UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DESIGN DI UN SISTEMA DI CARATTERIZZAZIONE DEGLI ATTUATORI DI UN DRONE

DAL CORTIVO ANDREA
TOMMASETTO ANDREA
CONTI ANDREA

DESCRIZIONE DEL PROGETTO:

L'interesse verso il mondo delle robotica aerea è in costante aumento e l'interesse tecnologico è volto a rendere i droni dei veicoli sempre più autonomi. Al fine di implementare algoritmi di controllo automatico efficaci, tuttavia, è necessario conoscere al meglio le caratteristiche aerodinamiche dei droni e le loro capacità in termini di attuazione.

Lo scopo del progetto è quello di progettare un banco di prova che permetta di caratterizzare le prestazioni degli attuatori (motori + eliche) dei droni.

Si sarà quindi in grado di:

- Impostare la velocità del motore affinché questo sviluppi una determinata forza verticale ed una determinata coppia di

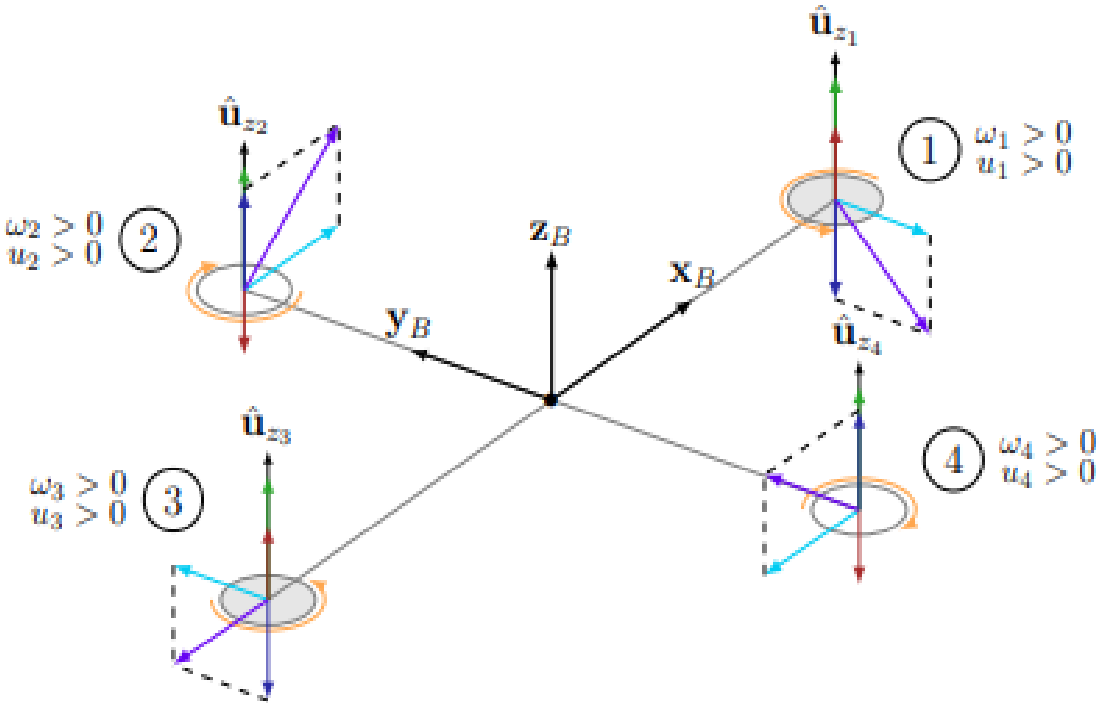
FASI DEL PROGETTO:

1. Studio della dinamica dei droni, con particolare attenzione alla dinamica degli attuatori al fine di capire la relazione tra le grandezze di ingresso e uscita di interesse.
2. Design del banco di prova: scelta delle componenti e dei materiali
3. Realizzazione del banco prova (hardware e software)
4. Sviluppo degli algoritmi per la lettura e l'analisi dei dati
5. Validazione del sistema

STUDIO DELLA DINAMICA DEI DRONI

L’elica di un motore trasforma la potenza fornita dal motore in una spinta aerodinamica attraverso l’azione di forze che si generano durante la rotazione. Tra le principali forze ci sono la **spinta** e la **torsione**. In un drone ogni rotore produce una coppia, e per questo ogni rotore ha bisogno di un altro rotore che abbia una rivoluzione opposta per compensare il primo.

ANALISI DELLE FORZE

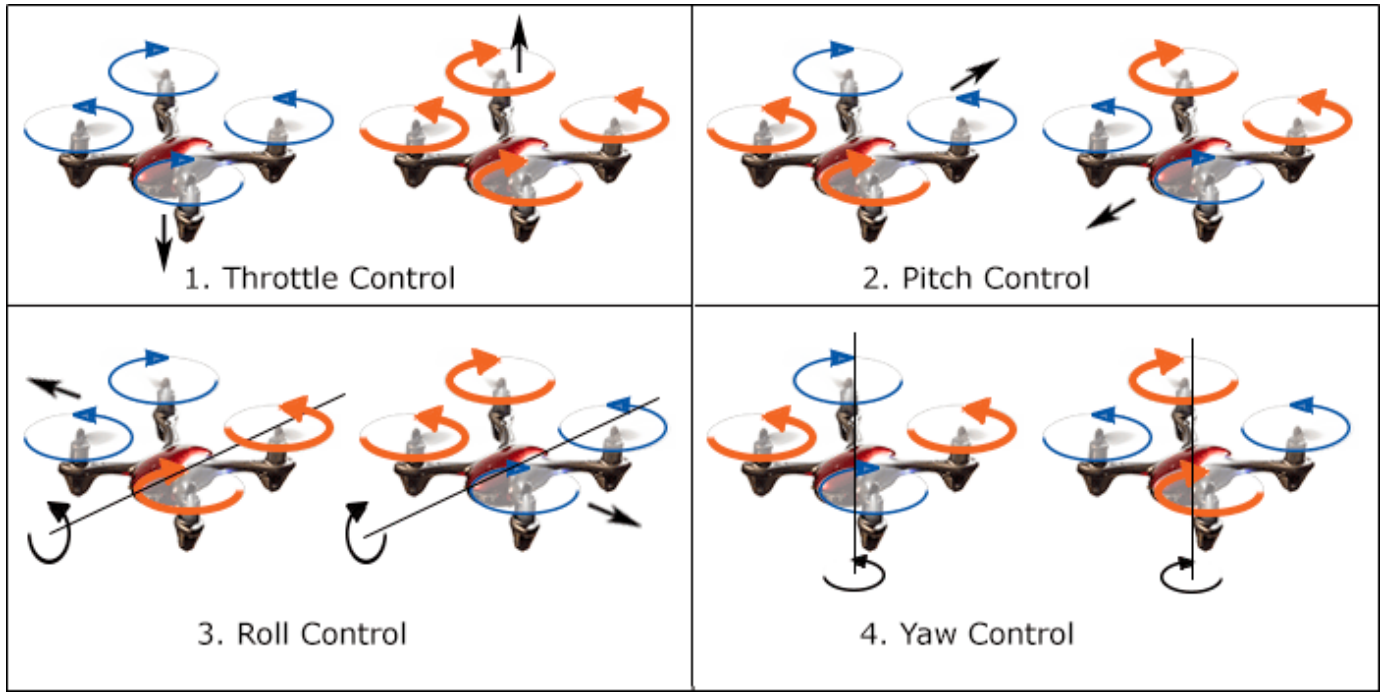


Nell'immagine qui a fianco si vede lo schema delle forze che agiscono su un rotore. In verde abbiamo la spinta che deve sempre essere verso l'alto, quindi avendo motori CW e CCW questo dipendo dalla scelta dell'elica. Mentre in viola abbiamo la sommatoria del momento di drag e del momento di spinta: $T_i = T_i^t T_i^d$.

Per avere una situazione corretta e di stabilità:

$$\sum_{i=1}^{n} T_i = 0$$

ANALISI DEI MOVIMENTI



E’ possibile ottenere quattro possibili movimenti del drone andando a variare la velocità dei motori.

DISPOSITIVI E SOFTWARE UTILIZZATI NEL PROGETTO BANCO PROVA

DISPOSITIVI:

- Brushless Motor : Tarot 4108 380kv 6S, 2206-kv2300
- ESC: Tekko F4 35 A
- Celle di Carico: TAL220
- Arduino Mega e Uno
- Display Nextion
- Display lcd
- Scheda acquisizione N.I. PCIe-6321

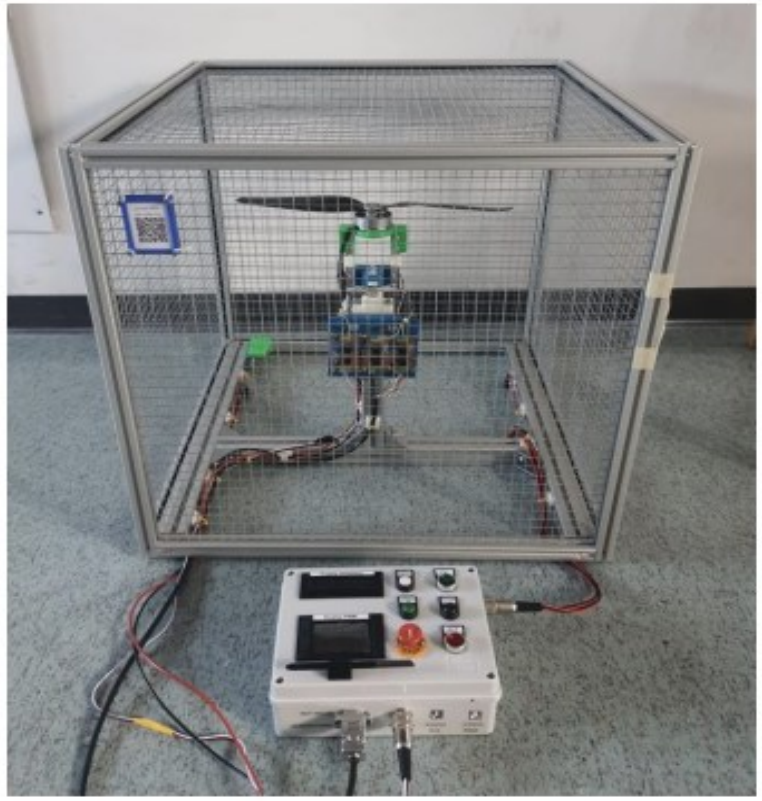
SOFTWARE:

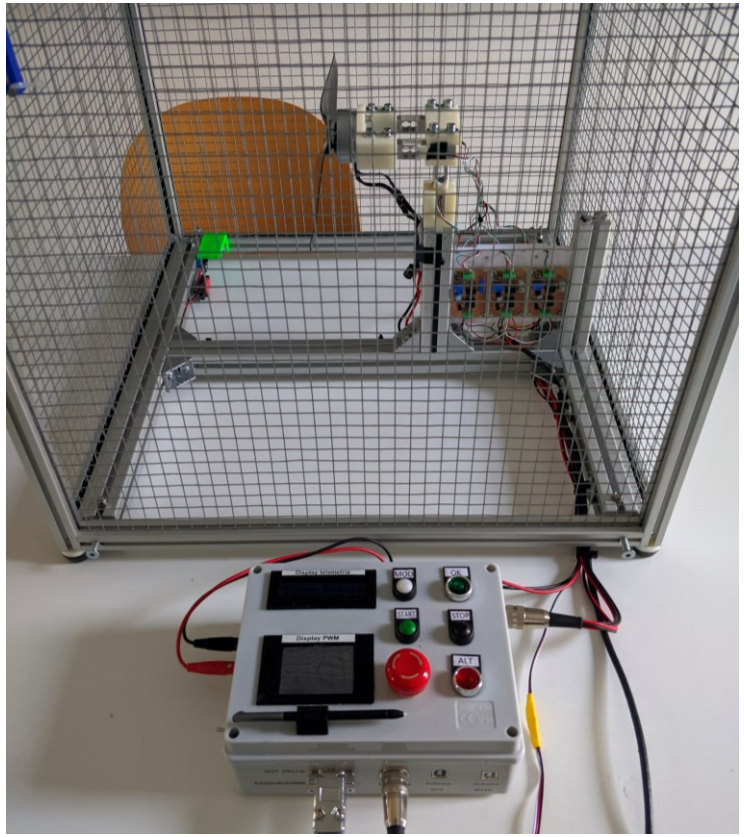
- Matlab, Simulink
- Arduino Ide
- Fusion 360, Solidworks
- Cura
- Kicad
- Falstad
- Blhelisuite32

