

UNIVERSITÀ DEGLI STUDI DI TRENTO  
DIPARTIMENTO DI INGEGNERIA INDUSTRIALE



---

**TESI: COMPENSAZIONE DINAMICA DI UN SENSORE DI  
TEMPERATURA PT100**

---

**PRODOTTO DA**  
Matteo Bonato, Matteo Bonetto, Enrico Michelato

Anno Accademico 2021-2022

# Sommario

<b>1 Presentazione</b>	<b>3</b>
1.1 Obiettivo . . . . .	3
1.2 Individuazione matematica della funzione di trasferimento . . . . .	3
1.2.1 Considerazioni . . . . .	3
<b>2 Circuito</b>	<b>4</b>
2.1 Descrizione del circuito . . . . .	4
2.2 Generatore di corrente . . . . .	5
2.3 Amplificatore invertente 1 . . . . .	5
2.4 Amplificatore invertente 2 . . . . .	6
2.5 Amplificatore strumentale . . . . .	7
<b>3 Identificazione della costante di tempo</b>	<b>8</b>
3.1 Importazione Dati . . . . .	8
3.2 Primo Metodo . . . . .	8
3.3 Secondo Metodo . . . . .	9
3.4 Terzo Metodo . . . . .	10
3.5 Quarto Metodo . . . . .	12
3.6 Considerazioni . . . . .	12
<b>4 PCB</b>	<b>13</b>
4.1 Descrizione Generale . . . . .	13
4.2 Specifiche elettriche e meccaniche . . . . .	14
4.2.1 Thermal relief . . . . .	14
4.2.2 Teardrop . . . . .	14
4.3 File Gerber . . . . .	14
<b>5 Custodia</b>	<b>16</b>
5.1 Descrizione . . . . .	16
<b>6 Compensazione dinamica</b>	<b>17</b>
6.1 Introduzione . . . . .	17
6.2 Effetto dei disturbi . . . . .	17
<b>7 Metodo 0</b>	<b>18</b>
7.1 Introduzione . . . . .	18
7.2 Funzione di Windowing . . . . .	18
7.3 Finestra di Hamming . . . . .	18
7.4 Funzione di trasferimento . . . . .	19
7.5 Campionamento . . . . .	20
7.6 Fast Fourier Transform . . . . .	20
7.7 Limiti del programma . . . . .	22
7.8 Efficienza dell'algoritmo . . . . .	22
<b>8 Metodo 1</b>	<b>23</b>
8.1 Introduzione . . . . .	23
8.2 Elaborazione teorica su Mathematica . . . . .	23
8.3 Programma MATLAB . . . . .	24
8.4 Risultati . . . . .	26
<b>9 Metodo 2</b>	<b>27</b>
9.1 Introduzione . . . . .	27
9.2 Descrizione . . . . .	27
9.3 Programma MATLAB . . . . .	28
9.4 Risultati . . . . .	29



# 1 Presentazione

Prendiamo in esame una sonda Pt100, ossia un termistore, che in quanto tale presenta un valore di resistenza variabile in funzione della temperatura. In particolare, la Pt100 assume una resistenza di 100 Ohm alla temperatura di 0° C.

## 1.1 Obiettivo

L'obiettivo della tesi è innanzitutto creare un termometro utilizzando la Pt100 come sensore di temperatura. Tramite questo termometro, poi, vogliamo creare uno strumento che sia in grado di approssimare la temperatura del corpo in esame in pochi secondi, senza dover aspettare l'equilibrio termico.

## 1.2 Individuazione matematica della funzione di trasferimento

Sia la sonda dotata di massa  $\mathbf{M}$ , coefficiente termico  $\mathbf{c}$ , coefficiente di scambio termico convettivo  $\mathbf{h}$  e superficie di contatto  $\mathbf{A}$ . Data  $T_f$  la temperatura del fluido in cui è immersa, posso definire la potenza termica trasmessa tra essa ed il fluido in questione:

$$P = hA(T_f - T)$$

Ricordo inoltre che la potenza termica scambiata dalla Pt100 è anche pari alla derivata dell'energia termica scambiata dalla stessa:

$$P = \frac{d}{dt} McT$$

Ponendo uguali le due equazioni appena determinate ottengo:

$$Mc \frac{d}{dt} T = hA(T_f - T) \quad (1)$$

Applico ora la trasformata di Fourier per ricavare la corrispondente equazione algebrica nel dominio di Fourier. Ricavo la funzione di trasferimento data dal rapporto tra la temperatura attuale della sonda e la temperatura del fluido nel quale essa è sommersa:

$$H(\omega) = \frac{T(\omega)}{T_f(\omega)} = \frac{1}{1 + i\omega \frac{Mc}{Ah}} \quad (2)$$

### 1.2.1 Considerazioni

Come atteso, ci troviamo di fronte ad un sistema dinamico Lineare Tempo Invariante di prim'ordine: ciò implica che, a differenza di un sistema statico, in presenza di un ingresso costante, l'uscita varia col passare del tempo.

Pertanto, per ricavare la costante di tempo, possiamo sottoporre il sistema ad un ingresso noto, come quello a gradino, e attendere che il sistema si assesti, in modo che esso possa tendere al valore di temperatura asintotica ( $T_f$ ).

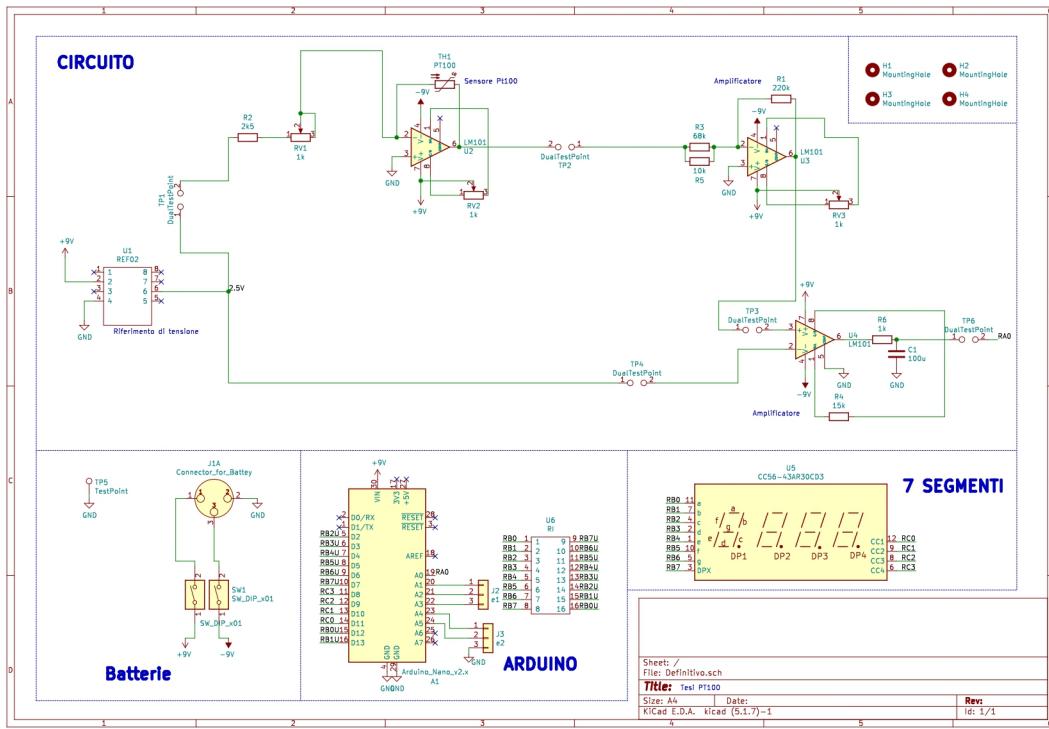
Il raggiungimento dell'equilibrio avviene per  $t > t_{assestamento} = 5 \frac{Mc}{Ah}$ .

Grazie a quest'ultima equazione, si può ricavare la costante di tempo dalla manipolazione dell'equazione (1).

Ad esempio:

$$tau = \frac{t}{\log \frac{T_0 - T_f}{T(t) - T_f}} \quad (3)$$

## 2 Circuito

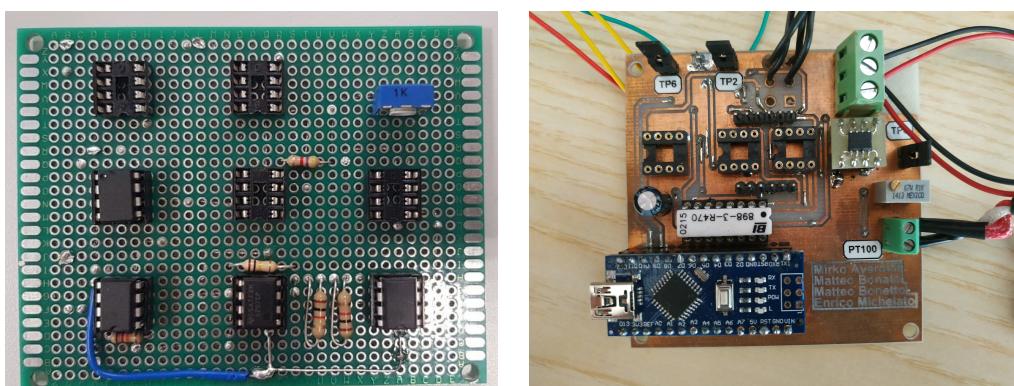


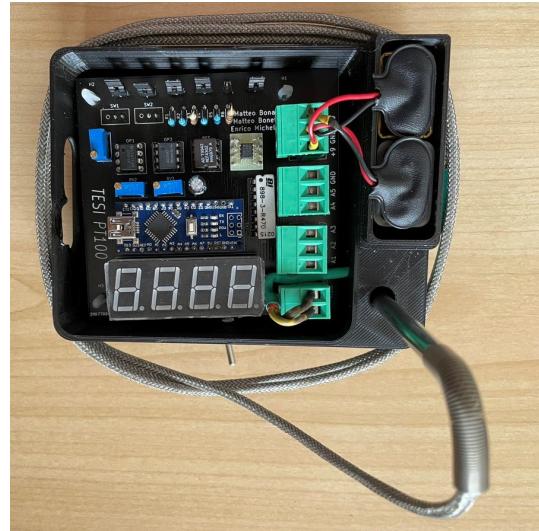
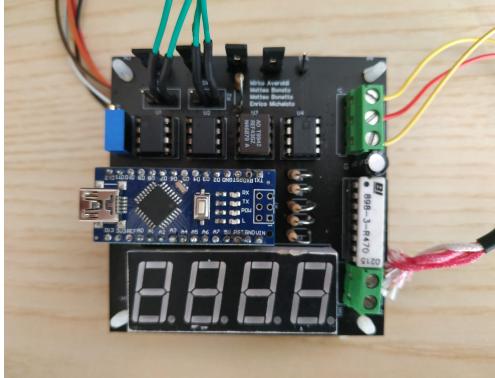
### 2.1 Descrizione del circuito

Il circuito che andrà a comporre l'intero PCB, il cui schematico è qui riportato, può essere diviso in quattro sezioni:

- slot per il collegamento delle pile, ossia l'alimentazione del PCB
- il circuito principale del PCB, che permette di convertire la temperatura della Pt100 in un valore di tensione
- un microprocessore, Arduino NANO, che elabora il segnale di tensione sopra citato
- un display 7 segmenti, per leggere la temperatura della sonda Pt100

Il circuito è stato prima assemblato su breadboard e in seguito saldato su millefori, e poi su PCB, come illustrato nelle immagini sottostanti.





