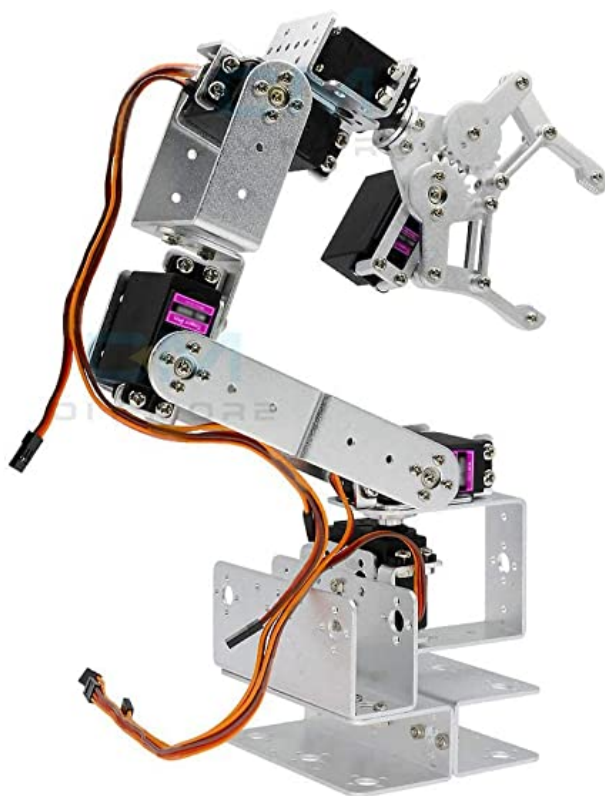




**TOR VERGATA**  
UNIVERSITÀ DEGLI STUDI DI ROMA



## **Relazione tirocinio Scorbot**

Simone Arcari, matricola: 0292185

19/02/2023

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Studio del modello matematico</b>	<b>3</b>
2.1	Denavit-Hartenberg . . . . .	3
2.2	Cinematica diretta . . . . .	4
2.3	Cinematica inversa . . . . .	5
2.4	Formulario . . . . .	6
<b>3</b>	<b>MATLAB</b>	<b>7</b>
3.1	Descrizione codice MATLAB . . . . .	7
3.1.1	Codice MATLAB . . . . .	7
<b>4</b>	<b>Processing</b>	<b>8</b>
4.1	Comunicazione seriale . . . . .	8
4.2	Render 3D . . . . .	10
4.3	Modalità e comandi . . . . .	10
4.3.1	Comadi base . . . . .	10
4.3.2	Manual Mod . . . . .	11
4.3.3	Inverse Kinematic Mod . . . . .	11
4.3.4	Frame Mod . . . . .	11
<b>5</b>	<b>Arduino</b>	<b>12</b>
5.1	schema elettrico . . . . .	12
5.2	codice . . . . .	12
	<b>Bibliografia</b>	<b>13</b>

# 1 Introduzione

Lo scopo del progetto è quello di controllare la cinematica di un robot di tipo Scorbob mediante simulazione 3D in ambiente Processing con comunicazione seriale tra Processing e la scheda di sviluppo Arduino UNO, che si occupa del controllo finale degli attuatori utilizzati per la movimentazione dei giunti meccanici.

## 2 Studio del modello matematico

Lo Scorbob utilizzato presenta una struttura leggermente differente da quella solitamente utilizzata in ambito didattico. Inoltre, siccome uno degli obiettivi iniziali era invertire il verso di riferimento degli angoli  $\theta_i$ , si è reso indispensabile riformulare tutti i passi per il calcolo della cinematica inversa.

### 2.1 Denavit-Hartenberg

Il robot presenta 5 DOF (degree of freedom), è perciò necessario ricorrere al modello di Denavit-Hartenberg per lo studio della geometria dello Scorbob. Per soddisfare l'obiettivo di invertire il verso degli angoli  $\theta_i$  sono stati scelti i versi degli assi  $z_i$  in direzione opposta rispetto alla rappresentazione degli stessi nella formulazione classica. Effettuati tutti i procedimenti richiesti si giunge alla seguente tabella:

i	$\theta_i$	$d_i$	$a_i$	$\alpha_i$
1	$\theta_1^*$	0	0	$\Pi/2$
2	$\theta_2^*$	0	$l_2$	0
3	$\theta_3^*$	0	$l_3$	0
4	$\theta_4^*$	0	$a_4$	$\Pi/2$
5	$\theta_5^*$	$d_5$	0	0

