Campionamento con Arduino

L’obiettivo di questo laboratorio è quello di utilizzare la scheda Arduino Uno per campionare un segnale sinusoidale e stimarne le caratteristiche principali.

Quello che noi sappiamo è che Arduino ha un ADC che può essere configurato per lavorare a diverse velocità, con una conseguente riduzione di accuratezza; il nostro fine è proprio quello di verificare questa diminuzione di accuratezza in condizioni dinamiche e l’effetto del multiplexer.

Scriveremo un programma per Arduino in modo che campioni la sinusoide, permettendo all’utente di scegliere il prescaler dell’ADC, del timer, il pin della scheda Arduino ed il numero di campioni. Come vedremo successivamente sarà molto importante trovare il legame tra questi valori per il corretto funzionamento del programma. Successivamente dovremo creare un programma in C# in grado di salvare i dati ricevuti su un file. Come ultima cosa utilizziamo un nostro script in Matlab per elaborare i dati ricevuti.

Per verificare il funzionamento inviamo all’ADC di Arduino una sinusoide che conosciamo, la campioniamo e ne scriviamo un modello matematico che dovrebbe teoricamente spiegare l’andamento dei campioni:

y = A•sin(wt + φ) + C

questo è un modello in cui teniamo in considerazione anche l’offset della sinusoide. --Campionando ottengo una serie di valori distanziati di ∆t.

**DESCRIZIONE ARDUINO**

Il compito di Arduino è quello di campionare i valori di tensione della sinusoide distanti ∆t, in modo da poter poi ricostruire il segnale correttamente. Si potrà variare la velocità di campionamento dell’ADC scrivendo all’interno dei registri di Arduino. Quando scelgo il numero di campioni (almeno 20 per una ricostruzione del segnale abbastanza corretta) è molto importante tener conto della velocità di campionamento dell’ADC settata tramite il prescaler: infatti ogni operazione di campionamento dura per 13 colpi di clock come possiamo leggere dal datasheet del nostro oggetto.

Un’altra cosa fondamentale da considerare è l’intervallo di campionamento che, per avere una stima corretta della sinusoide, deve essere della durata del periodo.

Nella nostra analisi abbiamo preso in considerazione una sinusoide di 100 Hz, con ampiezza di 450 mV, da 0 V a 1 V. In queste condizioni di lavoro abbiamo deciso di prendere 20 campioni, abbiamo settato il prestaler dell’ADC a 128, consentendoci di campionare ad una frequenza pari a 16 MHz/2^128. Oltre a questo, se consideriamo ancora i 13 colpi di clock dovuti al campionamento otteniamo che la durata totale di un campionamento è di circa 104 microsecondi. Settando il prescaler del timer a 256 facciamo in modo che si scateni l’interrupt della soglia (che è quello che fa partire un campionamento nuovo) dopo circa 500 microsecondi. In questo modo, se dovessimo portare al massimo le prestazioni del nostro sistema, potremmo effettuare circa 90 campionamenti in un periodo.

Utilizziamo come tensione di riferimento quella a 1.1 V, per minimizzare l’errore di campionamento.

ELABORAZIONE DEI CAMPIONI

Dopo il campionamento passiamo i valori misurati a Matlab per determinare i parametri del modello (ampiezza e frequenza) e per calcolare la differenza tra i campioni reali e i valori stimati tramite il modello in modo da trovare i residui.

Dal valore di questi residui riusciamo a capire la bontà del nostro campionamento, infatti se sono di piccoli valori vuol dire che è avvenuto in maniera corretta.

??Generalmente minimizziamo la somma delle differenze quadratiche tra i valori misurati e stimati del modello:

Ipotizziamo che tutte le cause di incertezza siano aleatorie e le singole misure siano scollegate; trascuriamo nella nostra misura le incertezze legate al campionamento.

