ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA, INFORMATICA E TELECOMUNICAZIONI

PROGETTO DI UN CIRCUITO DI CONDIZIONAMENTO DEL SEGNALE PER SISTEMI DI ANALISI REOLOGICA

Elaborato in

Elettronica Dei Sistemi Digitali

Relatore

Aldo Romani

Presentata da

Manuel Siboni

Correlatori

Luigi Ragni
Matteo Filippi

Sessione II

Anno Accademico 2014/2015

Sommario

Introduzione

Il lavoro di tesi proposto riguarda la progettazione di una rete di condizionamento di segnali analogici prodotti da un sistema di misura reologica.

La reologia è la scienza che studia il comportamento della materia sotto l'effetto di sollecitazioni. Essa si occupa dello studio di diversi materiali che presentano caratteristiche non omogenee che, se non tenute in considerazione, possono portare a comportamenti inattesi in fase di lavorazione e di controllo. La branca della reologia che è coinvolta nello studio di questa tesi è la reologia degli alimenti.

Lo studio delle caratteristiche degli alimenti porta a numerosi scopi quali:

- il controllo della qualità dei prodotti grezzi e dei processi di lavorazione;
- la progettazione di macchine per la lavorazione del prodotto stesso, che verranno modellate in base alle sue proprietà reologiche;
- l'accettabilità di un alimento, che permette di identificare le caratteristiche maggiormente apprezzate dai consumatori, etc.

Durante il periodo di tirocinio curriculare, svolto all'interno della Facoltà, mi è stato possibile lavorare sulla parte di programmazione dell'interfaccia grafica di comando e controllo del sistema di misura, realizzata grazie all'importante collaborazione del Dott. Matteo Filippi, tutor di questa tesi.

Alla prova dell'interfaccia grafica si è notato che i grafici contenuti al suo interno mostravano curve che, solitamente, erano caratterizzate da variazioni troppo piccole da poter essere apprezzate a occhio nudo. Così è nato lo scopo di questo nuovo progetto, che consiste nell'andare a incrementare la dinamica dei segnali rilevati durante le misure reologiche in modo da facilitarne l'interpretazione nei casi in cui le misure stesse siano di difficile lettura.

Per realizzare tutto ciò sono state svolte diverse fasi di studio e progettazione che qui elencherò a grandi linee, per poi passare successivamente nel dettaglio:

- Dapprima è stato possibile visionare lo strumento di misura per comprendere lo schema di principio dello strumento e i blocchi d'interesse da interfacciare con la rete di condizionamento da realizzare;
- In seguito è stato studiato il principio di funzionamento del circuito di condizionamento e, in base ai risultati sperimentali ottenuti, si è scelto quello più adatto, costruendolo su breadboard per la successiva simulazione;

- A questo punto si è passati alla programmazione firmware e software;
- Così facendo è stato possibile compiere delle simulazioni di funzionamento utilizzando gli strumenti di laboratorio;
- Infine, sono state tratte conclusioni sui possibili sviluppi futuri del progetto.

Le diverse fasi di questo progetto sono state divise per capitoli, per una corretta esposizione e lettura.

- **Capitolo 1:** il primo capitolo introduce il sistema di misura esistente e le motivazioni che hanno portato al nuovo progetto.
- **Capitolo 2:** nel secondo capitolo si entra più nello specifico descrivendo i blocchi di funzionamento del sistema esistente e lo studio del circuito di condizionamento.
- **Capitolo 3:** il terzo capitolo riguarda la programmazione software e firmware delle parti che compongono il circuito di condizionamento e l'implementazione con programma CAD del suo schema elettrico e del layout.
- **Capitolo 4:** il quarto capitolo espone i vari test effettuati a verifica delle specifiche ed i risultati ottenuti.
- Conclusioni e possibili sviluppi futuri.

CAPITOLO 1
Il sistema di misura

1. Introduzione

Il progetto di tesi complementa un altro progetto in via di sviluppo all'Università di Bologna riguardante un sistema di misure reologiche in grado di caratterizzare qualitativamente, attraverso l'analisi di parametri elettrici, la composizione di alimenti, in generale liquidi. Al giorno d'oggi, il consumatore presta una crescente attenzione ai processi di lavorazione delle materie prime e alla loro qualità finale percepita. Per questi motivi l'applicazione della reologia in ambito alimentare riveste sicuramente un ruolo di primaria importanza, poiché permette il controllo pre e post industriale delle variazioni delle specifiche dei prodotti. Attraverso l'analisi reologica è possibile caratterizzarne le proprietà in modo da distinguere prodotti soddisfacenti da altri che non rispettano i requisiti necessari per la loro produzione e messa in commercio. Il progetto esposto nell'elaborato è uno strumento del tutto generale che può trovare applicazione in sistemi di analisi nel dominio delle frequenze come, a puro titolo di esempio, quello descritto in [13].

2. Processo di misurazione

Il sistema di misura reologica è stato progettato con il compito di eseguire un'analisi spettrale su campioni, generalmente di liquido, in modo da caratterizzarne la risposta in frequenza a seconda dei loro parametri chimico-fisici.

Il procedimento di caratterizzazione del liquido è il seguente: inizialmente viene effettuata una misura di taratura, utilizzando l'aria o l'acqua come materiali di calibrazione. In seguito viene compiuta l'analisi sul liquido in esame e le due diverse misure scaturite dalle due misurazioni successive vengono sottratte. Così facendo, si ha una serie di misurazioni di diversi campioni tutte riconducibili a uno stesso riferimento, in modo da poter riconoscerne i diversi comportamenti in frequenza e confrontarli tra di loro.

CAPITOLO 2
Schema a blocchi

Il sistema di misura da cui deriva questo progetto sfrutta le proprietà elettriche degli alimenti, in particolare liquidi, per caratterizzarne il comportamento in frequenza, con lo scopo di ricavare informazioni sulla loro composizione e sulle loro proprietà d'interesse in base ai risultati delle misure. In Figura 1: schema a blocchi del sistema iniziale viene esposto in maniera concettuale lo schema a blocchi di interesse del sistema di misura. Questo schema comprende i blocchi di maggior interesse del sistema stesso.

É possibile suddividere lo schema iniziale in 3 blocchi principali:

1. Il blocco corrispondente al sensore;
2. Il blocco di gestione dei segnali, elaborazione degli stessi e conversione;
3. Il blocco di comando e lettura tramite PC.

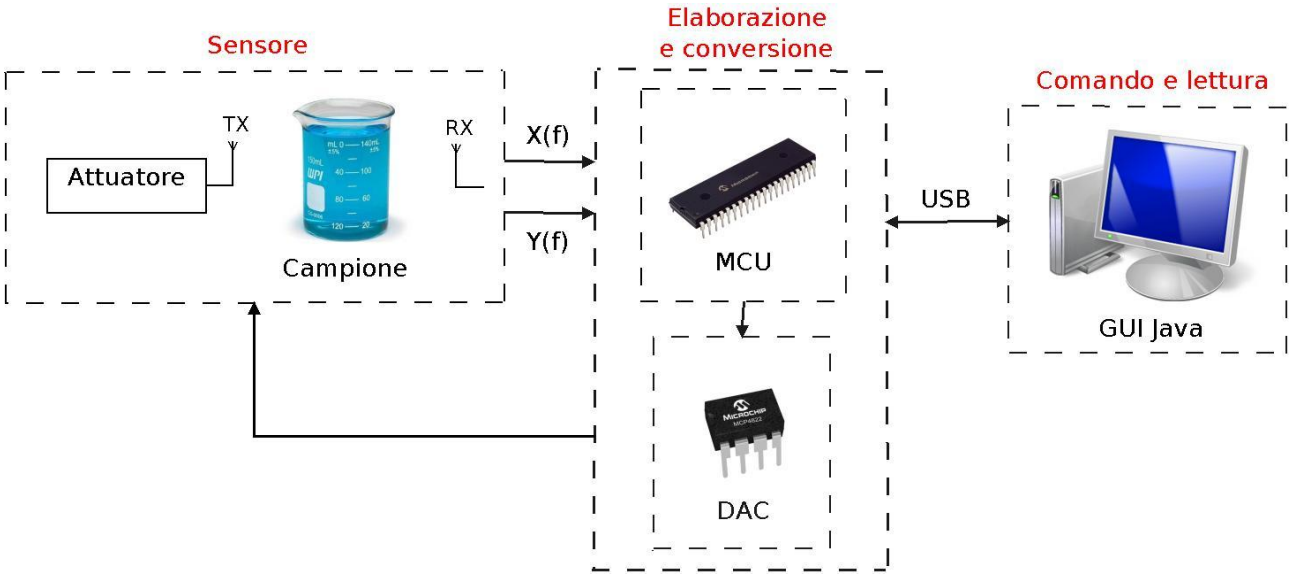

Figura 1: schema a blocchi del sistema iniziale [1] [2] [3]

Nel seguito, verranno esposti i singoli blocchi che compongono lo schema.

1. Sensore

Questo blocco circuitale è costituito da un trasmettitore che impone uno stimolo sinusoidale al campione, e da un ricevitore, che ne misura la risposta. Entrambi sono collocati all’interno

dell'involucro dello strumento. A valle di questi elementi, è posto un generatore di frequenza che riceve un segnale a rampa dal blocco di conversione al fine di generare uno sweep di frequenze dipendenti dalla tensione del segnale ricevuto. Il sensore, sottoposto a stimoli a diverse frequenze, produce la coppia di segnali X(f) ed Y(f) che contengono l'informazione della misura compiuta.

2. Elaborazione e conversione

Il blocco di elaborazione dei vari segnali è costituito da un microcontrollore che dialoga con un convertitore D/A. Il microcontrollore ha i compiti di produrre il segnale d'avvio del processo di misurazione ed elaborare la coppia di segnali X(f) ed Y(f) prodotti dal sensore. Esso dialoga con il blocco di controllo e lettura dal quale riceve i comandi ed al quale invia i valori misurati per la loro visualizzazione. Il convertitore DAC riceve il segnale digitale a rampa prodotto dal microcontrollore e lo converte in una rampa analogica della durata di 30 secondi che invia al generatore di frequenza.

3. Comando e lettura

Il blocco finale di comando e lettura del sistema permette di gestire tramite PC l'intero processo di misurazione. Esso comunica via USB con il microcontrollore, al quale invia i comandi, e dal quale riceve i dati delle misure. In pratica, queste funzionalità sono rese disponibili da un'interfaccia grafica (GUI) realizzata durante il precedente periodo di tirocinio curriculare. Per mezzo di essa è possibile avviare ogni singola misura e salvare su file di testo i valori della coppia di segnali misurati dallo strumento. L'interfaccia è in grado di leggere i valori misurati durante la fase di taratura e compiere la differenza tra le due misure. Le rispettive differenze prodotte vengono mostrate su due grafici visualizzati al suo interno. La GUI di partenza, come si può notare dalla Figura 2, era inizialmente costituita dalle parti di comando, di salvataggio e caricamento delle misure e dai rispettivi grafici.

Nella parte superiore dell'interfaccia grafica si possono notare i pulsanti che avviano le due diverse misure, quella di taratura ("ZERO Measure") e la seconda che è l'effettiva misura del campione ("SWEEP Measure"). Sotto questi pulsanti sono state introdotte due coppie di pulsanti che hanno il compito di salvare ogni singola misura e poterla successivamente caricare in modo da avere diversi riferimenti.

A destra dei pulsanti citati è stata aggiunta una barra di caricamento, che avverte l'utilizzatore dello stato del processo di misurazione. Nella parte inferiore della GUI sono stati realizzati i pulsanti per

il salvataggio/caricamento dei valori delle misure dei grafici visualizzati e, sotto, i grafici stessi dei relativi segnali. L'ascissa di ogni grafico rappresenta la frequenza, che viene misurata in GHz.

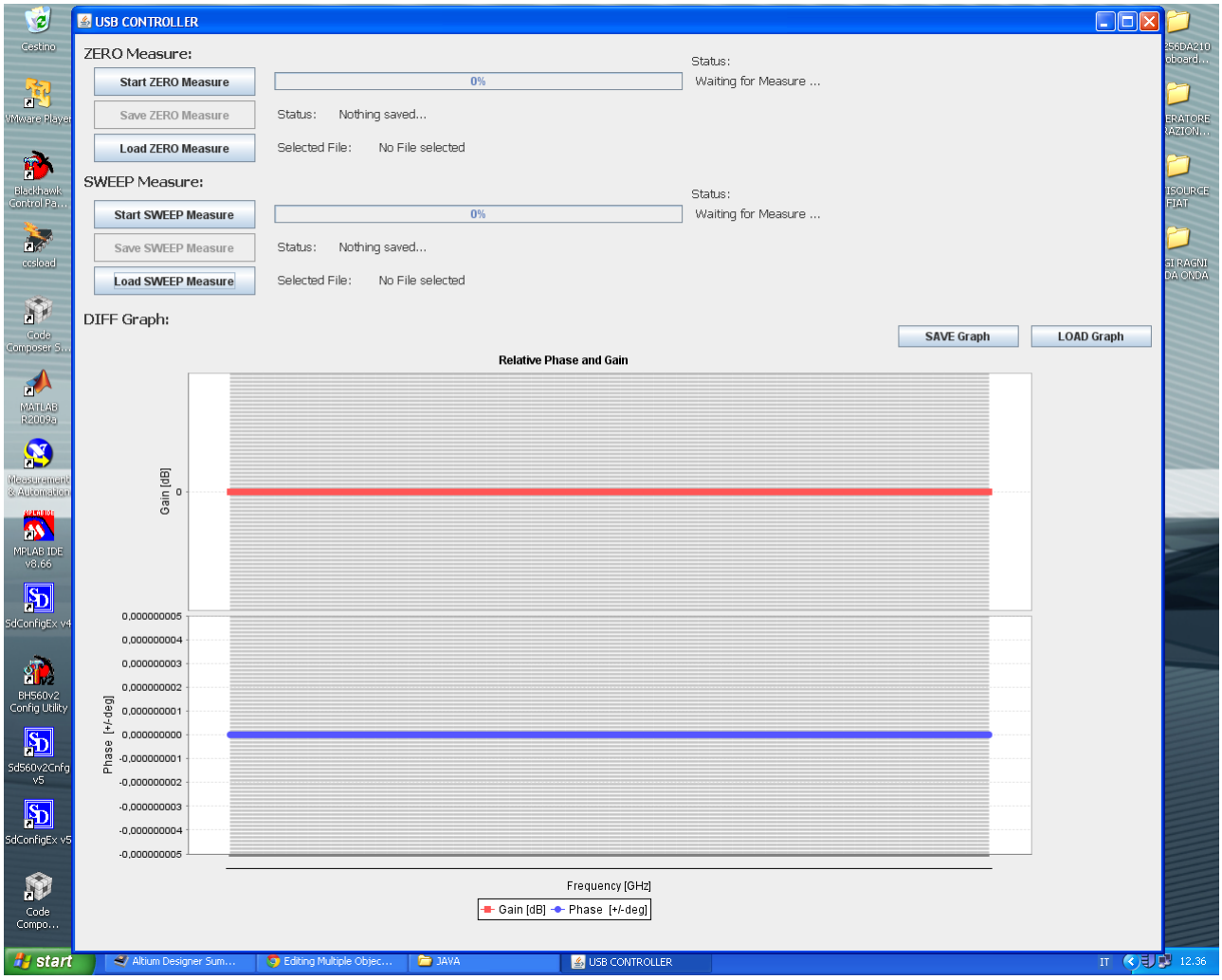

Figura 2: GUI Java di partenza

Il circuito di condizionamento

1. Introduzione

Richiamando il concetto esposto nell'introduzione principale al lavoro di tesi, l'obiettivo di questo progetto consiste nell'espandere i grafici contenuti nella GUI di comando e lettura aumentando la dinamica del segnale affinché la loro interpretazione risulti maggiormente comprensibile da chi ne usufruisce.