MATLAB ARDUINO AUTODESK® FUSION 360™ SOLIDWORKS BLHeli_32

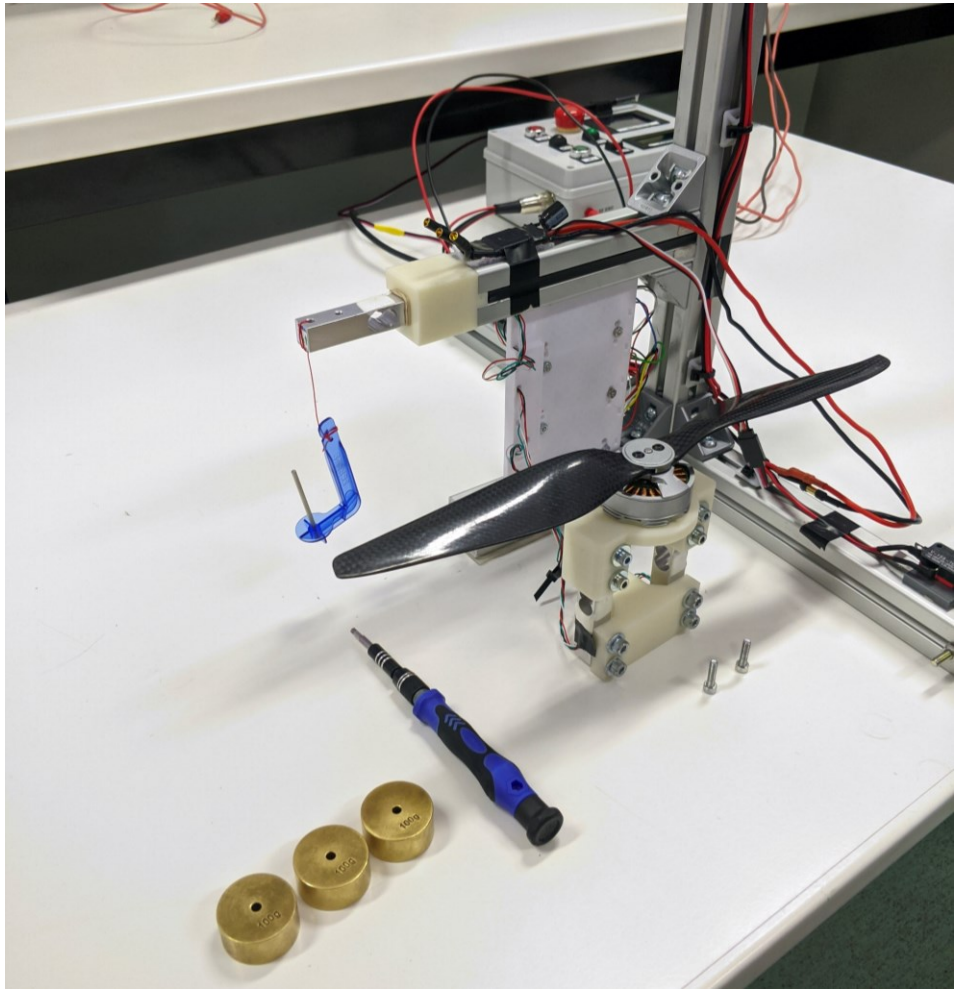
DESIGN DEL BANCO PROVA

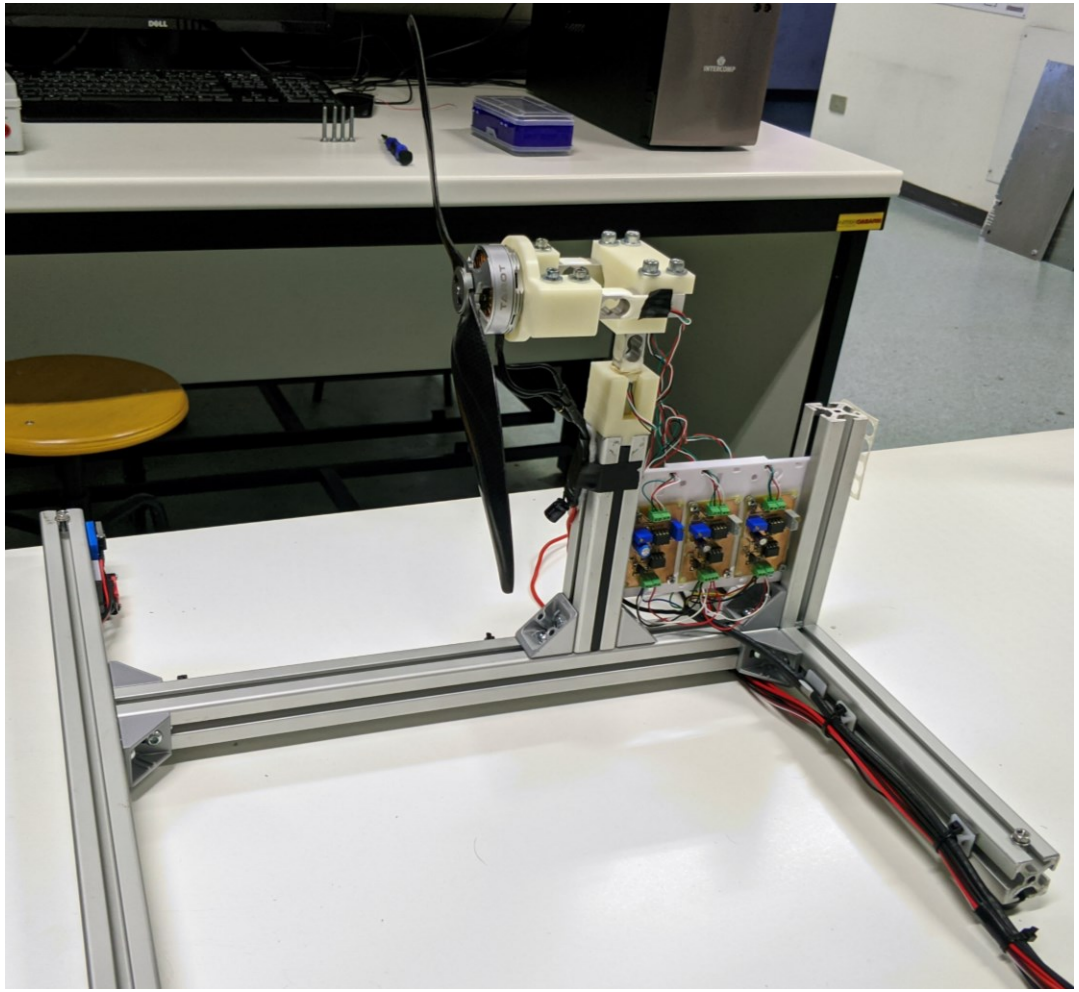
La struttura del banco prova è stata oggetto di modifiche su più fronti, con l'obiettivo principale di minimizzare le turbolenze durante la fase di prova.





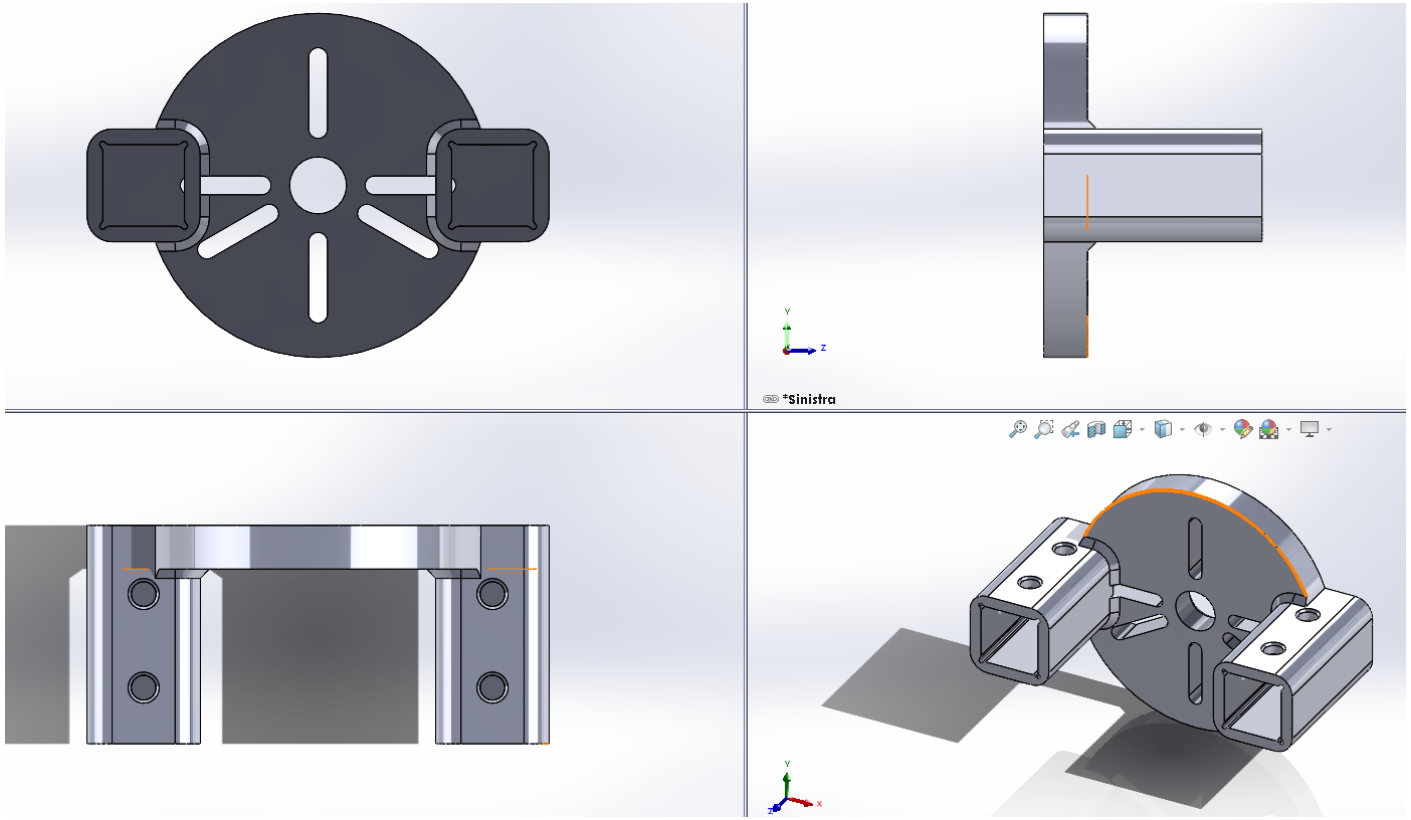
DESIGN DEL BANCO PROVA



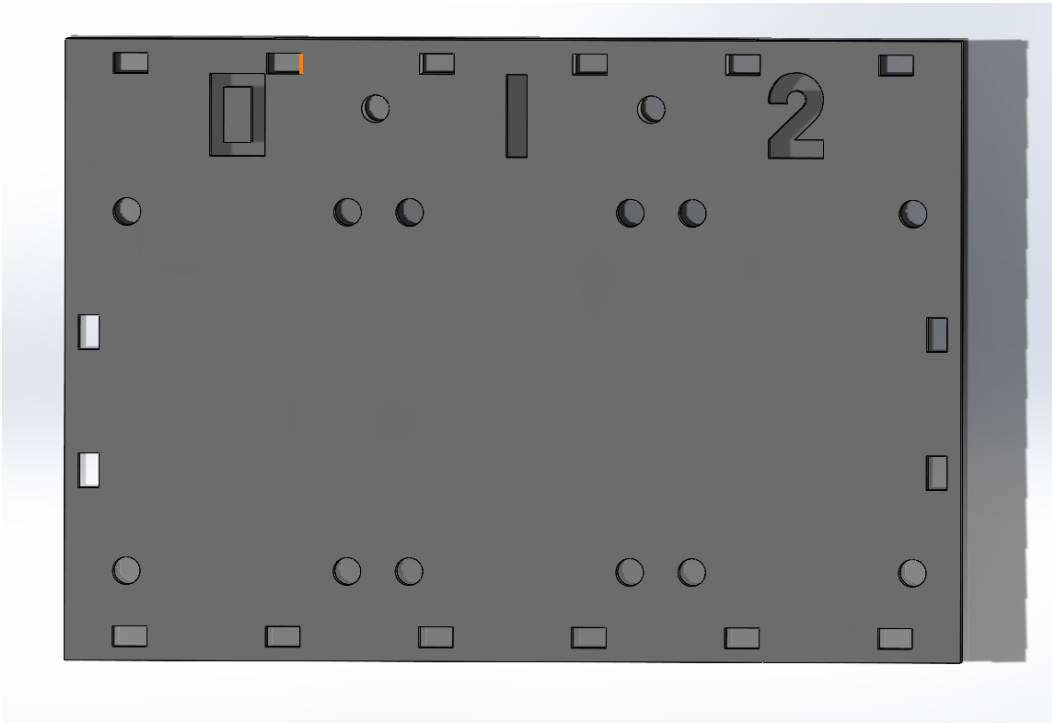


REALIZZAZIONE COMPONENTI 3D

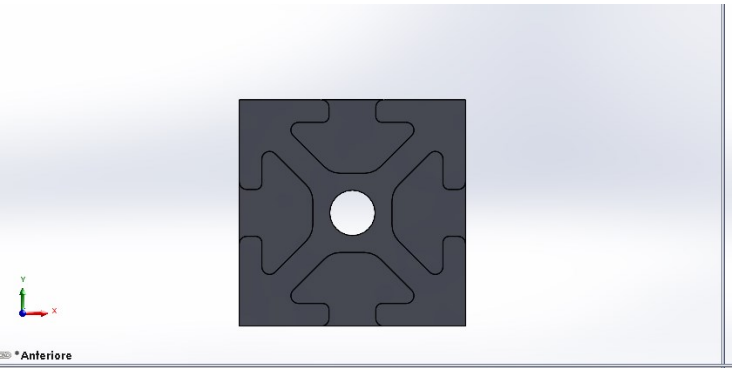
I componenti di giunzione tra telaio - celle di carico e celle di carico – attuatore sono stati riprogettati, pur mantenendo il concept originale, al fine di renderli più resistenti. In aggiunta, è stato fatto un disegno di assieme della struttura che risulta particolarmente utile durante la fase di progettazione di eventuali modifiche alla struttura