Il modello trigonometrico usato per calcolare i parametri è:

Da quale poi possiamo trovare:

Per trovare questi parametri useremo tre diversi metodi, derivanti da tre considerazioni diverse. In un primo momento, infatti, decidiamo di sfruttare il fatto che il nostro sistema in misura, se si conosce la frequenza, sia un sistema lineare, dunque risolvibile in forma matriciale come combinazione lineare delle misurazioni effettuate. Di seguito è riportato un estratto del file Matlab che implementa questo metodo:

U=zeros(nPoints,3);

U(:,1)=sin(omega\*t1);

U(:,2)=cos(omega\*t1);

U(:,3)=1;

ABC=U\samples'; %ABC=lambda

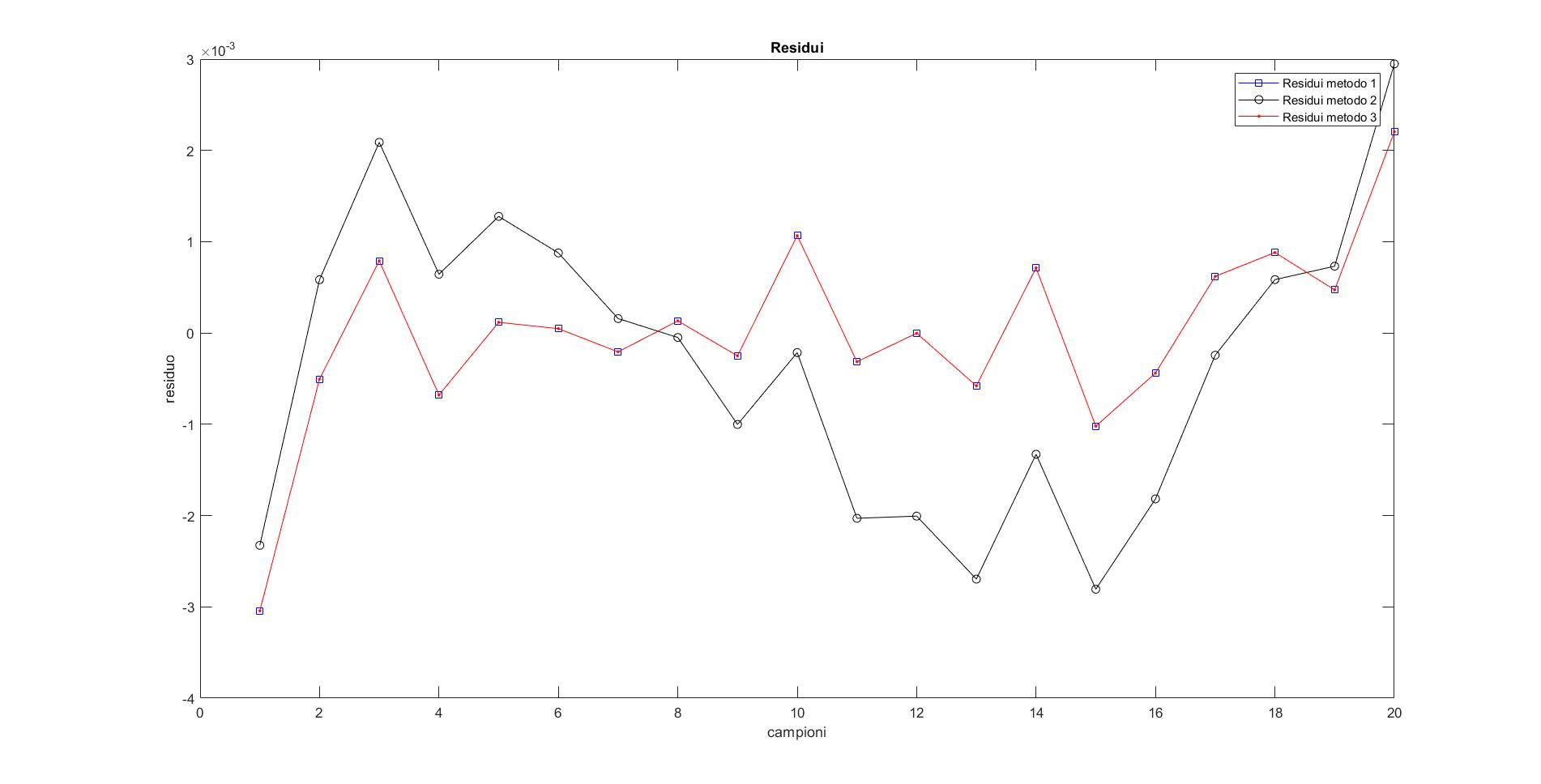
s=U\*ABC; %valori attesi

Successivamente ragioniamo sul fatto che in realtà questo primo metodo è applicabile se e solo se la frequenza del segnale è nota, ma in realtà quest’ultima dovrebbe essere un parametro ignoto. Per riuscire dunque a calcolare anche la frequenza utilizziamo una funzione particolare di Matlab, (lsqcurvefit), tramite la quale, inserendo dei valori iniziali, calcola tutte le nostre variabili. Dobbiamo usare uqesta tecnica perché la frequenza rende il sistema in misura un sistema non lineare.