In altri termini, l'operazione che tratta di espandere il grafico si può ricondurre a quella di amplificare i valori delle grandezze che lo compongono.

Così si è subito pensato di costruire un circuito di condizionamento del segnale derivante dalla guida d'onda che potesse amplificarlo all'interno del range che interessa espandere.

2. Studio del progetto

Per soddisfare l'esigenza di espandere i grafici si è pensato ad un controllo da realizzare all'interno della GUI che permetta di selezionare una coppia di valori in ordinata, il valore minimo e massimo del range desiderato, per poi amplificarne i valori compresi tra questi due, così da coprire l'intera escursione in ordinata di ogni grafico.

Come è possibile riconoscere in Figura 3, i valori del grafico che non rientrano nel range da espandere saranno tagliati dai valori massimo e minimo che il grafico può fornire; essi sono al di fuori del range, per cui sono inutili al fine dell'espansione ed il fatto di tagliarne i valori non implica una grossa perdita di informazione d'interesse.

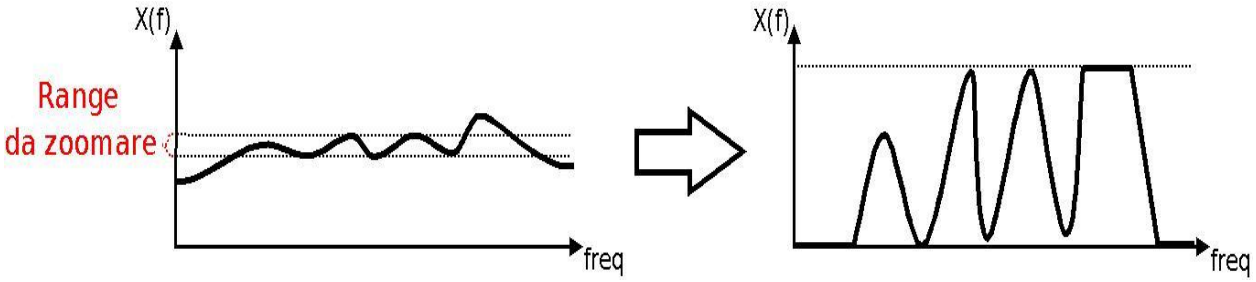

Figura 3: esempio di zoom del grafico

2.1. Il circuito amplificatore

Da subito si è studiato il comportamento atteso dal circuito di condizionamento: se il suo obiettivo è quello di amplificare un segnale e ricondurlo all'intero range disponibile, allora il suo compito principale sarà svolto da un circuito amplificatore. Il circuito amplificatore scelto è l'amplificatore operazionale, per il suo elevato guadagno (idealmente infinito) e la possibilità di compiere molteplici operazioni tra i segnali che gli vengono dati in pasto.

In termini di grandezze elettriche, il circuito amplificatore deve avere due compiti fondamentali:

- Impostare una tensione di offset corrispondente al valore minimo del range da espandere;
- amplificare il segnale all'interno del range in modo da coprire l'intera escursione di tensione disponibile dall'alimentazione del sistema.

Dati i suoi compiti, si è studiato il possibile utilizzo di un amplificatore operazionale in configurazione differenziale.

2.1.1. Amplificatore operazionale differenziale convenzionale

Data la sua semplicità costruttiva e di studio, è stato il primo circuito amplificatore differenziale preso in considerazione e se ne è studiata l'effettiva possibilità di utilizzo in base alle specifiche richieste.

Viene utilizzato con lo scopo di compiere la differenza tra due tensioni, ciascuna moltiplicata per una costante data dal rapporto tra i rispettivi resistori.

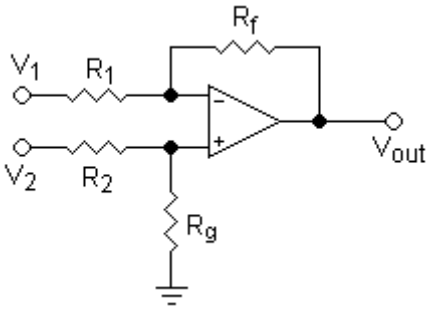

Figura 4: amplificatore operazionale in configurazione differenziale (con un operazionale) [4]

Quest`ultimo ha una caratteristica $Vin - Vout$ di questo tipo:

$$Vout = V2 \ \frac{Rf + R1 \ Rg}{Rg + R2 \ R1} \ - V1 \ \frac{Rf}{R1}$$

che si riduce, quando $R1 = R2$ e $Rf = Rg$, a:

$$Vout = \ V2 - V1 \ \frac{Rf}{R1}$$

e ulteriormente, se $Rf = R1$ e $Rg = R2$ (includendo quando $R1 = R2 = Rf = Rg$):

$$Vout = V2 - V1$$

Così, si è studiata la dipendenza tra la funzione di amplificazione e quella di differenza, andando a capire se fosse possibile utilizzare questa configurazione nel caso di interesse.

Partendo dall'equazione caratteristica riportata sopra e riscrivendola in modo da far comparire i rapporti tra le resistenze, ottengo:

$$V_{out} = V_2 \quad \frac{1 + \frac{R_f}{R_1}}{1 + \frac{R_2}{R_g}} \quad - V_{DD} \frac{R_f}{R_1}$$

Nell'equazione sopra, la tensione V_1 è stata sostituita con V_{DD}. Ora si sono sostituiti i rapporti tra le resistenze $R_f \ \ R_1$ e $R_2 \ \ R_g$ con i termini *x* e *y*, rispettivamente. Così facendo, *x* diventa il fattore per impostare la tensione di offset desiderata e il quoziente $(1 + x) \ \ (1 + y)$ rappresenta, invece, il guadagno di amplificazione e verrà chiamato *G* per semplificare i calcoli.

Studio del dominio

Il principio seguito per lo studio dei rapporti *x* ed *y* è il seguente: avendo come specifica di progetto una *VDD* pari a 5V, è stata fatta l'ipotesi di applicare all'ingresso positivo un segnale di tipo sinusoidale con valor medio V_o e valore di picco ΔV del tipo:

$$v \ t \ = V_O + \Delta \mathrm{V} \sin(\omega t)$$

Ipotizzando $v \ t_0 \ = V_O$, dovrei ritrovare in uscita $V_{out} \ t_0 \ = V_{DD} \ 2$, perciò scrivo:

$$GV_O - V_{DD} x = V_{DD} \ 2$$

Da cui ricavo:

$$x = \frac{GV_O}{V_{DD}} - \frac{1}{2}$$

L'obiettivo finale sarà quello di ricavare i termini *x* ed *y* in funzione delle tensioni V_O e ΔV, in modo da impostarli selezionando direttamente ampiezza e offset della sinusoide di studio in ingresso. Se l'obiettivo è quello di costruire in uscita un segnale che copre l'intero range di alimentazione, allora posso scrivere:

$$2\Delta V \cdot G = V_{DD}$$

Per cui:

$$G = \frac{V_{DD}}{2\Delta V}$$

Ora sostituisco nella formula di *x* :

$$x = \frac{V_O - \Delta V}{2\Delta V}$$

A questo punto devo trovare *y* sostituendo *x* nella formula di *G*:

$$y = \frac{V_O + \Delta V}{V_{DD}} - 1$$

Al fine di trovare il dominio per cui entrambi i valori *x* e *y* sono positivi o uguali a 0 si è utilizzato il software Excel.

Studio del dominio del primo circuito tramite Excel

Nel foglio di calcolo sono state costruite tre tabelle: in tutte è stato imposto in ascissa la grandezza *ΔV* corrispondente al valore di picco del segnale in ingresso e in ordinata la grandezza *V0* corrispondente alla tensione di offset.

I valori di ascissa sono stati presi con passo pari a 0.5V, mentre in ordinata il passo è di 0.25V in modo da avere più campioni da studiare. I valori coprono quasi l'intera escursione di tensione del circuito.

Sono state costruite tre diverse tabelle di calcolo: nella prima sono stati rappresentati i valori del rapporto R_f/R_1 , mentre nella seconda quelli del rapporto R_g/R_2. La terza tabella rappresenta il dominio di funzionamento del circuito, vale a dire i casi in cui entrambi i rapporti esistono e sono positivi e quindi fisicamente realizzabili. Per la corretta lettura del grafico sottostante, il valore 0 (blu scuro) rappresenta la non esistenza del dominio mentre il valore 1 (color azzurro) rappresenta l'esistenza del dominio. Qui di seguito è disegnato il grafico del dominio di funzionamento del primo circuito.

Conclusioni sul primo circuito

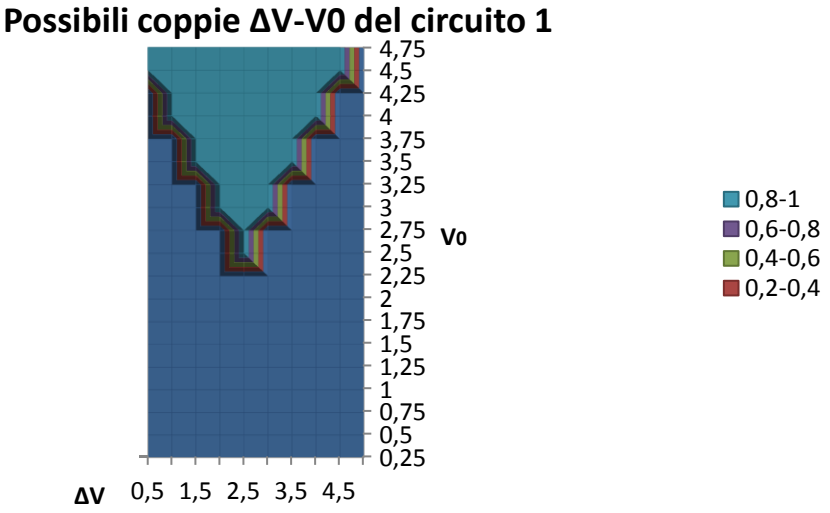

Figura 5: dominio del circuito differenziale convenzionale

Quello che si nota dal grafico in Figura 5 è che il dominio esiste solo per valori di V_0 superiori a 2.5V perciò, utilizzando questo tipo di circuito, non è possibile impostare un offset superiore a questo valore. Questa è una grossa limitazione al funzionamento richiesto, perché si desidera poter amplificare anche segnali che possiedono un offset inferiore a 2.5V. Purtroppo questo circuito non riesce a soddisfare ai requisiti di funzionamento e quindi si è passati allo studio di un nuovo circuito amplificatore operazionale differenziale con l'intenzione di trovare un dominio di funzionamento che copra l'intera escursione della tensione di offset.

2.1.2. Amplificatore differenziale a tre stadi

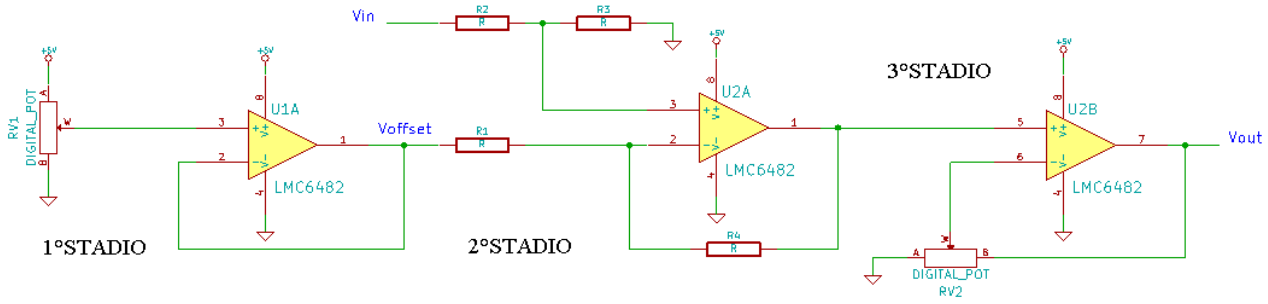

Figura 6: amplificatore differenziale a tre stadi con potenziometri digitali

Questa configurazione è composta da tre stadi distinti:

1. **1°STADIO**: costituisce un partitore di tensione ed è realizzato con un potenziometro digitale, che presenta il cursore collegato all'ingresso non invertente di un operazionale che funge da buffer, in quanto ha l'ingresso invertente collegato alla sua stessa uscita. In questo modo è possibile ottenere la tensione che successivamente sarà da togliere al segnale, cioè il valore minimo della tensione del segnale stesso, che chiameremo V_{REF}. Questo valore è realizzabile attraverso la programmazione del potenziometro digitale. Il buffer posto in seguito al potenziometro serve, inoltre, a disaccoppiare le resistenze del potenziometro digitale con quelle del secondo stadio, impedendo una qualsiasi somma di esse.
2. **2°STADIO**: è rappresentato dall'amplificatore operazionale differenziale "classico" sopra citato, con la particolarità che tutte le resistenze di cui è composto sono di ugual valore. Così facendo, la tensione di uscita è la differenza esatta delle due tensioni d'ingresso. Agli ingressi dell'amplificatore differenziale "classico" sono posti, rispettivamente, nell'ingresso non invertente il segnale *Vin* e in quello non invertente la tensione V_{REF}, proveniente dal primo stadio, da sottrarre a *Vin*. All'uscita di questo stadio ci sarà la tensione V_{DIFF} che ha l'andamento della V_{IN} a cui viene sottratto il valore minimo, quindi traslata verso il basso di V_{REF}.