## 2.2 Generatore di corrente

Per misurare la resistenza della sonda Pt100, è necessario innanzitutto che essa sia attraversata da una corrente nota.

Essendo, il nostro strumento un dispositivo mobile, deve poter essere alimentato a pile, le quali non forniscono un livello di tensione costante, necessario per poter avere una corrente costante.

Dunque è stato impiegato un **REF43GZ**, che a partire da una tensione in ingresso variabile, produce un output costante di 2.5 Volt (Figura 1).

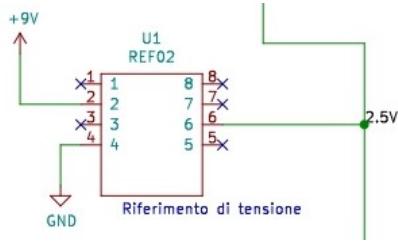
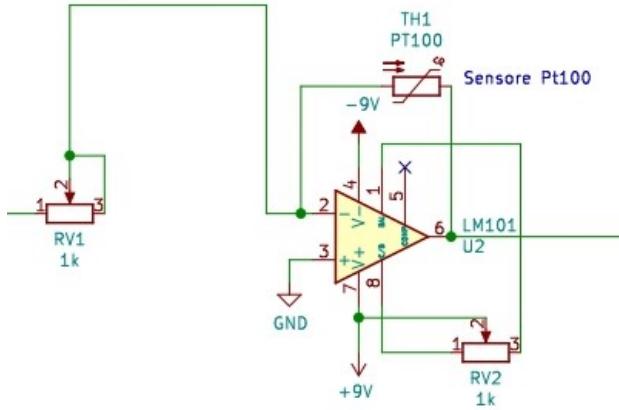


Figura 1: REF43GZ

## 2.3 Amplificatore invertente 1



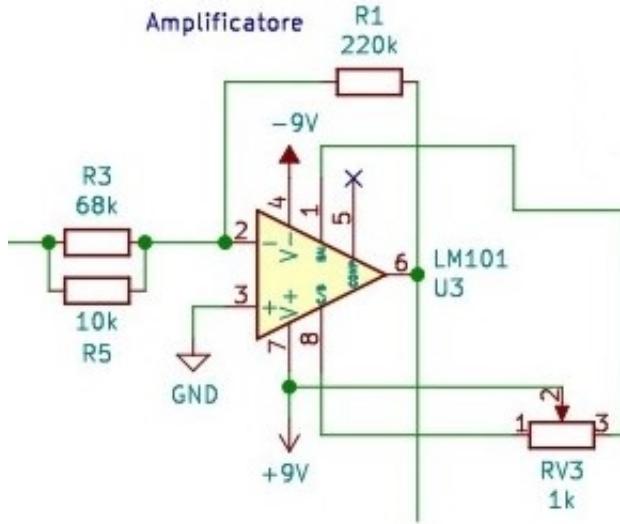
Il voltaggio in uscita dal **REF43GZ** è collegato all'ingresso invertente del primo amplificatore operazionale.

Grazie alla caratteristica del *cortocircuito virtuale*, i pin **2** e **3** sono connessi tra loro, e hanno quindi Voltaggio nullo (GND).

Per poter regolare la corrente che passa all'interno della sonda Pt100, è presente un potenziometro prima dell'Amplificatore invertente. La corrente viene quindi impostata per essere pari ad  $10^{-3} A$

Ai capi della sonda Pt100 sarà quindi presente un potenziale (negativo) pari a  $-R_{Pt100}(T)i = -10^{-3} R(T)$ , con  $-R_{Pt100}(T)$  resistenza della Pt100.

## 2.4 Amplificatore invertente 2



Il voltaggio in uscita dal primo amplificatore arriva quindi in ingresso al secondo amplificatore.

Imponendo l'uguaglianza tra la corrente in ingresso e quella in uscita otteniamo:

$$\frac{V_{in} - 0}{R_{in}} = \frac{0 - V_{out}}{R_{out}}$$

Ed isolando per ( $V_{out}$ ):

$$V_{out} = -\frac{R_{out}}{R_{in}} V_{in}$$

Inoltre, data la tensione d'ingresso  $V_{in} = -10^{-3} R_{Pt100}(T)$  e le resistenze  $R_{in}$  e  $R_{out}$ , rispettivamente pari a 8,72 KOhm e 220 KOhm:

$$V_{out} = \frac{R_{Pt100}(T)}{40} \quad (4)$$

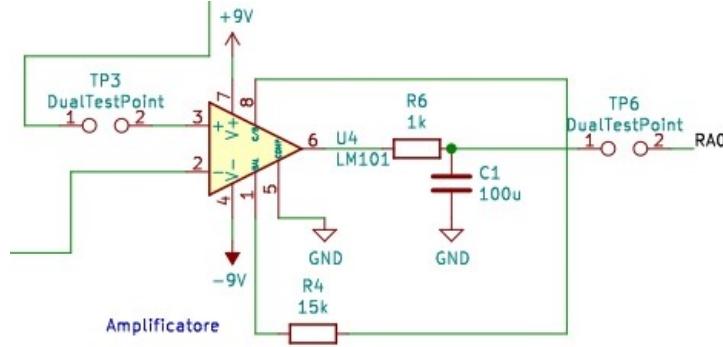
In aggiunta, come detto nella sezione 1, la resistenza della Pt100 ha andamento che, per il nostro scopo, è circa lineare con la temperatura.

Dunque, volendo rappresentare la relazione  $R_{Pt100} - T$  in un piano, essa sarà approssimabile ad una retta passante per (0 °C, 100 Ohm) e con coefficiente angolare positivo e pari a 0.38501  $\frac{\text{Ohm}}{\text{°C}}$ .

Possiamo quindi scrivere  $R_{Pt100} = 100 + 0.38501(T - 273.15)$  e sostituendola nella (4) otteniamo:

$$V_{out} = \frac{100 + 0.38501(T - 273.15)}{40} \quad (5)$$

## 2.5 Amplificatore strumentale



Questo tipo di amplificatore (**AD622**) prende come ingresso la differenza tra due tensioni, e la amplifica con guadagno **G**.

Le tensioni in ingresso nel nostro caso sono il voltaggio in uscita dal secondo OpAmp (5) e il voltaggio in uscita dal **Ref02**(1). Il voltaggio in uscita risulta quindi essere:

$$V_{out} = G(V_1 - V_2) = G\left(\frac{R(T)}{20} - 2.5\right) = G[2.5(1 + 0.0038501(T - 273.15) - 2.5]$$

A questo punto, imponendo  $V_{out} = 0$  possiamo individuare il valore minimo di temperatura misurabile, pari a  $268.15^{\circ}\text{K}$ , ossia  $-5^{\circ}\text{C}$  (tale valore di  $T_{min}$  può essere regolato cambiando i guadagni degli amplificatori invertenti).

Per conoscere invece la  $T_{max}$  misurabile dalla sonda Pt100, dobbiamo tenere presente che il valore massimo di tensione misurabile da Arduino è 5 V, pena il suo danneggiamento.

Sostituendo  $V_{out} = 5$  si può determinare il guadagno massimo dell'amplificatore strumentale:

$$V_{out} = 5 = G[5(1 + 0.0038501(T - 273.15) - 5]$$

$$G = -\frac{259.734}{273.15 - T_{max}}$$

Il guadagno è determinato dalla resistenza  $R_{gain}$  posta tra i pin 1 e 8 secondo la seguente relazione:

$$G = \frac{50.510^3 + R_g}{R_g} = -\frac{259.734}{273.15 - T_{max}}$$

Da questa equazione ricaviamo  $T_{max} = 323.019^{\circ}\text{K}$  ( $R_g$  è fissata a 15 kOhm per i calcoli riportati).

La sensibilità del nostro strumento è pari al range di temperatura misurabile diviso per la risoluzione di Arduino, ossia 10 bit ( $2^{11} - 1$  valori).

$$sens = \frac{T_{max} - T_{min}}{1023} = 0.04889\dots$$

In conclusione, ci rimane da controllare che le tensioni in uscita dagli OpAmp non superino i rispettivi livelli di saturazione positiva e negativa, cioè +9V e -9V.

Nel caso peggiore, ossia in corrispondenza di  $T_{max}$ , il voltaggio in uscita dal secondo OpAmp vale 5.96V, che rispetta pertanto le condizioni imposte.

### 3 Identificazione della costante di tempo

#### 3.1 Importazione Dati

```
7  for ii=1:numProve
8
9      dati=readmatrix(ii+".xlsx");
10     Tf(ii)=dati(end,2);
11     Ti(ii)=dati(1,2);
12     Data{ii}=dati(:,2).*(dati(:,2)>dati(1,2)+0.3);
13
14     kk=1;
15     for jj=1:length(Data{ii})
16
17         if(Data{ii}(jj)~=0)
18             Data{ii}(kk)=Data{ii}(jj);
19             kk = kk + 1;
20         end
```

Per determinare la costante di tempo  $\tau$  della sonda Pt100, la abbiamo sottoposta ad un gradino di temperatura immersendola in un fornelletto a temperatura costante.

Sono stati quindi registrati i valori di tensione in uscita dal circuito.

Dalla relazione precedentemente calcolata (2.5) abbiamo ricavato le corrispettive temperature della Pt100, poi salvate su un file excel ed importate su MatLab grazie al codice sopra mostrato.

Come si legge, infatti, da riga 9 del codice è stato creato un vettore contenente tutte le temperature per ogni prova.

Dopodiché vengono stabilite le temperature iniziali e finali per ogni prova (tale operazione è possibile solo perché abbiamo atteso un tempo sufficiente affinché la curva delle temperature si assestasse).

Successivamente, abbiamo azzerato tutti i valori del vettore di dati che non hanno almeno  $0.3^\circ$  in più rispetto al valore iniziale, poiché la Pt100 non era stata ancora immersa nel fornelletto.

Vengono infine riorganizzati i dati in un vettore di celle, con ogni cella corrispondente ad una prova diversa, grazie al ciclo compreso tra riga 15 e riga 20.