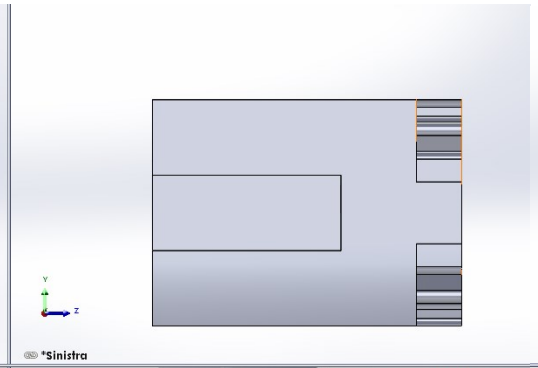


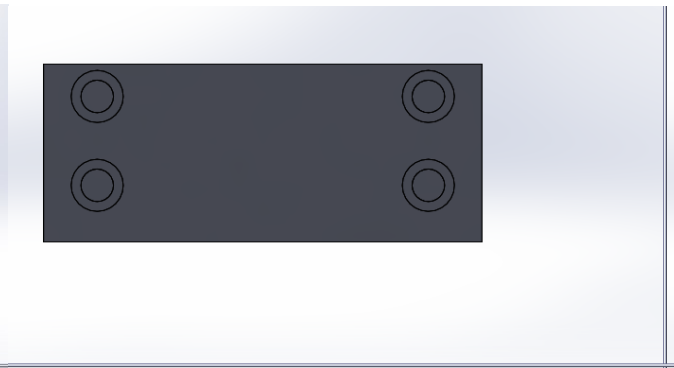
Supporto motore

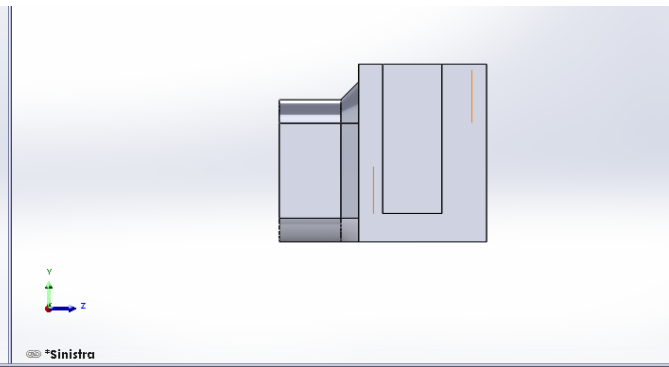


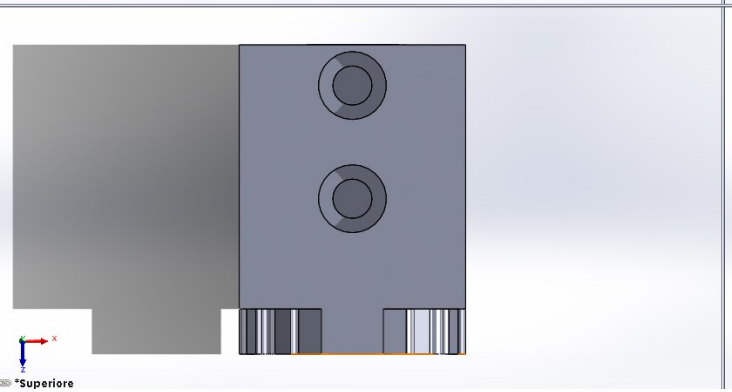
- **REALIZZAZIONE COMPONENTI 3D**

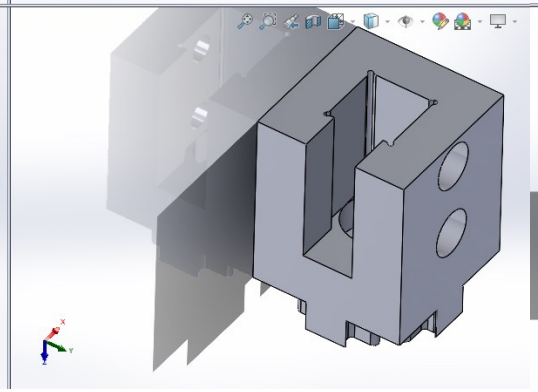


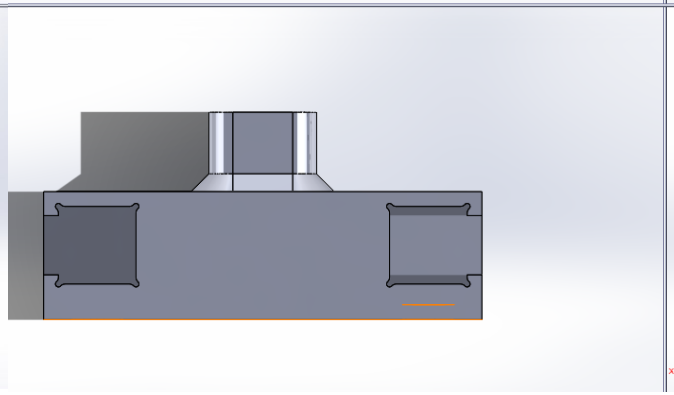


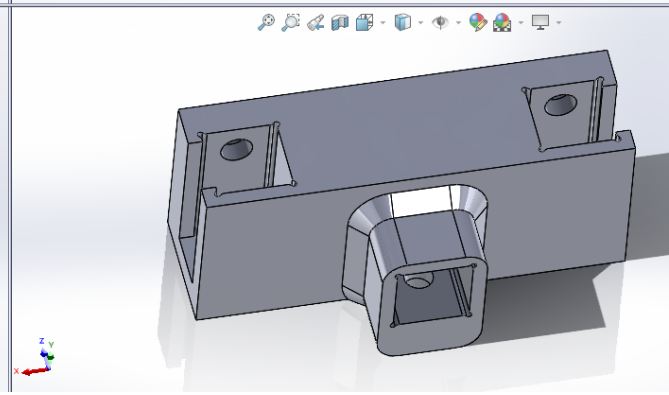








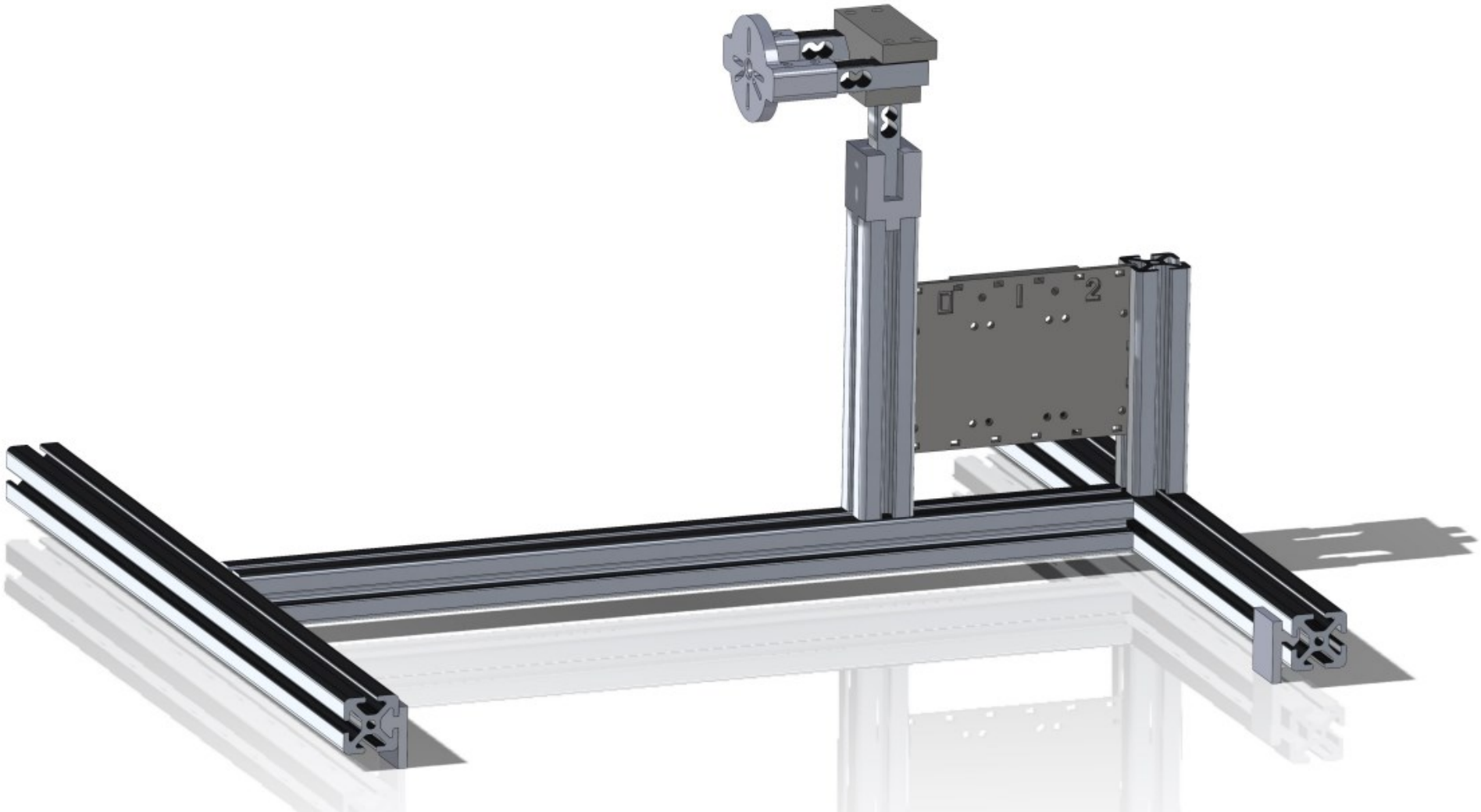




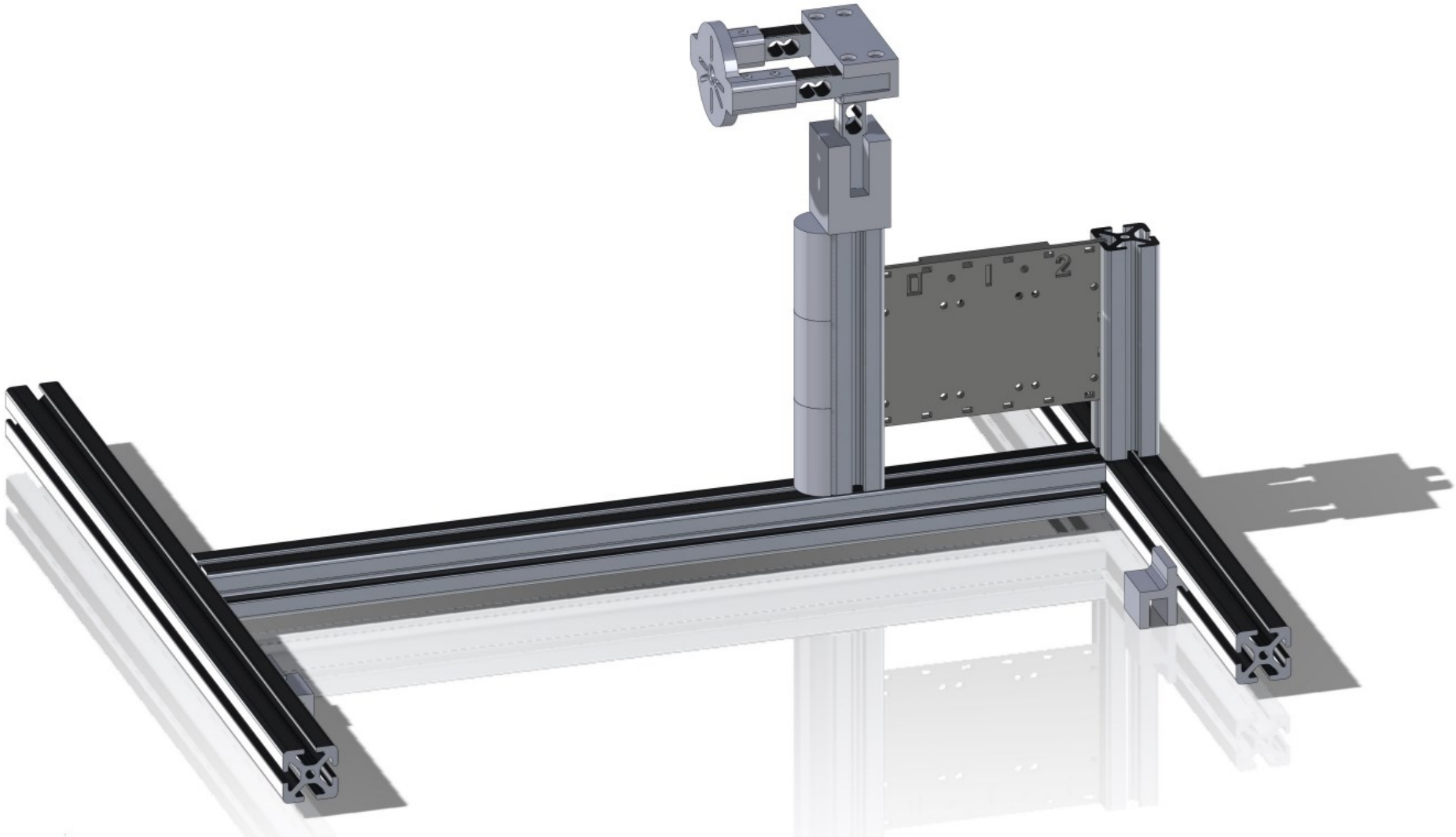
Supporto cella di carico 0

Supporto cella di carico 1 e2

- **ASSIEME 3D DELLA STRUTTURA**

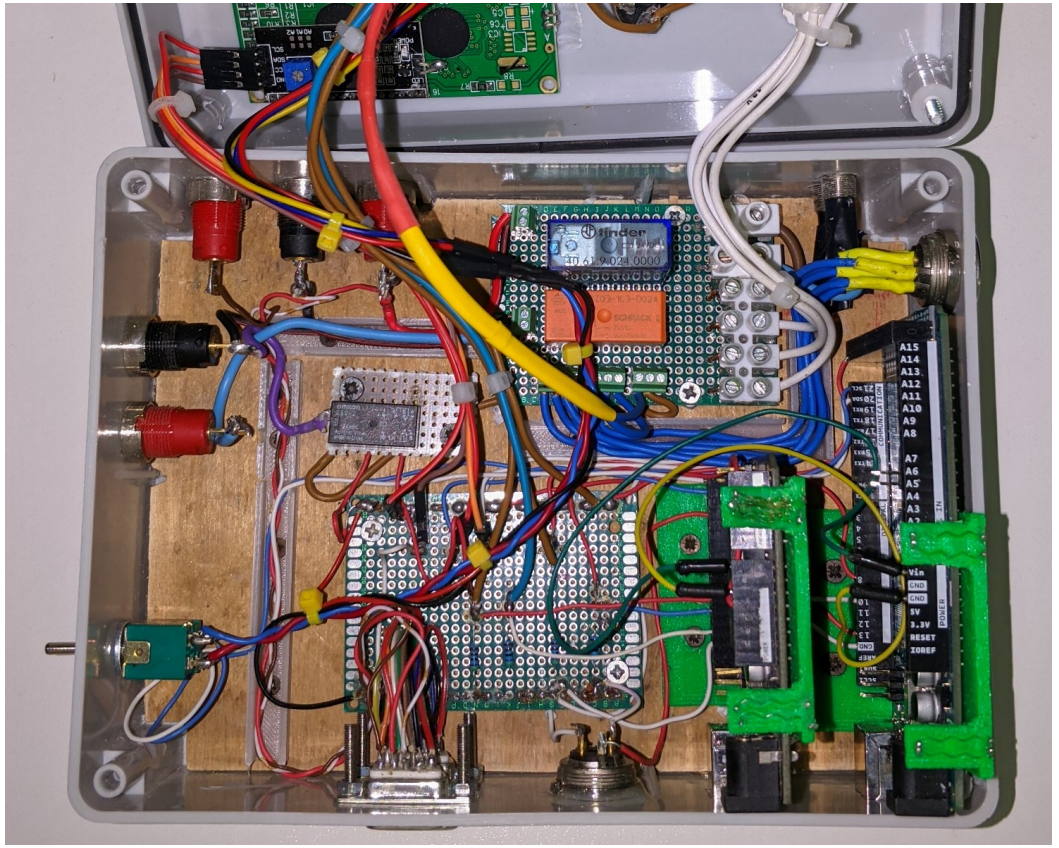


- **ASSIEME 3D DELLA STRUTTURA**



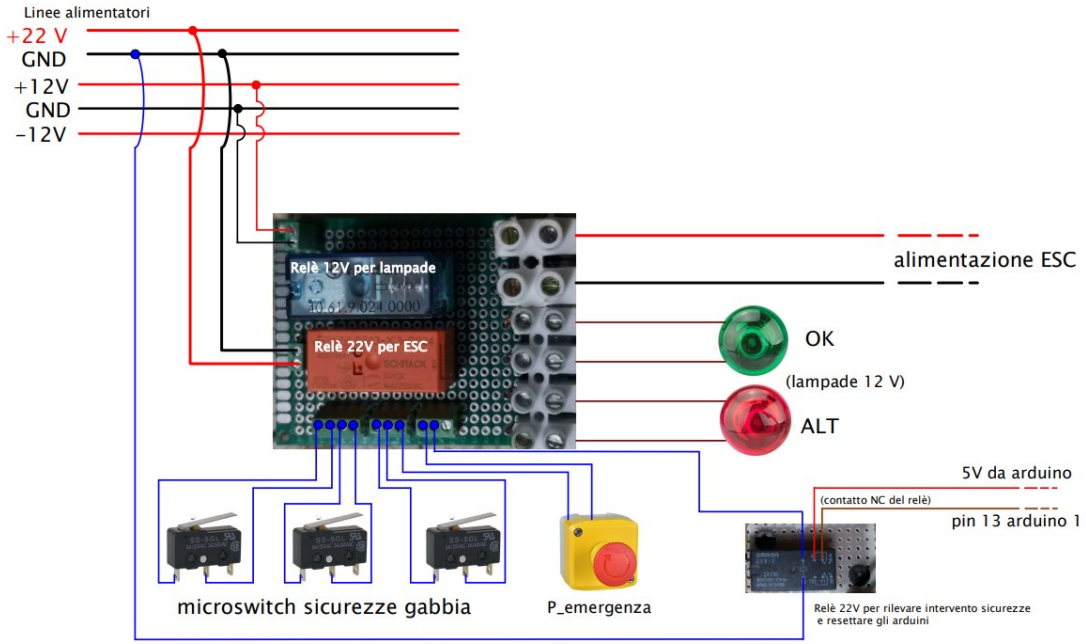
Quadro elettrico

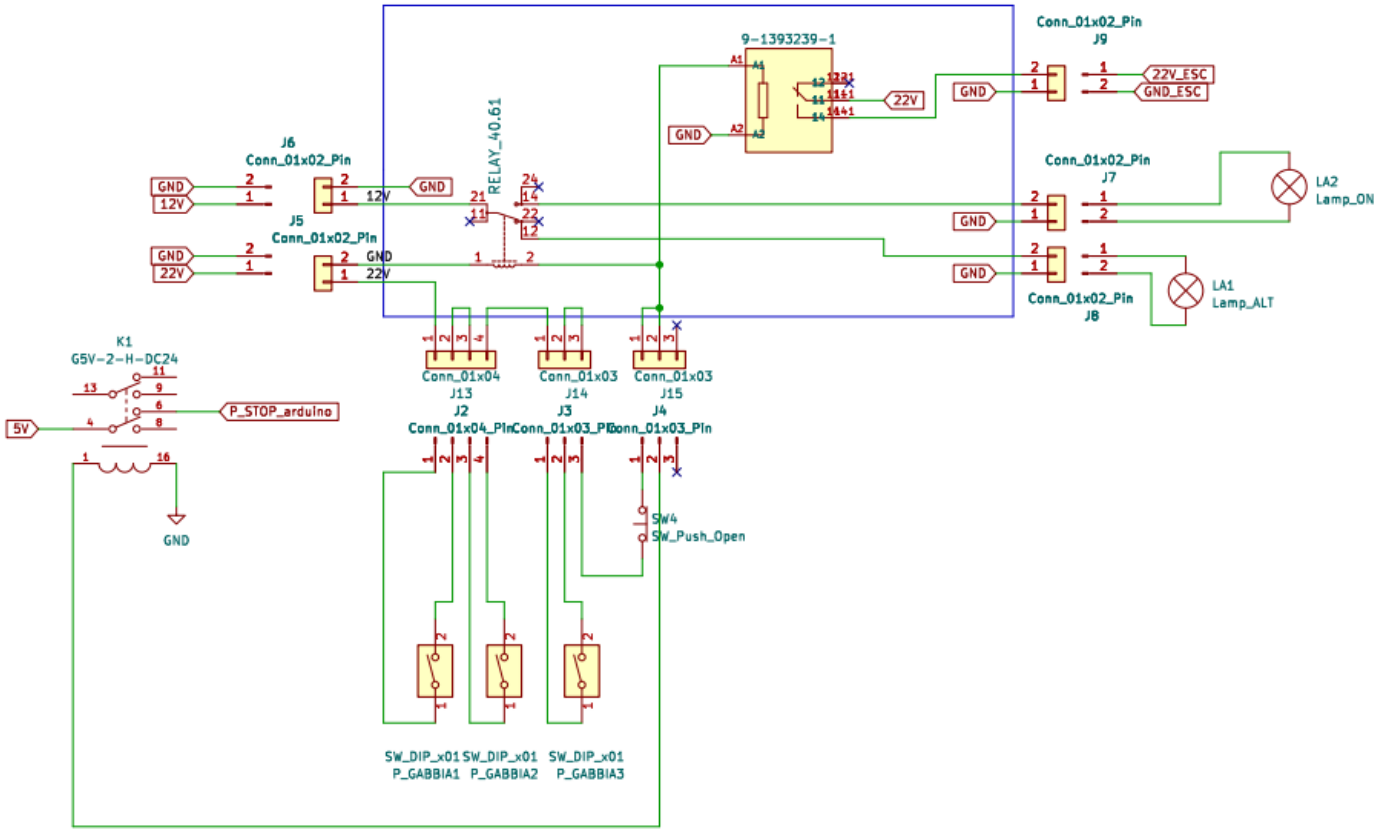
Il quadro elettrico non è stato soggetto a modifiche significative, ma per aumentare la sua affidabilità, alcuni cavi elettrici sono stati sostituiti e sono state aggiunte delle guaine termo-restringenti dove necessario. Inoltre, il tasto di stop di emergenza è stato sostituito con un modello commerciale.