3. **3°STADIO**: è composto da un amplificatore operazionale in configurazione non invertente che presenta l'ingresso non invertente collegato all'uscita del secondo stadio e l'ingresso invertente collegato al cursore del secondo potenziometro digitale: quest'ultimo viene utilizzato, ancora una volta, in configurazione di partitore di tensione tra GND, l'ingresso invertente e l'uscita dell'operazionale. Programmandone il valore, si decide il valore di guadagno di amplificazione del segnale V_{DIFF}. L'amplificatore non invertente diventa, così, a guadagno variabile. Come è possibile notare dalla Figura 7, il valore delle resistenze R_A ed R_B del secondo potenziometro digitale sono in funzione del dato D_n che sarà scritto al suo interno.

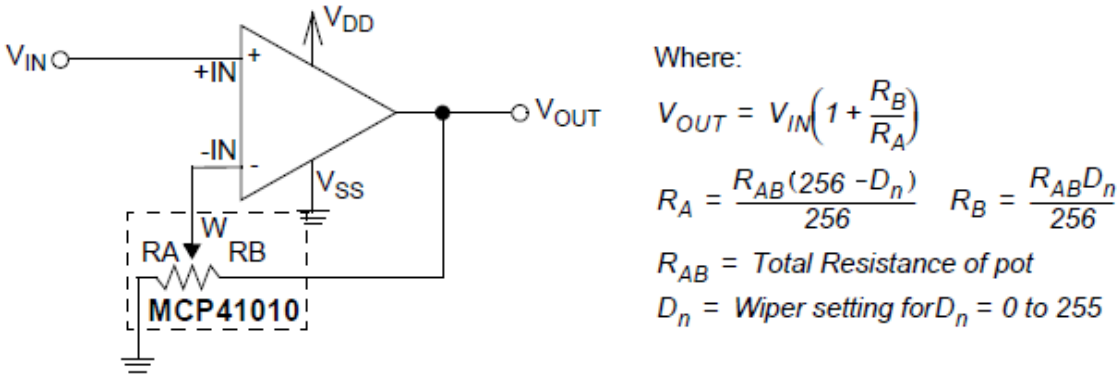

Figura 7: schema dell'amplificatore non invertente programmabile [5]

Studio del dominio del secondo circuito tramite Excel

Per lo studio del dominio si è utilizzato lo stesso procedimento del primo circuito, passando direttamente allo studio sul foglio di calcolo Excel.

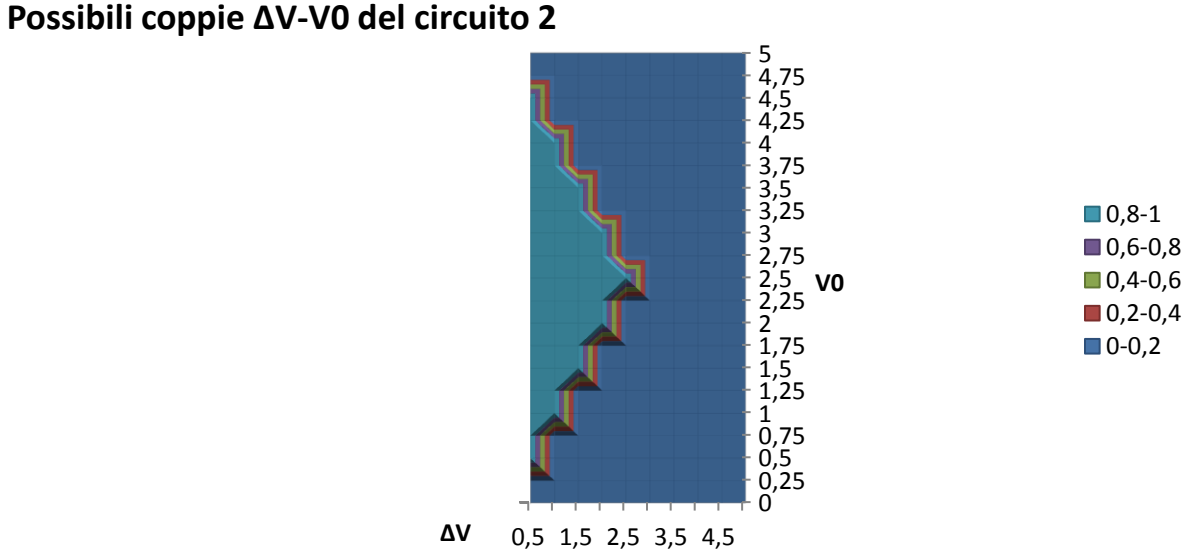

Figura 8: dominio del secondo circuito differenziale

Conclusioni sul secondo circuito

In questo caso si può notare dal grafico in Figura 8 che il circuito ha un corretto funzionamento per l'intera escursione della tensione *V0*, il che significa che esso può ricevere un'onda che presenta un offset qualunque (all'interno della tensione di alimentazione) e un valore di picco limitato da una parte da 0V e dall'altra dai 5V. Se si potesse utilizzare un passo minore si noterebbe dal grafico che la linea seghettata di confine diverrebbe, al limite, una linea continua.

Dai risultati ottenuti si è deciso di utilizzare questo tipo di circuito per compiere la differenza amplificata.

2.2. Il potenziometro digitale

Il fatto di dover avere diversi riferimenti di offset e diversi guadagni ha portato alla scelta di utilizzare dei potenziometri. Per compiere queste funzionalità è stato scelto di utilizzare dei potenziometri resistivi digitali. Questi ultimi si presentano sotto forma di circuiti integrati e, al contrario dei potenziometri analogici settabili attraverso lo spostamento manuale del cursore, sono programmabili digitalmente attraverso un microcontrollore. Questa importante caratteristica dei potenziometri digitali è necessaria per compiere, attraverso la GUI di comando e lettura, le funzionalità richieste dal progetto.
Nel progetto sono stati utilizzati dei potenziometri digitali Microchip MCP41010.

2.2.1. Descrizione generale

Essi sono dispositivi ad un canale, hanno un valore di resistenza nominale di 10 KΩ e risoluzione di 8 bit (256 possibili valori di resistenza e passo di circa 39 Ω). Lavorano con una tensione di alimentazione nominale di 5V e, come i potenziometri analogici, forniscono una tensione proporzionale alla tensione d'ingresso a seconda della posizione del cursore interno. Essi sono costruiti sotto forma di circuiti integrati di tipo PDIP (Plastic Dual In-Line Package) e dispongono dei seguenti pin:

1. *CS* (Chip Select):* attivo basso, abilita o disabilita l'esecuzione di un nuovo comando dopo che esso viene caricato nello shift register;
2. *SCK (Serial Clock):* è il pin che riceve il clock e fornisce il nuovo dato al pin SI sul fronte di salita del clock;
3. *SI (Serial Data Input):* è la porta seriale SPI di ingresso che fornisce il dato allo shift register;

4. *Vss*: pin di massa GND;
5. *PA0*: terminale A del potenziometro;
6. *PW0*: terminale corrispondente al cursore (*wiper*) del potenziometro;
7. *PB0*:terminale B del potenziometro;
8. V_{DD}: pin di alimentazione.

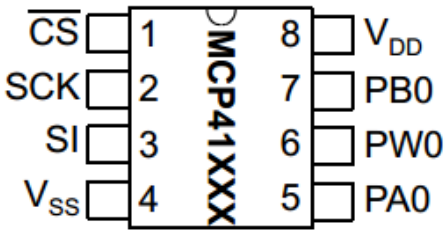

Figura 9: pinout MCP41010 [5]

La posizione del cursore varia linearmente e sono programmabili attraverso un'interfaccia che comunica con un microcontrollore attraverso lo standard di comunicazione SPI. Il consumo di potenza in modalità statica è inferiore ad 1µA e la tensione di alimentazione può variare da 2.7 a 5.5V. I potenziometri presentano una resistenza di cursore chiamata "*wiper resistance*". Questa resistenza, seppur piccola, è da considerare in fase di programmazione perché varia il valore sulla resistenza totale letta sul cursore. Il grafico sottostante in Figura 10 mostra che, in un campione di 400 circuiti potenziometri digitali, al codice 0x00 (che significherebbe 0 Ω) la wiper resistance ha un valore nominale di 53Ω. Nel caso d'interesse, avendo un potenziometro che misura 10kΩ di valore nominale e 256 possibili valori di resistenza, il minimo valore associato al codice 0x01 misura 10kΩ / 256 ≈ 39Ω. Questo valore è inferiore alla wiper resistance, perciò essa aumenta di poco più di un punto il codice attribuibile al valore di resistenza misurato. Questo accorgimento viene spiegato dalla Figura 11, dove D_n è il dato inviato e R_w la resistenza di wiper.

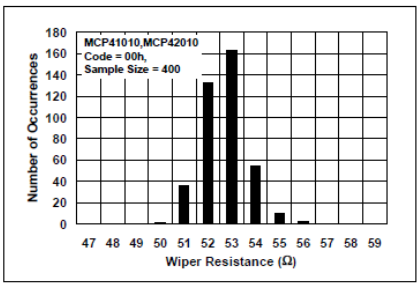

Figura 10: istogramma della wiper resistance [5]

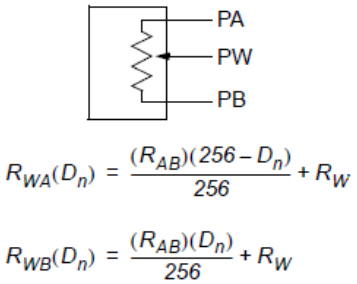

Figura 11: influenza della wiper resistance sulla resistenza misurata [5]

2.2.2. Schema a blocchi

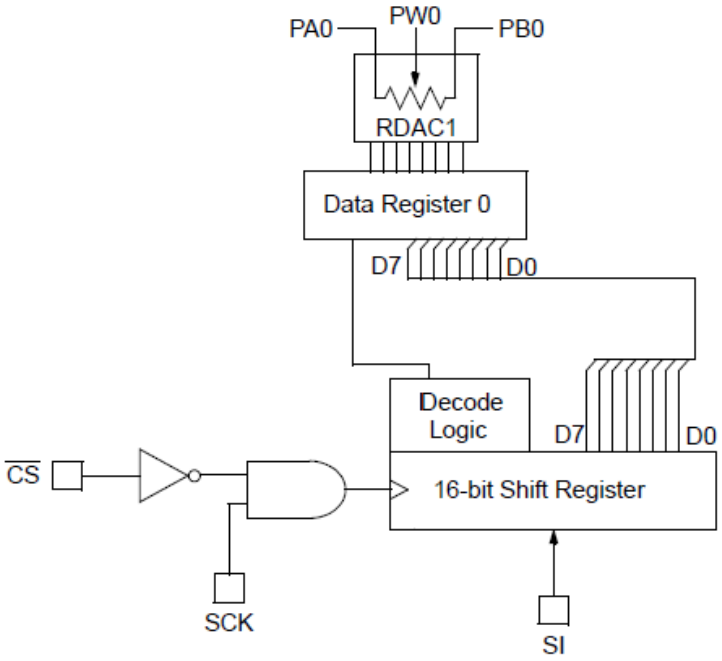

Figura 12: schema a blocchi MCP41010 [5]

Come si nota dalla precedente Figura 12, l'interfaccia SPI fornisce il dato sulla linea SI allo Shift Register a 16 bit, durante il fronte di discesa del segnale SCK, che è processato solo nel caso in cui il CS* del dispositivo assuma il valore logico 0. Successivamente, il dato viene fornito al Data Register a 8 bit che varia la resistenza del cursore in base alla lettura svolta.

2.3. Microcontrollore PIC18F4550

Il sistema di misura originale utilizza un microcontrollore Microchip PIC18F4550.

L'esigenza di dover comunicare col PC per ricevere e leggere i campioni di misura effettuati ha portato alla scelta di questo microcontrollore, dato che dispone di un modulo per la comunicazione USB che utilizza lo standard 2.0. Il microcontrollore prevede il collegamento di un oscillatore al quarzo da 12 MHz che gli fornisce il corretto segnale di clock. Un'ulteriore importante caratteristica del PIC sfruttata nel progetto è la presenza di un modulo che opera per mezzo del protocollo di comunicazione seriale SPI, utilizzato per comunicare con il DAC ed i potenziometri digitali.

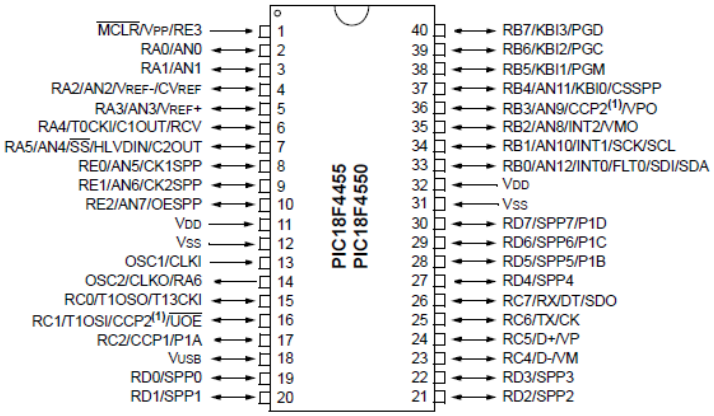

Figura 13: pinout del microcontrollore Microchip PIC18F4550 [6]

2.3.1. Modulo di comunicazione SPI

Descrizione generale

Il protocollo *SPI (Serial Peripheral Interface)* fornisce una comunicazione full-duplex seriale sincrona per lo scambio di dati, in genere, tra un microcontrollore ed una o più periferiche che dispongono dell`interfaccia *SPI*.