#### 3.2 Primo Metodo

```
41 for ii = 1:numProve
42     min(ii) = abs(Data{ii}(ii) -Tf(ii)-(Ti(ii) - Tf(ii))*0.368);
43 end
44
45 for ii = 1:numProve
46     for time = 1:length(Data{ii}(:,1))
47         if abs(Data{ii}(time,1) -Tf(ii)-(Ti(ii) - Tf(ii))*0.368)< min(ii)
48             min(ii) = abs(Data{ii}(time) -Tf(ii)-(Ti(ii) - Tf(ii))*0.368);
49             tau(ii) = time*Tc;
50         end
51     end
52 end
53
54 tau
55
56 tau = 1×6
57     93.2000    98.0000    94.5000    98.6000    98.0000    99.4000
58
59 Tau1 = mean(tau)
60
61 Tau1 = 96.9500
```

L'equazione che descrive l'andamento della temperatura nel tempo (ricavata dall'equazione 2) è:

$$T(t) = (T_i - T_f)e^{-\frac{t}{\tau}} + T_f \quad (6)$$

il valore di temperatura al tempo  $t = \tau$  vale quindi

$$T(\tau) = (T_i - T_f)e^{-1} + T_f$$

Si noti dunque che di questa formula appena descritta conosciamo tutte le variabili.

L'idea è quindi quella di trovare il valore di temperatura sperimentale più vicino possibile al valore teorico individuato, e da quello trovare il valore di tempo al quale la Pt100 raggiunge quella temperatura, che corrisponderà al valore di tau.

Basterà quindi determinare a che istante di tempo è stato campionato il valore di temperatura che si avvicina di più al valore teorico  $t(\tau)$  calcolato prima (3.2).

Per farlo inizializziamo ogni valore del vettore *min* come la differenza in valore assoluto tra il valore teorico e il primo dato sperimentale di ogni prova (riga 42).

Scorrendo la variabile *time* e compiendo la stessa operazione fatta in precedenza mi accerto che il valore che assume *min* sia ancora il minore rispetto al dato in posizione *time*. In caso contrario lo sostituisco proprio con quest'ultimo e salvo la posizione del vettore.

Moltiplicando la posizione del vettore in cui si trova un dato per il tempo di campionamento determino l'istante in cui tale dato è stato acquisito, determinando  $\tau$ .

### 3.3 Secondo Metodo

```

72 a=zeros(2,numProve); %coefficienti
73 %filtro
74 [B,A]=butter(4,0.3,"low");
75 for ii=1:numProve
76     filtrato=filter(B,A,Data{ii});
77     X{ii}=(150:1000)*Tc;
78     Y{ii}=log((filtrato(150:1000)-Tf(ii))/(Ti(ii)-Tf(ii)));
79     a(:,ii)=polyfit(X{ii},Y{ii},1);
80     tau(ii)=-1/a(1,ii);
81 end

```

Manipolando l'equazione (6) si ottiene la seguente:

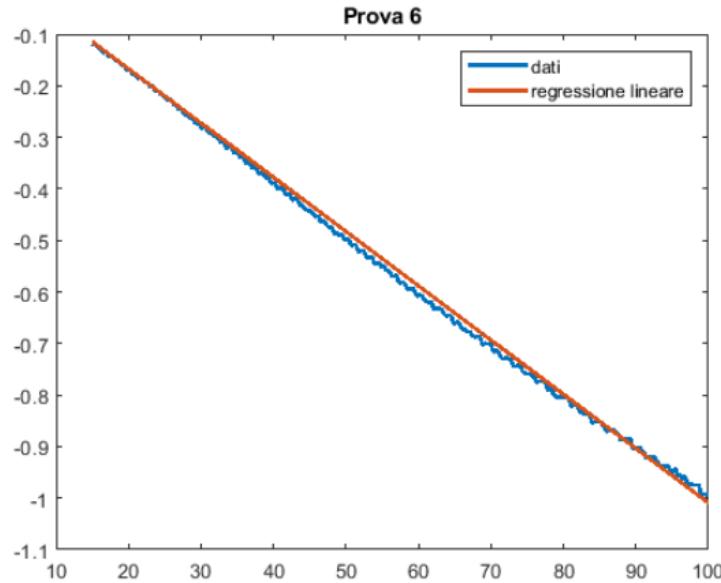
$$\log \frac{T(t) - T_f}{T_i - T_f} = -\frac{t}{\tau}$$

Rappresentando su un grafico  $\log \frac{T(t) - T_f}{T_i - T_f}$  sull'asse X, e il tempo sull'asse Y, otteniamo l'equazione di una retta che ha per coefficiente angolare  $-\frac{1}{\tau}$ .

Abbiamo quindi applicato un filtro passabasso (riga 74) per togliere del rumore indesiderato.

Infine, attraverso una regressione lineare (interpolazione) viene approssimato l'andamento della curva determinata con una retta.

Ricordando che il valore del coefficiente angolare di questa retta è  $-\frac{1}{\tau}$ , possiamo risalire ad un valore di  $\tau$ .



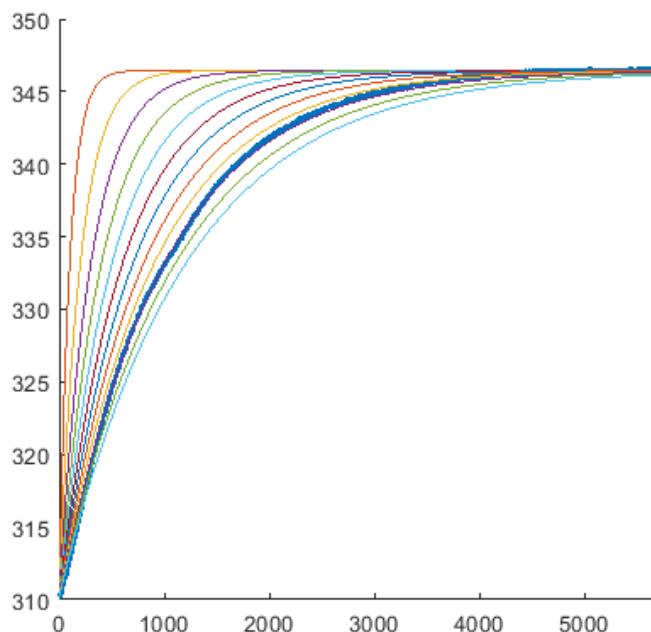
### 3.4 Terzo Metodo

Il terzo metodo che abbiamo implementato è il metodo dei minimi quadrati: sia  $T_s$  il valore sperimentale campionario,  $N$  la lunghezza del vettore di dati e  $\phi$  una funzione da noi definita come

$$\phi(\tau) = \sum_{s=0}^{s=N} (T_s - (T_i - T_f)e^{-\frac{sT_c}{\tau}} - T_f)^2$$

Il valore di  $\tau$  da noi ricercato è quello che minimizza la funzione  $\phi$ .

Vengono, in pratica, "simulate" tante prove, ognuna corrispondente ad una Pt100 con un valore di  $\tau$  diverso. Quella che più assomiglia alla prova da noi svolta sarà quella corrispondente al  $\tau$  che più si avvicina a quello della nostra sonda Pt100.



Per mezzo della variabile *scelta*, viene selezionata una determinata prova, mostrata nel grafico qui sopra assieme ad alcune delle curve ideali, che seguono l'andamento teorico. Tra tali curve quella che minimizza  $\phi$  è anche la funzione che meglio segue il comportamento della nostra curva sperimentale.

```

tot=0;
for ii = 1:numProve
    for time = 1:length(Data{ii}(:,1))
        min(ii)=min(ii)+power(Data{ii}(time) ...
            -Tf(ii)-(Ti(ii) - Tf(ii))*power(exp(1),-time*Tc/1),2);
    end
end

figure;

for ii = 1:numProve

    if ii==scelta
        hold on;
        plot(Data{ii}, 'LineWidth',2);
    end
    for tauVar=10:1:120

        tot=0;
        for time = 1:length(Data{ii}(:,1))
            tot=tot+power(Data{ii}(time) ...
                -Tf(ii)-(Ti(ii) - Tf(ii))*power(exp(1),-time*Tc/tauVar),2);

            if ii==scelta
                y(time)=Tf(ii)+(Ti(ii) ...
                    - Tf(ii))*power(exp(1),-time*Tc/tauVar);
            end
        end

        if mod(tauVar,10)==0 & ii==scelta
            hold on;
            plot(y);
        end

        if tot< min(ii)
            min(ii) = tot;
            tau3(ii) = tauVar;
        end
    end
end

```

### 3.5 Quarto Metodo

```
145 for ii=1:numProve
146     for time=1:length(Data{ii})
147         tauVar2(time)=time*Tc/log( abs( (Ti(ii)-Tf(ii)) / (Data{ii}(time)-Tf(ii)) ) );
148     end
149     tau4(ii)=mean(tauVar2);
150 end
151
152 tau4
```

L'ultimo metodo fa uso dell'equazione (3) ricavata precedentemente:

$$\tau_{\text{au}} = \frac{t}{\log \frac{T_0 - T_f}{T(t) - T_f}} \quad (7)$$

Dato che questa relazione può essere adoperata per ogni dato di ogni prova, è stato deciso di ottenere un  $\tau$  medio.

### 3.6 Considerazioni

Abbiamo ottenuto un valore di costante di tempo diverso a seconda della prova eseguita in laboratorio, e del metodo di calcolo utilizzato.

Il valore approssimato di  $\tau$  è ottenuto dalla media di tutti i valori determinati:

$$\tau_{\text{medio}} = 96.0.967s$$

Il tempo di acquisizione utilizzato (pari a 700s circa) risulta essere stato abbondantemente sufficiente affinché la temperatura della sonda si assesti a  $T_f \rightarrow \tau_{\text{medio}} = 96.0.967s \implies 700 > 5\tau_{\text{medio}}$ .