Schemi elettrici

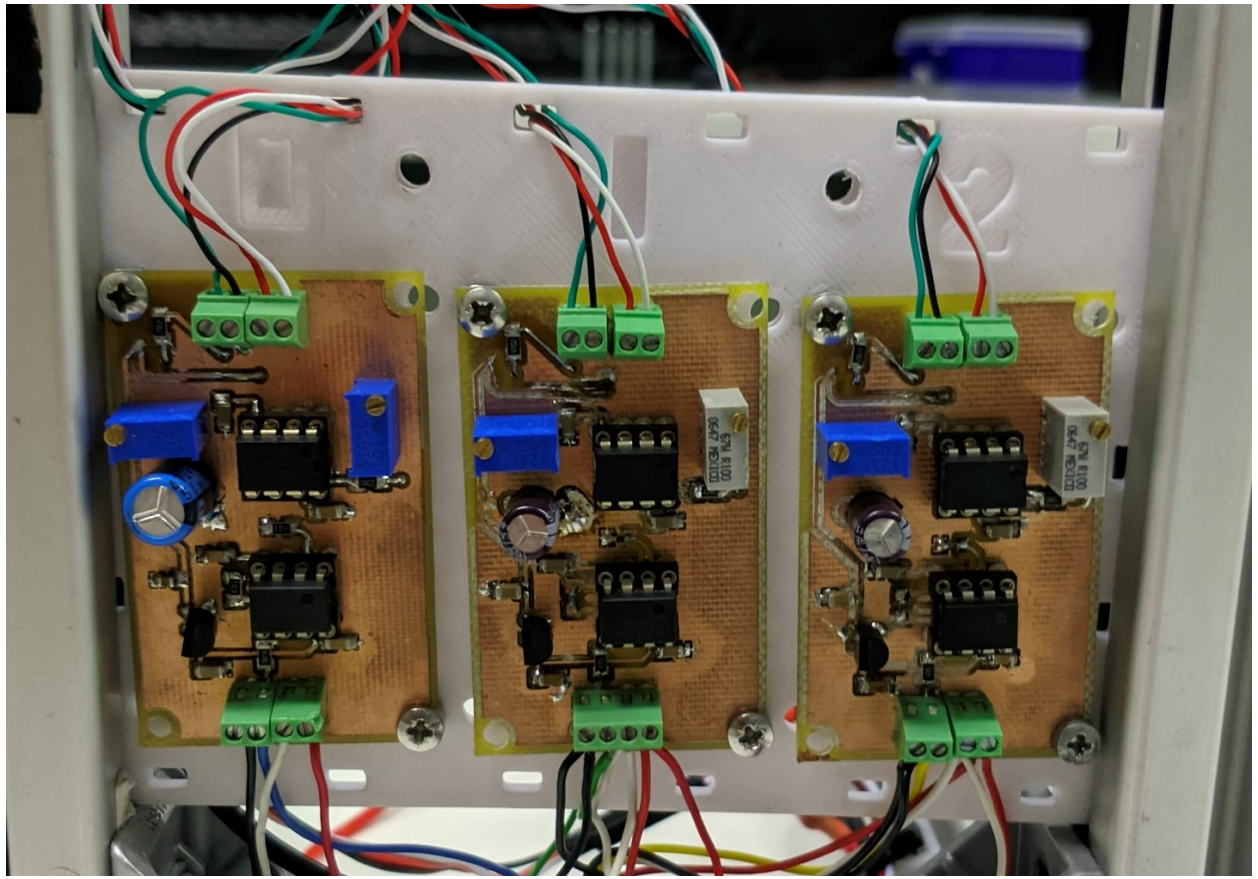
Nel quadro elettrico sono stati individuati diversi problemi nei circuiti di alimentazione e sicurezza. In particolare, la risoluzione del problema di sicurezza non è stata immediata a causa della presenza di uno schema elettrico poco dettagliato e a volte impreciso rispetto alla realtà. Per ovviare a questo problema, abbiamo creato degli schemi elettrici formali, con collegamenti più chiari per l'intero banco prova.





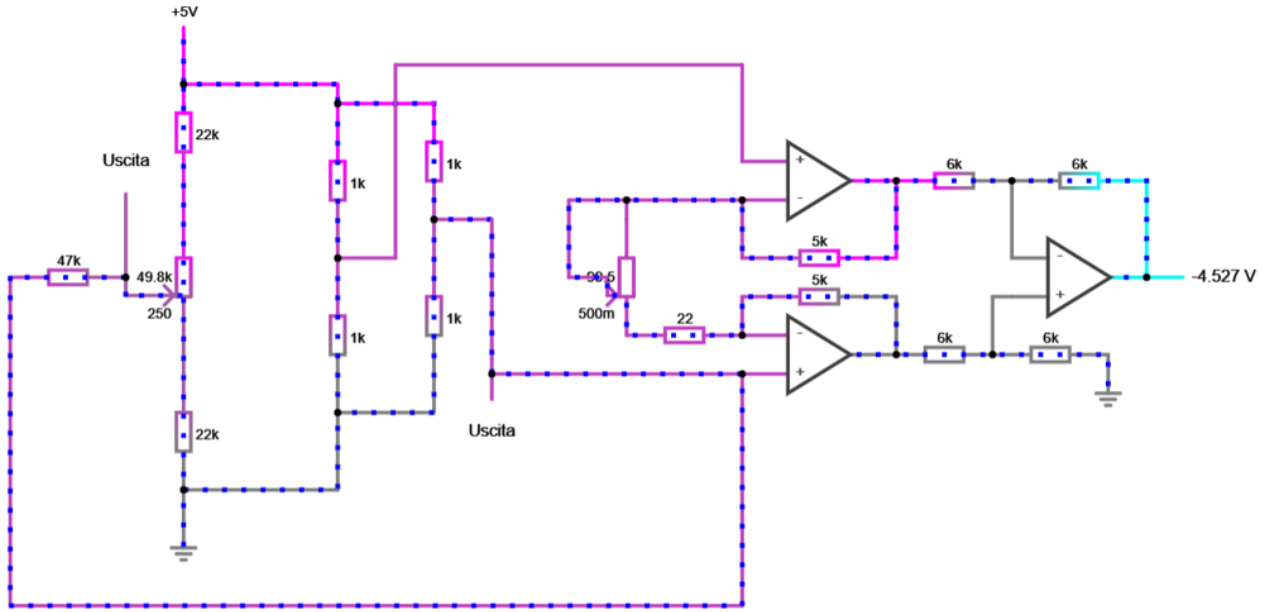
CIRCUITO AMPLIFICAZIONE SEGNALE DELLE CELLE DI CARICO

Per semplificare l'identificazione dei problemi e la rimessa in funzione del banco prova, abbiamo adottato una configurazione con tre schede di amplificazione del segnale delle celle di carico, ciascuna indipendente dalle altre. Inoltre, stiamo completando una quarta scheda a scopo di ridondanza.



CIRCUITO AMPLIFICAZIONE SEGNALE DELLE CELLE DI CARICO

Abbiamo eseguito una simulazione approfondita del circuito di amplificazione delle celle per analizzare gli effetti del partitore di tensione regolabile tramite trimmer. Questa regolazione consente di spostare l'offset della misurazione rilevata dalla cella di carico. Nella configurazione attuale, siamo in grado di regolare l'offset in un range compreso tra +4,5V e -4,5V (con amplificazione massima 500V/V).



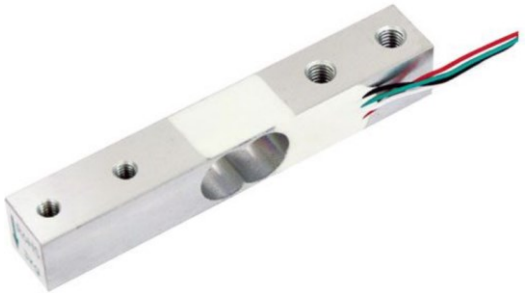
GAIN (V/V)	GAIN (dB)	R_G (Ω)
1	0	NC(1)
2	6	10000
5	14	2500
10	20	1111
20	26	526
50	34	204
100	40	101
200	46	50
500	54	20
1000	60	10
2000	66	5

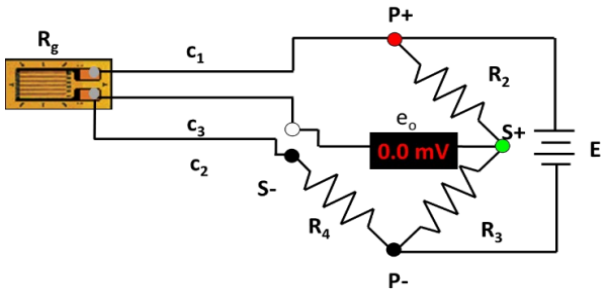
NOTE: (1) NC = No Connection.

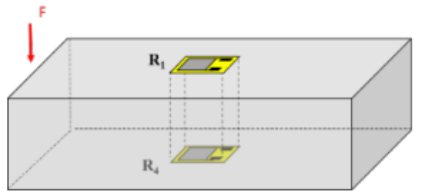
MISURA DI DEFORMAZIONE

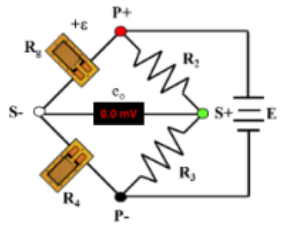
L'estensimetro è utilizzato per misurare la deformazione della cella. È possibile considerarlo come una resistenza variabile. Il legame tra deformazione e variazione di resistenza è espresso dalla seguente relazione: $\epsilon = \frac{1}{k}\frac{\Delta R}{R_g}$, in cui k è il gage factor, R_g la resistenza iniziale dell'estensimetro, ΔR la sua variazione. A causa dell'elevata sensibilità alla variazione di temperatura delle resistenze non è possibile calcolare direttamente la tensione ai capi dell'estensimetro, ma si mediare il segnale attraverso un ponte di resistenze chiamato Ponte di Wheatstone. Per misure di flessione si utilizzano due estensimetri longitudinali applicati ai lati opposti della cella.

I cavi rosso e nero sono utilizzati per l'alimentazione, i cavi verde e bianco per la lettura della tensione ai capi di S- e S+.









SVILUPPO DEGLI ALGORITMI PER LA LETTURA E L'ANALISI DEI DATI

In questa sezione sono presentati i 3 programmi per la lettura e l'analisi dei dati che provengono dal banco prova con il fine di riadattarli e prepararli per la fase finale di estrapolazione dei coefficienti di coppia-velocità e forza-velocità.

- Il primo algoritmo si occupa della *taratura delle celle* di carico;
- Il secondo algoritmo assolve il compito di acquisizione delle velocità e forze esercitate sulle celle di carico, per *calcolare la spinta e la coppia generata* durante la fase di test del motore.
- L'ultimo algoritmo *ricava i coefficienti* desiderati.

TARATURA CELLE DI CARICO

```
clear all;
close all;
load('K_parameters.mat', 'K');
K0=K(1,:);
K1=K(2,:);
K2=K(3,:);
numero_cella=input("Inserire numero cella da tarare (0,1,2): ");
numero_case=numero_cella+1;
time_step=40;
time_start=10;
num_step=7;
num_sim=1;
sim_time=num_step*time_step+1;
k=0;
j=0;
sum=0;
V_media = [0 0 0 0 0 0 0];
messaggio="pausa, premere per avviare la prova";
messaggio1 ="pausa, premi per stampare il grafico";
messaggio2 ="pausa, premi calibrazione cella successiva";

%%CARATTERIZZAZIONE DELLA CELLA DI CARICO CON STEP DI MASSA
massa=[0.005 0.105 0.205 0.305 0.205 0.105 0.005];
forza=massa*9.81;
```

Nella prima parte di programma vengono definite ed inizializzate le variabili utilizzate. In ‘K’ sono salvati i vari coefficienti angolari e offset delle prove. Si sceglie il numero di cella da tarare. Si definiscono i tempi delle prove in cui applicare il carico e il momento in cui leggere i valori di ogni prova. Il tempo di simulazione totale è di 281 secondi. V_media colleziona i valori medi delle tensioni lette.

```
while(1)

    messaggio
    pause
    numero_case=numero_cella+1;
    num_sim=1;
    sim_time=num_step*time_step+1;
    k=0;
    j=0;
    sum=0;
    V_media = [0 0 0 0 0 0 0];
```

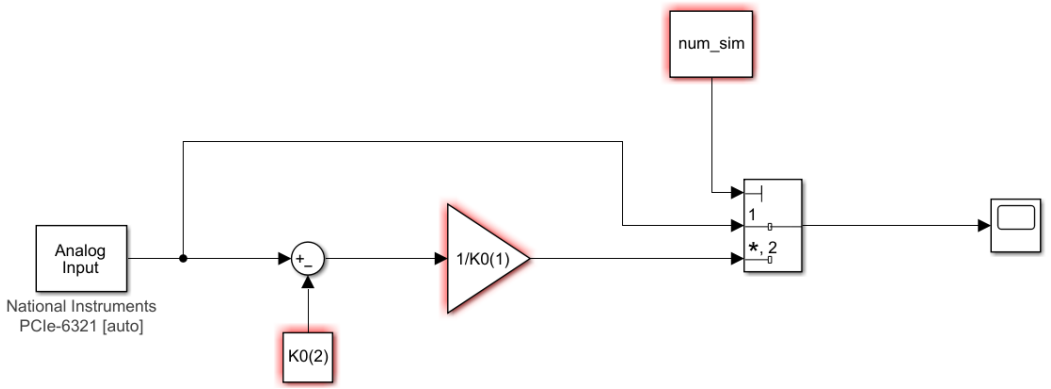
Il ciclo ‘while’ da la possibilità all’utente di scegliere di riefettuare la prova se questa non è andata a buon fine o di testare un’altra cella. All’inizio di ogni ciclo vengono reimpostate le variabili viste in precedenza.

```
switch numero_case

    case 1
        open_system('Simulink_Caratterizzazione0');
        set_param('Simulink_Caratterizzazione0',
                  'StopTime',string(sim_time),
                  'SimulationCommand','start');
        messaggio1
        pause

        for i=1:length(out.Volt.time)
            if(out.Volt.time(i) > (time_start+k*time_step))
                sum=sum+out.Volt.signals.values(i);
                j=j+1;
            end

            if(out.Volt.time(i)>(time_step*(k+1)))
                V_media(k+1)=sum/j;
                j=0;
                k=k+1;
                sum=0;
            end
        end

        figure(numero_case);
        grid on;
        title('Curva caratteristica cella di carico');
        plot(forza,V_media);
        hold on;
        plot(forza,V_media,'x');
        xlabel('F [N]');
        ylabel('V_{cella} [V]');
        legend({'Curva caratteristica', 'Media degli step'},'Location','southeast');
        saveas(figure(numero_case),'Curva_caratteristica_cella_di_carico0','jpg');

        %%DEFINIZIONE POLINOMIO CARATTERISTICO E TEST DELLA CELLA
        K(numero_case,:) = polyfit(massa,V_media,1)
        num_sim=2;
        sim_time=10000;
        bdclose('Simulink_Caratterizzazione0');
```

Questo è uno dei tre possibili ‘case’ che corrispondono al numero di celle impiegate. La struttura di ogni caso è la medesima per ogni cella quindi viene riportato solo il caso 1. Viene aperto il file di Simulink chiamato Simulink_Caratterizzazione0(riportato nella slide successiva) che acquisisce il segnale dalla cella. I dati letti vengono salvati in un array ‘out.Volt.time’. Quando il tempo di applicazione di un peso sulla cella supera i 10 secondi i valori di tensione vengono sommati. Durante i 10 secondi di assestamento del carico applicato alla cella si calcola la tensione media letta con il carico precedente. Successivamente viene stampata la retta ‘forza-tensione’ e salvate pendenza e offset della retta nella matrice ‘K’. È importante notare che K colleziona le pendenze e gli offset fittando le masse e non le forze. In fine si chiude il Simulink_Caratterizzazione0.

```
    first_run=input("Se la prova è valida,
                    digitare [1] per passare i parametri di calibrazione
                    al programma di acquisizione dati,
                    [0] per rieffettuare la taratura di una cella: ");

    if(first_run==1)
        save('K_parameters',"K");
        load('Workspace_acquisizione_spinta_tre_celle');
        run('acquisizione_spinta_tre_celle.m');
        break;
    else
        numero_cella=input("Inserire numero cella da tarare (0,1,2): ");
        save('K_parameters',"K");
        K
    end

end
```

Alla fine del ciclo ‘while’ si comunica all’utente se passare i parametri al programma successivo o effettuare la taratura di un’altra cella di carico.



Blocco Simulink relativo all'acquisizione della tensione in relazione al carico applicato alla cella.

