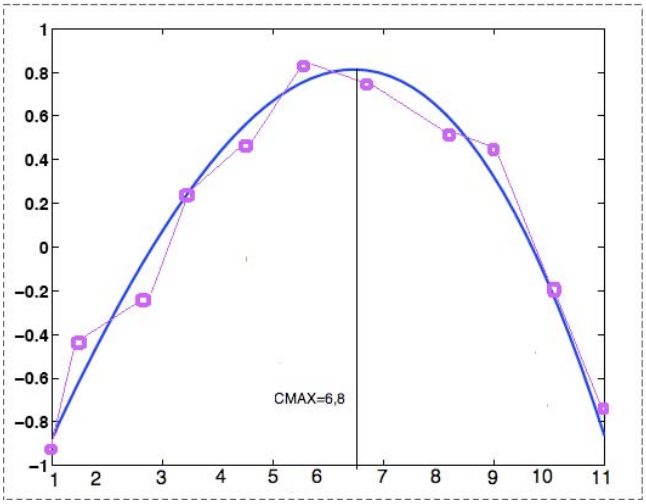
$$n_{MAX} \cong \frac{DimensioneBufferLettura}{f_{sample}} \cdot f_{signal} = \frac{63488}{f_{sample}} \cdot f_{signal} \qquad (5.4)$$



(a)



(b)



(c)

Figura 5.3: Differenza fra segnale interpolato linearmente e approssimato ai minimi quadrati (a); parabola che interpola 3 punti: interpolazione quadratica (b), il massimo si ha tra il campione 1 e 2 ed è pari a 1,8 ; approssimazione ai minimi quadrati di 11 punti(c), se i punti fossero interpolati linearmente il massimo sarebbe assegnato al 6° campione mentre in realtà è situato tra il 6° e il 7° ed è pari a C_{max}=6,8.

La relazione (5.3) viene verificata automaticamente dal VI il quale pensa a calcolare di volta in volta il numero corretto di punti. La relazione (5.2) deve essere verificata impostando manualmente sul pannello dello strumento il valore della soglia.
Usando la (5.3) è possibile stabilire quando i campioni c_{max} sono ottenuti per interpolazione:

$$f_{signal} > \frac{f_{sample}}{punti \cdot 3{,}5} = \frac{105{,}3K}{3 \cdot 3{,}5} = 10{,}03KHz$$

5.4.5 Elaborazione dei massimi

Dal valore di c_{max} è possibile risalire all' istante temporale nel quale si verifica semplicemente attraverso la relazione:

$$t_{\max} = t_0 + c_{\max} \cdot t_{sample} \qquad (5.5)$$

Conoscendo tutti gli istanti a cui si verifica un massimo è semplice ricavare la variazione dell'indice di rifrazione nel tempo. E' sufficiente infatti registrare *k* massimi per valutare una variazione dell'indice da monitorare pari a Dn (par. 1.6.2):

$$k = \frac{Dn \cdot h}{\lambda} \qquad (5.6)$$

e mettere in relazione l' istante temporale nel quale si verifica il *k-esimo* massimo e Dn.
Il *Peak Detector VI* (par 5.3.3) analizza i 63488 campioni (dimensione del buffer di lettura) inviati dal DAQ e fornisce il numero dei campioni (inteso come locazione)in cui si ha un massimo del segnale (c_{max}). Il numero dei massimi si ricava dalla relazione (5.4). Se la frequenza del segnale è tale da rendere impossibile individuare il massimo tra i 63488 campioni è necessario prelevare e fare analizzare al VI più elementi dalla coda (1 elemento=63488 campioni). La frequenza minima per cui è possibile individuare un massimo attraverso un solo elemento è data dalla relazione (5.4) impostando n_{MAX}=1 per cui si ha:

$$f_{signal} \cong \frac{f_{sample} \cdot 1}{63488} = 1{,}65Hz$$

Se si è interessati ad osservare variazioni dell' indice di rifrazione pari a Dn=10^{-3} con un portacampioni di 10 mm allora si dovranno contare 158 massimi (dalla relazione 5.6) quindi se il segnale ha frequenza pari a 1,65Hz prima di osservare la variazione desiderata trascorreranno (se la frequenza rimane costante):

$$t = 158 \cdot \frac{63488}{105.3KHz} = 95{,}262s$$

Se si suppone che la variazione totale dell'indice di rifrazione sia pari a 100^{-3} allora il tempo trascorso (se la frequenza rimane costante) è di 952,62s (circa 16minuti) ed il grafico Dn-t sarà costituito da 10 punti interpolati linearmente tra di loro.

Prelevando ed accodando 2 elementi della coda (1 elemento = 63488 campioni) la frequenza minima del segnale sarà di :

$$f_{signal} \cong \frac{f_{sample} \cdot 1}{63488 * 2} = 829mHz$$

in tal caso, il tempo necessario (se la frequenza rimane costante) per osservare la variazione totale dell' indice di rifrazione, supposta a pari a 100^{-3} ,sarà di circa 32 minuti .

La velocità di variazione dell'indice di rifrazione influisce quindi sulla modalità di gestione della coda, ovvero sul numero di elementi da prelevare dalla coda, affinché possa essere individuato un massimo all'interno di ogni elemento attraverso il *Peak Detector VI.* Ipotizzando variazioni dell' indice di rifrazione di circa 100^{-3} nell'arco di 10-15 minuti può essere sufficiente prelevare un elemento alla volta (h=10mm, Dn=10^{-3}) . Diversamente sarà necessario modificare il VI in modo che il numero di elementi da prelevare in funzione della frequenza del segnale è :

$$n°elementi \cong \frac{f_{sample}}{63488 \cdot f_{signal}} = \frac{f_{sample}}{63488 \cdot f_{signal}} \qquad (5.7)$$

La frequenza del segnale si ricava se si ipotizza di conoscere il tempo, t_{hp}, necessario affinché si abbia la massima variazione dell' indice di rifrazione Δn_{MAX} e la profondità del portacampioni: h , infatti:

$$\begin{cases} \Delta n_{\max} = \dfrac{k \cdot \lambda}{h} \\ k \cdot T_{signal} = t_{hp} \end{cases}$$

risolvendo il sistema si ottiene:

$$\frac{\Delta n_{\max} \cdot h}{\lambda} = \frac{t_{hp}}{T_{signal}}$$

risolvendo rispetto a T_{signal} si ottiene:

$$T_{signal} = \frac{t_{hp} \cdot \lambda}{\Delta n_{\max} \cdot h} \qquad (5.8)$$

Sostituendo l'espressione (5.8) nell' espressione (5.7) si ottiene:

$$n°elementi \cong \frac{f_{sample}}{63488 \cdot f_{signal}} = \frac{f_{sample}}{63488} \cdot \frac{t_{hp} \cdot \lambda}{\Delta n_{\max} \cdot h} \qquad (5.9)$$

Se t_{hp}=900 (15 minuti) , $\Delta n_{MAX}=100^{-3}$, h=10mm e la frequenza di campionamento di 105.3KHz allora il numero di elementi da prelevare dalla coda è pari a uno. Se invece la dimensione del portacampioni si dimezza allora è necessario prelevare dalla coda due elementi alla volta.