## 4 PCB

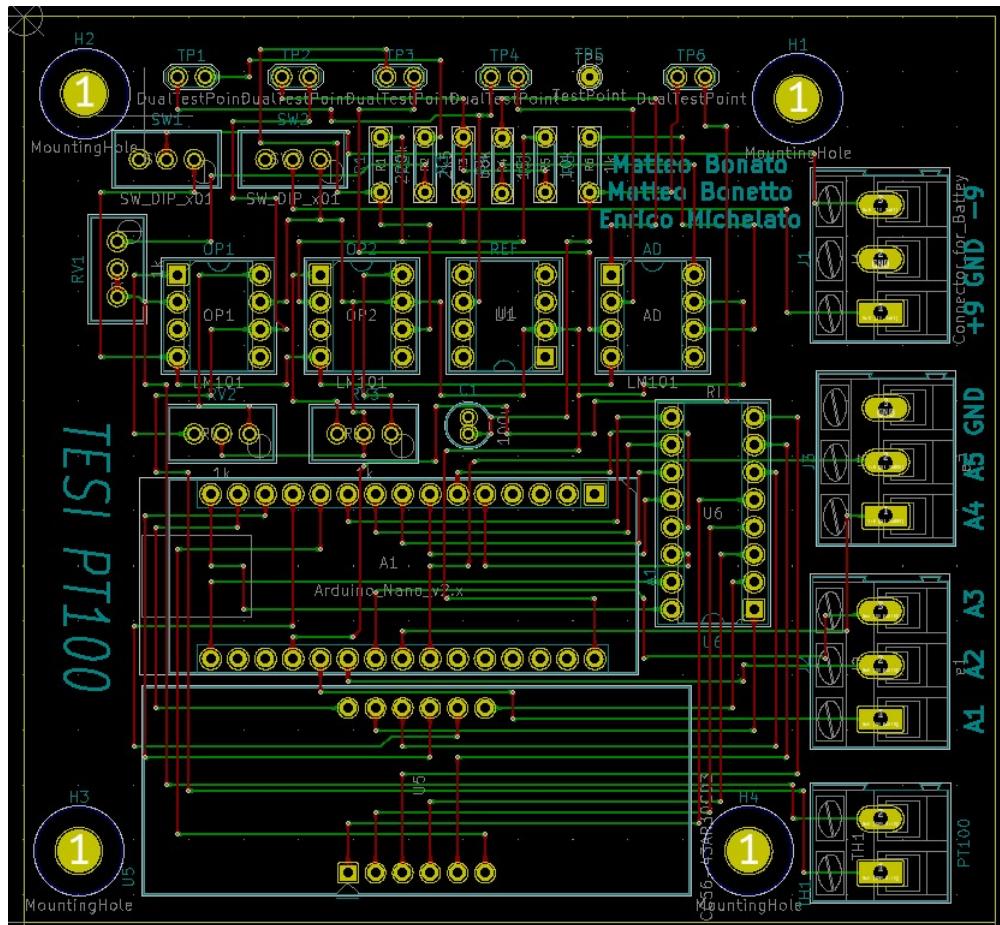


Figura 2: PCB

### 4.1 Descrizione Generale

Nella realizzazione del PCB (Printed Circuit Board) è stata determinata la disposizione dei dispositivi in modo da consentire un facile utilizzo dello strumento.

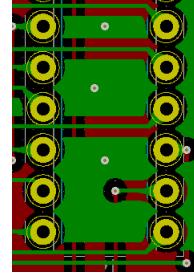
Un esempio sono la posizione centrale del display, la disposizione laterale di Arduino, che permette la programmazione senza doverlo necessariamente rimuovere dalla scheda, e la posizione dei test point, collocati verso l'esterno della scheda. Inoltre è importante che l'alimentazione sia rivolta verso l'esterno. Questo per facilitarne il collegamento con le pile che saranno posizionate nella custodia, esternamente al PCB.

Tale custodia verrà infine fissata al PCB tramite i fori di montaggio posizionati ai quattro angoli della scheda.

## 4.2 Specifiche elettriche e meccaniche

Come si nota da Figura 2, il routing è stato ottimizzato impiegando direzioni preferenziali a seconda che le piste appartenessero al top layer (piste in rosso) o al bottom layer (piste in verde).

Inoltre, tra le piste viene lasciata sempre una "clearance" minima, affinché risultino elettricamente isolate. Sono state aggiunte svariate *vias* (piccoli fori metallizzati) in modo da assicurare che tutte le zone tra i due layer abbiano lo stesso potenziale.



### 4.2.1 Thermal relief

Altro elemento presente nel PCB realizzato sono i thermal relief.



Essi permettono di isolare meglio termicamente, in modo che nel momento della brasatura, sia più semplice fondere il materiale d'apporto.

Come si vede dall'immagine, infatti, sebbene il pad dovrebbe essere del tutto immerso nel Bulk, al contrario si vengono a creare delle zone vuote in corrispondenza di dove dovrà avvenire la brasatura.

### 4.2.2 Teardrop

Infine, sono state aggiunte 279 teardrops:



Una teardrop è un trattamento post-processing che consiste nell'aggiunta di porzioni di rame per evitare che le sollecitazioni meccaniche sfocino nella formazione di cricche, le quali potrebbero perfino interrompere l'instradamento di una pista, impedendo il corretto funzionamento della scheda.

## 4.3 File Gerber

I file Gerber sono i layer fisici e le lavorazioni necessarie alla creazione della scheda.

I file Gerber che creiamo sono:

- i layer di rame top e bottom
- l'edge cuts, che mostra dove termina il PCB e quindi ne indica le dimensioni (nel nostro caso sono di 73.66mm\*71.12mm)
- la Silkscreen, ossia la serigrafia, che ci dà un'idea della posizione che è stata designata per ogni componente e di identificarli velocemente
- la Soldermask, che viene applicata su tutta la scheda, fatta eccezione per i pad e i fori metallizzati, per permettere un migliore isolamento (non solo elettrico, ma anche contro polvere e umidità)
- il file drill, che contiene le informazioni che indicano dove dovranno essere praticati i fori sia metallizzati, che quelli di montaggio

Le immagini a pagina 15 rappresentano corrispettivamente i file *Bottom Copper*, *Top Copper*, *SolderMask*, *Silkscreen* e *Drill File*.

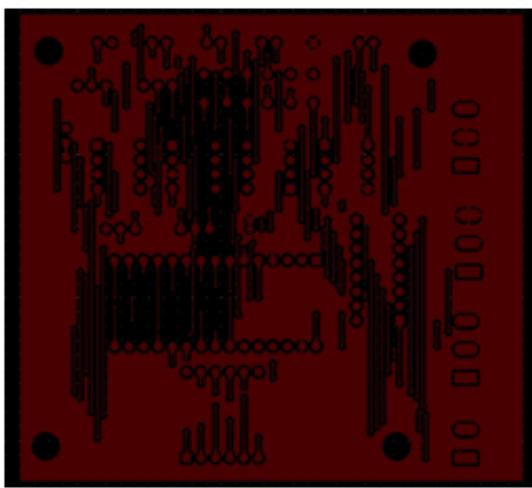
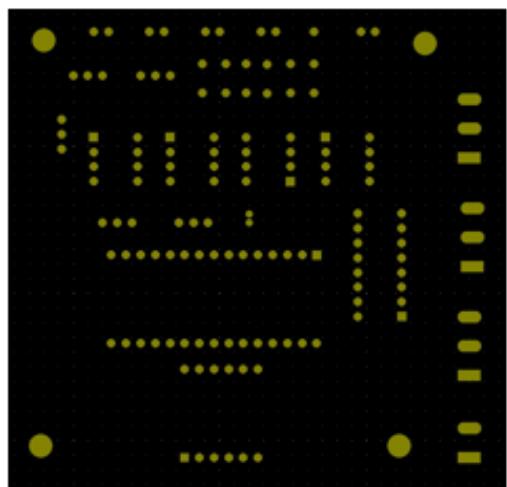
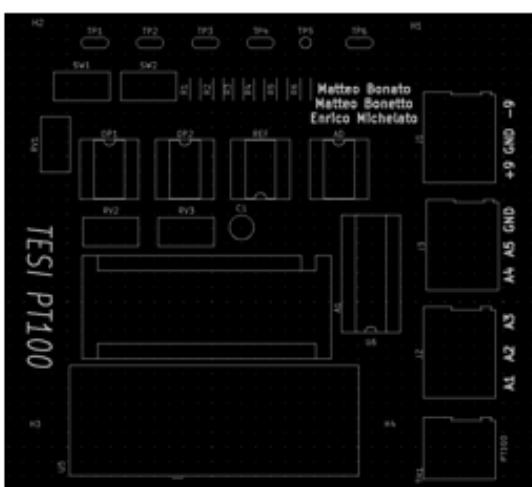
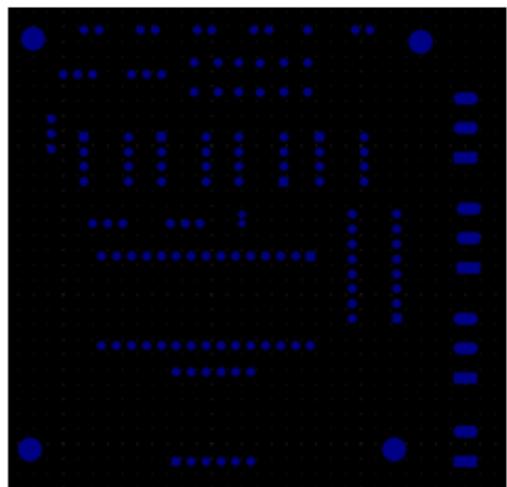
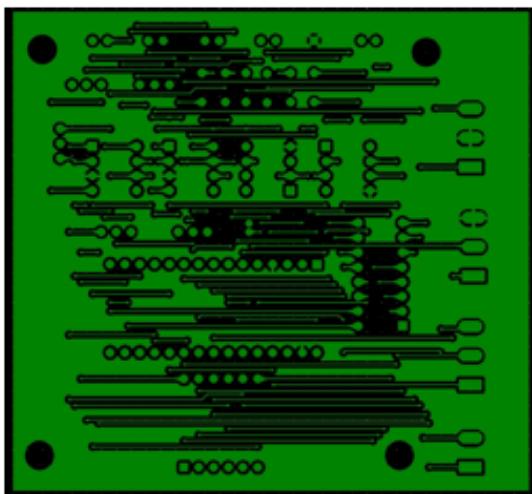


Tabella 1: Gerber files

## 5 Custodia

### 5.1 Descrizione

La custodia è completa di coperchio ed ha una importante funzione di protezione. Infatti è necessario proteggere la scheda da polvere, umidità, e, nel caso peggiore, da urti.

La custodia inoltre salvaguarda i componenti elettronici dalla polvere. Essa può formare uno strato sottile conduttivo che può pregiudicare la resistenza dell'isolamento superficiale tra i conduttori. Inoltre, la polvere che si accumula sui componenti attivi e i connettori di alimentazione può provocare danni irreversibili nel momento in cui i componenti si surriscaldano.

Abbiamo dovuto quindi ricercare un trade-off tra robustezza, dimensioni e massa complessiva. Abbiamo quindi adottato alcune accortezze, come evitare la presenza di punti angolosi o introdurre nervature che garantiscono basse deformazioni in seguito alla presenza di carichi.

La nostra custodia è stata prodotta tramite una stampante 3D (Ender-3). Quest'ultima utilizza PLA, un polimero con proprietà meccaniche mediocri ma che per il nostro utilizzo rappresentano un'ottima soluzione.

Per fissare la scheda alla custodia abbiamo predisposto quattro mounting holes da 3mm di diametro in prossimità degli angoli del PCB. Il corretto fissaggio è possibile grazie a degli appositi supporti.