La comunicazione avviene tra un componente detto *Master* e uno o più componenti detti *Slaves.* In generale, il *Master* è il microcontrollore mentre gli *Slaves* sono le periferiche. Nella comunicazione, il microcontrollore utilizza un modulo hardware chiamato *SSP (Synchronous Serial Port)* o *MSSP(Master Synchronous Serial Port).*

Linee e segnali di comunicazione

Le linee che costituiscono la comunicazione sono:

- *SCK (Serial Clock)*: è la linea che fornisce il clock trasmesso dal *Master* alle periferiche *Slaves* per sincronizzare la comunicazione;
- *MOSI (Master Out Slave In)*: è l`uscita del *Master* e l`ingresso degli *Slaves*;
- *MISO (Master In Slave Out)*: è l`ingresso del *Master* e l`uscita degli *Slaves*;
- *SS (Slave Select)*: è la linea di selezione dei vari *Slaves.*

Le linee *SCK, MISO* e *MOSI* sono condivise dai vari *Slaves*, mentre le linee *SS* sono diverse per ogni periferica e limitate dal numero di chip select.

I segnali che costituiscono la comunicazione sono:

- *SCK (Serial Clock)*: rappresenta il clock;
- *SDI (Serial Data Input)*: è il segnale ricevuto dall`interfaccia;

- *SDO (Serial Data Output)*:è il segnale spedito dall`interfaccia;
- *SS* (Slave Select*, attivo basso): è il segnale utile al *Master* per selezionare lo *Slave* corrispondente .

Nella comunicazione *SPI* il dato cambia durante il fronte di salita o di discesa di *SCK*; in questo modo viene sincronizzato con quest`ultimo. Di conseguenza la lettura del dato avviene sul fronte opposto di *SCK*.

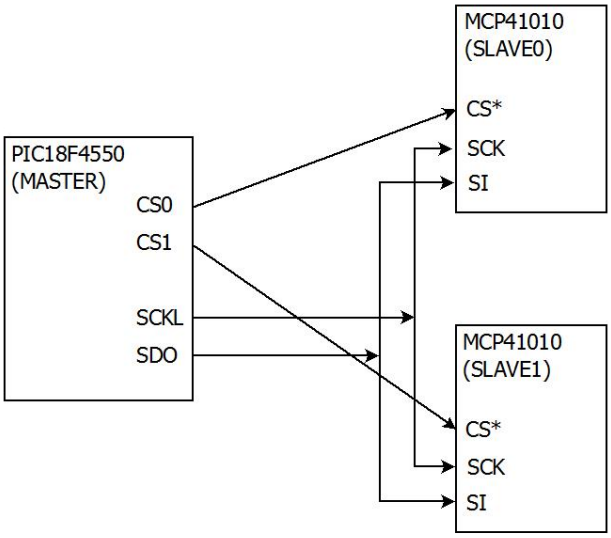

Figura 14: linee SPI tra PIC18F4550 e MCP41010

Impostazione dei registri del PIC

Per il corretto utilizzo del protocollo SPI da parte del PIC è necessario impostare i registri correttamente: il primo è il registro a 8 bit *SSPCON*.

Questo registro contiene i seguenti bit:

- <SSPOV>: "Synchronous Serial Port OVerflow" è settato quando c'è un errore di overflow;
- <SSPEN>: "Synchronous Serial Port Enable" abilita il modulo SSP;
- <CKP>: "ClocKPolarity" controlla la polarità del clock (quando CKP = 1 lo stato IDLE del clock sarà lo stato high, quando CKP = 0 quello low);
- <SSPM3 : SSPM0>: controllano se il modulo SSP è in una modalità SPI e aiutano a configurare la frequenza di clock nella modalità Master.

Un altro registro che controlla la comunicazione SPI è il registro *SSPSTAT* (*"Synchronous Serial Port STATus"*) che contiene 3 bits per il controllo SPI:

- *<SMP>: "data SaMPle timing"* controlla il timing del campionamento e deve essere mantenuto alto per ogni PIC che funge da *slave*. Quando il PIC è in *Master Mode* questo bit

controlla se il campionamento viene effettuato nel mezzo o alla fine della trasmissione del dato.

- *<CKE>: "ClocK Edge select"* questo bit controlla se il dato viene trasmesso sul fronte di salita o su quello di discesa del clock. L'unione tra *CKP* e *CKE* serve per controllare in quale modalità SPI opera il trasferimento: le modalità sono chiamate 0,1,2 e 3. Ogni modalità è descritta dalla coppia di bit CKP e CKE: *0,0, 0,1, 1,0 e 1,1* sono le configurazioni rispettive delle quattro modalità, dove il primo bit indica il bit *CKP* ed il secondo il bit *CKE*. Nel caso di studio la modalità SPI scelta è la modalità 1 (il dato è trasmesso sul fronte di discesa del clock che passa dallo stato alto allo stato *IDLE*).
- *<BF>: "Buffer Full"* quando il PIC attiva questo bit significa che il buffer *SSPBUF*, contenente i dati ricevuti via SPI, non è stato ancora letto. In entrambe le configurazioni *Master* o *Slave,* i dati devono essere sempre letti prima che nuovi dati vengano scritti o ricevuti. Se succedesse il contrario ci sarebbe un errore di *Overflow* e si attiverebbe il bit *SSPOV* che obbligherebbe al *reset* forzato del modulo *SSP* per mezzo del bit *SSPEN*.

2.3.2. Comunicazione SPI tra PIC e potenziometro digitale

La comunicazione tra i due componenti è possibile grazie all'interfaccia SPI e si può dividere in tre diverse fasi:

1. Il PIC seleziona il potenziometro al quale inviare il dato resettando la linea di CS* relativa per un tempo multiplo di 16 T_{CK} (se il CS* torna a livello logico alto prima della fine della scrittura del dato il processo rimane incompiuto);
2. Sulla linea dedicata viene prima trasmesso il *byte di comando* che contiene due bit di comando e due bit per la selezione del canale. I bit di comando determinano quale comando debba essere eseguito tra il comando di scrittura sul potenziometro ed il comando NOP ("No operation"). In questo caso i bit sono rispettivamente 0,1, coppia che rappresenta il comando di scrittura sul potenziometro. I bit di selezione sono trascurabili nel caso di studio perché i potenziometri usati utilizzano ad un unico canale.
3. Successivamente al byte di comando viene scritto il byte corrispondente al dato. Entrambi i byte vengono trasmessi nel dispositivo su ogni fronte di salita del clock.

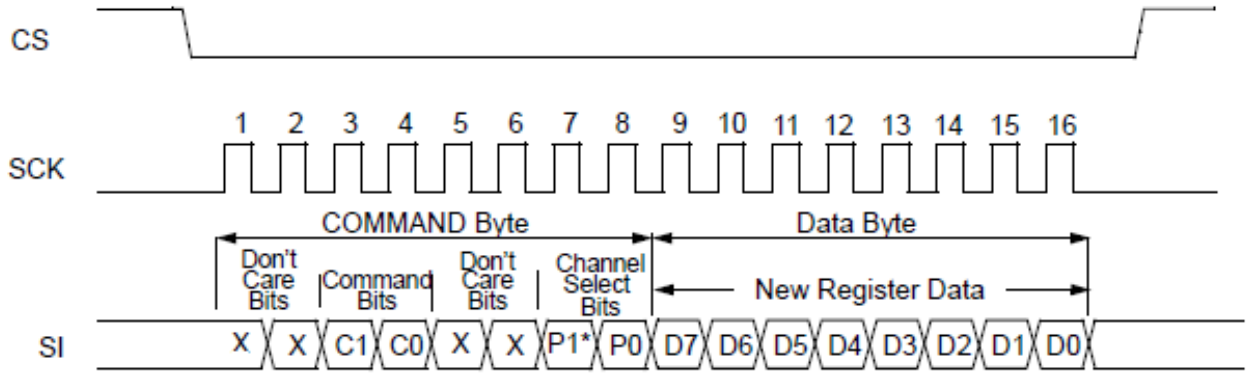

Figura 15: diagramma temporale per la scrittura del potenziometro digitale [5]

2.4. Convertitore D/A

Il convertitore D/A usato nel sistema di misura originario è l'MCP4822.

Non è di interesse per la progettazione, in quanto il suo studio è già stato svolto e non coinvolge il progetto direttamente. Comunque, è bene ricordare la funzione che ha, e cioè quella di convertire il segnale digitale a gradini generato dal microcontrollore che in una rampa analogica che viene fornita al generatore di frequenza. Come il potenziometro digitale, anche il DAC contiene un'interfaccia SPI per la comunicazione con il microcontrollore e quindi presenta i pin CS*, SCK e SDI proprio come il potenziometro.

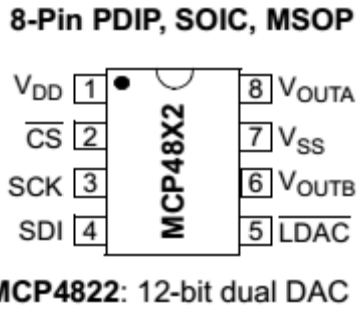

Figura 16: pinout del DAC MCP4822 [7]

CAPITOLO 3
Progettazione del circuito

Prima di comporre l'intero circuito comprensivo di quello di condizionamento del segnale, è stato necessario studiare la programmazione del potenziometro digitale.

1. Programmazione del potenziometro digitale

La programmazione del potenziometro digitale è una delle operazioni principali, volta a far funzionare il circuito di condizionamento sotto le specifiche desiderate dall'utilizzatore.
L'incaricato alla programmazione del potenziometro digitale è il microcontrollore, che deve inviare il dato proveniente dalla porta USB collegata al pc.

1.1. Prova di programmazione da firmware PICmicro

Come prima prova di programmazione, si è creato un file *"SPI_Write.c"* che sarà il firmware di prova con cui programmare il microcontrollore. Di seguito è riportato il listato del file, realizzato con il software MPLAB X IDE e i relativi commenti alle righe di codice che lo compongono.

```
/*
 * File:   SPI_Write.c
 * Author: Manuel Siboni
 * Created on 7 novembre 2014, 17.43
 */

#include <stdio.h>
#include <stdlib.h>
#include <spi.h>
#include "HardwareProfile.h"

void IO_Init(void){
//  set tris bits for serial port pins
    SPI_CLK = 0;    // Output
    SPI_SDI = 1;    // Input
    SPI_SDO = 0;    // Output
}
```

```
void SPI_Init(){
    // SPI Interrupt control
    PIR1bits.SSPIF = 0;    // Clear interrupt flag of PIR1: PERIPHERAL INTERRUPT REQUEST
                           REG 1
    PIE1bits.SSPIE = 0;    // Disable MSSP interrupt of PIE1: PERIPHERAL INTERRUPT ENABLE
                           REG 1
    INTCONbits.PEIE = 1;   // Enable peripheral interrupts
    SSPSTATbits.SMP = 0;   // Input sampled at middle of interval
    SSPSTATbits.CKE = 0;   // Data transmitted from low to high clock
    SSPCON1 = 0b00100010; // Enables serial port and configures SCK, SDO, SDI and SS as
                           serial port pins, Master mode, clock Fosc/64
}

void SPIWrite(unsigned char data){
    unsigned char TempVar;
    TempVar = SSPBUF;      // Clears BF
    PIR1bits.SSPIF = 0;    // Clear interrupt flag
    SSPBUF = data;         // Write byte to SSPBUF register
    while(!PIR1bits.SSPIF) {}; // Wait until bus cycle complete
}

void main(void){
    IO_Init();             // Initialize IO configuration
    SPI_Init();            // Initialize SPI configuration
    CSPOT = 0;             // Enable Pot
    // Sending command data byte
    unsigned char cmd;     // Create command byte
    cmd = 0x11;            // Initialize it to “00010001” (write mode)
    SPIWrite(cmd);         // Write command byte
    // Sending one data byte
    unsigned char data;    // Create data byte
    data=0x34;             // Initialize it to “00110100” (52 means ≈ 2kΩ)
    SPIWrite(data);        // Write data byte
    CSPOT = 1;             // Disable Pot
}
```

Il listato qui sopra comprende gli *#include* delle librerie necessarie e 3 procedure principali:

1. L’inizializzazione dei bit relativi alle porte ed alla modalità di trasmissione del dato.
2. La procedura *SPIWrite* che scrive il dato nel buffer del registro SSPBUF;
3. Il *main* che richiama le procedure di inizializzazione e va a scrivere il byte di comando e il byte del dato al potenziometro.

A questo punto si sono montati i componenti necessari alla prova su breadboard, quali:

- Microcontrollore PIC18F4550 e relativo connettore per PicKit3;
- Potenziometro digitale MCP41010.

Infine, si è programmato il microcontrollore con l'utilizzo del PicKit3; esso è un debugger che permette la programmazione del microcontrollore tramite il software MPLAB X IDE e può fornire la tensione di alimentazione di +5V al circuito. Così facendo, si è misurato l'effettivo valore di resistenza ai capi del potenziometro digitale con un multimetro, e si è scoperto che cambiava valore ogni volta che si andava a variare il dato da scrivere all'interno del suo registro.