Concludendo possiamo affermare che potrebbe essere necessario modificare la gestione della coda (par. 5.4.1) in funzione dei risultati che ci aspettiamo e della dimensione del portacampioni, che al momento della progettazione del VI non sono certi. Questa operazione non richiede la riprogettazione del VI ma il semplice inserimento di un ciclo for-loop, cosa assai semplice e veloce.

L'algoritmo che permette di analizzare i massimi individuati dal *Peak Detector VI* è implementato nella funzione *Calcola Dn-t VI*. Questo VI permette di "contare" i massimi contenuti in un array di lunghezza variabile contenente tutti i campioni c_{max}. La lunghezza dell'array dipende dal numero di massimi individuati, e quindi dalla frequenza del segnale. La funzione di conteggio è realizzata estrapolando dall' array il valore c_{max} contenuto all' indice *k* ricavato dalla relazione (5.6) e moltiplicandolo per il tempo di campionamento e sommandolo al tempo trascorso(5.5). Il VI in questione è in grado di gestire il caso in cui debba estrarre dall' array più valori indicizzando autonomamente gli array senza concatenarli come rappresentato in figura 5.4. Una volta elaborato il primo array si calcola l'offset come:

$$Offset = DimensioneArray - valore\_puntatore(k) \qquad (5.10)$$

All' analisi dell' array successivo il puntatore non sarà più dato dal valore di *k* e dai suoi multipli ma dal nuovo valore calcolato come:

$$k' = k - offset \qquad (5.11)$$

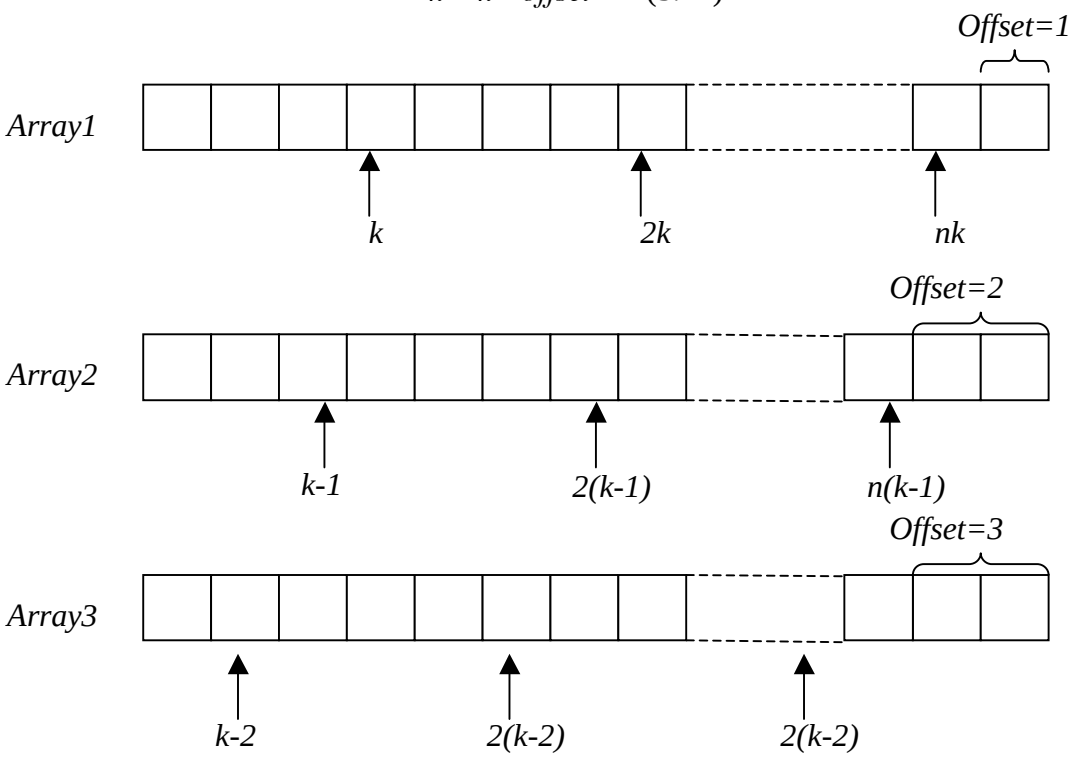


Figura 5.4: Ricerca e indicizzazione all' interno degli arrays contenenti i valori c_{max} nel caso in cui la dimensione dell'array sia maggiore di k .

Calcola Dn-t VI è inoltre in grado di gestire il caso in cui la dimensione dell'array non sia tale da permettere la lettura del valore contenuto all' indice *k* perché la dimensione dell'array è inferiore a *k* come nel caso di Figura 5.5. In tale caso, se la dimensione del primo array è maggiore del valore del puntatore *k* questi riviene calcolato come:

$$k' = k - DimensioneArray \qquad (5.12)$$

altrimenti si estrae il valore c_{max} contenuto all'indice *k* e si calcola il nuovo valore del puntatore come:

$$k' = k - (DimensioneArray - k) \qquad (5.13)$$

Se gli arrays seguenti hanno dimensione superiore al valore del puntatore allora si estrae il valore contenuto all'indice ricavato con la relazione (5.13),altrimenti il valore del puntatore verrà aggiornato (relazione 5.12) fin tanto che il valore dell'indice non è inferiore alla dimensione dell'array.

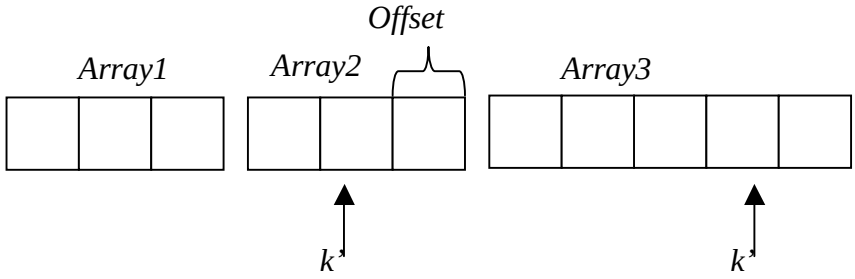


Figura 5.5: Gestione del puntatore nel caso in cui la dimensione dell'array sia inferiore al valore di k (numero di massimi necessari per avere una variazione dell'indice di rifrazione pari a Dn).

Calcola Dn-t VI calcola poi, attraverso la relazione (5.5), gli istanti t_{max} , e gli associa alla variazione Dn corrispondente dell' indice di rifrazione come in Figura 5.6 (par. 1.6.2). Ogni punto di coordinate (t_{max},Dn) viene riportato sul grafico Dn-t. I punti sono interpolati in maniera lineare e l'aggiornamento del grafico è istantaneo. Al termine della misura, se prima di avviare il test è stata selezionata questa opzione, i punti rappresentati su questo grafico saranno salvati su un file di testo su due colonne separate, in modo da semplificare l'esportazione dei dati in programmi di analisi.