Facendo ulteriori ragionamenti, però, abbiamo dedotto che fosse inutile dover calcolare tutte le nostre variabili tramite questa funzione Matlab, infatti l’unica variabile non lineare è la frequenza. Decidiamo dunque di complicare il modello al quale applichiamo la funzione lsqcurvefit in modo da calcolare prima, attraverso il primo metodo, i parametri A B e C, e una volta ottenuti questi si applica la funzione lsqcurvefit solo per calcolare la frequenza.

È interessante comparare questi tre metodi, per vedere quale è più preciso. Per fare ciò abbiamo messo a confronto i residui ottenuti da queste diverse modalità.

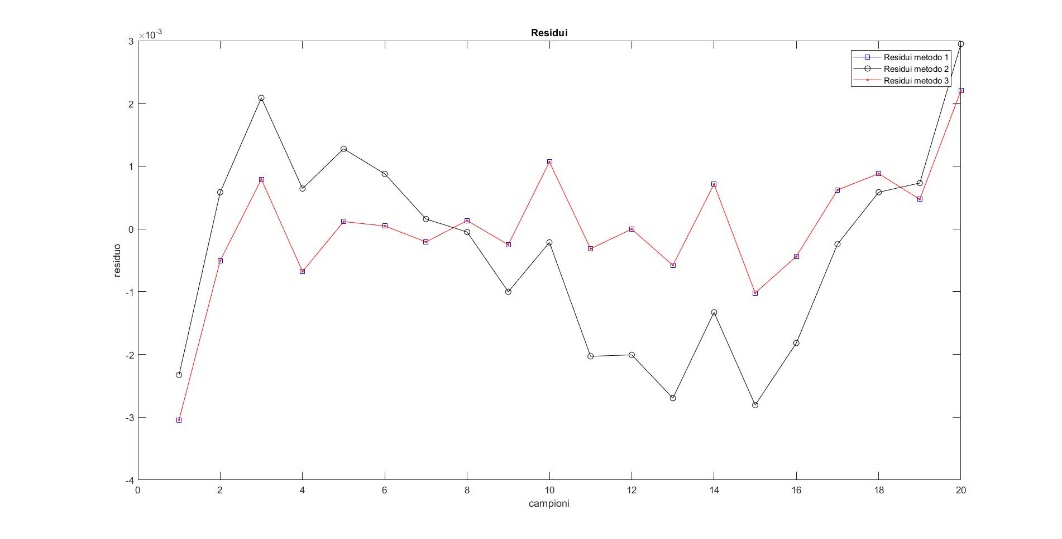
Di seguito riportiamo il grafico dei tre residui ottenuto con il prescaler impostato a 128, ovvero con una frequenza dell’ADC pari a 125KHz.



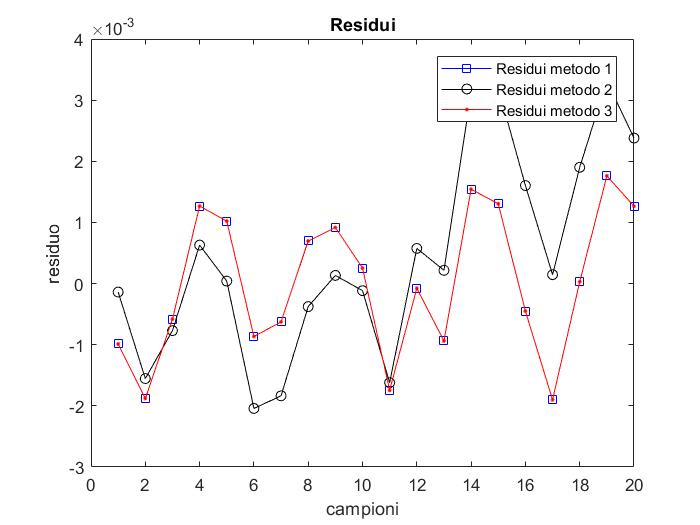
Possiamo notare che i valori dei residui del metodo uno e del metodo tre sono completamente sovrapposti e sono generalmente più bassi del metodo due. Questo avviene perché, nel seocndo metodo, la funzione di matlab trova i coefficienti che meglio approssimano la funzione non lineare, dunque ricade in un errore che deriva dall’algoritmo di Levenberg-Marquardt, utilizzato per calcolare i parametri. Invece nel metodo uno e nel metodo tre si risolve un sistema lineare, ottenendo dunque per i tre parametri A B e C un risultato più preciso.