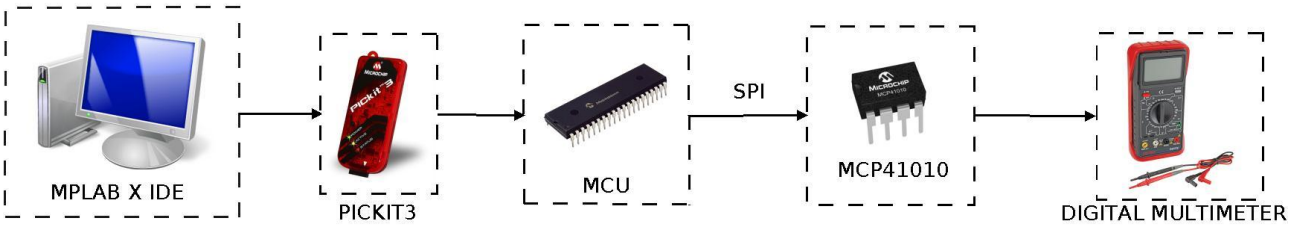

Figura 17: schema a blocchi della prova di programmazione del potenziometro digitale da firmware [3][10][1][2][9]

1.2. Prova di programmazione tramite "PuTTY"

Come seconda prova di scrittura è stato utilizzato l'emulatore di terminale "PuTTY".

PuTTY è un client che utilizza diversi protocolli di rete come SCP, SSH, Telnet ed è in grado di gestire in remoto un sistema informatico grazie ad una funzione di emulatore di terminale su porta seriale COM.

Una volta collegata la breadboard al pc si è proceduto ad instaurare la comunicazione attraverso la relativa porta seriale impostando i vari parametri quali:

- Baud rate : 9600 simboli/s;
- Data bits: 8 bit;
- Stop bits: 1 bit;
- Bit di parità: nessuno.

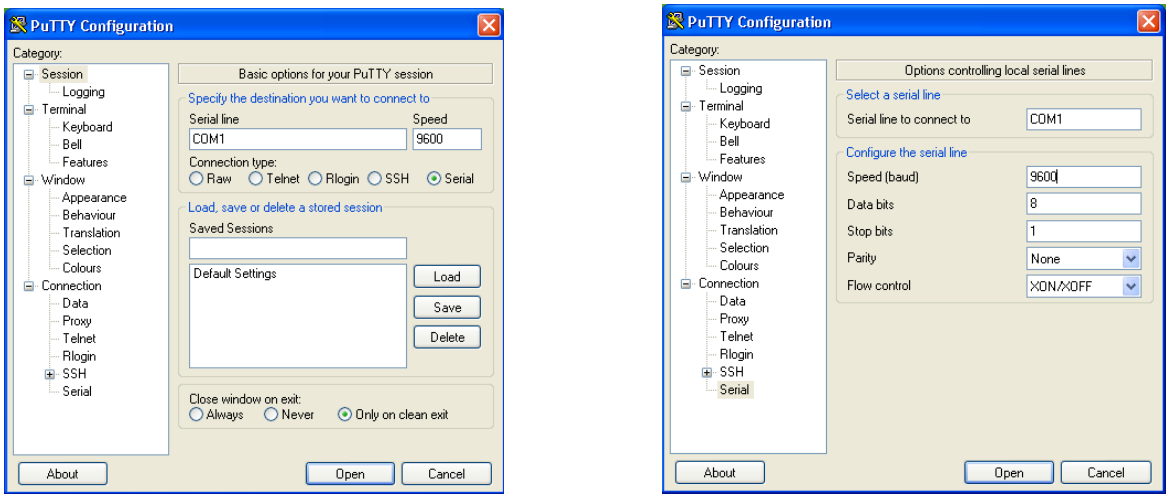

Figura 18: finestre PuTTY di apertura sessione e di configurazione dei parametri

PuTTY apre un collegamento seriale alla porta COM relativa alla breadboard di test. Per la corretta programmazione è stato necessario modificare il firmware del microcontrollore, aggiungendo l'istruzione da inviare al potenziometro riportata, in sintesi, di seguito.

```
case 0x53:      // 0x53 = character "S"                    //  if "S" is pressed
                USB_In_Buffer[0] = USB_Out_Buffer[0];      //  clean out buffer
                CSPOT_OFFSET = 0;
                WriteSPI(0x13);                            //  write command byte
                WriteSPI(0x4D);                            //  0X4D = 77 => 3kohm
                CSPOT_OFFSET = 1;
```

Come si può notare, il set di istruzioni viene fatto partire alla digitazione del carattere "S". Esso comprende in ordine: lo scaricamento del buffer d'uscita in modo da non inviare dati fasulli, l'abilitazione della linea del potenziometro, la scrittura del byte di comando e del data byte e la disabilitazione finale della relativa linea.

A questo punto, come è stato scritto nel firmware del microcontrollore, è stato necessario digitare il carattere "S" da tastiera perché quest'ultimo inviasse il dato al potenziometro. Conoscendone il valore, si è passati alla verifica della scrittura, andando a misurare tramite un multimetro digitale il valore di resistenza ai pin e si è verificato il corretto procedimento.

1.3. Prova di programmazione in linguaggio Java

Finalmente, dopo essere riusciti a capirne il metodo, si è passati a programmare il potenziometro digitale in linguaggio Java. Questa prova è quella quasi definitiva, in quanto la GUI di comando e lettura del sistema è stata scritta proprio in questo linguaggio e quindi è bastato integrare, successivamente, le nuove righe di codice sviluppate all'interno del progetto esistente.

Per quanto riguarda il circuito su breadboard costruito in precedenza, è stato necessario costruire il cavo USB di collegamento tra il pc ed circuito stesso; si è mantenuta la porta USB dal capo del pc e si sono collegati i 4 fili interni alla guaina del cavo ad un connettore a 4 pin che è stato innescato nella breadboard. A questo punto non è stata più necessaria l'alimentazione dal PicKit3, in quanto fornita direttamente dalla porta USB. Come controllo della corretta alimentazione è stato posto un LED in serie ad una resistenza tra i pin +5V e GND del connettore USB.

1.3.1. Prova con JButton

In partenza, si è scelta la modalità di programmazione più semplice per il potenziometro digitale: si è pensato di costruire un pulsante (*JButton*) a lato della GUI che, alla pressione, avrebbe inviato un carattere preimpostato al microcontrollore (il carattere "S", come in precedenza). Quest'ultimo,

all'arrivo del carattere, avrebbe inviato un valore impostato all'interno del suo firmware (corrispondente al dato da processare) al potenziometro, che avrebbe cambiato il valore di resistenza. La novità di questa prova sta nel fatto che è stata introdotta per la prima volta la GUI Java. Essa è stata costruita all'interno di una classe Java che appartiene al progetto comprendente diverse altre classi, tra le quali le classi che scrivono e ricevono tramite la porta seriale. La classe che scrive sulla porta USB è stata modificata inserendo l'istruzione che invia il carattere "S" alla pressione del pulsante *"SendValue"*.

1.3.2. Prova con JButton e JTextField

Nella seconda prova con Java si è sfruttato l'utilizzo di una casella di testo (*JTextField*) dove poter andare a scrivere il dato. Così facendo, si è potuto eliminare il codice del firmware del microcontrollore nel quale veniva scelto il dato da inviare, visto ora la scelta è stata assegnata alla GUI. Per prima cosa si è costruita la casella di testo contenente il dato; quest'ultimo è stato reso disponibile al metodo che scrive sulla seriale così, oltre a scrivere il carattere "S", è stato necessario creare il comando di scrittura del dato prelevandolo dalla casella di testo.
Nella classe che crea la GUI:

```
if(pulsantePremuto == sendData){                    // If JButton sendData is pressed
       String dataString = dataTextField.getText();  // Get text from textfield
       data = Integer.parseDouble(dataString);       // Convert it to Double
...}
```

Nella classe che scrive sulla seriale:

```
data = USBManager.getData();                                  // Get the value of data
this.out.write(0x53);                                         // Write "S"
System.out.println("MANDATO CARATTERE INVIO VALORI");         // Print to screen for check
this.out.write((int)data);                                    // Write data
```

Nel firmware è stata introdotta una parte che, arrivato il carattere "S" dal pc, prende il dato contenuto nel buffer d'uscita, lo mette nel buffer d'ingresso e lo scrive alla porta SPI insieme al byte di comando.

```
case 0x53:      // 0x53 = S                  // If "S" is pressed
       USB_In_Buffer[0] = USB_Out_Buffer[0];  // Buffer out downloading
       putrsUSBUSART("-- ricevuto comando di scrittura su digipot --\r\n");    // Check
       if(mUSBUSARTIsTxTrfReady()){     // If transfer completed and now ready to receive
              CSPOT = 0;                        // Enable digipot chip select
              WriteSPI(cmd);                    // Write command byte
              WriteSPI(USB_Out_Buffer[0]);      // Write data byte
```

```
        CSPOT = 1;                          // Disable digipot chip select
        putrsUSBUSART("-- dato processato nel digipot -- \r\n");          // Check
}
```

Lo schematico del circuito realizzato finora si può osservare in Figura 19.

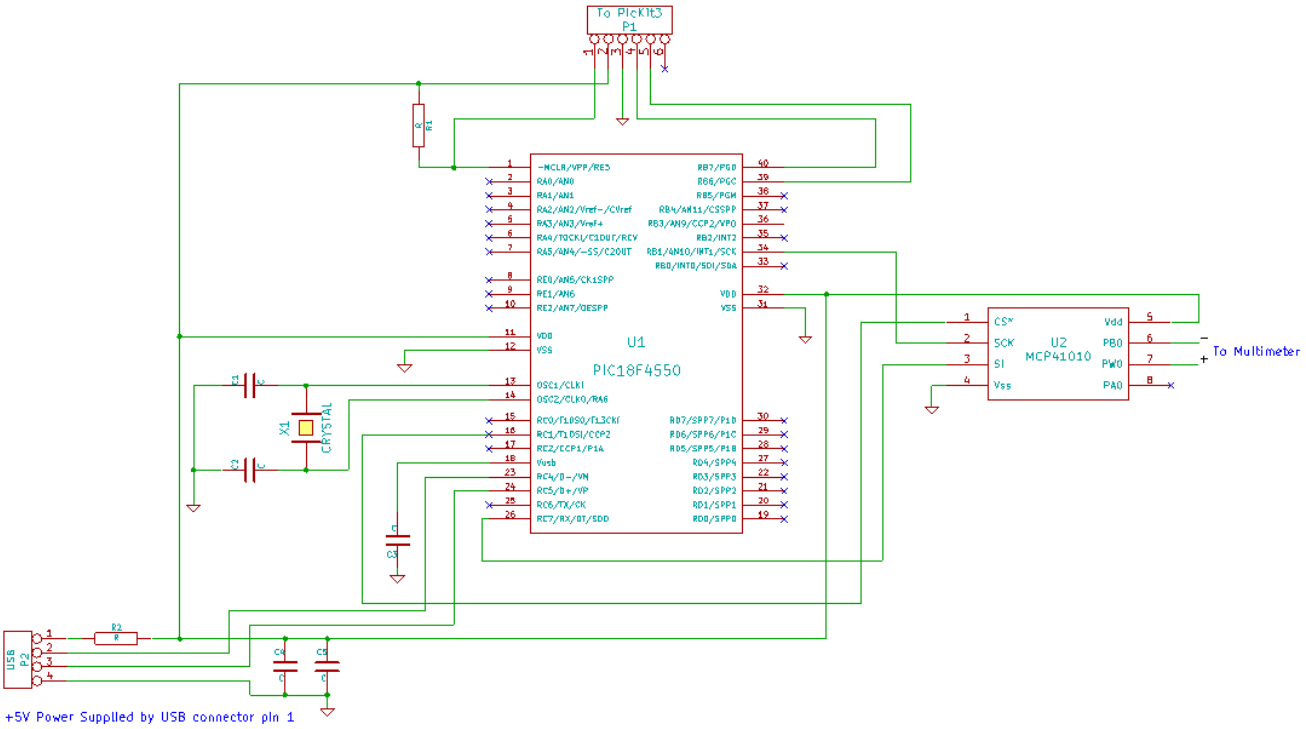

Figura 19: schematico del circuito della prima prova con Java

2. Costruzione su breadboard

Dopo aver constatato l'effettiva programmazione del potenziometro digitale, si è passati all'integrazione dei restanti componenti del circuito completo e alla programmazione finale di ognuno di essi. Per programmare i vari componenti del circuito è stata necessaria la sua costruzione su breadboard. Il DAC ed i due potenziometri digitali utilizzati si presentano sotto forma di circuiti integrati e dato che possiedono entrambi una linea di Chip Select (CS*), in fase di costruzione, è stato necessario collegare tre pin liberi del microcontrollore ai pin di Chip Select degli integrati. Di seguito sono riportate le righe di codice che definiscono le porte di Chip Select dei vari componenti.

```
#define   CSDAC              PORTBbits.RB4   // RB4 is pin 37
#define   LDAC               PORTBbits.RB2   // RB2 is pin 35
#define   CSPOT_OFFSET       PORTCbits.RC1   // RC1 is pin 16
#define   CSPOT_GAIN         PORTCbits.RC2   // RC2 is pin 17
```

Per mantenere un corretto posizionamento sono stati posti, da sinistra verso destra: il connettore USB, il microcontrollore, il DAC, i due potenziometri digitali ed infine gli operazionali e la relativa rete di resistenze. Un'immagine del circuito montato si può esaminare in Figura 20.