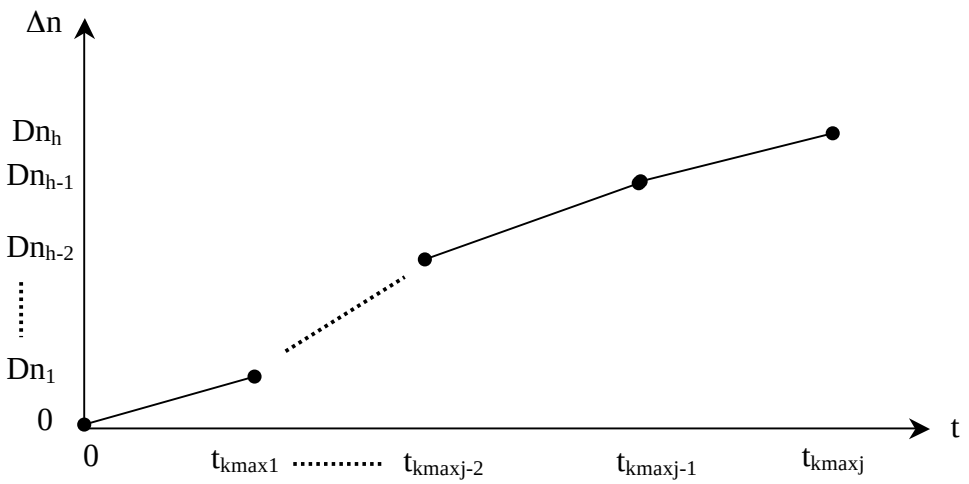


Figura 5.6: Rappresentazione delle coppie di punti (t_{max},Dn) sul grafico che rappresenta la variazione dell'indice di rifrazione in funzione del tempo.

5.5 Il pannello di controllo del VI.

Il pannello di controllo del VI è suddiviso in tre Tab:

- Calibrazione e test time
- Testing
- USB Setting/Diagnostic

L'ordine delle tab rispetta l'ordine delle operazioni che devono essere effettuate per effettuare una misura. L'ultima tab è stata inserita per facilitare il debugging della software e la ricerca di errori che possono verificarsi durante la misura.

5.5.1 Calibrazione e test time

In questa Tab è possibile:

- **Impostare la durata della misura** tramite una manopola o inserendo il valore in HH:MM:SS da tastiera nel display sottostante.
- **Visualizzare lo stato di avanzamento** del test in termini di tempo trascorso attraverso due indicatori. Uno grafico (barra di avanzamento %) ed uno strumentale (conto alla rovescia).
- **Stoppare il test** prima del tempo impostato.
- **Abilitare/disabilitare il filtro** per eliminare il rumore con frequenza superiore a 50Hz
- **Visualizzare l'andamento della tensione** durante la fase di allineamento (calibrazione) dell' interferometro.
- **Visualizzare il valore** continuo ed efficace della tensione monitorata.
- Abilitare/disabilitare **l'aggiornamento del grafico.**

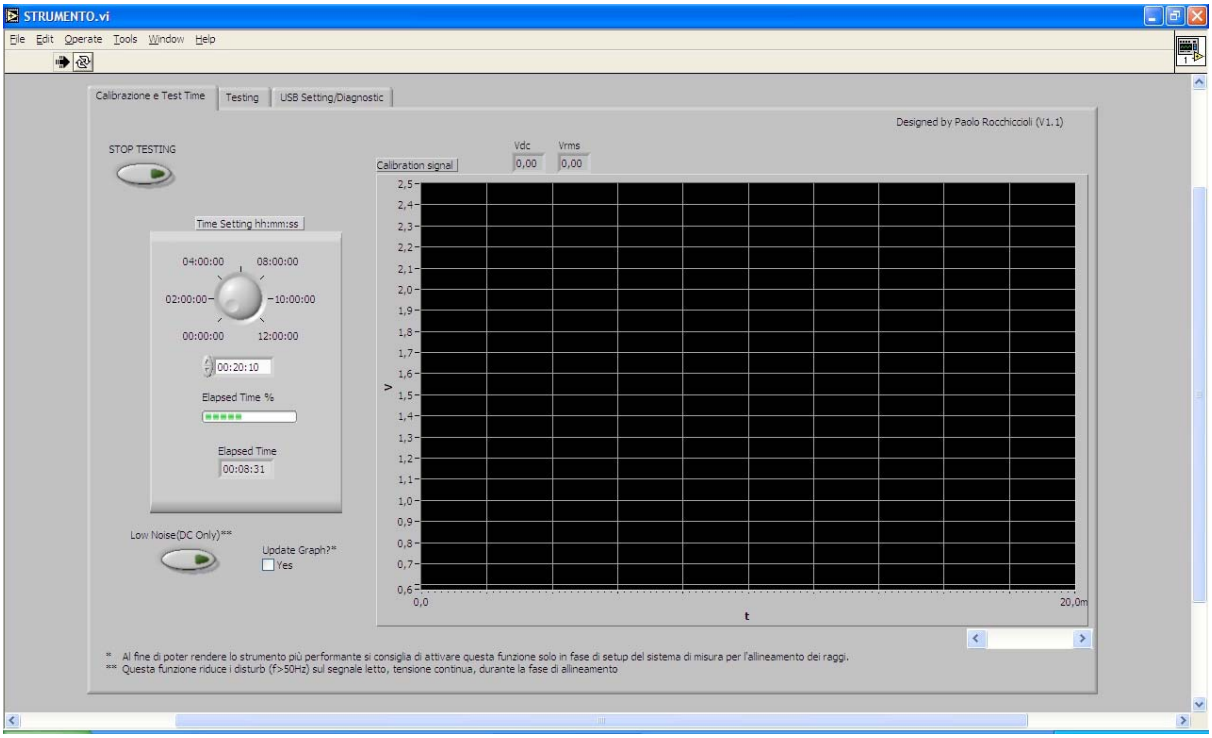


Figura 5.7: Tab di Calibrazione e test time

5.5.2 Testing

Questa Tab è suddivisa a sua volta in altre 2 Tab:

- Grafico Dn-t
- Andamento del segnale presente sul canale 1 del DAQ.

In questa Tab è possibile:

- Impostare **"Resolution Dn"**: la minima variazione dell'indice di variazione che si è intenzionati a monitorare.
- Impostare la profondità del portacampioni contenete l'idrogelo**: h**.
- Impostare il tempo di campionamento del DAQ : **tsample**.
- Visualizzare il numero di massimi: **nmax** , necessari affinché , con i parametri impostati, si verifichi una variazione Dn dell' indice di rifrazione.
- Visualizzare **Dn** la minima risoluzione che è possibile monitorare e che dipende dalla profondità del portacampioni (par. 1.5).
- Visualizzare la frequenza di campionamento: **fsample** .
- Impostare il metodo di ricerca dei massimi : **peaks** .
- Impostare la soglia secondo la relazione (5.2) : **threshold** .
- Visualizzare il numero di campioni : **width** necessari per una corretta ricerca del massimo (relazione 5.3)
- Visualizzare il numero dei massimi individuati e quelli da trovare, ricavati secondo la relazione (5.4).
- Visualizzare la frequenza del segnale che si sta acquisendo.
- Abilitare/disabilitare il salvataggio del grafico Dn-t.
- Impostare in quale cartella si vuole salvare il file .txt qualora sia stato abilitato il salvataggio del grafico.