[Questi residui dovrebbero apparire come un rumore a valori medio nullo, se così non fosse ed apparisse una sorta di trend evidente allora dovremmo supporre che la scelta del nostro modello potrebbe non essere corretta. ]Per fare in modo che la stima sia corretta è necessario che il numero di campioni sia molto maggiore del numero dei parametri.

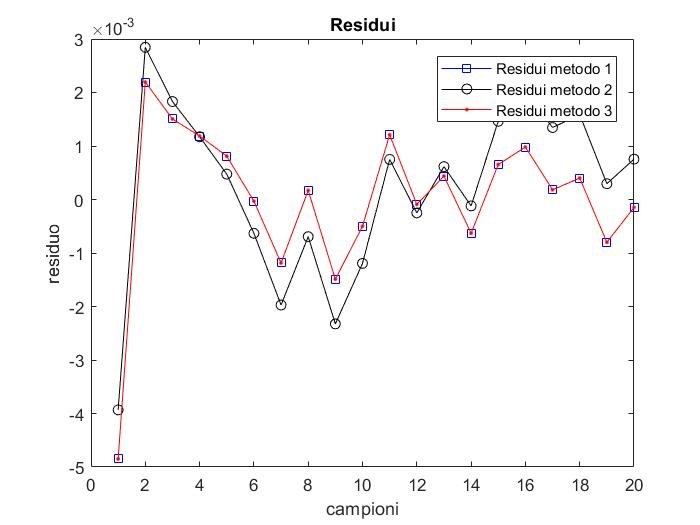
Una ulteriore analisi è quella di analizzare l’andamento dei residui al variare della frequenza dell’ADC. Notiamo dai grafici riportati in seguito come il residuo aumenti con l’abbassarsi del prescaler, ovvero alzando la frequenza.