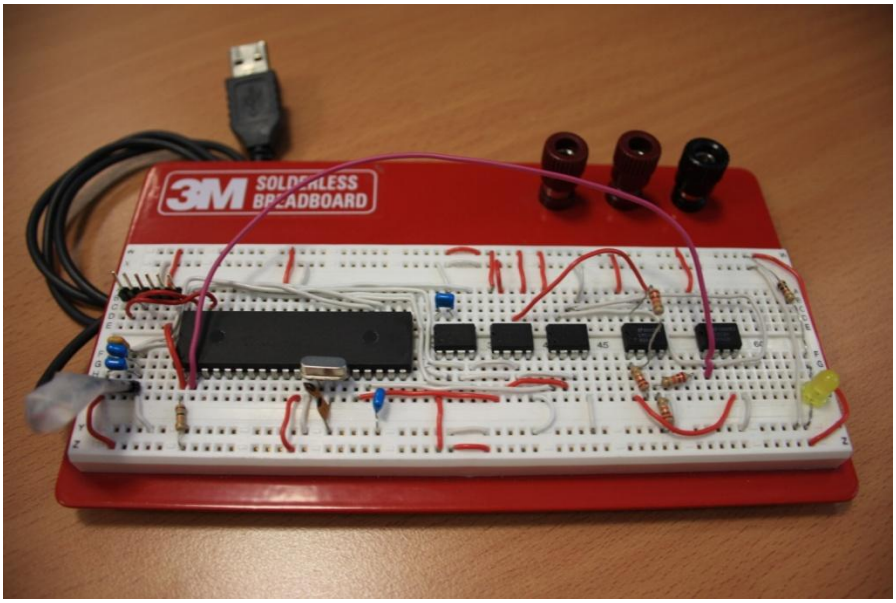

Figura 20: breadboard del circuito completo

3. Programmazione del circuito di condizionamento

3.1. Introduzione

Il circuito di condizionamento prevede l'uso di due potenziometri digitali, perché ognuno possiede un solo canale di resistenza. Per questa ragione la loro programmazione necessita di due data byte: il primo conterrà il valore su cui si imposterà il potenziometro che rileva l'offset (il valore minimo del segnale) che è definito come il primo stadio del circuito di condizionamento del segnale, mentre il secondo conterrà il valore su cui si imposterà il potenziometro che regola il guadagno del terzo stadio, cioè l'amplificatore non invertente a guadagno variabile.

3.2. Calcolo dei valori da scrivere sulla porta USB

I data byte devono variare a seconda dei valori entro la quale si desidera espandere il grafico della GUI, perciò essi devono essere calcolati in funzione dei valori stessi. Questo calcolo è stato affidato al file Java. Come prima cosa, si è aggiunta nella GUI una casella di testo; a questo punto le due caselle sarebbero servite a contenere il valore minimo e massimo del segnale X(f) riportato nel

primo grafico che si desidera zoomare. Il pulsante, questa volta, si sarebbe chiamato “Send Gain Values”. Per il calcolo dei valori sono stati svolti i passaggi elencati di seguito.

Nella classe GUI.Java:

in questa classe sono stati creati tutti i componenti della GUI e i campi necessari al calcolo dei valori. Queste sono le nuove operazioni compiute al suo interno:

1. creazione campi Double statici *gainMIN* e *gainMAX* che contengono i valori minimo e massimo del range di X(f) da zoomare presi da *gainMinString* e *gainMaxString*;

```
String gainMinString = gainMin.getText();
String gainMaxString = gainMax.getText();

gainMin = Double.parseDouble(gainMinString);
gainMax = Double.parseDouble(gainMaxString);
```

2. creazione campi Double statici *voltMIN* e *voltMAX* contenenti i valori di tensione relativi a quelli del segnale X(f). Il relativo valore di gain è stato moltiplicato per la tensione massima raggiungibile dal segnale (cioè la tensione di alimentazione di 5V) e diviso per il valore massimo raggiungibile dal grafico corrispondente al massimo del segnale (34.276);

```
voltMIN = 5*gainMIN/34.276;
voltMAX = 5*gainMAX/34.276;
```

Nella classe USBSerialWriter.Java:

questa classe contiene il codice per la scrittura sulla seriale. Le nuove operazioni compiute sono:

1. creazione campi *offset* e *gain* corrispondenti ai valori da inviare:

```
private static double      offset;
private static double      gain;
```

2. creazione metodi *setOffset()* e *setGain()* che operano sulle variabili *voltMIN* e *voltMAX*, provenienti dalla classe *GUI.Java*, per settare i campi *offset* e *gain*:

```
public static void setOffset(){
       double voltMin = USBManager.getVoltMIN();
       offset = 256*(voltMin/5);
}

public static void setGain(){
       double voltMin = USBManager.getVoltMIN();
       double voltMax = USBManager.getVoltMAX();
       gain = 256-(256*(voltMax-voltMin)/5);
}
```

I valori contenuti nei campi *offset* e *gain* sono proprio quelli da scrivere all’interno dei potenziometri digitali del circuito di condizionamento del segnale.

3.3. Scrittura dei valori sulla porta USB

Dato che in precedenza il collegamento alla seriale era gestito dai thread di lettura e scrittura contenuti nelle classi *USBSerialReader.Java* e *USBSerialWriter.Java* ora, che si vuole solamente scrivere sulla porta, è stato necessario distinguere il caso della misurazione con il caso dell'invio dei valori attraverso delle variabili che cambiano a seconda della procedura selezionata. Così facendo, la classe di scrittura va ad inviare l'esatto carattere alla seriale a seconda dei valori di queste variabili. Una di queste variabili è *transmitValues* che dice se si vuole effettuare la misura o scrivere i dati. Qui di seguito vengono riportate le nuove righe di codice più importanti scritte nella classe *USBSerialWriter.Java*.

```
if(transmitValues == false){                          // if measure occures
            try{
                  this.out.write(0x52);               // send "R"
                  System.out.println("MANDATO CARATTERE INIZIO MISURA");     // check
            }catch ( IOException e ){
                  e.printStackTrace();
                  System.exit(7);
            }
}else{                                    // if digitalpot programming occurs
      if(sendZeros == false){
            setOffset();                  // set offset digitalpot data byte
            setGain();                    // set gain digitalpot data byte

            try{
                  this.out.write(0x53);                     // send "S"
                  System.out.println("MANDATO CARATTERE INVIO VALORI");      // check
                  this.out.write((int)offset);              // send offset data byte
                  this.out.write((int)gain);                // send gain data byte
                  System.out.println(""+offset);            // check
                  System.out.println(""+gain);              // check
            }catch ( IOException e ){
                  e.printStackTrace();
                  System.exit(7);
            }
      }else{
            try{
                  this.out.write(0x53);                     // send "S"
                  System.out.println("pot resettati");      // check
                  this.out.write(0);                        // send 0 to offset digitalpot
                  this.out.write(0);                        // send 0 to gain digitalpot
            }catch ( IOException e ){
                  e.printStackTrace();
                  System.exit(7);
            }
      }
}
```

Reset dei valori dei potenziometri

Per riportare in uscita lo stesso valore d'ingresso, così da non misurare il segnale modificato ma quello originale, si è integrata la GUI con il pulsante *"SendZeros"* che scrive due zeri ai potenziometri. Come è possibile notare dal circuito di condizionamento del segnale che si forma,

riportato in Figura 21, se ai potenziometri vengono applicati due zeri, il circuito riporta in uscita proprio il valore dell'ingresso (a meno di piccole cadute di tensione legate alle resistenze di wiper).

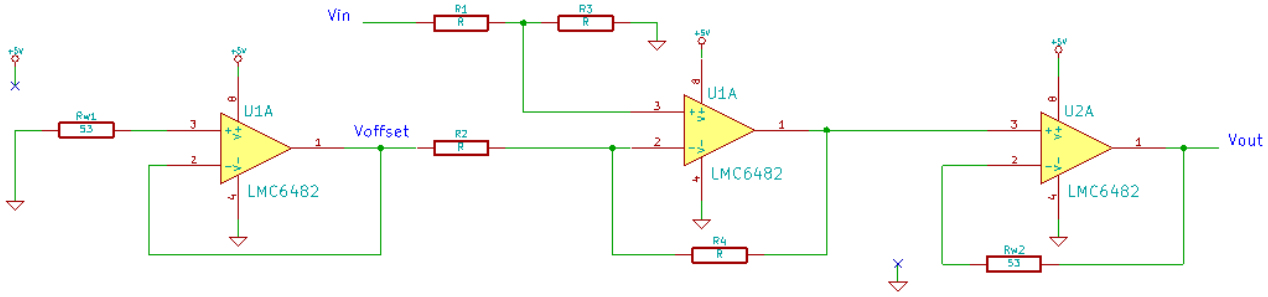

Figura 21: circuito nel caso di reset dei potenziometri digitali

3.4. Ricezione dati da USB e scrittura sui potenziometri

Conclusa la parte di programmazione Java relativa ai dati da inviare sulla porta USB, si è passati a programmare il firmware del microcontrollore affinché fosse in grado di ricevere gli stessi dati e spedirli ai potenziometri correttamente.

Ricezione dei dati da USB

Dato che il microcontrollore si aspetta il carattere "S" per entrare nella routine di invio dei dati ai potenziometri, si è dovuto distinguere l'invio del carattere di inizio misura con l'invio del dato che vale "S" (0x53 = 83 in decimale). La distinzione avviene grazie alle variabili *DigitalOffsetPotCommand* e *DigitalGainPotCommand.* Di seguito la parte del firmware che compie la distinzione.

```
case 0x53:                                      // 0x53 = S
if(DigitalOffsetPotCommand == FALSE && DigitalGainPotCommand == FALSE){
       USB_In_Buffer[0] = USB_Out_Buffer[0];   // buffer out downloading
       putrsUSBUSART("-- ricevuto comando di scrittura su digipot --\r\n");     // check
       DigitalOffsetPotCommand = TRUE;         // offset digitalpot is ready to receive
}
if(DigitalOffsetPotCommand == TRUE && DigitalGainPotCommand == FALSE){
       if(mUSBUSARTIsTxTrfReady()){
              USB_In_Buffer[0] = USB_Out_Buffer[0];
              ProcessDataPOT();
              putrsUSBUSART("-- ricevuto carattere S: dato processato in OffPot-- \r\n");
       }
}else{
       if(mUSBUSARTIsTxTrfReady()){
              USB_In_Buffer[0] = USB_Out_Buffer[0];
              ProcessDataPOT();
```

```
            putrsUSBUSART("-- ricevuto carattere S: dato processato in GainPot-- \r\n");
      }
}
break;
default:
if(DigitalOffsetPotCommand == TRUE && DigitalGainPotCommand == FALSE){
      if(mUSBUSARTIsTxTrfReady()){
            USB_In_Buffer[0] = USB_Out_Buffer[0];
            ProcessDataPOT();
            putrsUSBUSART("-- ricevuto carattere diverso da S: dato processato in OffPot
      -- \r\n");
      }
}
if(DigitalOffsetPotCommand == FALSE && DigitalGainPotCommand == TRUE){
      if(mUSBUSARTIsTxTrfReady()){
            USB_In_Buffer[0] = USB_Out_Buffer[0];
            ProcessDataPOT();
            putrsUSBUSART("-- ricevuto carattere diverso da S : dato processato in
            GainPot -- \r\n");
      }
}
```

Scrittura dei dati ai potenziometri

Di seguito la parte di firmware che scrive il command byte e il data byte sulla linea SPI corretta, distinta per mezzo dell'utilizzo delle stesse variabili *DigitalOffsetPotCommand* e *DigitalGainPotCommand*.

```
void ProcessDataPOT(void){
    if(DigitalOffsetPotCommand == TRUE){      // if data byte is for offset digitalpot
        CSPOT_OFFSET = 0;                     // enable its CS*
        WriteSPI(cmd);                        // write command byte
        WriteSPI(USB_Out_Buffer[0]);          // write data byte
        CSPOT_OFFSET = 1;                     // disable its CS*
        DigitalOffsetPotCommand = FALSE;
        DigitalGainPotCommand = TRUE;
    }else{                                    // if data byte is for gain digitalpot
        CSPOT_GAIN = 0;                       // enable its CS*
        WriteSPI(cmd);                        // write command byte
        WriteSPI(USB_Out_Buffer[0]);          // write data byte
        CSPOT_GAIN = 1;                       // disable its CS*
        DigitalGainPotCommand = FALSE;
    }
}
```

3.5. Costruzione dei nuovi grafici nella GUI

Come parte finale della fase di programmazione, si è modificato il codice Java relativo alla costruzione dei grafici. Queste sono le righe di codice che creano i nuovi valori da andare a disegnare negli assi di X(f) ed Y(f).

```
YGAINMIN = GUI.getGainMin();
YGAINMAX = GUI.getGainMax();
for(int i=0;i<List.size();i++){
       String y = new String (""+((Costants.FSTEP*i)+Costants.FSTART));
       double value = YGAINMIN - (List.get(i)[ch]*((YGAINMIN-YGAINMAX)/34.2));
       chDataset.addValue(value, x1, y);
}
```

Esse ricalcolano i valori da andare a disegnare nei grafici in base al range che si desidera espandere grazie alle variabili *YGAINMIN* e *YGAINMAX*.

4. Schema elettrico e layout del circuito

Di seguito, in Figura 22, è mostrato l'intero layout su PCB del circuito.

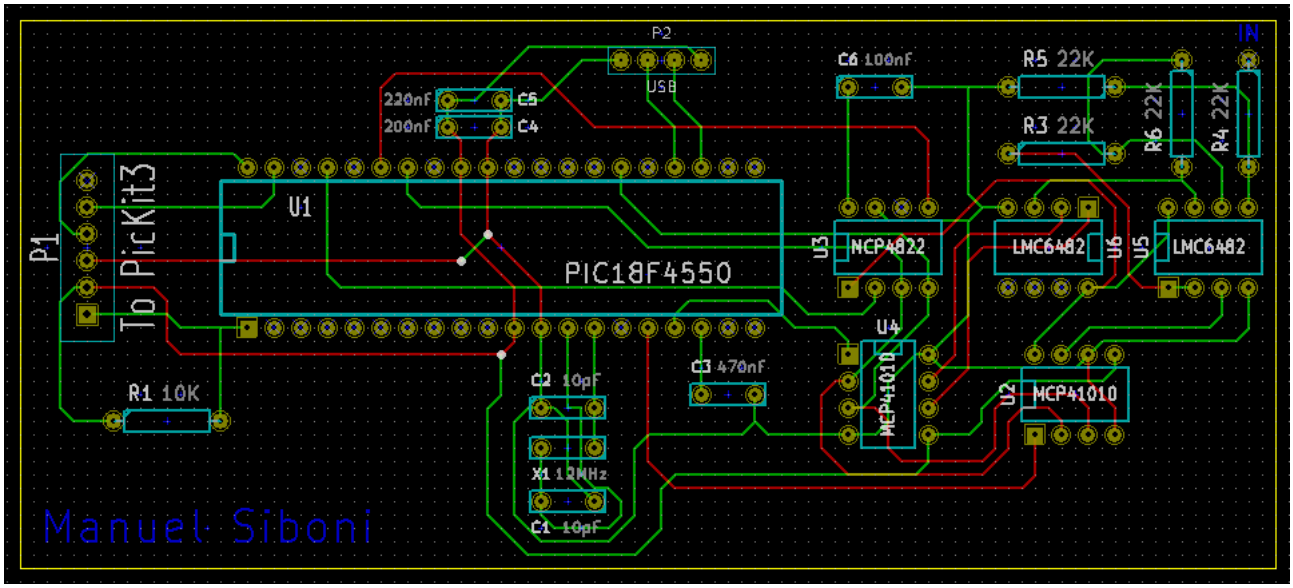

Figura 22: Layout del progetto

Nella Figura 23 sottostante viene mostrato lo schema elettrico dell'intero circuito.