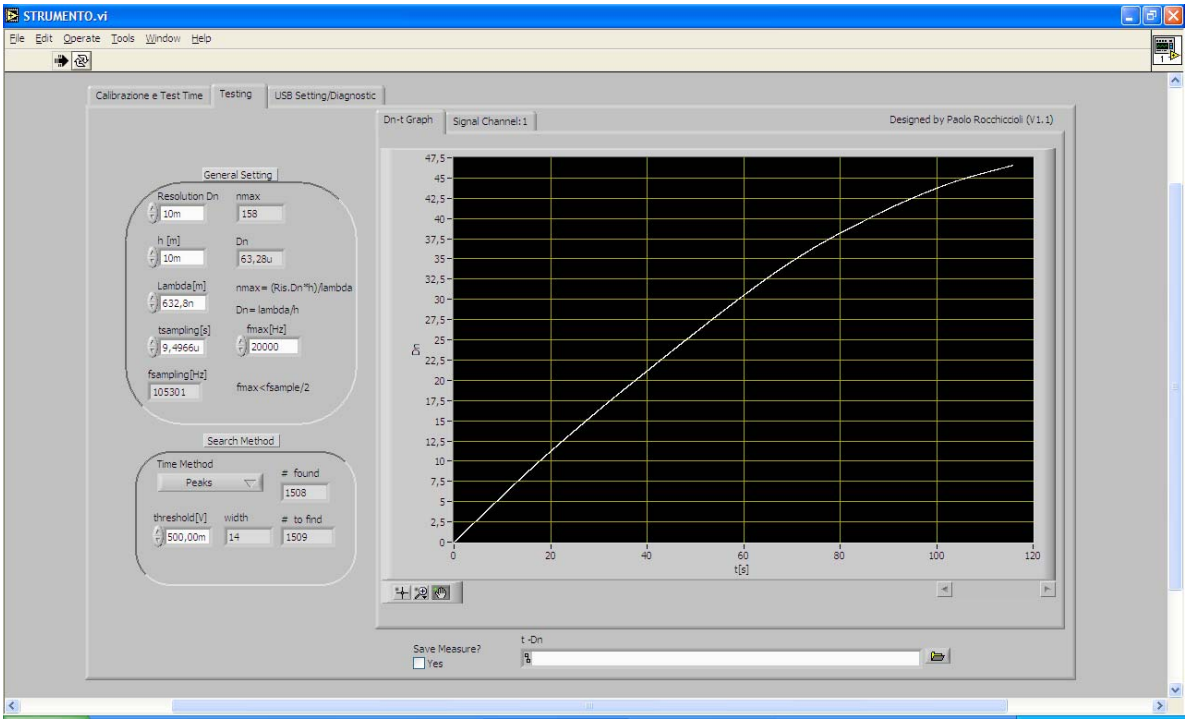


Figura 5.8: Tab di testing e visualizzazione del grafico Dn-t. Il grafico rappresentato è stato ottenuto variando la frequenza di un generatore di segnale da 20KHz a 20Hz, si spiega così l'elevata variazione (47,5) dell'indice di rifrazione.

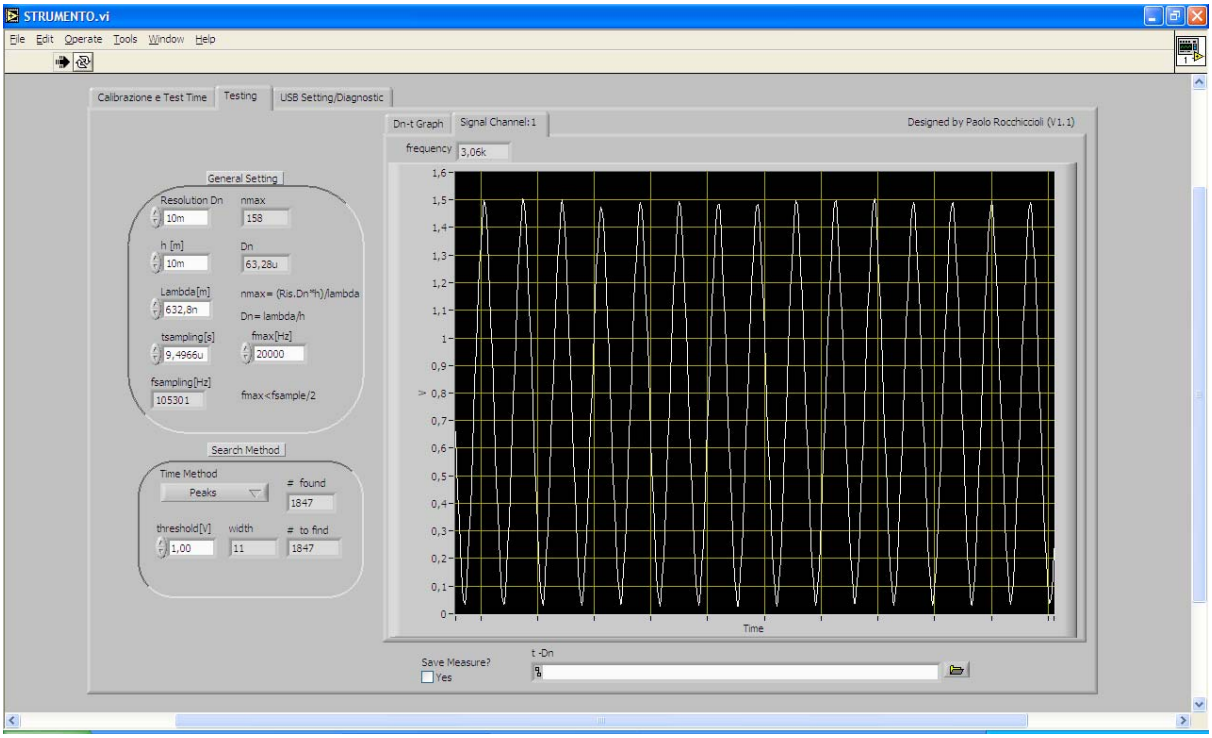


Figura 5.9: Tab testing e rappresentazione del segnale proveniente dal canale 1 del DAQ.

5.5.3 USB Setting/diagnostic

In questa Tab è possibile:

- Impostare il **latency timer.**
- Impostare la **dimensione dei buffer** allocati dal driver D2XX.
- Impostare il **controllo di flusso** del bus USB.
- Individuare attraverso il **vettore di diagnostica**, qualora l'inizializzazione della comunicazione con il DAQ non sia andata a buon fine, la causa del malfunzionamento.
- Monitorare lo **stato del buffer di lettura** attraverso la visualizzazione parziale dei dati ricevuti e il numero di byte letti.
- **Controllare la velocità di svuotamento** della coda e monitorarne lo stato.