Nella progettazione del PCB e custodia abbiamo adottato utili accorgimenti per agevolare l'utilizzo del prodotto finale:

1. foro centrale per visualizzare al meglio i valori del display
2. foro sul coperchio per permettere di accedere ai test point
3. sede per le batterie da 9V
4. sede dove riporre la PT100
5. foro laterale che permette di collegare la porta USB del nostro Arduino nano

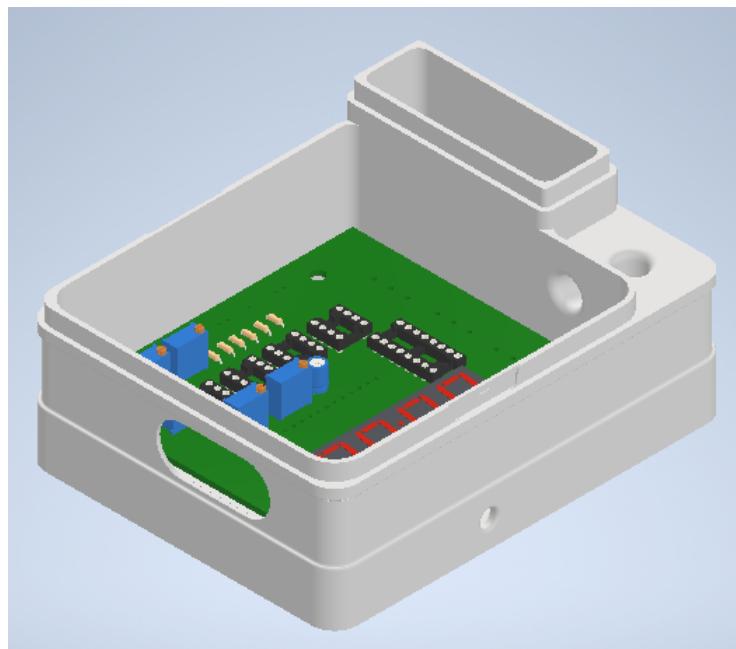


Figura 3: Custodia

## 6 Compensazione dinamica

### 6.1 Introduzione

Dato un sistema dinamico regolato da equazioni differenziali lineari, la caratteristica che determina l'uscita in base ad un ingresso è la funzione di trasferimento.

In un sistema di misura, l'ingresso  $u(t)$  viene inteso essere il misurando, mentre l'uscita corrisponde con il segnale che lo strumento fornisce.

Chiaramente è necessario avere una certa conoscenza dello strumento utilizzato per effettuare la misura, ed in particolare della sua funzione di trasferimento.

Quindi, nel dominio delle frequenze, vale la seguente relazione:

$$U(\omega) = H(\omega) * Y(\omega)$$

dove  $U(\omega)$  è l'uscita,  $H(\omega)$  la funzione di trasferimento, e  $Y(\omega)$  l'ingresso.

Nel nostro caso specifico,  $Y(\omega)$  è la temperatura che noi vogliamo sapere, cioè la temperatura del fornelletto.  $H(\omega)$  è la funzione di trasferimento dello strumento che utilizziamo per misurare il fornelletto, ossia la Pt100.  $U(\omega)$  è la misura dell'uscita del circuito, che noi quindi conosciamo.

La compensazione dinamica è proprio il processo che permette, avendo a disposizione l'uscita dello strumento, di stimare l'andamento del misurando e quindi effettuare l'operazione di misura.

$$Y(\omega) = \frac{U(\omega)}{H(\omega)} \quad (8)$$

### 6.2 Effetto dei disturbi

Altro effetto da considerare allorquando si considera l'operazione di compensazione dinamica è quello degli ingressi di disturbo di tipo interferente.

Nel caso in cui l'effetto di tali ingressi interferenti sia importante ed abbia componenti in frequenza molto estese nello spettro, l'operazione di compensazione dinamica rischia di amplificare il rumore piuttosto che ricostituire le armoniche originarie del misurando.

Poiché tale situazione è comune nelle situazioni pratiche della misurazione, la compensazione dinamica viene effettuata in intervalli di frequenza limitati, ove il modulo della funzione di trasferimento è certamente noto con buona accuratezza e l'effetto degli ingressi interferenti è sicuramente minore dell'effetto delle componenti armoniche del segnale.

# 7 Metodo 0

## 7.1 Introduzione

La terza soluzione adottata per compiere la compensazione dinamica consiste nell'effettuare uno studio offline dell'uscita del circuito. Tutti i calcoli vengono svolti da Arduino, e poi mostrati su un grafico da MATLAB.

L'uscita viene campionata a gruppi di 128 elementi, con una frequenza di campionamento di 10 Hz. Su ogni gruppo di campioni viene effettuata una FFT, per determinare lo spettro delle frequenze del segnale.

$$u(t) \rightarrow U(\omega) \quad (9)$$

Grazie al teorema della risposta in frequenza, è possibile ricostruire l'ingresso del segnale, conoscendo la funzione di trasferimento dello strumento che lo legge, come spiegato nel capitolo precedente. Conoscendo la funzione di trasferimento della PT100, ricostruisco nel dominio delle frequenze il segnale di ingresso.

$$Y(\omega) = \frac{U(\omega)}{H(\omega)} \quad (10)$$

Una volta determinato il segnale d'ingresso nel dominio delle frequenze, viene effettuata una FFT inversa, che ci permette di ricostruire lo stesso segnale nel dominio del tempo.

$$Y(\omega) \rightarrow y(t) \quad (11)$$

A questo punto viene mostrato il risultato così calcolato, ed il procedimento ricomincia.

## 7.2 Funzione di Windowing

Una funzione di Windowing (funzione finestra) è una funzione che vale zero al di fuori di un certo intervallo. Per esempio, una funzione che è costante all'interno dell'intervallo è chiamata finestra rettangolare.

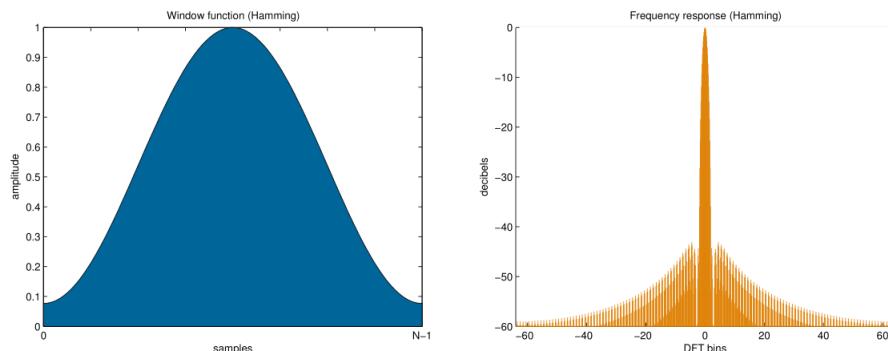
Quando un'altra funzione è moltiplicata per una funzione finestra, il risultato assume valori nulli al di fuori dell'intervallo: tutto ciò che resta è la "vista" attraverso la finestra.

Noi ci serviamo di una funzione di Windowing per eliminare le componenti di rumore in alta frequenza, in modo da lavorare in un intervallo in cui la funzione di trasferimento della Pt100 è ben nota.

## 7.3 Finestra di Hamming

In particolare, la funzione finestra da noi utilizzata è la funzione di Hamming.

La funzione di Hamming è un coseno rialzato, che vale -75 dB al di fuori dell'intervallo considerato



## 7.4 Funzione di trasferimento

Come descritto nel capitolo (1.2), la funzione di trasferimento della sonda di temperatura Pt100 è la seguente:

$$H(\omega) = \frac{T(\omega)}{T_f(\omega)} = \frac{1}{1 + i\omega \frac{Mc}{Ah}} \quad (12)$$

Per implementare questa funzione su Arduino ci siamo serviti della programmazione ad oggetti. Abbiamo creato un oggetto di tipo FDT, che avesse come unico parametro la costante di tempo  $\tau$ .

```
class FDT{
    private:
        float T;
    public:
        FDT() {
            this->T = TIME_CONSTANT;
        }

        FDT(float t) {
            if(t <=0) this->T = 0;
            else this->T = t;
        }

        void setT(float timeC) {
            this->T = timeC;
        }
}
```

Questo oggetto ha 2 metodi:

- Il primo metodo ci consente di determinare la fase della funzione di trasferimento ad una certa frequenza. Data la FdT precedentemente mostrata, la sua fase sarà pari a:

$$Phase(\omega) = - \arctan\left(\omega \frac{Mc}{Ah}\right) \quad (13)$$

Il metodo che calcola la fase sarà quindi costruito così:

```
float Phase(float f){
    float P = - atan2(f*this->T,1);
    return P;
}
```

- Il secondo metodo ci consente di determinare l'ampiezza della funzione di trasferimento ad una certa frequenza. Data la FdT precedentemente mostrata, la sua ampiezza sarà pari a:

$$Magnitude(\omega) = \sqrt{1^2 + \omega^2 \left(\frac{Mc}{Ah}\right)^2} \quad (14)$$

Il metodo sarà quindi implementato come segue:

```

float Magnitude(float f){
    float M = 1.0/sqrt(1.0 + pow(f*this->T,2.0));
    return M;
}

```

## 7.5 Campionamento

Per effettuare lo studio di un segnale nel dominio della frequenza, è innanzitutto necessario effettuare il campionamento del suddetto segnale.

Essendo il segnale in questione una variazione di temperatura, abbiamo ritenuto sufficiente campionare ad una frequenza di 1000 Hz, sebbene Arduino potesse arrivare fino a 10 kHz.

Il numero scelto di campioni è pari a 128, come spiegato meglio nel capitolo 7.7.

```

#define SAMPLES 128
#define SAMPLING_FREQUENCY 10
#define TIME_CONSTANT 96.0967

```

Il valore letto da Arduino, tuttavia, è un valore di voltaggio, chiaramente. Per ricondurci alla corrispettiva misura di temperatura, compiamo il seguente conto:

$$T(t) = 5.42341 + 9.12414 * V(t) - 273.15 \quad (15)$$

A questo punto, abbiamo riempito il vettore vReal di 128 campioni di temperature, corrispondenti alla temperatura registrata da Arduino nell'intervallo di tempo.

## 7.6 Fast Fourier Transform

Per effettuare la trasformazione nel dominio della frequenza del segnale appena registrato, ci affidiamo alla libreria ArduinoFFT, disponibile sull'IDE stesso di Arduino.

Attraverso il comando ArduinoFFTWindowing, viene effettuato il filtraggio del segnale in uscita per mezzo del filtro di Hamming, spiegato nel capitolo 7.3.

```
FFT.Windowing(vett1, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
```

Questa funzione di Windowing va a togliere 75 dB a tutte le componenti che si trovano a frequenza maggiore di quella di campionamento.