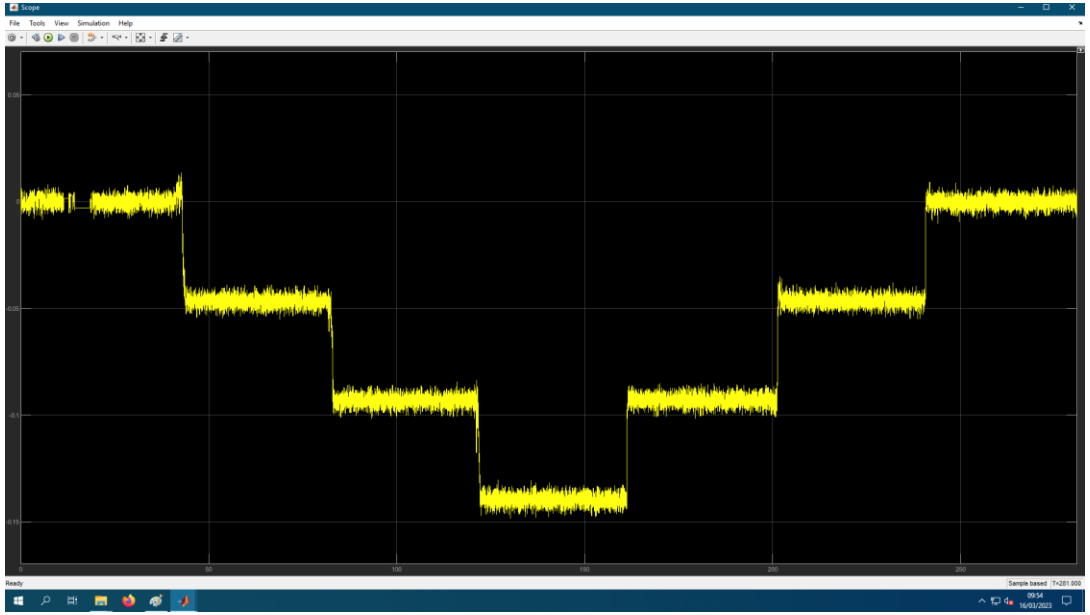
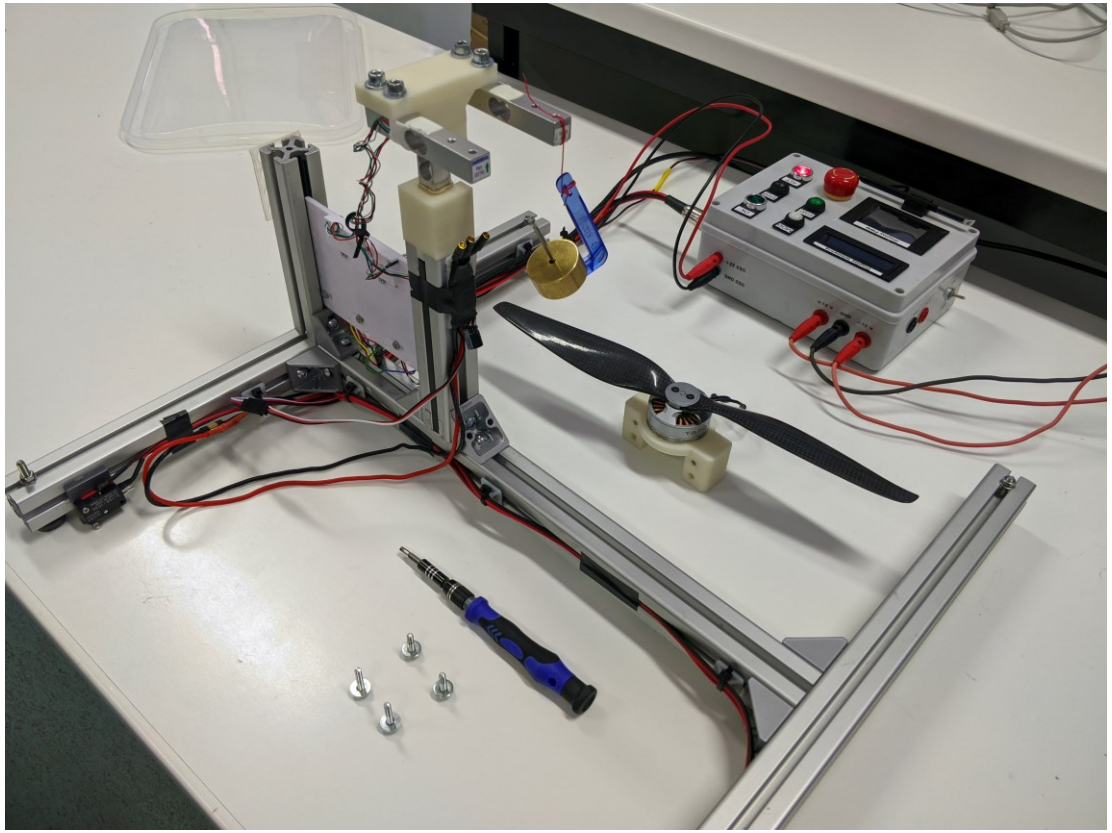


Grafico visualizzato dallo scope.

Come effettuare la prova di taratura



La prova di taratura è svolta applicando in successione dei pesi partendo da 0.005kg fino a 0.305kg e poi scaricando il sistema per tornare alle condizioni inziali. È importante scegliere un cavo non troppo rigido per non rischiare di tenere sollevato il peso o non riuscire a estinguere le oscillazioni innescate nei 10 secondi che precedono l'acquisizione del segnale. Il carico essendo fluttuante in aria è sensibile a fluttuazioni innescate da movimenti bruschi da parte degli operatori.

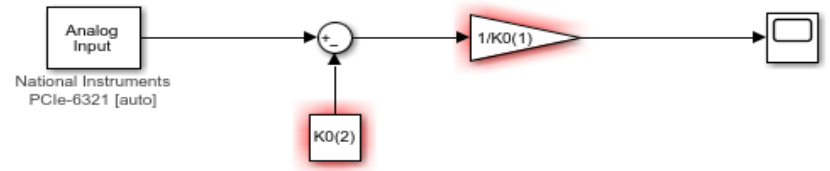
CALCOLO DELLA SPINTA DEL MOTORE

```
clear all;
close all;

time_start = 10;
time_stop = 10;
time_step= time_start + 10 + time_stop;
duty = [50; 55; 60; 65; 70; 75; 80; 85; 90;];
num_step = length(duty);
sim_time=num_step*(time_step);

%variabili per i cicli
k=0;
j=0;
i=0;

%TARATURA
%definizione variabili usate per taratura
messaggio ="pausa, premi per iniziale la taratura";
messaggio1 ="pausa, premi per iniziale la prova quando compare key matlab";
messaggio2 ="pausa, premi per visualizzare i grafici";
taratura=1;
durata_taratura=5;

load('K_parameters', "K");
K0 = K(1,:);
K1 = K(2,:);
K2 = K(3,:);

K0
K1
K2


sum_tara0=0;
sum_tara1=0;
sum_tara2=0;
tara_media0=0;
tara_media1=0;
tara_media2=0;
```

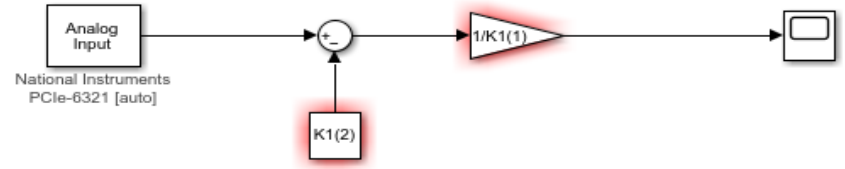
Vengono definite e inizializzate le variabili utili al calcolo della spinta e la coppia prodotta dal motore in fase di test. Per ogni acquisizione di forza sviluppata dal motore sulle celle ad ogni duty si ignorano i dati raccolti durante i primi e gli ultimi 10 secondi. La taratura iniziale delle celle di carico dura 5 secondi. Viene caricata la matrice ‘K’ dei parametri calcolati durante la prova precedente.

```
%ANALISI FORZA SVILUPPATA DAL MOTORE A DIVERSI DUTY
%definizione variabili
F0= zeros(2750,1);
F1= zeros(2750,1);
F2= zeros(2750,1);
velocita = zeros(2750,1);
Forza_media0 = zeros(length(duty),2);
Forza_media1 = zeros(length(duty),2);
Forza_media2 = zeros(length(duty),2);
Forza_netta0 = zeros(length(duty),2);
velocita_media_rpm = zeros(length(duty),2);
velocita_media_angolare = zeros(length(duty),2);
braccio = 0.028;
coppia1=zeros(length(duty),2);
coppia2=zeros(length(duty),2);
coppia_media = zeros(length(duty),2);
messaggio
pause

%TARATURA
if taratura == 1
    open_system('Simulink_taratura_celle');
    set_param('Simulink_taratura_celle','StopTime',string(durata_taratura),'SimulationCommand','start');
    messaggio1
    pause

    for i=1:length(tara0.time)
        sum_tara0=sum_tara0+tara0.signals.values(i);
        sum_tara1=sum_tara1+tara1.signals.values(i);
        sum_tara2=sum_tara2+tara2.signals.values(i);
        j=j+1;
    end
    tara_media0=sum_tara0/j;
    tara_media1=sum_tara1/j;
    tara_media2=sum_tara2/j;
    j=0;
    taratura=0;
    save_system('Simulink_taratura_celle');
    close_system('Simulink_taratura_celle');
end
```

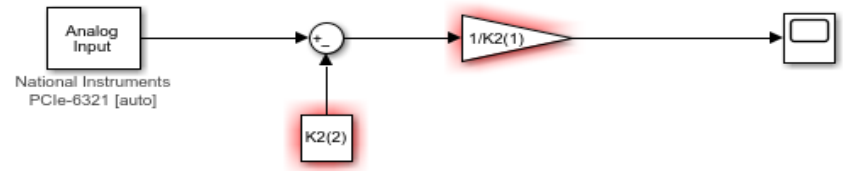
cella 0



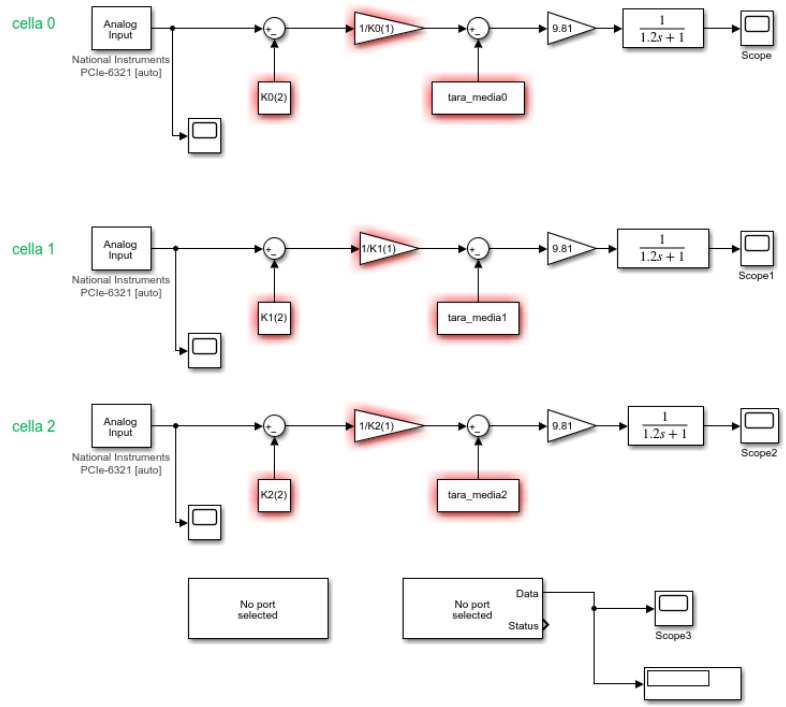
cella 1

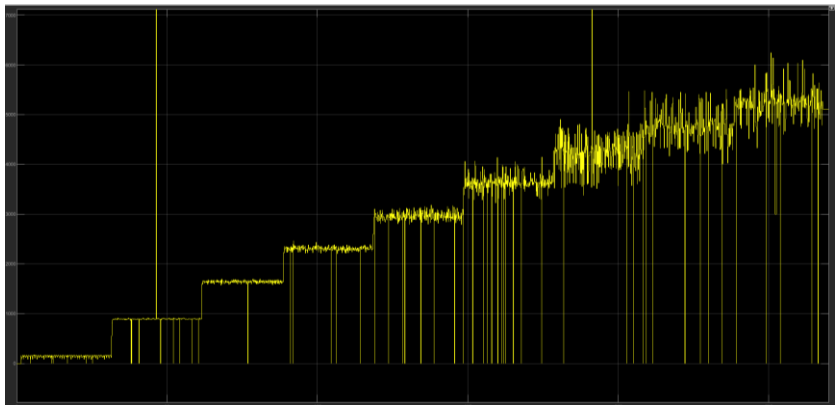


cella 2



```
if taratura==0
    open_system('Simulink_Analisi_Forze');
    set_param('Simulink_Analisi_Forze','StopTime',string(sim_time),'SimulationCommand','start');
    messaggio2
    pause

    %accumula valore forze nello step di duty e j conta numero misurazioni
    for i=1:length(out.Forza0.time)
        if (((out.Forza0.time(i) > (time_start+k*time_step))&& (out.Forza0.time(i) < ((k+1)*time_step-time_stop))))
            j=j+1;
            F0(j)=out.Forza0.signals.values(i);
            F1(j)=out.Forza1.signals.values(i);
            F2(j)=out.Forza2.signals.values(i);
            velocita(j)=abs(out.Velocita.signals.values(i));
        end
        %calcola la forza media date le precedenti misurazioni acquisite in 10+k*30
        if(out.Forza0.time(i)>=(time_step*(k+1)-time_stop))
            velocita((velocita==0) | (velocita > 10000)) = NaN;
            velocita = fillmissing(velocita, 'nearest');
            %calcolo media per lo step di duty
            Forza_media0(k+1,1)=(sum(F0))/(length(F0));
            Forza_media1(k+1,1)=(sum(F1))/(length(F1));
            Forza_media2(k+1,1)=(sum(F2))/(length(F2));
            velocita_media_rpm(k+1,1)= (sum(velocita))/(length(velocita));
            %calcolo varianza per lo step di duty
            Forza_media0(k+1,2)=(sum((F0-Forza_media0(k+1,1)).^2))/(length(F0));
            Forza_media1(k+1,2)=(sum((F1-Forza_media1(k+1,1)).^2))/(length(F1));
            Forza_media2(k+1,2)=(sum((F2-Forza_media2(k+1,1)).^2))/(length(F2));
            velocita_media_rpm(k+1,2)=(sum((velocita-velocita_media_rpm(k+1,1)).^2))/(length(velocita));

            j=0;
            k=k+1;
            F0= zeros(2750,1);
            F1= zeros(2750,1);
            F2= zeros(2750,1);
            velocita =  zeros(2750,1);
        end
    end
```





```
%finita la prova calcola le coppie in ogni duty date le Fmedie e il braccio
%CALCOLO DELLE COPPIE MISURATE DALLE CELLE VERTICALI E DELLA FORZA NETTA
coppia1 = [Forza_media1(:,1) * braccio , Forza_media1(:,2)*(braccio^2)];
coppia2 = [Forza_media2(:,1) * braccio , Forza_media2(:,2)*(braccio^2)];
for i=1:length(duty)
    coppia_media(i,1)=(abs(coppia1(i,1))+ abs(coppia2(i,1)))/2;
    coppia_media(i,2)=(coppia1(i,2) + coppia2(i,2)) * 0.25;
    Forza_netta0 = Forza_media0; %- (peso_totale * 9.81); aggiunto su simulink
    velocita_media_angolare(i,1) = ((velocita_media_rpm(i,1))*2*pi)/60;
end

%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1);
plot (velocita_media_angolare(:,1),Forza_netta0(:,1));
xlabel('Velocità [rad/s]');
ylabel('Forza Motore [N]');
hold on;

figure(2);
plot (velocita_media_angolare(:,1),coppia_media(:,1));
xlabel('Velocità [rad/s]');
ylabel('Coppia di Drag [Nm]');
hold on;

cf = polyfit(velocita_media_angolare(i,1).^2, Forza_netta0(i,1), 1);
ct = polyfit(velocita_media_angolare(i,1).^2, coppia_media(i,1), 1);

coefficiente_forza = cf(1);
coefficiente_coppia = ct(1);

n=input('inserire il numero della prova: ');
save('Workspace_acquisizione_spinta_tre_celle');
filename = sprintf('%s_%d','Risultati_prova',n);
save(filename, "Forza_netta0", "coppia_media",
               "velocita_media_angolare",
               "coefficiente_forza", "coefficiente_coppia");
n=n+1;

end
```

```
close_system('Simulink_Analisi_Forze');

risultati_ok=input("Per elaborare il risultato combinando le varie prove precedentemente effettuate premere [1],
                    altrimenti [0] per effettuare altre bancature");
if(risultati_ok==1)
    load('Workspace_analisi_risultati.mat')
    run('analisi_risultati.m');
else
    run('acquisizione_spinta_tre_celle.m');
end
```

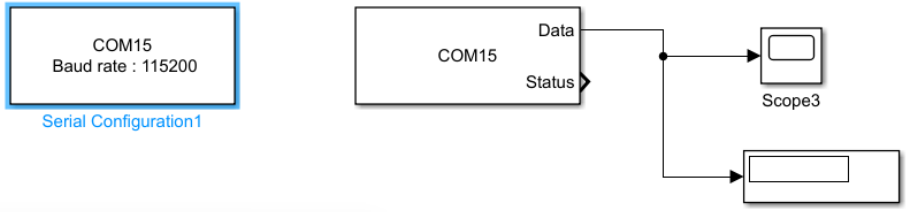
Con l'obiettivo di calcolare i coefficienti di coppia e forza in relazione al quadrato della velocita abbiamo modificato la funzione che trova il coefficiente angolare della retta.