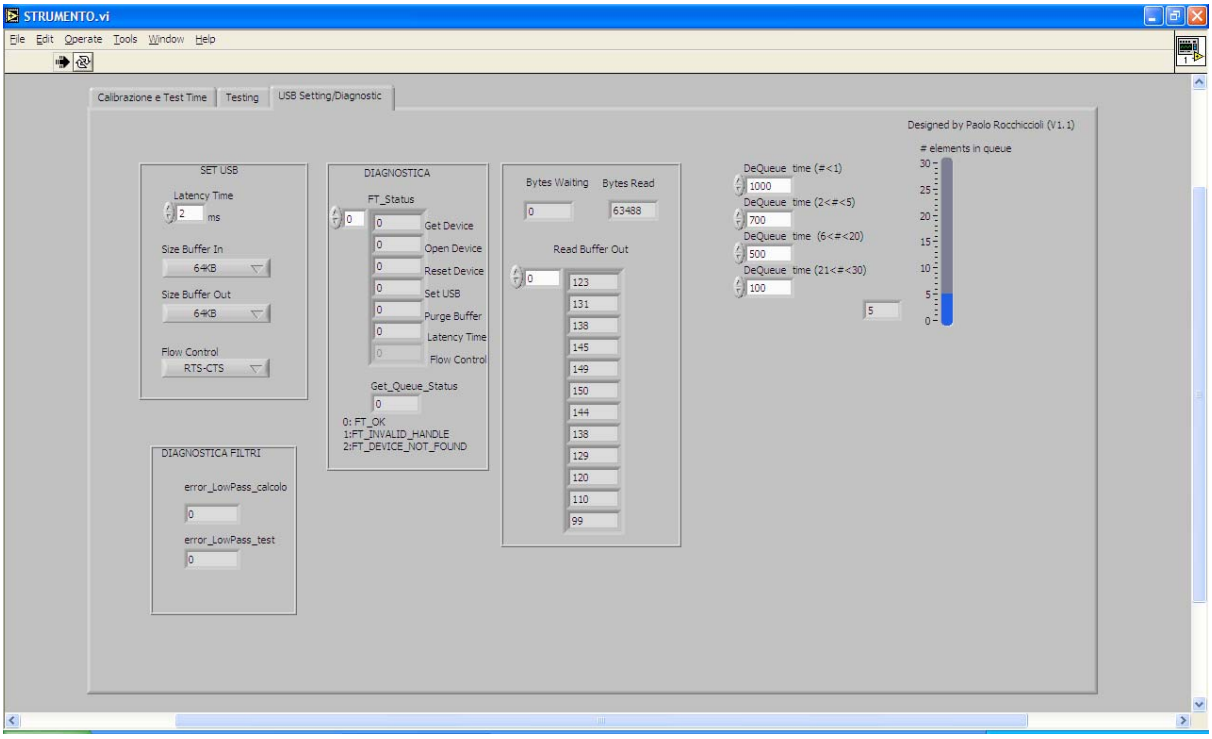


Figura 5.10: Tab USB Setting/Diagnostic.

5.6. Installazione del DAQ e del VI

Per installare il DAQ è necessario scaricare i driver D2XX all' indirizzo http://www.ftdichip.com/ . Collegato il DAQ al PC questo fa partire la procedura di installazione del dispositivo. A questo punto è necessario eseguire una installazione personalizzata andando a selezionare la cartella in cui sono stati salvati i driver D2XX. La stessa procedura dovrà essere nuovamente eseguita. Al termine dell' installazione potrebbe essere necessario riavviare il sistema operativo. Il driver installati permettono di scollegare il DAQ dal PC senza dover effettuare nessuna rimozione sicura dell'hardware.

Per evitare di dover installare LabVIEW sul calcolatore impiegato per fare la misura, riducendo così anche i costi delle licenze, si è deciso di distribuire l'applicazione sottoforma di file ***.exe*** . Per essere eseguita però è necessario installare il ***LabVIEW run-time engine*** scaricabile gratuitamente all' indirizzo :

http://joule.ni.com/nidu/cds/fn/p/sn/n23:4.28.5000/.

Per rendere più immediata questa procedura è stato creato un ***installer_Strumento (SetUp_Strumento.exe)*** che permette di installare il *run-time engine* e l'applicazione ***Strumento.exe***. Dopo aver riavviato il sistema sarà possibile iniziare ad usare lo strumento per fare le misure.

5.7 Debugging del software

Per effettuare il "collaudo" del client software è stato impiegato, collegandolo al canale 1 del DAQ, un generatore di onde sinusoidali a frequenza variabile 20Hz-20KHz ed ampiezza pari a ±0,22Vcc. Per far si che la forma d'onda generata fosse di ampiezza compresa 0 e 0,22Vcc il pin 2 dell' ICL8038 è stato collegato ad un partitore resistivo collegato ad un generatore di tensione negativo e variabile in modo da rendere il valor medio pari alla metà dell'ampiezza massima della sinusoide. Prima di effettuare il "debugging finale" ogni singolo subVI richiamato nel VI è stato collaudato singolarmente nella maniera più idonea, a seconda della funzione svolta. Il tempo di collaudo, grazie alla modularizzazione del VI e alla presenza della Tab di Diagnostica, è stato notevolmente ridotto.

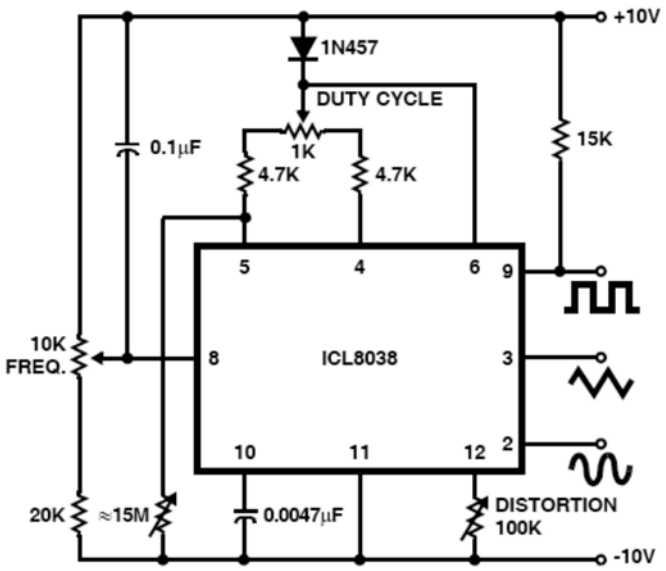


Figura 5.10: Schema elettrico del generatore di onde sinusoidali impiegato per collaudare il client software.