A questo punto, dato che rimangono solo le componenti in frequenza più significative e prive di rumore, possiamo effettuare la FFT.

La funzione ArduinoFFTCompute prenderà il vettore di temperature vett1, effettuerà su di esso la FFT, e inserirà nei vettori vett1 e vett2 i corrispondenti valori di parte reale ed immaginaria.

```
FFT.Compute(vett1, vett2, SAMPLES, FFT_FORWARD);
```

Grazie al teorema della risposta in frequenza, possiamo facilmente determinare l'ampiezza e la fase del segnale di ingresso, dato che conosciamo la funzione di trasferimento dello strumento e il segnale in uscita.

Calcolo l'ampiezza dell'ingresso come rapporto tra l'ampiezza del misurando  $u(t)$  ad una certa frequenza  $f$ , e l'ampiezza della funzione di trasferimento alla stessa frequenza. Salvo il valore nella variabile temporanea temp1.

```
temp1 = sqrt(pow(vett1[f],2)+pow(vett2[f],2)) / fdt.Magnitude(f); //magn input
```

Calcolo la fase dell'ingresso come differenza tra la fase del misurando  $u(t)$  ad una certa frequenza  $f$ , e la fase della funzione di trasferimento alla stessa frequenza.

```
temp2 = atan2(vett2[f], vett1[f]) - fdt.Phase(f); //phase input
```

Ripeto questo procedimento per tutte le frequenze.

```
for (int f = 0; f < SAMPLES; f++){
    temp1 = sqrt(pow(vett1[f],2)+pow(vett2[f],2)) / fdt.Magnitude(f); //magn input
    temp2 = atan2(vett2[f], vett1[f]) - fdt.Phase(f); //phase input
    vett1[f] = temp1*cos(temp2); //vett1->real input
    vett2[f] = temp2*sin(temp2); //vett2->imag input
}
```

A questo punto le variabili temp1 e temp2 contengono rispettivamente l'ampiezza e la fase del segnale di ingresso.

Determino la parte reale ed immaginaria del segnale di ingresso moltiplicando l'ampiezza rispettivamente per il seno e coseno della fase.

```
vett1[f] = temp1*cos(temp2); //vett1->real input
vett2[f] = temp2*sin(temp2); //vett2->imag input
```

Ora posso effettuare la trasformazione inversa, e tornare nel dominio del tempo.

```
FFT.Compute(vett1, vett2, SAMPLES, FFT_REVERSE); //temp input
```

Questa funzione prende come ingresso i 2 vettori vett1 e vett2 contenenti rispettivamente parte reale ed immaginaria del segnale, e restituisce il vettore vett1 contenente i valori di temperatura nel dominio del tempo.

A questo punto, prendiamo gli ultimi 5 valori calcolati, ne facciamo la media, e mostriamo il risultato sul display.

```
InReal[i] = InReal[i]*10;
T_int=T_finale;
sevseg.setNumber(T_int,1);
```

## 7.7 Limiti del programma

La FFT è un potente algoritmo, che ha tuttavia dei limiti. Non è infatti possibile effettuare la FFT in maniera dinamica, ossia utilizzando un campione alla volta. E' necessario ogni volta attendere che vengano registrati un numero sufficiente di campioni, e su di essi effettuare la trasformata.

Il risultato della compensazione effettuata con questa modalità sarà quindi inevitabilmente appartenente al passato. Attraverso il clock di Arduino abbiamo potuto stimare la durata di una compensazione: 3 decimi di secondo.

## 7.8 Efficienza dell'algoritmo

Dato che Arduino non possiede il più potente dei processori, abbiamo dovuto effettuare uno studio sull'efficienza dell'algoritmo utilizzato. Per ricostruire il segnale in ingresso al circuito nel dominio del tempo è possibile ricorrere a diversi metodi.

Un altro metodo che è possibile utilizzare è quello della convoluzione. E' possibile dimostrare la seguente relazione:

$$x(t) \otimes y(t) \rightarrow X(\omega) * Y(\omega) \quad (16)$$

dove l'operatore  $\otimes$  indica l'operazione di convoluzione, mentre l'operatore  $\rightarrow$  indica l'operazione di trasformazione. Applico la relazione appena definita al nostro specifico caso, indicando con  $u(t)$  e  $i(t)$  i rispettivi segnali di uscita ed ingresso, e con  $h(t)$  la funzione di trasferimento della PT100.

$$i(t) = u(t) \otimes \frac{1}{h(t)} \rightarrow I(\omega) * \frac{1}{U(\omega)} = I(\omega) \quad (17)$$

Da questa relazione si può notare che l'operazione di convoluzione consente di elaborare il segnale in uscita e ricostruire quello in ingresso rimanendo nel dominio del tempo, senza quindi effettuare alcuna trasformata di Fourier.

Abbiamo analizzato quindi la complessità dei 2 differenti metodi. Dato un segnale costituito da un numero  $n$  di campioni, l'operazione di convoluzione necessita di effettuare  $n$  operazioni per ogni campione. La totale complessità algoritmica  $o(n)$  richiesta quindi per elaborare un segnale di  $n$  campioni tramite la convoluzione è pari a:

$$o(n) = n^2 \quad (18)$$

Nel nostro particolare caso, dato un segnale di 128 campioni, la ricostruzione del segnale di ingresso richiederebbe 16384 operazioni.

D'altro canto, la compensazione dinamica effettuata tramite la trasformata di Fourier risulta essere molto meno dispendiosa. Un'operazione di trasformazione su  $n$  campioni richiede  $\log(n)$  operazioni. Dato che l'operazione di trasformata deve essere effettuata 2 volte, la totale complessità dell'algoritmo di compensazione risulta essere pari a:

$$o(n) = n \log(n) \quad (19)$$

Nel nostro particolare caso, dato un segnale di 128 campioni, la ricostruzione del segnale di ingresso richiederebbe 540 operazioni. La compensazione dinamica effettuata nel dominio delle frequenze risulta quindi essere 30 volte più efficiente.

# 8 Metodo 1

## 8.1 Introduzione

L'idea su cui si basa questa soluzione prevede di dividere il segnale in intervalli ( $t_{int}$ ), abbastanza ampi da contenere più campioni ( $t_c$ ), e salvare il valore di tensione in ingresso rilevato al primo campionamento.

A questo punto, ipotizziamo che da un intervallo all'altro non vi siano sbalzi della temperatura d'ingresso significativi (essendo comunque  $T_{int}$  circa un secondo non ci aspettiamo, nel campo delle misure termiche, variazioni a queste frequenze) e così facendo possiamo considerare ogni intervallo soggetto ad ingressi a sé stanti.

## 8.2 Elaborazione teorica su Mathematica

Per verificarne il possibile corretto funzionamento prima di implementare il codice su MATLAB e testarlo sul piano pratico, ipotizziamo di ricevere come temperatura di ingresso una funzione sinusoidale.

Per mostrare come tale segnale influenza la temperatura della Pt100 definiamo la funzione di trasferimento della Pt100 (prima riga di codice di Figura 8.2) e lo moltiplichiamo per la trasformata dell'ingresso sinusoidale. Dopodiché antitrasformiamo l'intero prodotto per ottenere l'uscita in funzione del tempo e chiamiamo tale funzione **eq**.

In questo esempio abbiamo voluto testare una situazione molto più penalizzante rispetto a quelle che verranno realmente sperimentate: infatti, sono stati associati a  $t_{int}$  e  $t_c$ , rispettivamente, i valori di 10 e 1 secondo.

Da equazione 6, ricavata proprio dall'ingresso a gradino:

$$T(t) = (T_i - T_f)e^{-\frac{t}{\tau}} + T_f \quad (20)$$

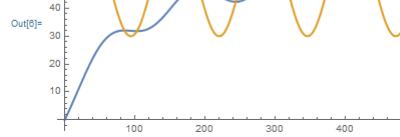
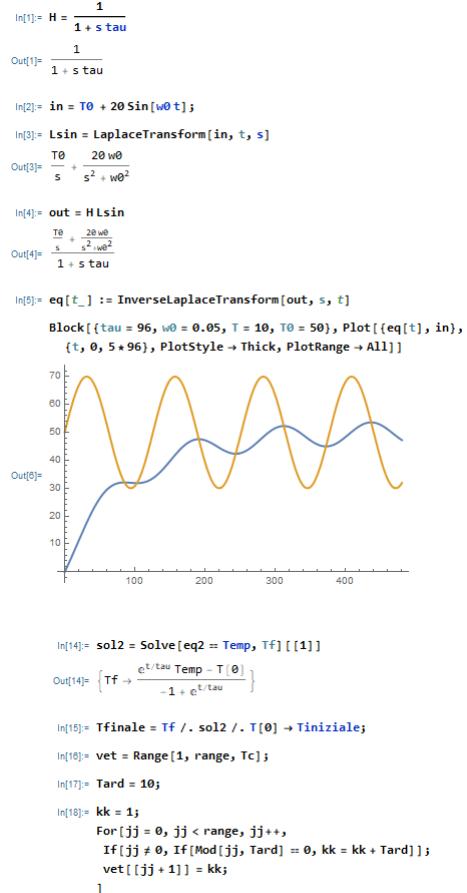
isoliamo  $T_f$ , che rappresenta la temperatura del fluido in ingresso e sostituiamo  $T_0$ , che corrisponde alla temperatura iniziale della Pt100, con la temperatura corrispondente al primo valore di ogni singolo intervallo.

Per farlo, inizializziamo un vettore, che poi indicherà la posizione del vettore **eq**, con valori che partono da 1 e incrementano di  $t_{int}$  ogni  $t_{int}$  volte (ovvero nel nostro caso: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 22, ecc...]).

Infine, come possiamo notare dalla figura sottostante, in ogni intervallo la temperatura in ingresso ricavata (rappresentata dai puntini rossi) ha andamento esponenziale, come ci aspettavamo, tuttavia non si distanzia molto dal valore atteso, rendendo il metodo accettabile e dunque valido.

Inoltre, desideriamo mostrare un valore stimato di temperatura per ogni intervallo; dunque, in accordo anche con le ipotesi, consideriamo l'ingresso come costante nell'intervallo e facendo una media di tutti i campionamenti dell'intervallo otteniamo una stima dell'ingresso.