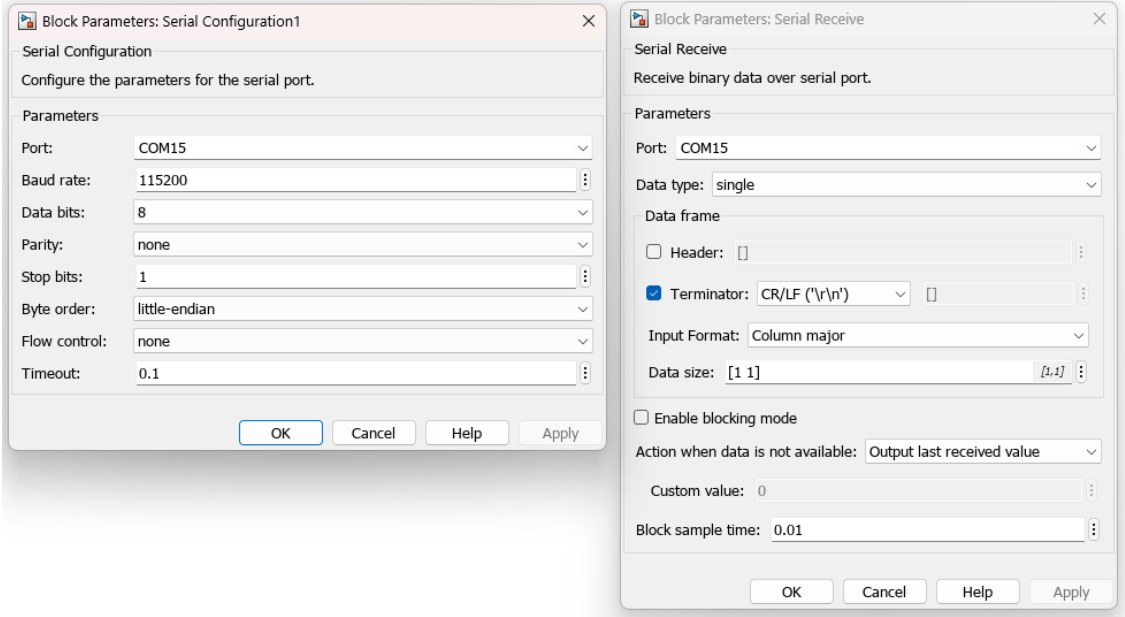
Comunicazione seriale Arduino-Matlab

Attraverso la libreria Simulink Instrumental Control Toolbox è stata implementata la comunicazione seriale tra Arduino Mega e Matlab. Grazie a questa modifica del software di acquisizione dati, la procedura di testing è stata resa più efficiente eliminando la necessità di utilizzare Coolterm e la fase di elaborazione dei dati relativi alla velocità alla fine del test.

La telemetria utilizzata per la prova è prelevata dall'ESC, che utilizza il protocollo KISS 32-bit.
La comunicazione della telemetria avviene con un baud rate di 115200 e viene inviato un pacchetto di 8 byte ogni 900µs.
La velocità è rappresentata dai byte 7 e 8 del pacchetto e ha una risoluzione di 100 rpm.
Per la conversione della velocità da Erpm a rpm relativi al motore in prova, viene utilizzato il numero di coppie polari. La velocità viene inviata come un uint16 e deve essere moltiplicata per 100 e poi divisa per il valore di (npoli/2).
Questa operazione viene eseguita dall'Arduino Mega che salva il risultato su una variabile denominata trueRPM di tipo double (4 byte).

```
byte *b=(byte *) trueRPM;
Serial.write(b, 4); //invia 4 byte alla porta seriale
Serial.write('\r');
Serial.write('\n');  //invia sequenza che termina riga /r/n
delay(10);
```

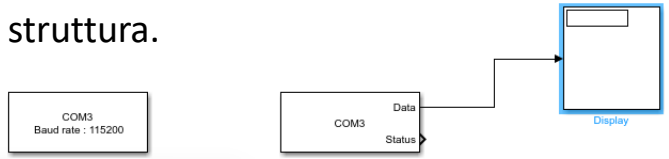


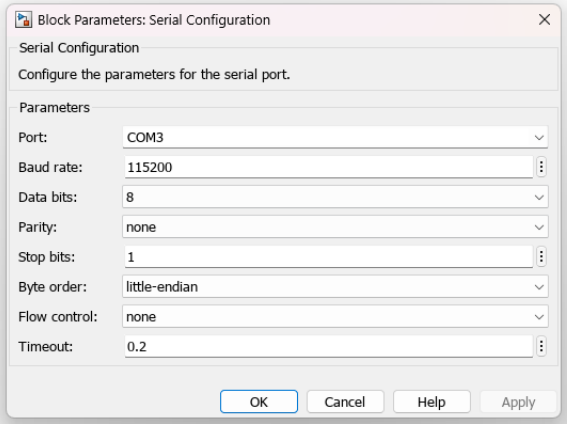


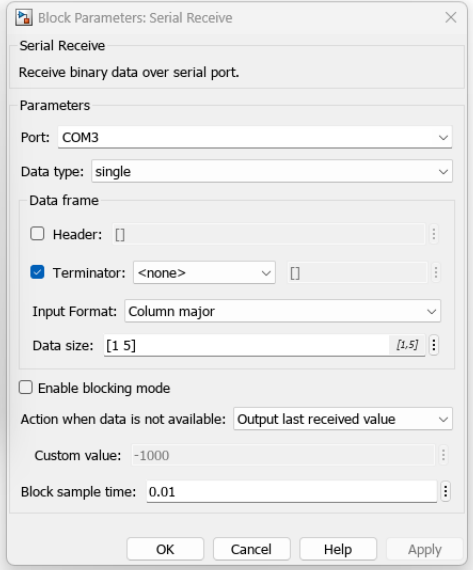
Comunicazione seriale Arduino-Matlab

Inoltre, attualmente si sta progettando la possibilità di inviare i segnali provenienti dalle celle di carico all'Arduino Mega, in modo da non dipendere dalla scheda National Instrument.

Bisogna però fare attenzione affinché non si verifichino tensioni negative all'ingresso degli input analogici dell'Arduino Mega.
Pertanto, sarà necessario agire sulla taratura dell'offset delle celle di carico e sull'orientamento delle stesse durante la fase di montaggio sulla struttura.








```
1
2  float trueRPM = 0;
3  float cella0;
4  float cella1;
5  float cella2;
6  float dataArray[5];
7  long time=0;
8  long prevTime=0;
9
10 void setup() {
11   Serial.begin(115200);
12 }
13
14 void loop() {
15   cella0 = analogRead(A0)*1.00;
16   cella1 = analogRead(A1) * 1.00;
17   cella2 = analogRead(A2) * 1.00;
18   data += 0.0001;
19   dataArray[0]= cella0;
20   dataArray[1]= cella1;
21   dataArray[2]= cella2;
22   dataArray[3]= trueRPM;
23   dataArray[4]= (time-prevTime)*1.00;
24   time = micros();
25   byte *b = (byte *) dataArray;
26   for(byte i=0; i<sizeof(dataArray); i++){
27     Serial.write(b[i]);
28   }
29   prevTime = time;
30   delay(10);
31 }
32
33
```

ANALISI DATI

Per l'analisi dei dati ottenuti durante le prove abbiamo creato uno script Matlab, chiamato analisi_risultati, che ci ha permesso di eliminare l'uso di Coolterm, inoltre con questo script si ha la possibilità di analizzare insieme i dati di più prove in modo da avere dei risultati finali migliori e più affidabili.

```
n=input('inserire il numero della prova: ');
save('Workspace_acquisizione_spinta_tre_celle');
filename = sprintf('%s_%d','Risultati_prova',n);
save(filename, "Forza_netta0", "coppia_media", "velocita_media_angolare", "coefficiente_forza", "coefficiente_coppia");
n=n+1:
```

Poi, a seconda dei risultati ottenuti e dal numero di prove effettuate, questi dati vengono aperti ed elaborati da un ulteriore programma:

```
risultati_ok=input("Per elaborare il risultato combinando le varie prove precedentemente effettuate premere [1],altrimenti [0] per effettuare altre bancature");
if(risultati_ok==1)
    load('Workspace_analisi_risultati.mat')
    run('analisi_risultati.m');
else
    run('acquisizione_spinta_tre_celle.m');
```

SCRIPT analisi_risultati

```
for i=1:9
    dati_duty = risultati_forza((j*numero_prove+1):((j+1)*numero_prove),:)
    valori_finali(i,1)= sum(dati_duty(:,1)) / numero_prove;
    valori_finali(i,2) = sum(dati_duty(:,2)) / (numero_prove^2);

    dati_duty = risultati_coppia((j*numero_prove+1):(i*numero_prove),:);
    valori_finali(i,3)= sum(dati_duty(:,1)) / numero_prove;
    valori_finali(i,4) = sum(dati_duty(:,2)) / (numero_prove^2);

    dati_duty = risultati_velocita((j*numero_prove+1):(i*numero_prove),:);
    valori_finali(i,5)= sum(dati_duty(:,1)) / numero_prove;

    %verifica coefficienti di coppia e forza
    forza_media_cf(i) = -(valori_finali(i,5).^2) * coefficiente_forza_medio;
    coppia_media_cp(i) = (valori_finali(i,5).^2) * coefficiente_coppia_medio;

    errore_medio_forza(i)=abs(forza_media_cf(i)-valori_finali(i,1));
    errore_medio_coppia(i)=abs(coppia_media_cp(i)-valori_finali(i,3));

    j=j+1;

end
```

In questo script, andiamo a calcolare i valori medi di forza e coppia di ogni duty cycle, siamo quindi in grado di confrontare i dati ottenuti con i valori che dipendono dai coefficienti di coppia e forza:

$$F_m = k_F \cdot v^2$$

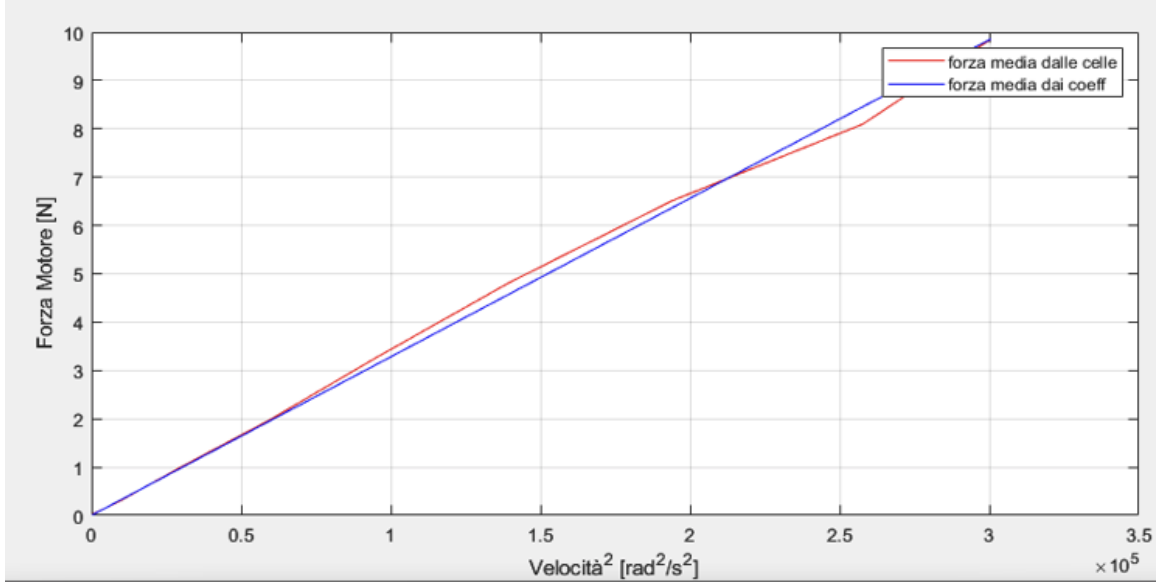
$$C_m = k_c \cdot v^2$$

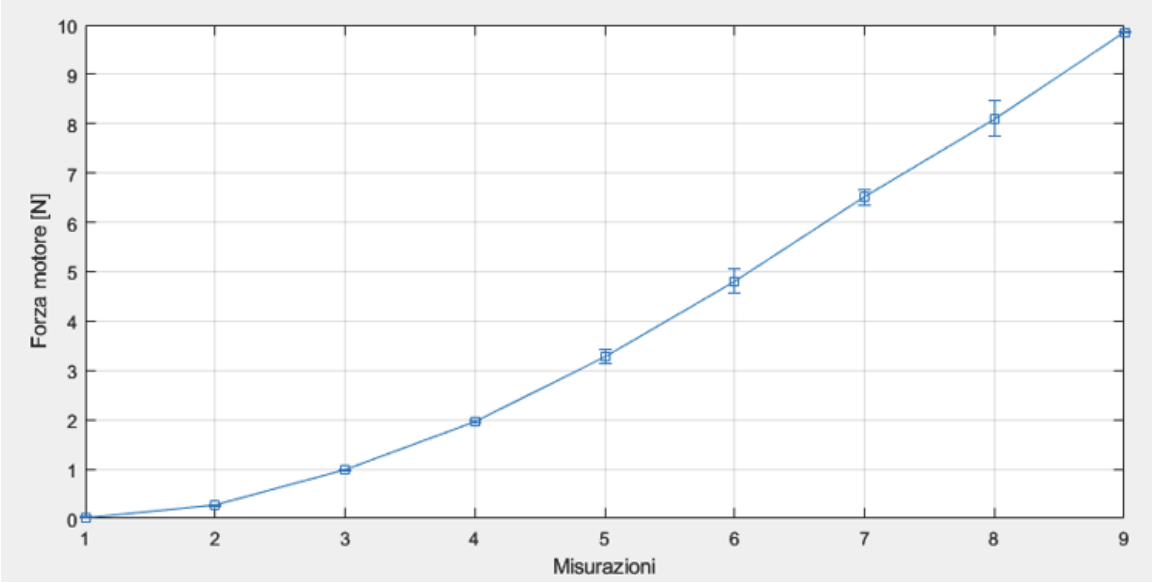
ESEMPIO DI WORKSPACE MATLAB

Import	Name	Size	Bytes	Class
☑	Forza_netta0	9x2	144	double
☑	coefficiente_coppia	1x1	8	double
☑	coefficiente_coppia_medio	1x1	8	double
☑	coefficiente_forza	1x1	8	double
☑	coefficiente_forza_medio	1x1	8	double
☑	coppia_media	9x2	144	double
☑	coppia_media_cp	9x1	72	double
☑	dati_duty	4x2	64	double
☑	errore_medio_coppia	9x1	72	double
☑	errore_medio_forza	9x1	72	double
☑	filename	1x17	34	char
☑	forza_media_cf	9x1	72	double
☑	i	1x1	8	double
☑	j	1x1	8	double
☑	n	1x1	8	double
☑	numero_prove	1x1	8	double
☑	risultati_coefficiente_coppia	4x1	32	double
☑	risultati_coefficiente_forza	4x1	32	double
☑	risultati_coppia	36x2	576	double
☑	risultati_forza	36x2	576	double
☑	risultati_velocita	36x2	576	double
☑	valori_finali	9x5	360	double
☑	velocita_media_angolare	9x2	144	double

	1	2	3	4	5
1	0.0225	1.7949...	5.2663...	6.9304...	15.7814
2	0.2702	3.9553...	0.0015	7.7874...	93.2063
3	0.9811	4.7105...	0.0039	1.4451...	171.21...
4	1.9607	5.6858...	0.0073	3.4244...	243.10...
5	3.2692	2.2385...	0.0119	1.1745...	308.61...
6	4.8129	3.2503...	0.0178	8.6747...	373.45...
7	6.5059	2.8810...	0.0256	5.7537...	440.24...
8	8.0975	8.8960...	0.0324	8.9165...	507.69...
9	9.8532	9.7469...	0.0413	8.4984...	548.16...