Si può osservare che nella tabella figura il parametro  $a_4$ , ciò è dovuto alla posizione dell'handeffector, che risulta assemblato con una leggera traslazione rispetto all'asse  $x_4$ . Da ogni riga della tabella si può ora ricavare la matrice del cambio delle coordinate tra  $O_{i-1}$  e  $O_i$  che indicheremo con  $T_{i-1}^i$ . Questa matrice è nota ed è la seguente:

$$T_{i-1}^i = T_z(\theta_i, d_i) * T_x(\alpha_i, a_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Si prosegue con il calcolo della matrice del cambiamento delle coordinate da  $O_0$  a  $O_5$ :

$$T_0^5 = T_0^1 * T_1^2 * T_2^3 * T_3^4 * T_4^5 = \begin{bmatrix} c_1 c_{234} c_5 + s_1 s_5 & -c_1 c_{234} s_5 + s_1 c_5 & c_1 s_{234} & a_4 c_1 c_{234} + l_3 c_1 c_{23} + l_2 c_1 c_2 + d_5 c_1 s_{234} \\ s_1 c_{234} c_5 - c_1 s_5 & -s_1 c_{234} s_5 - c_1 c_5 & s_1 s_{234} & a_4 s_1 c_{234} + l_3 s_1 c_{23} + l_2 s_1 c_2 + d_5 s_1 s_{234} \\ s_{234} c_5 & -s_{234} s_5 & -c_{234} & a_4 s_{234} + l_3 s_{23} + l_2 s_2 + d_1 - d_5 c_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matrice è stata scritta nella sua forma con notazione compatta dove valgono le seguenti relazioni:

$$\begin{aligned}
c_1 &= \cos\theta_1, & s_1 &= \sin\theta_1 \\
c_2 &= \cos\theta_2, & s_2 &= \sin\theta_2 \\
c_5 &= \cos\theta_5, & s_5 &= \sin\theta_5 \\
c_{23} &= \cos(\theta_2 + \theta_3), & s_{23} &= \sin(\theta_2 + \theta_3) \\
c_{234} &= \cos(\theta_2 + \theta_3 + \theta_4), & s_{234} &= \sin(\theta_2 + \theta_3 + \theta_4)
\end{aligned}$$

## 2.2 Cinematica diretta

La matrice del cambiamento delle coordinate  $T_0^5$  si compone di una sottomatrice quadrata  $R_0^5$ , che indica la rotazione di  $O_5$  rispetto a  $O_0$  e un vettore colonna  $\vec{p}$ , che indica la traslazione di  $O_5$  rispetto a  $O_0$ . Questa traslazione rappresenta le coordinate dell'handeffector e si ricava nel seguente modo:

$$T_0^5 * = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_4 c_1 c_{234} + l_3 c_1 c_{23} + l_2 c_1 c_2 + d_5 c_1 s_{234} \\ a_4 s_1 c_{234} + l_3 s_1 c_{23} + l_2 s_1 c_2 + d_5 s_1 s_{234} \\ a_4 s_{234} + l_3 s_{23} + l_2 s_2 + d_1 - d_5 c_{234} \\ 1 \end{bmatrix}$$

Solo le prime tre componenti rappresentano le coordinate  $[x_d, y_d, z_d]$ , la quarta è superflua. In conclusione la terna di coordinate dell'handeffector rispetto a  $O_0$  risulta:

$$\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} = \begin{bmatrix} a_4 c_1 c_{234} + l_3 c_1 c_{23} + l_2 c_1 c_2 + d_5 c_1 s_{234} \\ a_4 s_1 c_{234} + l_3 s_1 c_{23} + l_2 s_1 c_2 + d_5 s_1 s_{234} \\ a_4 s_{234} + l_3 s_{23} + l_2 s_2 + d_1 - d_5 c_{234} \end{bmatrix}$$

Dallo studio della struttura del robot ci si rende conto che le coordinate del polso risultano:

$$\begin{bmatrix} \textcolor{red}{x}_{polso} \\ \textcolor{green}{y}_{polso} \\ \textcolor{blue}{z}_{polso} \end{bmatrix} = \begin{bmatrix} l_3 c_1 c_{23} + l_2 c_1 c_2 \\ l_3 s_1 c_{23} + l_2 s_1 c_2 \\ l_3 s_{23} + l_2 s_2 + d_1 \end{bmatrix}$$