In presenza di un segnale costante e in assenza di rumore, non potremmo dare un'esatta collocazione del segnale tra i due valori discreti che lo contengono. Tuttavia, grazie alla presenza del rumore che



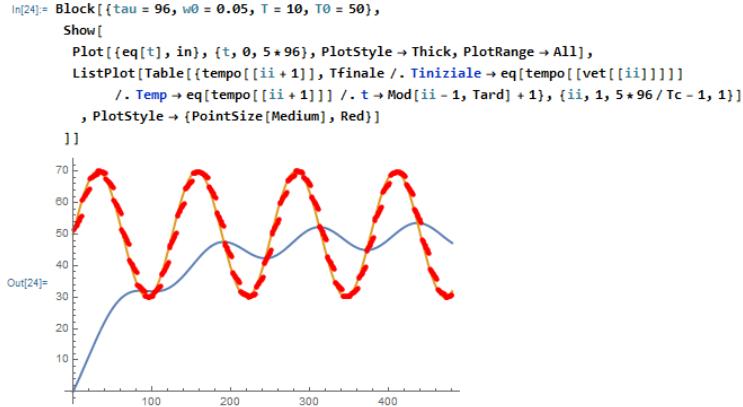
```

In[14]:= sol2 = Solve[eq == Temp, Tf][[1]]
Out[14]= {Tf -> e^(t/tau) Temp - T[0] / -1 + e^(t/tau) }

In[15]:= Tf finale = Tf /. sol2 /. T[0] -> Tiniziale;
In[16]:= vet = Range[1, range, Tc];
In[17]:= Tard = 10;
In[18]:= kk = 1;
For[jj = 0, jj < range, jj++,
  If[jj != 0, If[Mod[jj, Tard] == 0, kk = kk + Tard]];
  vet[[jj + 1]] = kk;
  ]

```

fa oscillare il valore del segnale da un valore discreto all'altro, facendo la media dei campioni in un determinato intervallo, otteniamo una stima della posizione all'interno dei due valori discreti che lo contengono, migliorandone la risoluzione.



### 8.3 Programma MATLAB

```
clc;
clear;

%Scelta della porta, DIPENDE DALLA PORTA DEL PC
porta = 'COM3';
ard = arduino(porta,'Nano3');

%Scelta del pin di lettura
pinAnalog = 'A0';

%Comando che permette il grafico in tempo reale
compensata = animatedline('Color', [0 1 1]);
reale= animatedline('Color', [1 0 0]);

%Parametri
tintervallo = 5; %tempo trascorso tra un punto e l'altro sul grafico online
tau = 96.0967; %parametro caratteristico della nostra sonda
```

Il programma utilizza la comunicazione seriale per ottenere i dati dall'USB ed elaborarli tramite MatLab.

Infatti, come si nota dal codice, è stato creato un oggetto (**ard**) che consente di stabilire una connessione con Arduino NANO.

In seguito sono stati creati altri due oggetti (**compensata** e **reale**), che ci consentono di creare un'animazione, che al momento della loro dichiarazione sono privi di dati.

```

for tempo = tintervallo:tintervallo:1000
    T_iniziale = 270.728 + 23.5705 * readVoltage(ard, pinAnalog) - 273.15; %ricavo temperatura
    % iniziale tramite relazione V-T nota
    tic; % inizio timer
    T_medio = 0; %media cumulativa in tempo reale delle temperature compensate
    b = toc;
    ii = 0;
    T = T_iniziale; %inizialmente T=Tiniziale
    while b < tintervallo
        b = toc; % controllo timer
        T = 270.728 + 23.5705 * readVoltage(ard, pinAnalog) - 273.15; %ricavo un nuovo istante di tempo
        T_finale = (T-T_iniziale*exp(-b/tau))/(1-exp(-b/tau)); %calcolo temperatura compensata,
        % la chiamiamo finale perchè si intende di fine transitorio

        %Non prendo i primi 55 campioni, faccio la media sui campioni
        %successivi
        if(ii>55)
            T_medio = ((T_medio*(ii-55)) + T_finale)/(ii - 54);
        end
        ii = ii + 1;
    end

    %proprietà grafico
    axis([0,tempo,0,100])
    addpoints(compensata,tempo,T_medio)
    addpoints(reale,tempo,T);
    readVoltage(ard, pinAnalog);
    drawnow limitrate;

```

Come anticipato dalla precedente sezione, deve essere mostrato un valore di temperatura stimata ad ogni intervallo e per farlo viene creato un ciclo che inizializza la variabile **tempo**. Tale variabile, riporta il tempo trascorso da quando è stato prelevato il primo dato da Arduino.

Pertanto, siccome per ogni ciclo passa un intervallo, la variabile deve essere incrementata della quantità  $Tc * intervallo$ , dove **intervallo** è la variabile che rappresenta il numero di campionamenti necessari a formare un intervallo.

A questo punto viene letto il voltaggio dal pin di Arduino e viene convertito in un valore di temperatura in gradi celsius, dopodiché viene resettato il timer **a** grazie al comando *tic* e inizializzata a zero la variabile **t**, che rappresenta il tempo trascorso dall'inizio di ogni intervallo.

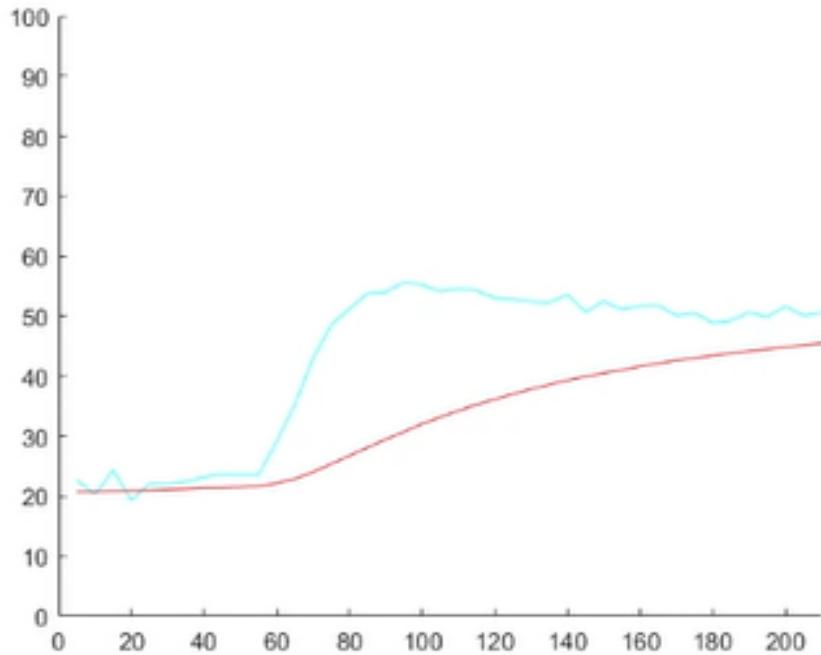
Successivamente, nel ciclo while viene reiterato il controllo del timer precedentemente avviato fino a che quest'ultimo raggiunga il valore di tempo di campionamento **Tc**.

Verificatasi questa condizione, viene resettato il timer, viene incrementato **t** di **Tc**. Viene poi applicata l'equazione 6 per trovare il valore approssimato di temperatura del fluido (**Tfinale**) e ne viene fatta una media con tutti gli altri valori di **Tfinale**.

Infine, vengono aggiunti all'animazione i valori di temperatura compensata e di quella reale della Pt100 e viene mostrato il grafico.

## 8.4 Risultati

La compensazione dinamica effettuata con questo metodo è riuscita, e nella foto qui sotto vi è un esempio



Sull'asse delle ascisse è presente il tempo [s], mentre sull'asse delle ordinate si trova la temperatura [C°]. Circa dopo 60 secondi la sonda Pt100 è stata messa nel fornelletto, sottoponendola dunque ad un gradino di temperatura.

La curva rossa rappresenta la temperatura della Pt100, che sale lentamente fino ad arrivare al valore di assestamento, ossia la temperatura del fornelletto (circa 50 gradi).

In azzurro è rappresentata la temperatura compensata, risultato del codice descritto sopra.

Come si può notare, la temperatura compensata già dopo 20 secondi riesce a predire abbastanza accuratamente la temperatura del fornelletto, mentre la sonda Pt100 ci impiega almeno altri 3 minuti.

## 9 Metodo 2

### 9.1 Introduzione

L'ultimo metodo utilizzato per effettuare la compensazione dinamica si basa, come il primo metodo (Capitolo 7.1), sull'utilizzo della trasformata di Laplace, ma con un approccio differente.

### 9.2 Descrizione

Come ricavato nel Capitolo 1, la funzione di trasferimento che lega l'ingresso e l'uscita del nostro sistema è la seguente:

$$H(\omega) = \frac{T(\omega)}{T_f(\omega)} = \frac{1}{1 + i\omega \frac{Mc}{Ah}} \quad (21)$$

Ricordo inoltre che al posto di  $\frac{Mc}{Ah}$  si può sostituire  $\tau$ .

La funzione di trasferimento diventa quindi la seguente:

$$H(\omega) = \frac{T(\omega)}{T_f(\omega)} = \frac{1}{1 + i\omega\tau} \quad (22)$$

Isolando quindi la variabile di interesse, ossia la temperatura finale  $T_f(\omega)$  otteniamo:

$$T_f(\omega) = T(\omega) + \tau i\omega T(\omega) \quad (23)$$

Nel dominio di Laplace, moltiplicare la variabile  $T(\omega)$  per *iomega*, corrisponde, nel dominio del tempo, a fare la derivata di  $T(\omega)$ .

Trasformando nuovamente l'equazione precedente e tornando quindi nel dominio del tempo, otteniamo:

$$T_f(t) = T(t) + \tau \frac{d}{dt} T(t) \quad (24)$$

Tra i vari modi di effettuare una derivata, noi abbiamo scelto il più semplice, che poi si è rivelato anche essere il più efficace. La derivata di  $T(t)$  viene svolta semplicemente campionando 2 valori di  $T(t)$  e dividendo la loro differenza per il tempo di campionamento  $t_c$ .

$$T_f(t) = T(t) + \tau \frac{d}{dt} T(t) = T(t) + \tau \frac{T(t + t_c) - T(t)}{t_c} \quad (25)$$

### 9.3 Programma MATLAB

```
clc;
clear;

%Scelta della porta, DIPENDE DALLA PORTA DEL PC
porta = 'COM3';
ard = arduino(porta, 'nano3');

%Scelta del pin di lettura
pinAnalog = 'A0';

%Comando che permette il grafico in tempo reale
compensata = animatedline('Color',[0 1 1]);
reale = animatedline('Color', [1 0 0]);

%Parametri:
tintervallo = 3; %tempo di campionatura
tau = 96.0967; %parametro caratteristico della nostra sonda
```