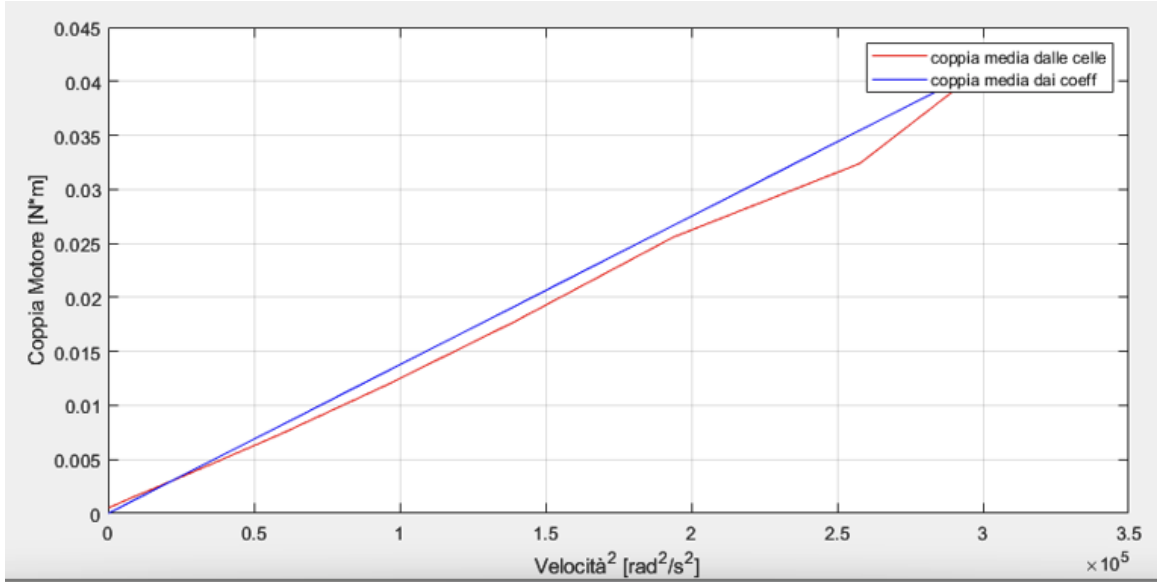
CURVE DI FORZA MOTORE TAROT 6S4108KV380 (senso antiorario)

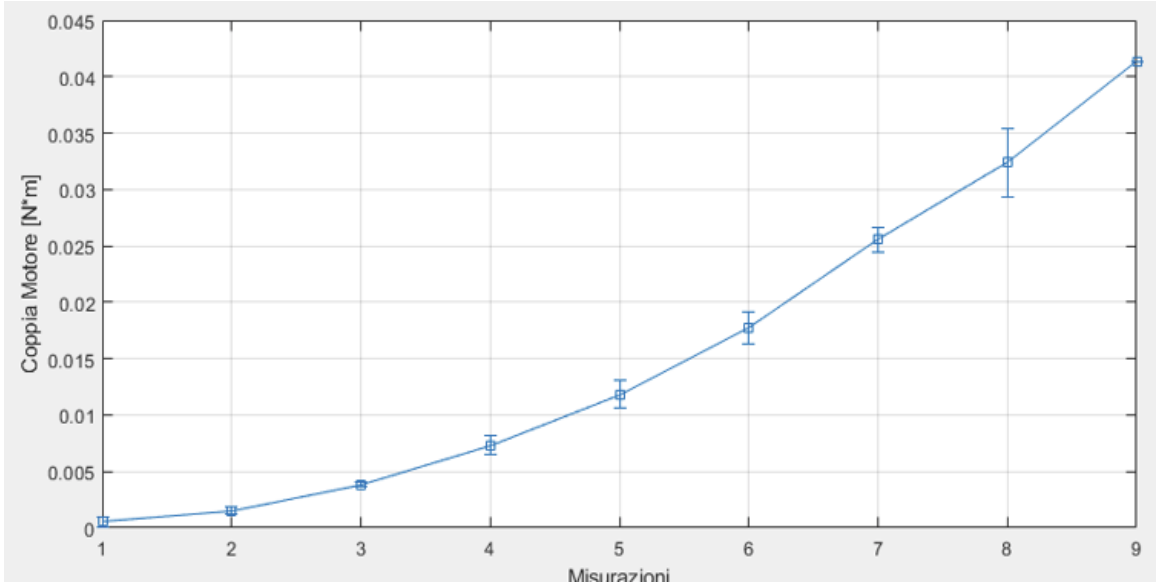




Nella prima curva viene vengono confrontati i valori di forza ottenuti dalle celle di carico con quelli ottenuti grazie al coefficiente di forze, mentre nella seconda curva viene mostrato l'errore medio delle misurazioni di forza.

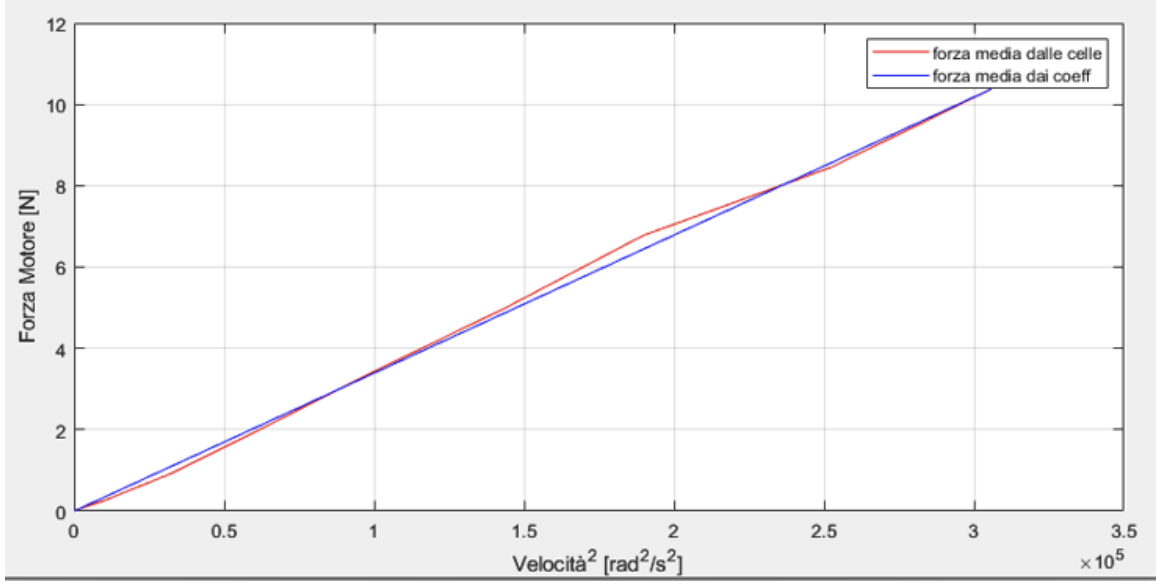
CURVE DI COPPIA MOTORE TAROT 6S4108KV380 (senso antiorario)

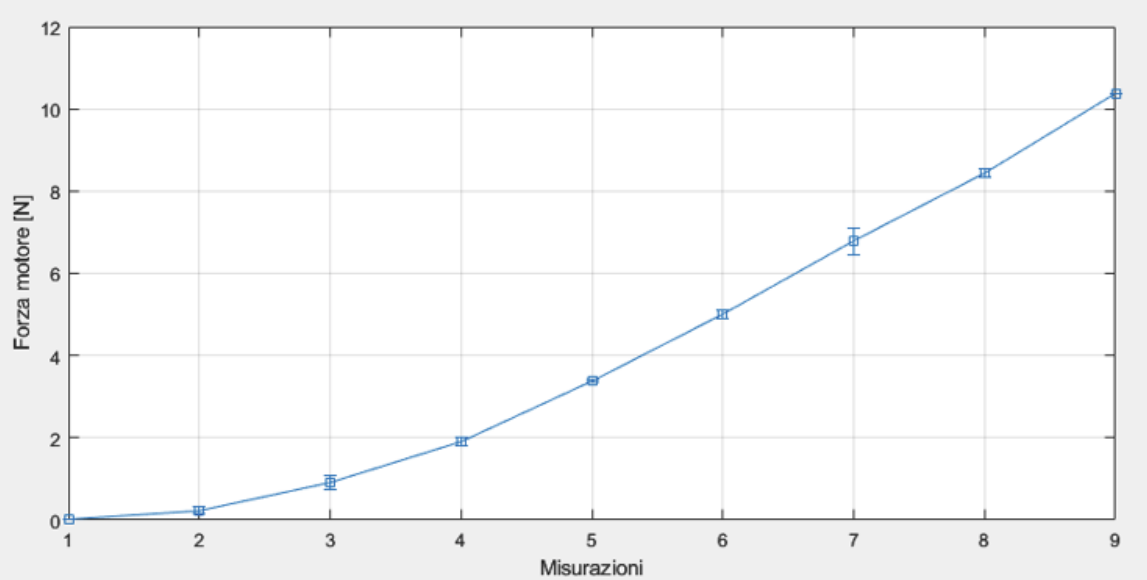




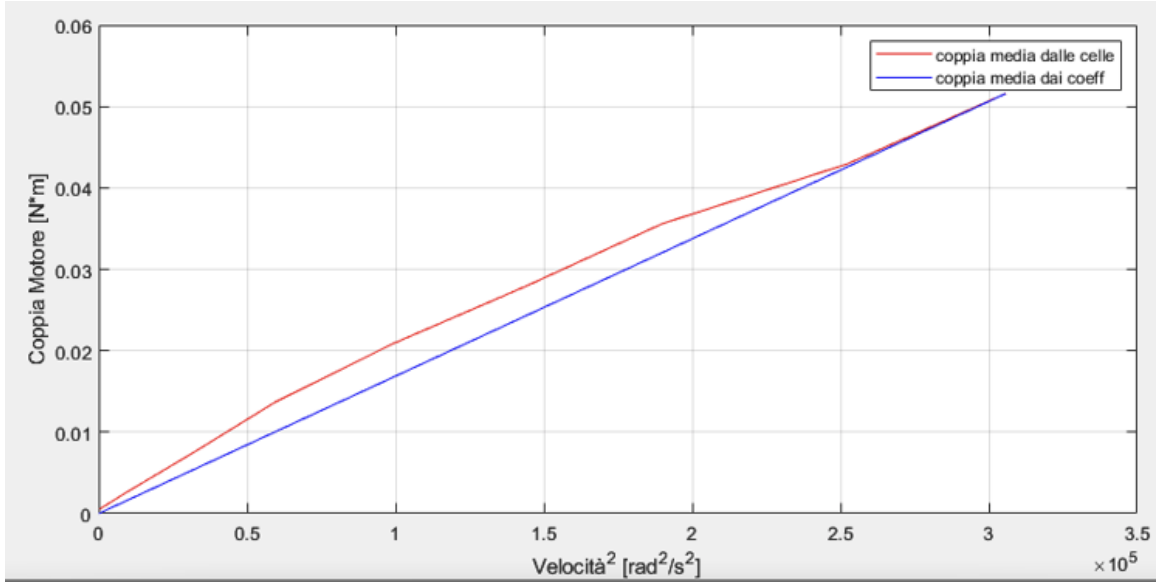
In questi curve invece vengono prima confrontati i valori di coppia ottenuti dalle celle di carico con quelli ottenuti grazie al coefficiente di coppia, mentre nella seconda curva viene mostrato l'errore medio delle misurazioni di coppia.

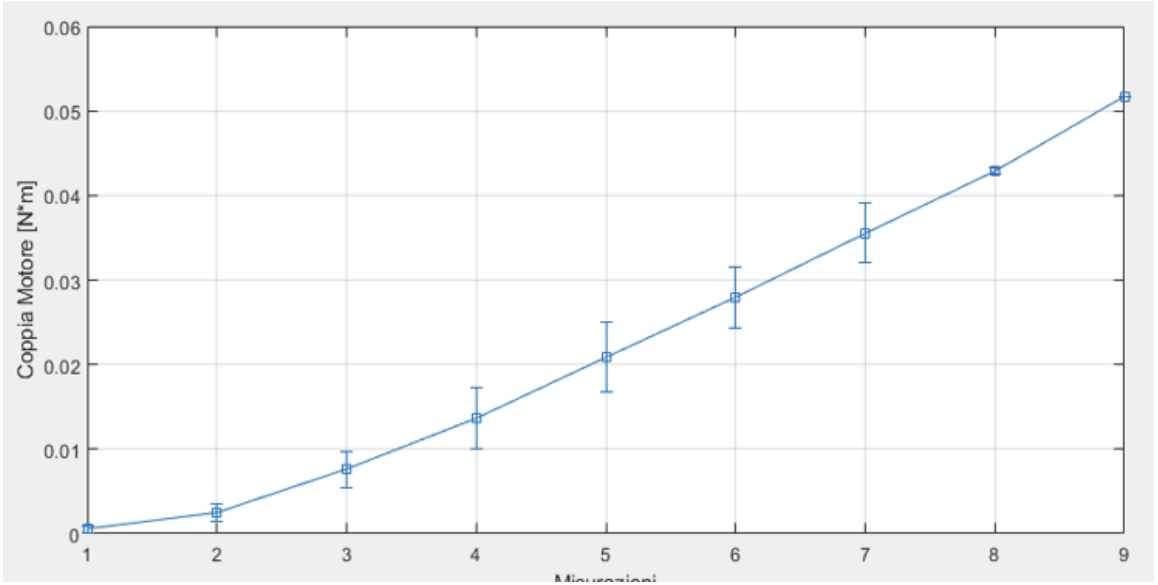
CURVE DI FORZA MOTORE TAROT 6S4108KV380 (senso orario)



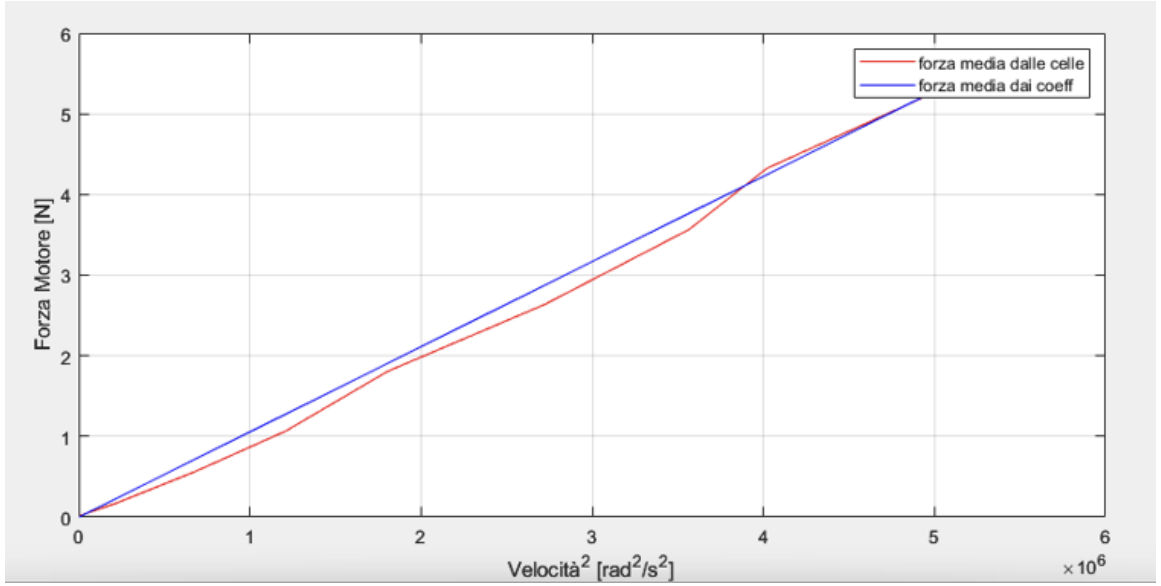


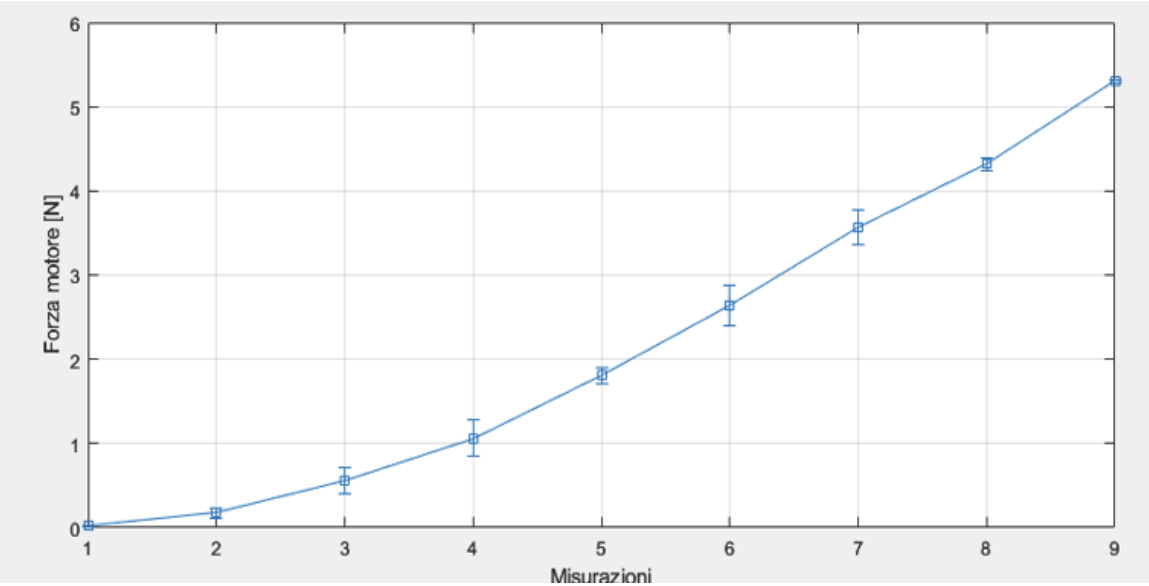
CURVE DI COPPIA MOTORE TAROT 6S4108KV380 (senso orario)