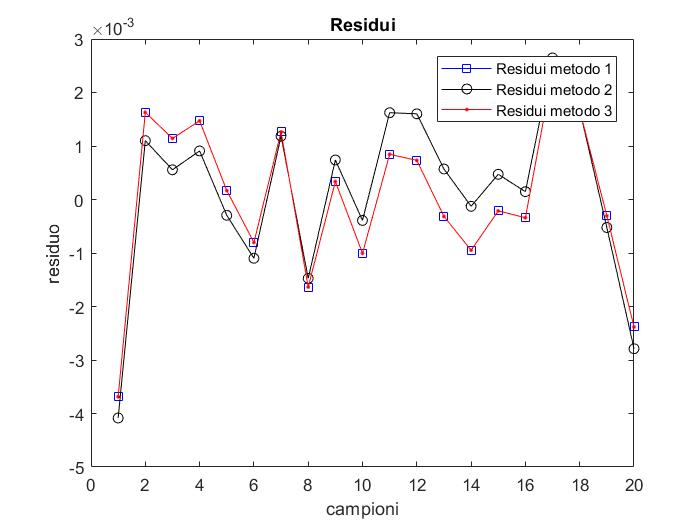
Prescaler 128



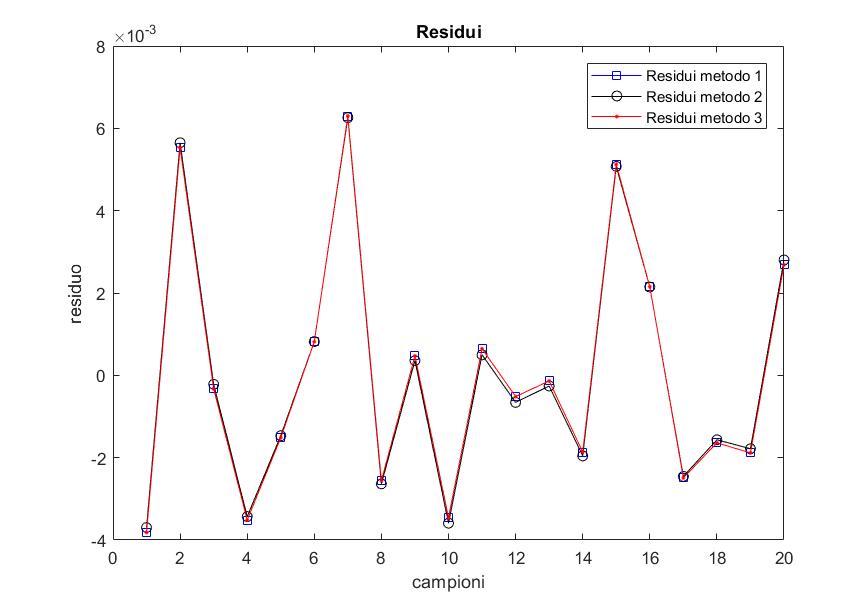
Prescaler 64



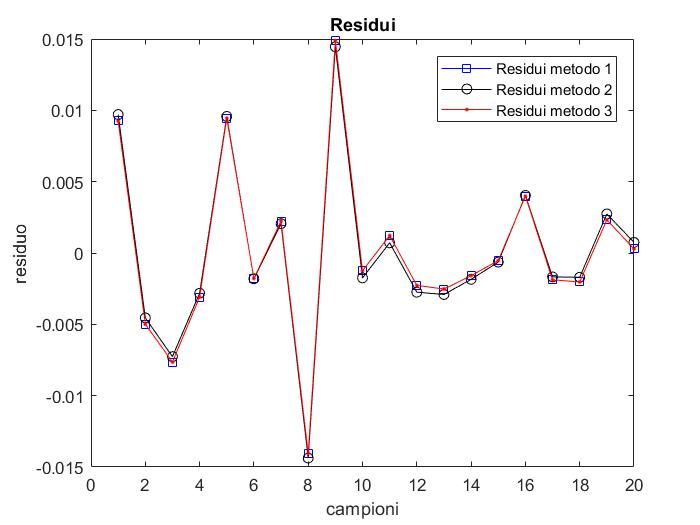
Prescaler 32



Prescaler 16



Prescaler 8



Prescaler 4

Inoltre da questi grafici possiamo notare che la differenza tra i tre metodi, man mano che si alza la frequenza di campionamento si assottiglia. Questo andamento è da attribuire al fatto che le condizioni di campionamento peggiorano, dunque i campioni sono sempre meno precisi e diventano la principale causa dello scostamento dai valori attesi, rendendo quasi ininfluente con quale metodo vengo calcolati.

**DESCRIZIONE ARDUINO**

Il compito di Arduino è quello di campionare i valori di tensione della sinusoide distanti ∆t, in modo da poter poi ricostruire il segnale correttamente. Varierò la velocità di campionamento dell’ADC scrivendo all’interno dei registri di Arduino. Quando scelgo il numero di campioni (almeno 20 per una ricostruzione del segnale abbastanza corretta) è molto importante tener conto della velocità di campionamento dell’ADC settata tramite il prescaler: infatti ogni operazione di campionamento dura per 13 colpi di clock come possiamo leggere dal datasheet del nostro oggetto.

Un’altra cosa fondamentale da considerare è l’intervallo di campionamento che, per avere una stima corretta della sinusoide, deve essere della durata del periodo.

Nella nostra analisi abbiamo preso in considerazione una sinusoide di 100 Hz, con ampiezza di 1 V, da 0 V a 1 V. In queste condizioni di lavoro abbiamo deciso di prendere 20 campioni, abbiamo gettato il prestaler dell’ADC a 128, consentendoci di campionare ad una frequenza pari a 16 MHz/2^128. Oltre a questo, se consideriamo ancora i 13 colpi di clock dovuti al campionamento otteniamo che la durata totale di un campionamento è di circa 104 microsecondi. Settando il prescaler del timer a 256 facciamo in modo che si scateni l’interrupt della soglia (che è quello che fa partire un campionamento nuovo) dopo circa 500 microsecondi. In questo modo, se dovessimo portare al massimo le prestazioni del nostro sistema, potremmo effettuare circa 90 campionamenti in un periodo.