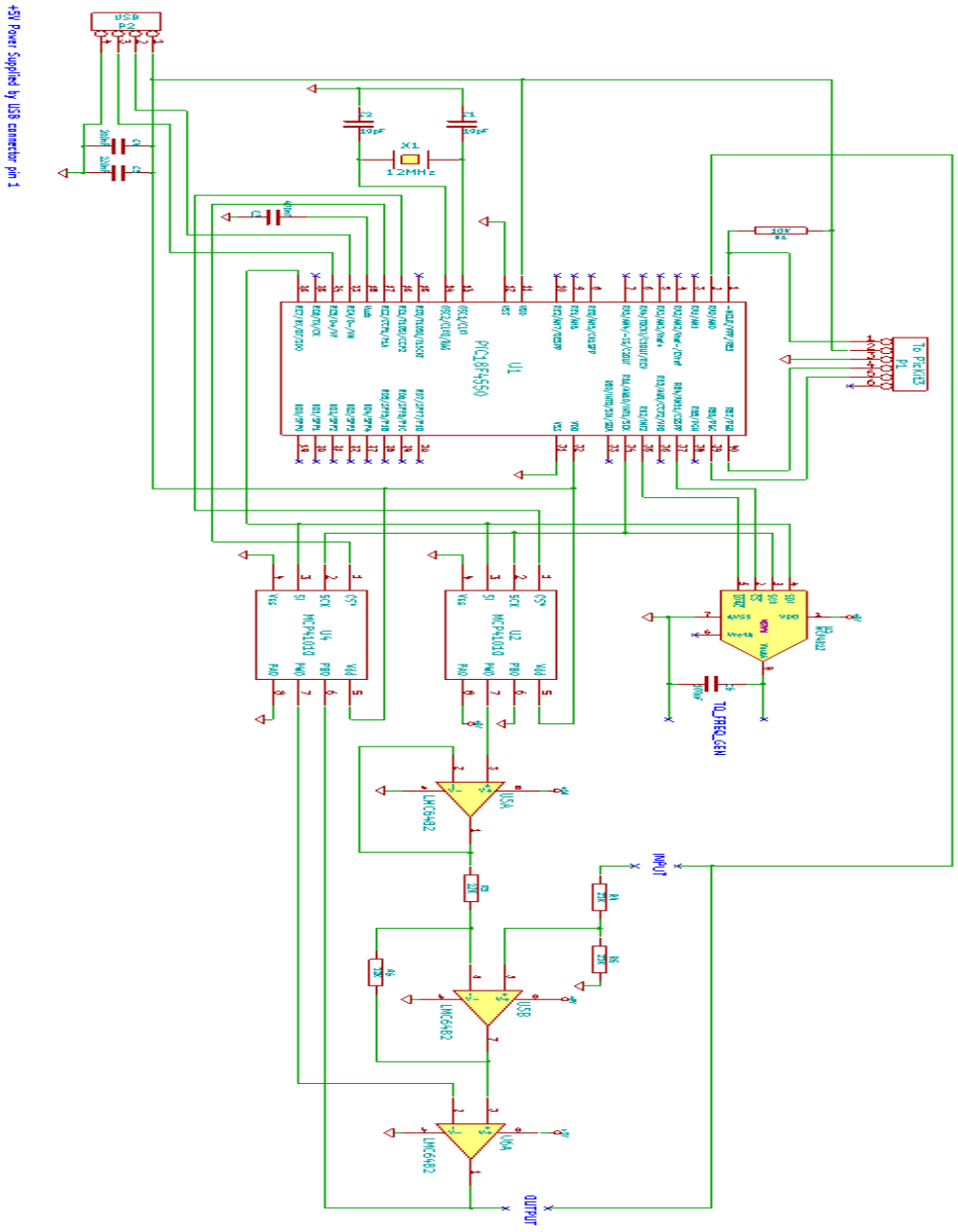

Figura 23: schema elettrico del circuito completo montato nella breadboard

CAPITOLO 4
Test effettuati e risultati ottenuti

Dopo aver programmato il firmware del microcontrollore ed il software Java si è finalmente passati alla fase di test del circuito. Questa è la parte di verifica delle funzioni che si sono volute implementare nel progetto.

1. Strumenti di laboratorio utilizzati

Per testare la breadboard si sono utilizzati i seguenti strumenti elettronici disponibili nel laboratorio:

- Oscilloscopio Tektronix MSO 3014: è stato utilizzato per leggere le forme d'onda all'ingresso e all'uscita del circuito di condizionamento. Grazie ad esso, inoltre, è stato possibile misurare le forme d'onda per avere una lettura più chiara e verificare il corretto procedimento.

Figura 24: oscilloscopio Tektronix MSO 3014 [11]

- Alimentatore stabilizzato: è stato utilizzato per creare un segnale variabile tra 0 e +5V da usare come ingresso del circuito di condizionamento, e quindi simulare il sensore del sistema di misura.

Figura 25: alimentatore stabilizzato da laboratorio [12]

2. Schema a blocchi del test

Lo schema a blocchi del circuito composto per il test è riportato di seguito in Figura 26.

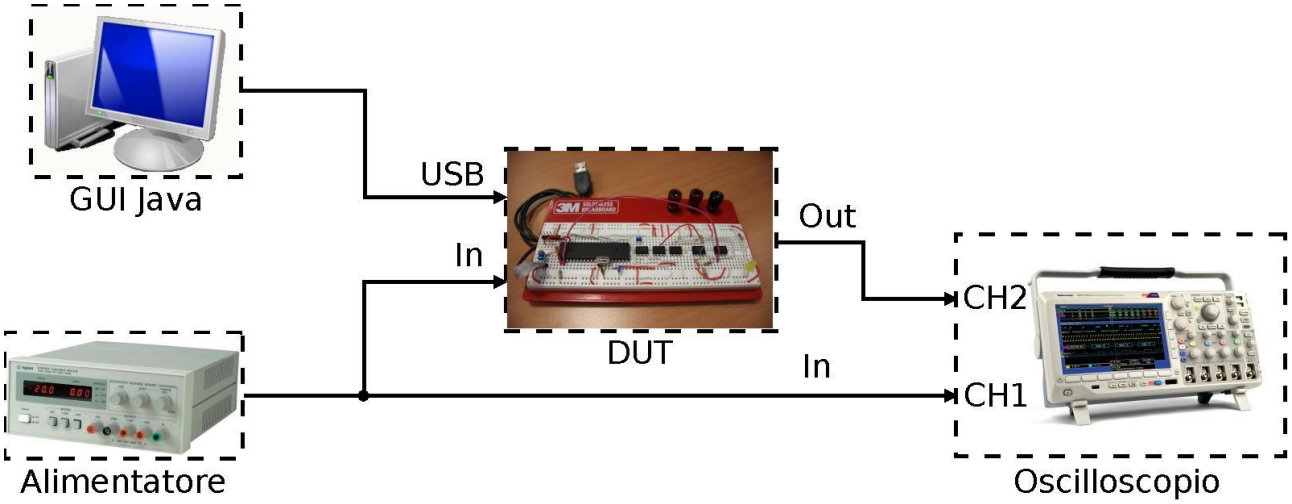

Figura 26: schema a blocchi del circuito composto per il test [3] [11] [12]

L'alimentatore fornisce al DUT *("Device Under Test")*, rappresentato dal circuito della breadboard, una tensione $V_{In}(t)$ attorno ad 1V che, nella parte centrale della fase di misurazione, compie variazioni di pochi mV; queste variazioni sono proprio quelle che si desidera espandere nel grafico.

3. Test di misurazione

Dopo aver fornito al DUT la $V_{In}(t)$, si è fatta partire la misura dalla GUI.

Di seguito vengono riportate le operazione svolte durante la misura:

1. Inizialmente è stata compiuta la taratura dello strumento. Per graficare la stessa forma d'onda $V_{In}(t)$ si è dovuta compiere una prima misura in cui l'ingresso al circuito è stato collegato a GND. Dato che entrambi i grafici compiono la differenza tra le misure di sweep e quelle di zero, si è salvata la misura appena compiuta per il successivo disegno dei grafici.
2. In secondo luogo si è svolta la misura di sweep dello strumento. Il terminale dell'alimentatore è stato, così, collegato all'ingresso del DUT e, prima di iniziare la misura, sono stati resettati entrambi i potenziometri digitali premendo il pulsante *SendZeros* dalla GUI; questa azione è necessaria ogni volta che si vuole misurare direttamente i valori dell'ingresso, senza che il circuito di condizionamento li alteri in alcun modo.
3. Andando a caricare i file di testo .txt relativi alle due misurazioni vengono automaticamente disegnati i grafici. Essi sono riportati in Figura 27. Il grafico di X(f) è disegnato in rosso, mentre Y(f) in blu. Come si può notare dalla figura, il grafico che riproduce X(f) compie

delle variazioni molto piccole tra i valori 6 e 7. Questo range è proprio quello che si desidera amplificare.

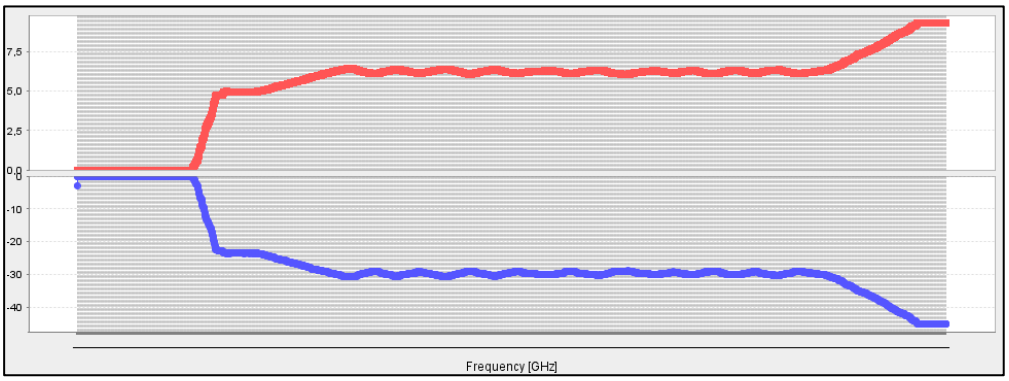

Figura 27: grafici della misurazione pre-zoom

4. Per amplificare il range è necessario immettere il suo valore minimo e massimo all'interno delle caselle di testo della parte della GUI chiamata *"Zoom Measure"* e premere il pulsante "*Send Gain Values"*. A questo punto, i valori del range saranno processati dal software Java che creerà la coppia di dati da inviare alla seriale. Il microcontrollore riceverà questi dati e li scriverà nei potenziometri, andando a modificare il segnale da misurare.

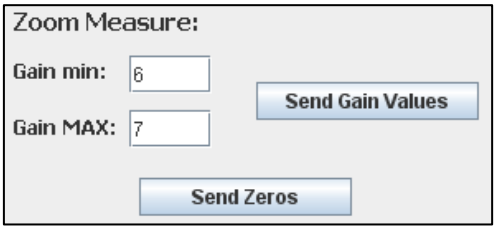

Figura 28: pannello Zoom Measure della GUI

5. Infine si è compiuta una seconda misura mantenendo il più inalterato possibile il segnale $V_{In}(t)$proveniente dall'alimentatore. Come è possibile vedere nella Figura 29, questa volta il grafico mostra le variazioni amplificate di X(f) ed Y(f), che vanno a coprire l'intera escursione possibile.

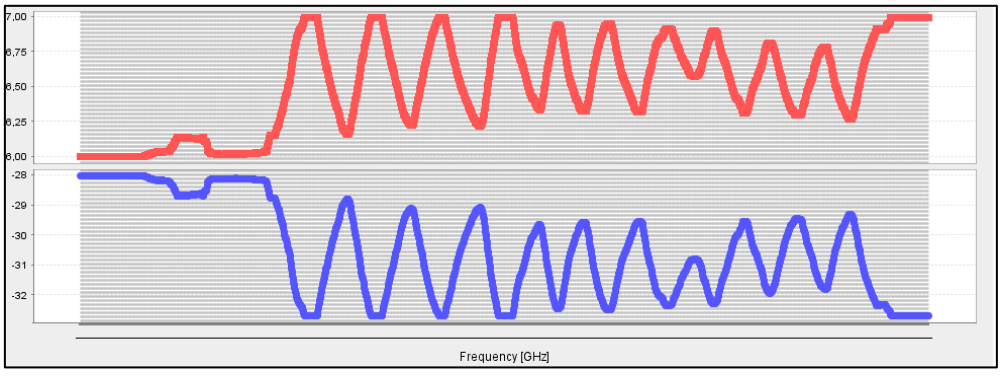

Figura 29: grafici della misurazione post-zoom

4. Forme d’onda da oscilloscopio

Nelle figure sottostanti sono state riportate le forme d’onda coinvolte nella misura, misurate dall’oscilloscopio: in giallo la forma d’onda $V_{IN}(t)$ dell’alimentatore ed in viola l’uscita del circuito di condizionamento del segnale $V_{IN}(t)$. Nello schermo dello strumento ogni riga è posta ad 1V, mentre ogni colonna a 4 secondi: in questo modo si è potuta riportare la misura completa, in quanto essa ha durata di 30 secondi.