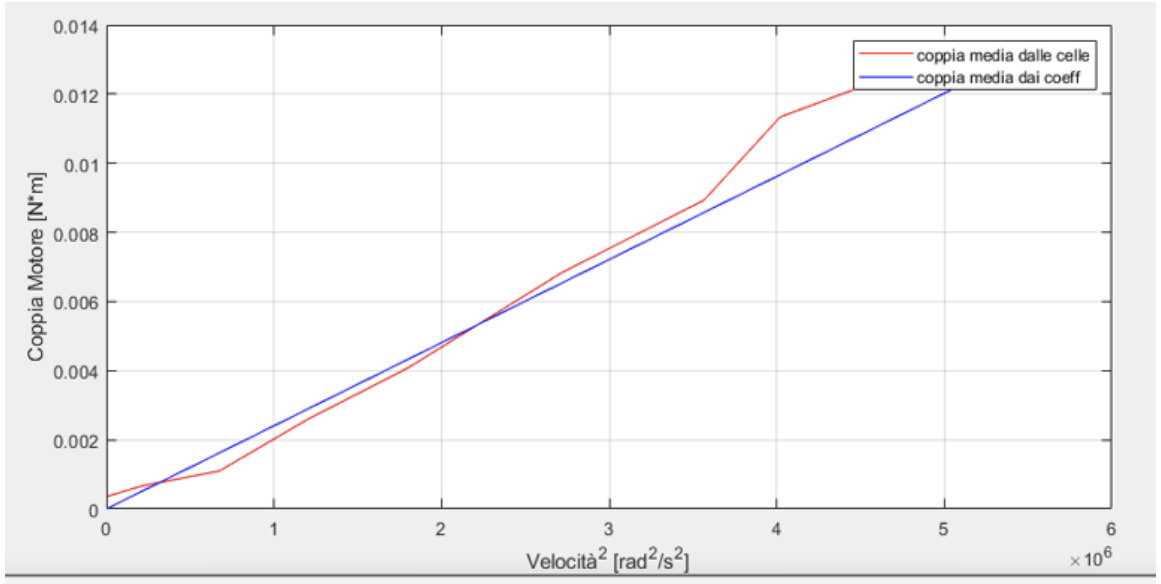


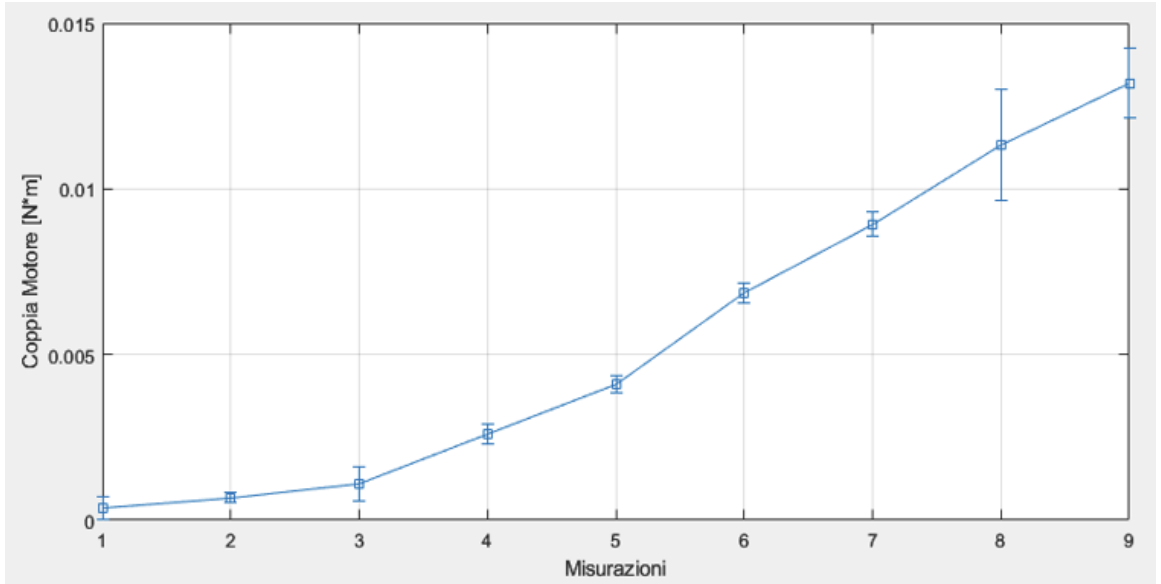
CURVE DI FORZA MOTORE 2206-KV2300 (senso antiorario)



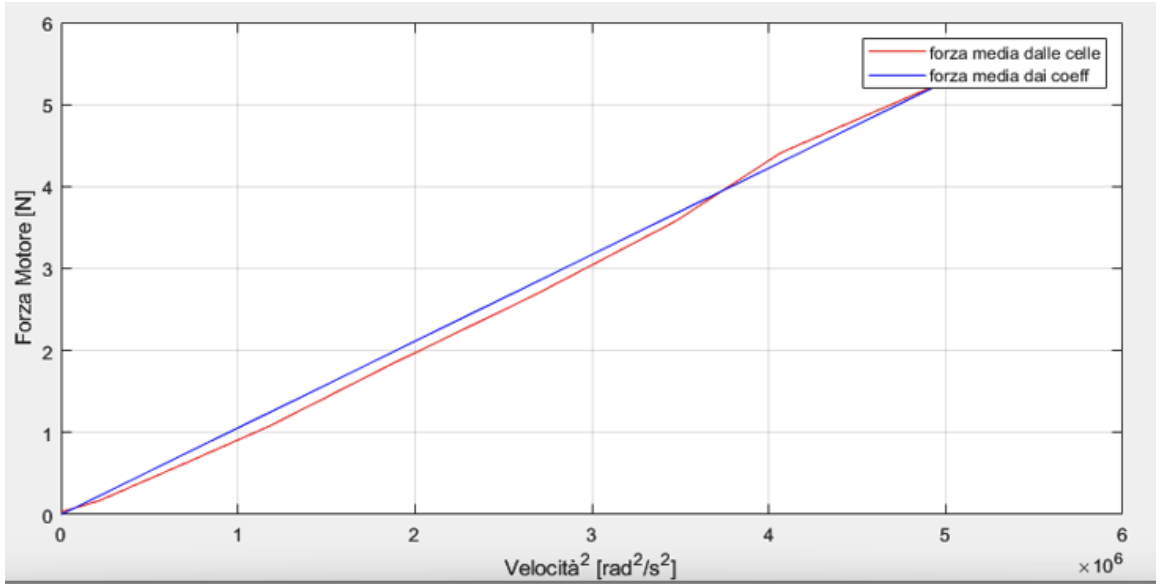


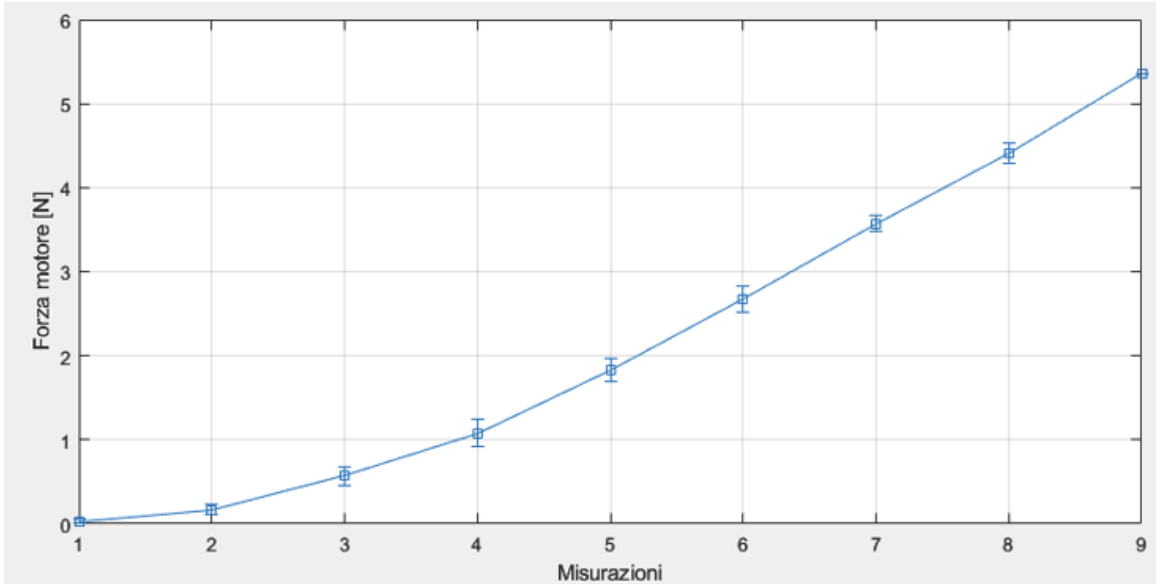
CURVE DI FORZA MOTORE 2206-KV2300 (senso antiorario)



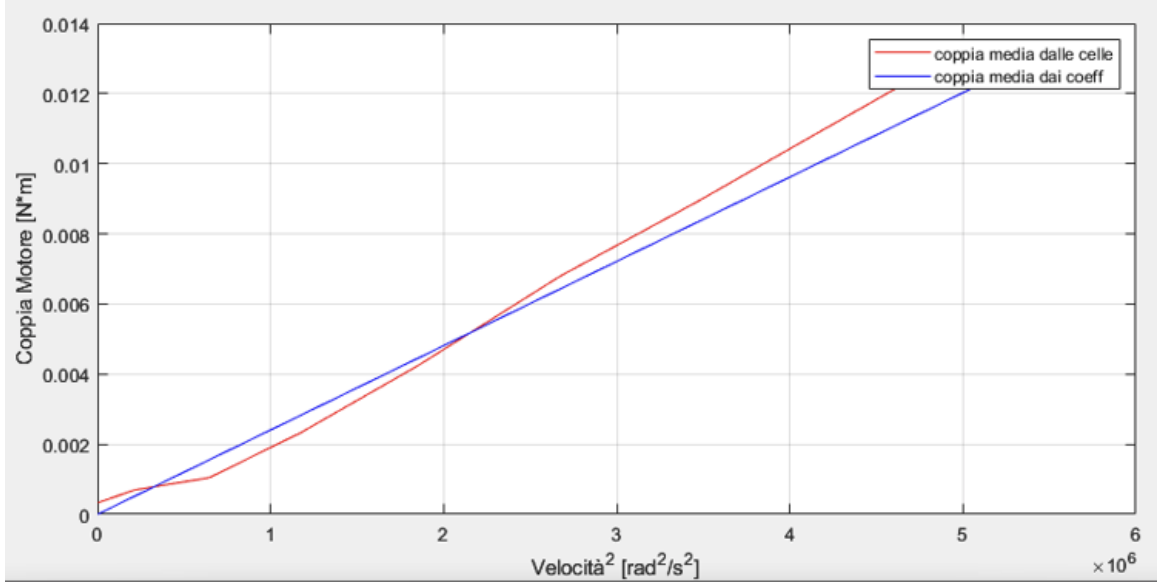


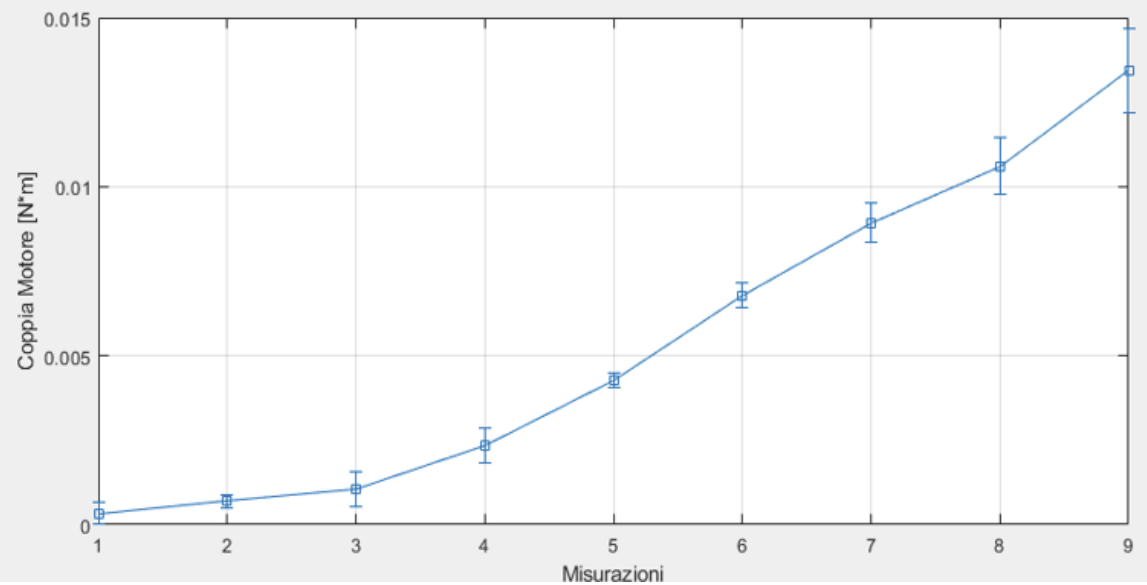
CURVE DI FORZA MOTORE 2206-KV2300 (senso orario)





CURVE DI FORZA MOTORE 2206-KV2300 (senso orario)





DOCUMENTAZIONE

Al fine di consentire una continuità al progetta ed una maggiore facilità di comprensione del sistema agevolando dunque l'esecuzione della prova abbiamo elaborato una guida all'utilizzo del banco prova. Inoltre, in allegato, si trovano:

- Disegni dei componenti 3d modificabili ed assieme 3d completo della struttura;
- Schemi elettrici di tutti i circuiti in cui sono rappresenti tutti i vari collegamenti;
- Tutti i datasheet dei vari dispostivi presenti;
- Risultati da noi ottenuti

SVILUPPI FUTURI E CONCLUSIONI

Eseguendo le diverse prove per la caratterizzazione dei due motori abbiamo notato che il sistema banco prova può essere ulteriormente migliorato su più aspetti:

- Snellire i supporti delle celle di carico 1,2;
- Irrobustire supporto cella di carico 0;
- Modifica supporto motore QAV250 2206KV2300 (legati a problemi di turbolenza);
- Eliminazione scheda acquisizione N.I. PCIe-6321 → comunicazione seriale dei segnali provenienti dalle celle di carico tramite Arduino mega;
- Riduzione braccio coppia di drag;
- Introdurre sistema di sincronizzazione automatico tra Arduino Uno e Matlab;
- Conversione da rpm a rad/s fatta fare a Matlab in modo da consentire il cambiamento di poli(MOTORE) più facilmente;

Grazie alla prova si è constato che i coefficienti calcolati restituiscono delle forze e coppie coerenti con quelle misurate e quindi l'obiettivo di validare il modello è riuscito.

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

SI RINGRAZIANO

MICHIELETTO STEFANO
MICHIELETTO GIULIA
BERTONI MASSIMILIANO
LOSCO ROBERTO