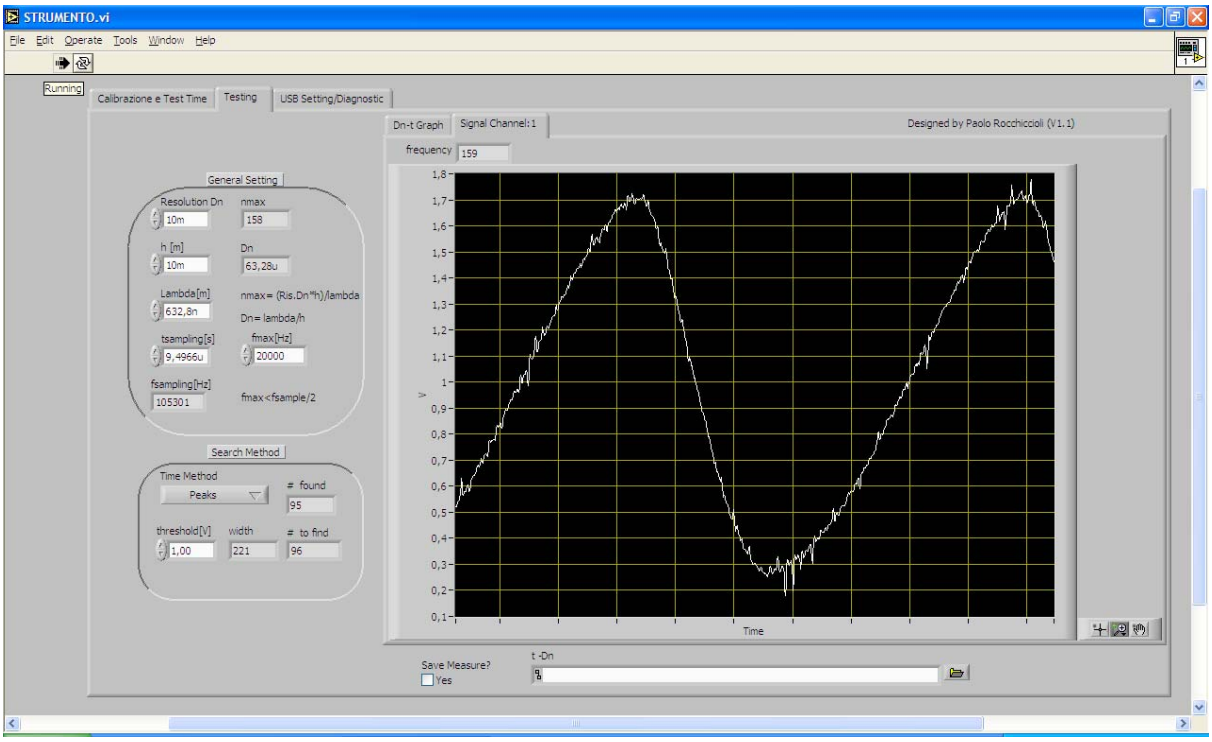


Figura 5.11: Test del programma con segnale a bassa frequenza (159Hz) generato con il circuito di figura 5.10. Il numero di massimi da trovare corrisponde al numero di massimi trovati all'interno dell' elemento (63488Byte) prelevato dalla coda. Il valore della frequenza, l'ampiezza e l'andamento del segnale corrispondono a quelli misurati con l'oscilloscopio Fluke 123.

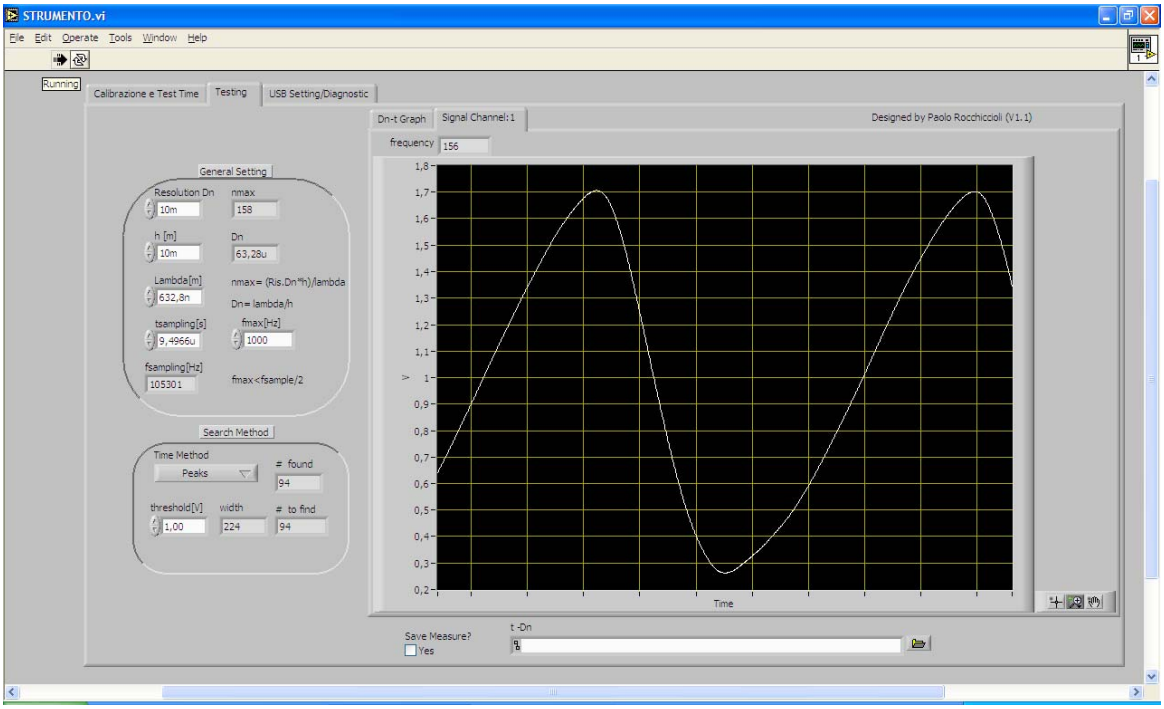


Figura 5.12: Verifica del corretto funzionamento del filtro passa basso. Il solito segnale usato in figura 5.11 qui viene filtrato. Il filtro passa basso in questo caso ha banda 1KHz anziché 20KHz. E' possibile modificare il valore della banda(fmax) durante la misura a seconda della frequenza del segnale da acquisire.