4.1. Screenshot pre-zoom

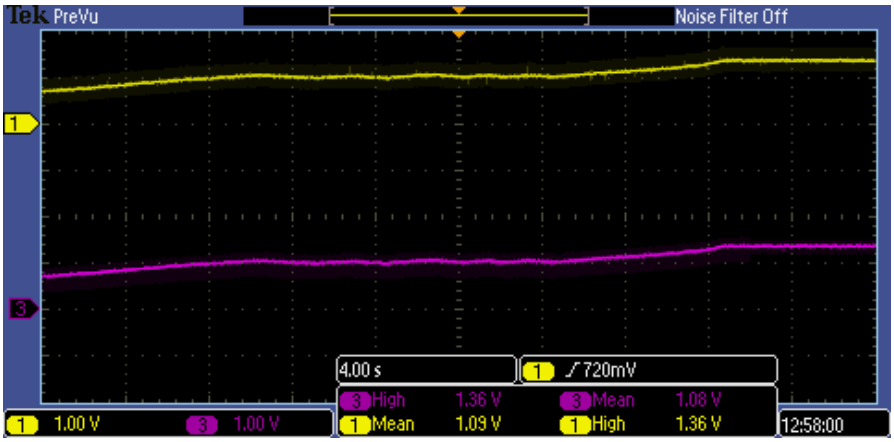

Figura 30: screenshot dall'oscilloscopio delle forme d'onda pre-zoom

Come è possibile notare dal primo screenshot dell’oscilloscopio in Figura 30, antecedente allo zoom, le due forme d’onda sono pressoché identiche ed hanno valore medio pari a circa 1.08V e valore massimo di 1.36V. Questo è stato possibile grazie al reset dalla GUI dei potenziometri digitali.

4.2. Screenshot post-zoom

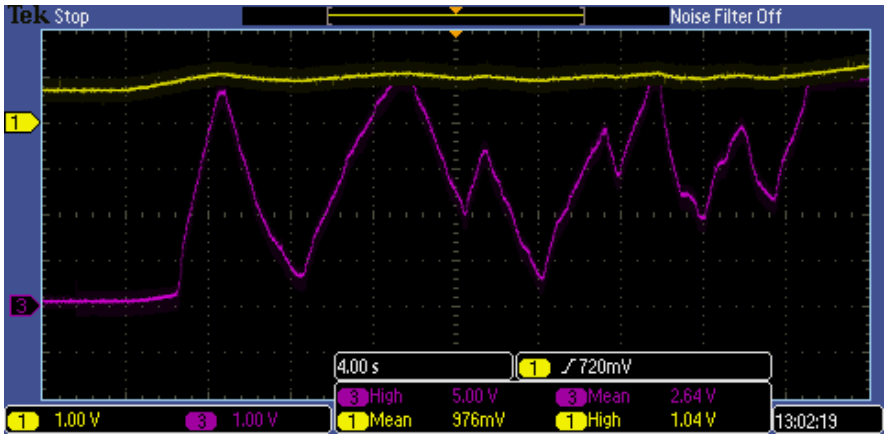

Figura 31: screenshot dall'oscilloscopio delle forme d'onda post-zoom

Nel secondo screenshot in Figura 31 si nota, invece, che l'uscita in viola ha dei picchi in corrispondenza delle variazioni del segnale $V_{IN}(t)$; questo lo si può ulteriormente notare dal riquadro che compie le misure dei valori medi e massimi delle onde. L'ingresso varia, come in precedenza, attorno al valore di 1V mentre l'uscita possiede 2.64V di valore medio e 5V di valore massimo in corrispondenza dei picchi superiori.

4.3. GUI Java finale

In Figura 32 è mostrata la GUI finale, con il pannello *Zoom Measure* integrato in alto a destra.

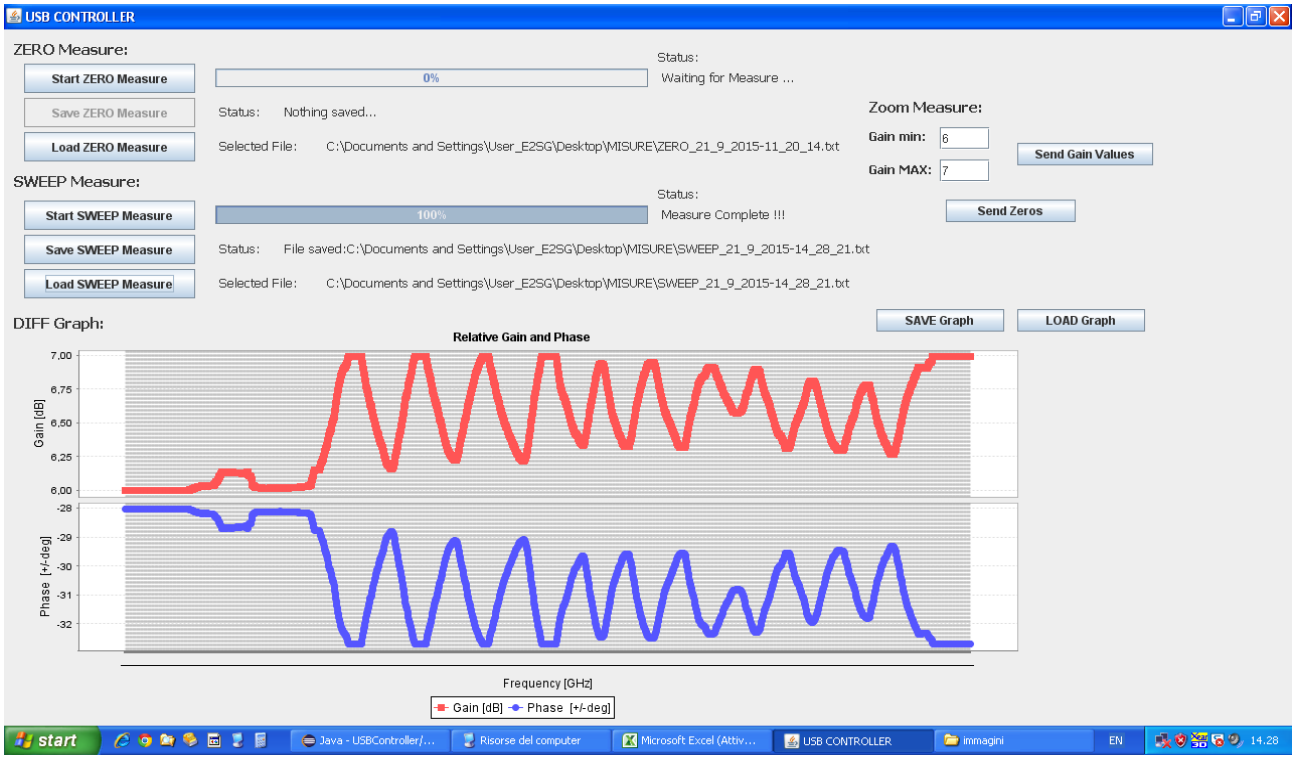

Figura 32: GUI Java finale

5. Risultati ottenuti

Come è possibile notare dai grafici, il circuito di condizionamento permette il loro zoom esatto tramite il controllo Zoom Measure interno alla GUI. É possibile, inoltre, compiere lo zoom più volte per ottenere un'analisi sempre più approfondita delle forme d'onda che compongono i grafici.

I risultati ottenuti sono soddisfacenti, in quanto aiutano l'utilizzatore del sistema di misura a comprendere meglio le piccole variazioni delle forme d'onda che lo strumento rileva.

Conclusioni

Il progetto è stato svolto mediante una serie di attività collegate:

- dapprima si è studiato il sistema di misura reologica ed il circuito di condizionamento del segnale da integrare ad esso che soddisfacesse la richiesta di compiere lo zoom dei grafici;
- successivamente si è passati alla realizzazione dello stesso su breadboard utilizzando i potenziometri digitali, in modo da avere la possibilità di programmare i valori di resistenze e quindi il guadagno di amplificazione e la tensione di offset del circuito;
- in seguito è stata svolta la programmazione dei componenti del circuito che comunicano tra di loro e del software che permette il controllo e la lettura dei dati provenienti dalle misure;
- infine, sono stati effettuati i test di misurazione del sistema a verifica delle specifiche desiderate in fase di studio, traendo risultati soddisfacenti.

Lo schematico del progetto disegnato al software CAD si presta ad essere trasformato in un progetto di layout su scheda PCB per la sua integrazione con il progetto del sistema di misura reologica. Inoltre, come ulteriore possibile sviluppo futuro, si può pensare ad un controllo, magari a cursore direttamente sul grafico, che riesca ad espandere il segnale in una banda di frequenze, piuttosto che in un range di valori dello stesso come in questo progetto, così da poter compiere lo studio di diversi segnali tutti riferiti ad un dominio fisso. Inoltre, il circuito di condizionamento del segnale realizzato è in grado di gestire un solo segnale d'ingresso, mentre il sistema di misura fornisce una coppia di segnali alla sua uscita; perciò un ulteriore sviluppo di esso riguarderebbe la costruzione di un secondo circuito, parallelo a quello progettato, che sia in grado di attuare la stessa procedura di zoom del secondo segnale fornito dal sistema di misura.

Bibliografia

[1] *http://c1b3rh4ck.blogspot.it/2014/01/fuses-for-pic18f4550.html*

[2] *http://www.microchip.com/wwwproducts/Devices.aspx?product=MCP4822*

[3] *http://www.fancyicons.com/free-icon/108/hardware-icon-set/free-computer-icon-png*

[4] *http://www.elemania.altervista.org/amplificatori/operazionale/opamp17.html*

[5] *MCP41010 Datasheet - Microchip Technology*

[6] *PIC18F4550 Datasheet - Microchip Technology*

[7] *http://tahmidmc.blogspot.it/2014/10/pic32-spi-using-mcp4822-12-bit-serial.html*

[8] *http://www.microchip.com/_images/ics/medium-MCP41010-PDIP-8.png*

[9] *http://www.harborfreight.com/ac-dc-digital-multimeter-37772.html*

[10] *http://artofcircuits.com/product/pickit-3-programer-debugger-clone*

[11] *http://www.arttool.ru/catalog/kontrolno-izmeritelnie-pribori/oscillografi/seriya-dpomso3000/mso3014/*

[12] *http://www.conrad.it/ce/it/product/128050/Alimentatore-da-laboratorio-regolabile-Keysight-Technologies-E3630A-0-6-VDC-1-25-A-35-W-3-x*

[13] *Paganelli, R.P., Romani, A., Golfarelli, A., Magi, M., Sangiorgi, E., Tartagni, M. Modeling and characterization of piezoelectric transducers by means of scattering parameters. Part II: Application on colloidal suspension monitoring (2010) Sensors and Actuators, A: Physical, 160 (1-2), pp. 1-8.*

Ringraziamenti

Questo giorno è sicuramente il più intenso, da quando ho iniziato questo percorso di studi ormai 5 anni fa. Non si può dire che la sofferenza non sia mancata, ma ora mi accorgo che grazie ad essa questo traguardo ha un gusto più dolce. Per quanto possa essere scontato dirlo, ho sempre sognato il giorno della laurea e ora mi è possibile viverlo grazie a tutte quelle persone che mi hanno supportato in questo percorso.

Innanzitutto, vorrei ringraziare coloro che mi hanno seguito nel periodo di tesi: il Prof. Aldo Romani ed il Dott. Matteo Filippi. Per merito vostro, sono riuscito a raggiungere il traguardo che ci siamo prefissati nel giorno in cui è nato il progetto della mia tesi e questa è una soddisfazione che mi accompagnerà nel mio percorso di studi e mi sarà di grande aiuto per affrontare il mondo del lavoro. Grazie per la vostra disponibilità e pazienza nei miei confronti. Un ringraziamento va anche agli altri dottorandi dello studio di ricerca di Via Venezia 260.

Non è stato sicuramente facile, a volte, sopportare le lunghe lezioni del corso, ma ad alleggerire le mie giornate da studente ci hanno pensato i miei colleghi e, soprattutto, amici col quale ho condiviso la mia vita dentro e fuori dalla facoltà. Perciò, grazie a Barto, Fede, Moro, Nick, Dom, Cava, Bronz, Manza, Zava, Fabio, Valmo e Andrea, che mi hanno aiutato negli studi e, soprattutto, insegnato che nella vita c'è sempre il tempo per una marrrraffa!

Mi sento, inoltre, di ringraziare tutti i miei amici che mi hanno permesso di staccare, ogni tanto, la spina dallo studio e condividere con loro le mie passioni, soprattutto quella musicale. Sarebbero troppi da elencare, per cui perdonatemi se elencherò solo i nomi dei gruppi musicali. Per questo, ringrazio tutti gli amici degli Alterego, dei Summer Hits e dei Mald'estro; è magnifico trovare il feeling e creare musica con voi. Un grazie sincero anche a tutti i colleghi e amici della Piscina Comunale di Forlì per aver condiviso la passione per il nuoto e avermi permesso di fare esperienza in un bel gruppo.

Il mio ringraziamento più sentito è per la mia famiglia: grazie di cuore a mio fratello Ramòn, mia mamma Patrizia, mio babbo Franco, mio nipote Samu (grazie per farmi rimanere sempre un po' bambino come te!), la mia cognatina Gè e la mia nonna Rina. Oggi mi rendo conto che tutti gli incitamenti che avete speso nei miei confronti sono stati essenziali per coronare il mio desiderio di laurearmi. Questo giorno è sicuramente anche vostro, per cui festeggiamolo insieme.

Infine, ma non per ordine di importanza, vorrei ringraziare la mia ragazza, la Gio. Grazie infinitamente per esserti sorbita ogni mia delusione, per aver festeggiato ogni mio successo e per essere stata sempre al mio fianco. Spero che, come nel mio percorso di studi, mi accompagnerai anche nella vita come tu hai sempre saputo fare e, insieme, condivideremo le avventure più belle.

Se c'è una persona che avrebbe voluto vedermi oggi è la mia nonna Dina, per cui dedico questo giorno a lei.
Oggi, grazie a tutti voi, è il giorno più gratificante della mia vita.