La connessione tramite porta seriale e l'inizializzazione delle variabili è identica a quella del Metodo 1 (Capitolo 8).

```
%Ricavo la temperatura iniziale
T_iniziale = 270.728 + 23.5705 * readVoltage(ard, pinAnalog) - 273.15;
pause(tintervallo); %attendo tintervallo definito sopra
T = 270.728 + 23.5705 * readVoltage(ard, pinAnalog) - 273.15; %ricavo il secondo campione
% di temperatura con relazione T-V nota
T_finale = T + tau * ((T - T_iniziale)/tintervallo); %calcolo il primo valore compensato
% sulla base delle ultime 3 righe

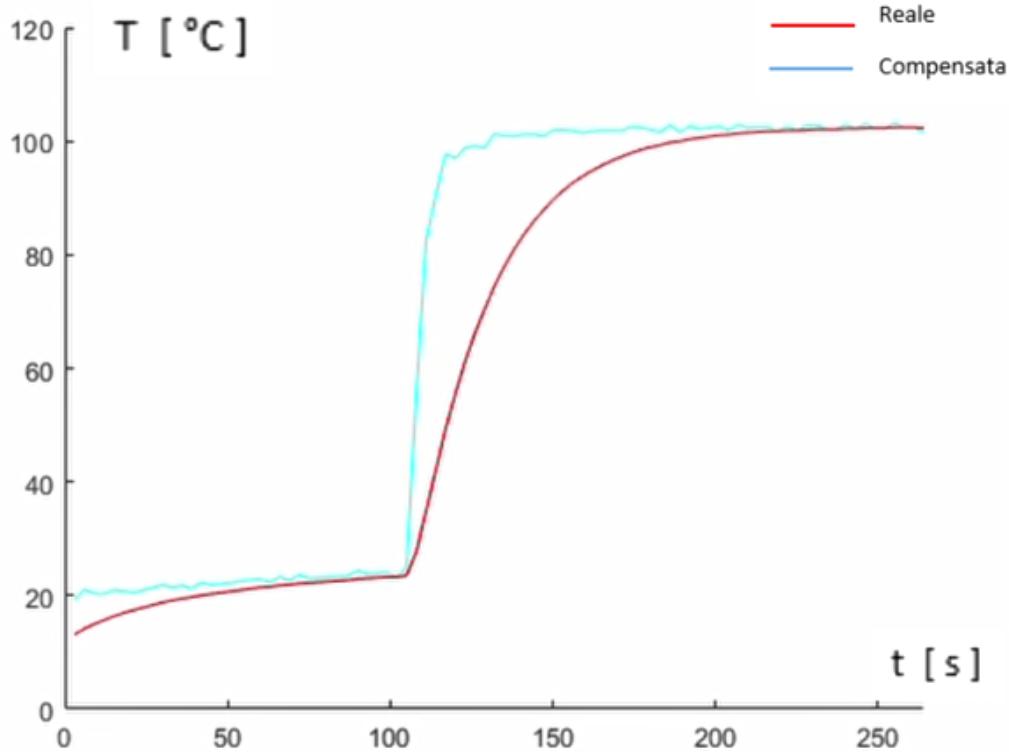
%Creo un ciclo per svolgere il tutto in loop
for tempo = tintervallo:tintervallo:1000
    T_iniziale = T;
    pause(tintervallo)
    T = 270.728 + 23.5705 * readVoltage(ard, pinAnalog) - 273.15;
    T_finale = T + tau * ((T-T_iniziale)/tintervallo);

    %definisco caratteristiche del grafico
    axis([0,tempo,0,100])
    addpoints(compensata,tempo,T_finale)
    addpoints(reale,tempo,T);
    drawnow limitrate
end
```

Nelle prime 8 righe di codice, semplicemente utilizzo l'equazione prima determinata (25) per determinare il primo dato da cui iniziare. Nel ciclo for sottostante, semplicemente itero questa operazione. Nella parte finale del codice, stampo sul grafico i risultati.

## 9.4 Risultati

La compensazione dinamica effettuata con questo metodo ha funzionato, e qui sotto ne sono mostrati i risultati:



Circa dopo 100 secondi la sonda Pt100 è stata messa nel fornelletto, sottoponendola dunque ad un gradino di temperatura.

La curva rossa rappresenta la temperatura della Pt100, che sale lentamente fino ad arrivare al valore di assestamento, ossia la temperatura del fornelletto (circa 100 gradi).

In azzurro è rappresentata la temperatura compensata, risultato del codice descritto sopra.

Come si può notare, la temperatura compensata già dopo 20 secondi riesce a predire abbastanza accuratamente la temperatura del fornelletto, mentre la sonda Pt100 ci impiega almeno altri 3 minuti.

## 10 Conclusioni

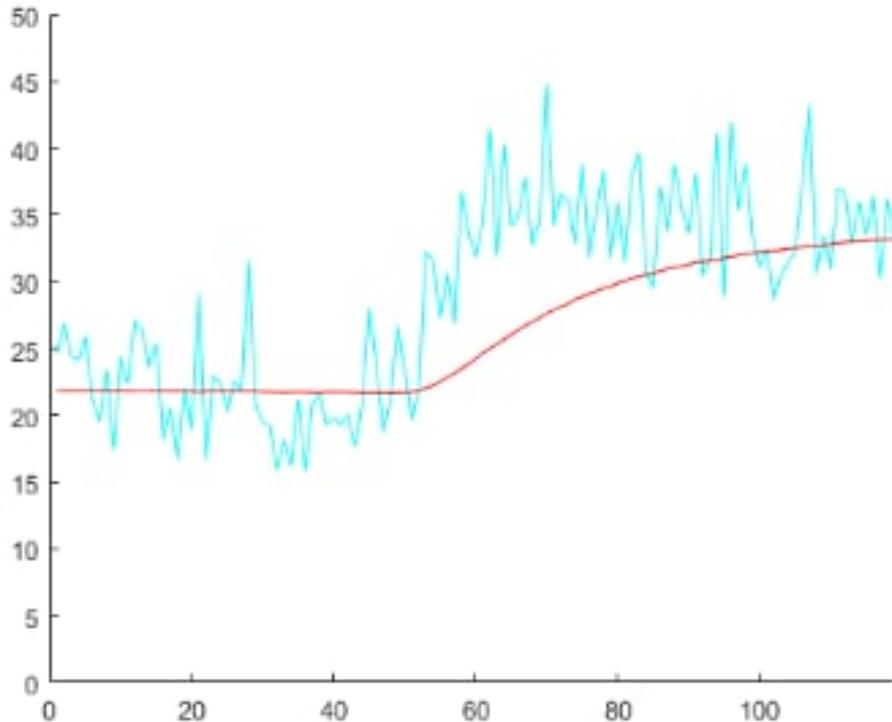
Come si può notare dai grafici dei risultati prima mostrati, le curve delle temperature compensate (linee azzurre), appaiono, anche se poco, "sporcate" da un rumore in alta frequenza.

Il rumore in alta frequenza è rimasto, seppur in misura molto minore, anche dopo aver messo un filtro passa-basso dopo l'uscita del circuito.

Il filtro in questione è un filtro R-C del primo ordine, con frequenza di taglio a 10 Hz.

Senza il suddetto filtro, la temperatura della Pt100 veniva registrata con un rumore piuttosto elevato, e ciò causava delle oscillazioni piuttosto brusche della curva della compensata.

Questo fenomeno si può notare bene da questo grafico:



Il rumore, come si può notare dai grafici dei risultati, non è scomparso completamente.

Questo a causa dell'inversione della dinamica.

La Pt100, infatti, appare nel dominio di Laplace come un filtro passa-basso:

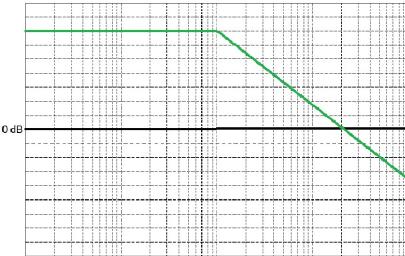
$$H(\omega) = \frac{T(\omega)}{T_f(\omega)} = \frac{1}{1 + i\omega\tau} \quad (26)$$

Come ci si aspetta, infatti, la Pt100 riesce a seguire delle oscillazioni di temperatura a basse frequenze, mentre non fa passare tutte le oscillazioni di temperatura in alta frequenza.

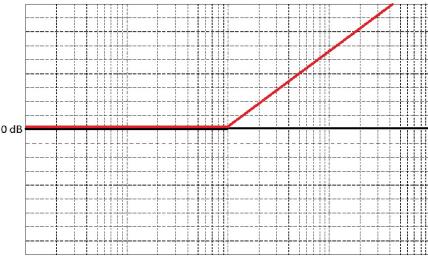
Quando si effettua l'inversione della dinamica (compensazione dinamica), come si vede dall'equazione sottostante, si va a moltiplicare la variabile misurata  $T(\omega)$  per l'inverso di  $\frac{1}{1 + i\omega\tau}$ , ossia  $1 + i\omega\tau$ .

$$T_f(\omega) = T(\omega)(1 + i\omega\tau) \quad (27)$$

Nel dominio di Laplace,  $1 + i\omega\tau$  è infatti l'inverso di un filtro passa-basso (non è un filtro passa-alto), come si può notare dall'immagine sottostante.



Filtro passa-basso



Inverso di un filtro passa-basso

Moltiplicare la variabile misurata  $T(\omega)$  per  $1 + i\omega\tau$  va quindi ad amplificare, con un guadagno molto elevato, le alte frequenze di  $T(\omega)$ .

Tra le varie soluzioni proposte per risolvere questo problema, la più accreditata consiste nell'utilizzare un filtro R-C di ordine superiore (2,3 o 4).

Con un filtro di ordine superiore, infatti, la tensione letta da Arduino risulta più pulita dal rumore in alta frequenza.

Con meno rumore in alta frequenza, durante l'inversione della dinamica, le alte frequenze, che prima venivano amplificate notevolmente, sono ora quasi assenti, e anche se amplificate non sono quindi così rilevanti.

Il problema di questa soluzione è che, utilizzando un filtro di ordine superiore, esso non può più essere trascurato durante l'inversione della dinamica.

L'inversione della dinamica effettuata finora, infatti, prevedeva la compensazione della sola sonda Pt100, perché ipotizzavamo che il filtro da noi utilizzato non avesse troppa influenza sul segnale.

Se avessimo usato un filtro piuttosto invasivo (2°, 3° o 4° ordine), bisognerebbe considerare anche quest'ultimo nella compensazione dinamica.

Nell'equazione sottostante si considera l'inversione della dinamica della Pt100 associata ad un filtro passa-basso del 2° ordine con costante di tempo  $\tau_f$ .

$$T_f(\omega) = T(\omega)(1 + i\omega\tau)(1 + i\omega\tau_f)^2 \quad (28)$$

Invertire la dinamica di un'ulteriore filtro passa-basso amplificherebbe ancor di più le alte frequenze. Se da un lato, quindi, un filtro di ordine maggiore di 1 filtra meglio le alte frequenze, durante l'inversione della dinamica le va ad amplificare notevolmente, come visto prima.

Il filtro scelto da noi rappresenta quindi un compromesso tra rimozione delle alte frequenze e amplificazione delle stesse durante l'inversione della dinamica: la copertura è corta.