Utilizziamo come tensione di riferimento quella a 1.1 V.

**EFFETTO MUX**

Arrivati a questo punto il nostro obiettivo è quello di stimare il valore di un condensatore campione (sceglieremo un condensatore uguale a quello misurato durante il secondo laboratorio Arduino carica-scarica).

Per fare questo campioniamo una sinusoide, generata da un generatore di funzioni, su due pin differenti tramite Arduino; abbiamo modificato leggermente il programma precedente in maniera da commutare da un canale all’altro ogni qualvolta fosse possibile effettuare una misura. Arduino prende alternativamente i campioni della sinusoide sui due canali, ovvero prende un campione prima sul canale uno e successivamente sul canale due, e cosi via.

Campionando su più canali incontriamo diversi problemi:

• Time-skew: i dati non sono misurati nello stesso istante;

• Crosstalk: il segnale su un canale influenza gli altri;

• Settling- time: lo stadio di ingresso deve avere il tempo di raggiungere il valore misurato.

[Vedremo anche che questi problemi si aggravano all’aumentare della frequenza di campionamento, vedremo infatti come la stima del condensatore diventa sempre più imprecisa.]

Per vedere questi effetti nelle nostre misure poniamo quindi un condensatore tra i canali e il GND - in modo che i segnali abbiamo valor medio nullo - e vedremo uno sfasamento tra i segnali campionati tra A0 e A1.

Per stimare questo sfasamento devo utilizzare i metodi usati precedentemente per poter trovare i parametri A, B, C e w; e di conseguenza trovare la fase delle due sinusoidi campionate. Trovata la fase dai parametri dobbiamo ragionare su un ulteriore effetto: i campioni presi sui due canali non sono corrispondenti ma hanno uno sfasamento dovuto al tempo di campionamento. Questo sfasamento può essere calcolato nel modo seguente:

Con:

, con fc pari alla frequenza di campionamento effettiva

, con f0 pari alla frequenza della sinusoide in ingresso

Una volta calcolato lo sfasamento, questo deve essere aggiunto alla fase calcolata per il secondo canale. Una volta preso questo accorgimento, si può passare al dominio dei fasori per effettuare i calcoli e trovare la componente resistiva della capacità da cui ricaveremo poi il valore di C : Zc = 1/(jwC).

Questo sarebbe il modello ideale, ma noi dobbiamo tenere conto del fatto che Zc ha una componente reale.

Dunque per trovare il valore del condensatore useremo la seguente formula:

Per calcolare Zc invece risolvo il circuito nel dominio dei fasori e ricavo:

=

R= (21970.22 ± 0,23) Ω

Il circuito è stato realizzato con una Val pari a 5V perché, nonostante non sia stabile e abbia un’incertezza del 10 %, nel calcolo della Zc il contributo dell’incertezza si semplifica grazie al termine V1/V0.

Scelgo le tensioni delle sinusoidi in maniera che sia il più vicino possibile a Val in modo da sfruttare tutta la dinamica dell’ADC, per la scelta della pulsazione (di w) invece cerco di mettermi vicino al polo del mio sistema in maniera da rendere minima l’incertezza della mia misura.

Volessi stimare gli effetti del MUX singolarmente dovrei agire in questo modo:

• settling-time: pongo un canale a 4V e l’altro a 1V e osservo quanto tempo impiega per avere il valore costante di tensione commutando da un canale all’altro.

• time-skew: posso stimarlo in maniera diretta, utilizzando dei timer per vedere quando una misura viene effettuata su un canale e calcolando la distanza in tempo rispetto al campionamento successivo sull’altro canale.

•cross-talk: pongo una tensione fissa su A1 e collego A0 al ground; misurando il canale 0 posso poi stimare quanto sia l’effetto di CH1 su questo.

Campionando le sinusoidi sui due canali inserendo il prescaler pari a 128, otteniamo i seguenti risultati:

Zc= 50.67 + 1633.1i

C=( 0.97± 0.35)μF

Per trovare l’incertezza del condensatore abbiamo considerato l’incertezza sulle misure di e di, che sono dovute all’ADC dell’Arduino (considerando ) e l’incertezza sulla resistenza , secondo il metodo deterministico.

Questo valore del condensatore è da confrontare con quello calcolato nella terza esperienza di laboratorio.