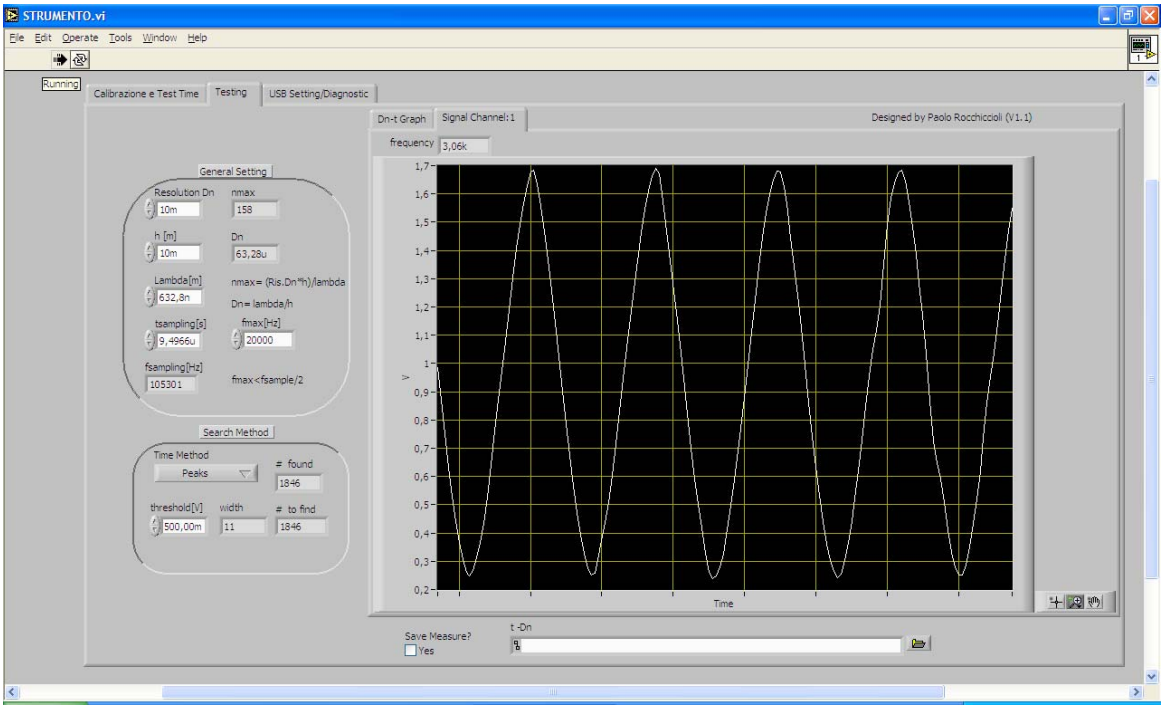


Figura 5.13. Test con segnale sinusoidale a 3063Hz. Anche in questo caso il numero di massimi rilevati corrisponde al numero di massi da rilevare all'interno dei 63488Byte(campioni) analizzati.

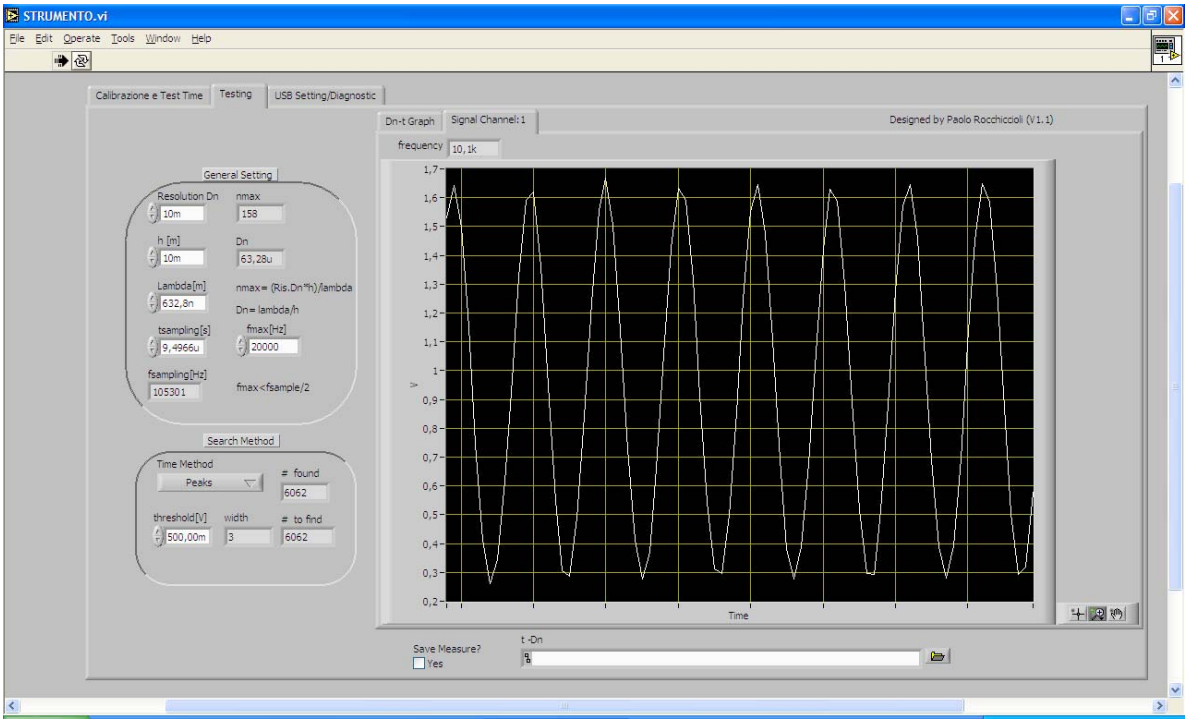


Figura 5.14: Test con segnale sinusoidale a 10KHz. Anche in questo caso il numero di massimi rilevati corrisponde al numero di massi da rilevare all'interno dei 63488Byte(campioni) analizzati. Il numero di campioni per periodo è ridotto, quindi la visualizzazione del segnale peggiora (interpolazione lineare tra punti)

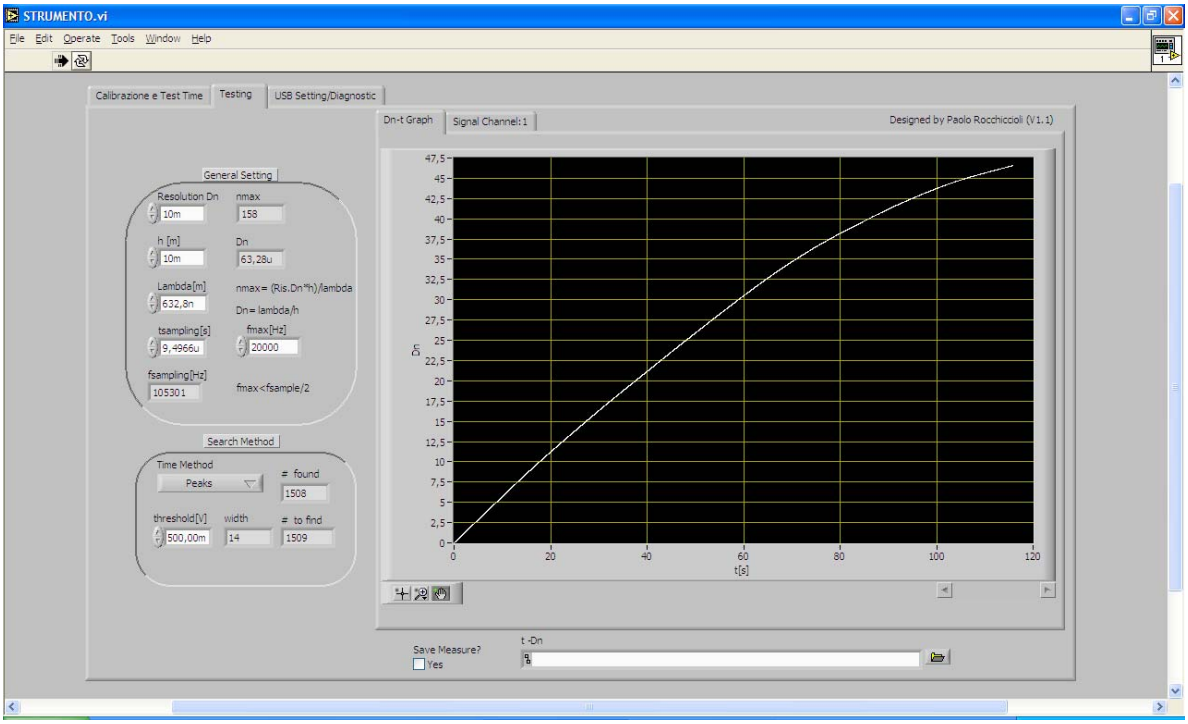


Figura 5.15: Test di visualizzazione del grafico Dn-t. Il grafico rappresentato è stato ottenuto variando la frequenza del generatore di segnale da 20KHz a 20Hz, si spiega così l'elevata variazione (47,5) dell'indice di rifrazione.

Conclusione

In questo lavoro sono state descritte le fasi di progettazione e realizzazione di un sistema di elaborazione ed acquisizione dati da un interferometro ottico, nella configurazione di Mach-Zehnder, finalizzato a caratterizzare la cinetica del meccanismo di reticolazione di un idrogelo. Il trasduttore è stato provato con il raggio laser che verrà usato durante le misure cambiando le caratteristiche dell'ottica interposta tra il puntatore ed il sensore, in modo da modificare la potenza incidente sul fotodiodo. I risultati della prova sono stati positivi ed hanno confermato la bontà delle scelte progettuali effettuate, in termini di guadagno, facilità di posizionamento del trasduttore all'interno del sistema di misura e nell' allineamento dell' interferometro. La scheda di acquisizione dati (DAQ) è stata programmata, attraverso il firmware che risiede nell' MCU, per acquisire dati da un solo canale con una frequenza di campionamento pari a 105.3KS/s. In futuro sarà possibile , se necessario, sfruttare tutti i canali (4) del DAQ e campionare ogni canale a circa 26KS/s. La frequenza di campionamento può arrivare a 150KS/s o fino a 200-250KS/s su un calcolatore dedicato. La presenza di 8 I/O digitali configurabili rende il DAQ idoneo a pilotare un microposizionatore,un servomotore, un motore passo-passo oppure per rilevare la posizione del portacampione attraverso dei sensori di prossimità. Qualora si fosse interessati a caratterizzare la cinetica del meccanismo di reticolazione in funzione della posizione si può ad esempio aggiornare il firmware dell'MCU, inserendovi una routine in grado di gestire un motorino passo-passo in modo che questo sposti il porta campioni rispetto al raggio laser di una certa quantità ad intervalli di tempo prestabiliti. Questi parametri possono essere scelti anche attraverso il VI, inviandoli al DAQ prima di iniziare la misura.

Si può pensare inoltre di monitorare le variazioni della temperatura sfruttando un canale del DAQ oppure usando un sensore con interfaccia I^2C collegato a due ingressi digitali liberi. Nel secondo caso è possibile memorizzare le misure nella EEPROM interna all' MCU fino ad un massimo di 512Byte ed inviarle al VI al termine della misura. Qualora invece si fosse interessati ad eseguire una vera e propria caratterizzazione della cinetica del meccanismo di reticolazione in funzione della temperatura, il microcontrollore può

essere usato per implementare un controllore di tipo P, PI , PD o PID. Sfruttando così gli 8 I/O digitali per collegarvi un sensore di temperatura ,il circuito di controllo di una ventola ed una resistenza corazzata è possibile controllare la temperatura del portacampioni posto all'interno di un contenitore sigillato e trasparente al raggio laser.
Il punto di forza di questo DAQ è il basso costo di realizzazione, da un decimo ad un ventesimo, rispetto a quelli attualmente in commercio con frequenza di campionamento di 25-50KS/s per canale. Nella maggioranza dei DAQ in commercio, a differenza di quello realizzato, la programmabilità è legata interamente al software (LabVIEW). Ciò ha permesso di ridurre i costi di realizzazione anche se è necessario un aggiornamento del firmware interno al DAQ. Questa operazione in termini di studio, è ridotta al minimo ed è legata al *know-how* del progettista, mentre dal punto di vista hardware (programmatore) richiede un dispendio di poche decine di euro. In applicazioni future, si può ad esempio pensare di disporre di più versioni del firmware, da scaricare, attraverso il connettore di programmazione *In-System* nel DAQ, a seconda delle caratteristiche del sistema di misura o di progettare un firmware in grado modificare la frequenza di campionamento ed il canale di acquisizione in funzione dei parametri impostati dal pannello dello strumento. La prototipizzazione estrema del PCB e la modularizzazione del VI hanno permesso di eseguire il collaudo finale in minor tempo e far si che la libreria (subVI) realizzata in questo lavoro di tesi, possa essere utilizzata per progettare un VI con caratteristiche diverse o rivisitare l'attuale.

Bibliografia:

[1] *"Optical interference Study of Physiological Fluids Transportation through Semipermeable Membranes"*. **Journaul of Molecular Liquids 106/1 (2003) 81-88**

[2] "*Cuprophane membrane permeability to urea studied by optical interference method"*. **Journaul of Molecular Liquids (10 gennaio 2007)**

[3] *"Misura della variazione dell'indice di rifrazione dell'acqua in funzione della temperatura mediante tecniche interferometriche "* ***Dott. Francesco Michelotti Università di Roma "La Sapienza Dipartimento di Energetica Laboratorio di Fotonica Molecolare.***

http://w3.uniroma1.it/cattedra_michelotti/Esercitazione3.pdf

[4] Silicon Photodetectors serie 5T Datasheet- **Centronics** www.centronic.co.uk

[5] "*AD820 Datasheet*" – **Analog Device** www.analog.com

[6] *"Op Amp Applications Handbook"* **Walt Jung-** Analog Device www.analog.com

[7] *"Op Amps for every one- Design reference"*- **Ron Mancini** - Texas Instruments www.ti.com

[8] "*BUF04 Datasheet*" - **Analog Device** www.analog.com

[9] "*Universal Serial Bus Revision 2.0 specification"- Rev2.0 27 aprile 2000 www.usb.org*

[10] *"Appunti di sistemi operativi"*- **Prof. Anastasi**- Università di Pisa

[11] *"ATMEGA16(L) Datasheet"*- **Atmel Corporation**- www.atmel.com

[12] *"FT245 Datasheet"*- **FTDI chip** – www.ftdichip.com

[13] "*D2XX Programmer's guide"*- **FTDI chip**- www.ftdichip.com

[14] *"FT245BM Designers Guide Version 2.0"* – **FTDI chip** -www.ftdichip.com

[15] *"AD7825 Datasheet"* – **Analog Device** – www.analog.com

[16] "*Peak detection using LabVIEW and Measurement Studio" – **NI Developer Zone**- Tutorial- 6 Dicembre 2006.*