Ed è quindi possibile riscrivere la terna di coordinate nel seguente modo:

$$\begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} = \begin{bmatrix} a_4 c_1 c_{234} + d_5 c_1 s_{234} + \textcolor{red}{x}_{polso} \\ a_4 s_1 c_{234} + d_5 s_1 s_{234} + \textcolor{green}{y}_{polso} \\ a_4 s_{234} - d_5 c_{234} + \textcolor{blue}{z}_{polso} \end{bmatrix}$$

Infine è anche possibile ricavare l'orientamento dell'handeffector tramite lo studio degli angoli di Roll e Pitch. In particolare chiameremo  $\omega_d$  l'angolo di Roll e  $\beta_d$  l'angolo di Pitch, quest'ultimo definito come l'angolo tra l'orizzontale e l'asse  $z_5$ .

$$\begin{aligned}
\omega_d &= \theta_5 \\
\beta_d &= \frac{\pi}{2} - \theta_2 - \theta_3 - \theta_4
\end{aligned}$$

## 2.3 Cinematica inversa

L'obiettivo principale del progetto è di controllare il robot affinché raggiunga una posizione desiderata, ovvero fornendo una terna di coordinate  $[x_d, y_d, z_d]$  deve essere possibile calcolare  $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]$ . Inoltre, bisogna fornire un orientamento desiderato all'handeffector tramite gli angoli di Roll e di Pitch, rispettivamente  $\omega_d$  e  $\beta_d$ .

Considerando come noti:  $\vec{p}_e = [x_d, y_d, z_d]^T$  e  $\omega_d, \beta_d$  ricaviamo  $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5]$

L'angolo  $\theta_1$  si ottiene calcolando:  $\text{atan2}(y_d, x_d)$

Riprendiamo le equazioni di  $x_d$  e  $y_d$ :

$$\begin{aligned} x_d &= a_4 c_1 c_{234} + l_3 c_1 c_{23} + l_2 c_1 c_2 + d_5 c_1 s_{234} = a_4 c_1 c_{234} + d_5 c_1 s_{234} + \textcolor{red}{x}_{polso} \\ y_d &= a_4 s_1 c_{234} + l_3 s_1 c_{23} + l_2 s_1 c_2 + d_5 s_1 s_{234} = a_4 s_1 c_{234} + d_5 s_1 s_{234} + \textcolor{green}{y}_{polso} \end{aligned}$$

Ed ora sviluppiamo:  $\textcolor{red}{x}_{polso} c_1 + \textcolor{green}{y}_{polso} s_1 = (c_1^2 * s_1^2)(l_3 c_{23} + l_2 c_2) = l_3 c_{23} + l_2 c_2$

Introduciamo due variabili ausiliarie che ricaveremo esplicitamente in seguito:

$$\begin{aligned} A_1 &= \textcolor{red}{x}_{polso} c_1 + \textcolor{green}{y}_{polso} s_1 = l_3 c_{23} + l_2 c_2 \\ A_2 &= \textcolor{blue}{z}_{polso} - d_1 = l_3 s_{23} + l_2 s_2 \end{aligned}$$

Sviluppiamo la somma dei quadrati e troviamo  $\theta_3$ :

$$A_1^2 + A_2^2 = l_3^2 + l_2^2 + 2l_3 l_2 c_3 \rightarrow c_3 = \frac{A_1^2 + A_2^2 - l_3^2 - l_2^2}{2l_3 l_2} \rightarrow \theta_3 = \arccos\left(\frac{A_1^2 + A_2^2 - l_3^2 - l_2^2}{2l_3 l_2}\right)$$

Riprendiamo le definizioni di  $A_1$  e  $A_2$  impostando il sistema nelle incognite  $c_2$   $s_2$ :

$$\begin{cases} A_1 = l_3 c_{23} + l_2 c_2 \\ A_2 = l_3 s_{23} + l_2 s_2 \end{cases} \Rightarrow \begin{cases} A_1 = l_3(c_2 c_3 - s_2 s_3) + l_2 c_2 \\ A_2 = l_3(c_2 s_3 + s_2 c_3) + l_2 s_2 \end{cases} \Rightarrow \begin{cases} A_1 = (l_3 c_3 + l_2)c_2 - l_3 s_3 s_2 \\ A_2 = l_3 s_3 c_2 + (l_3 c_3 + l_2)s_2 \end{cases}$$

Risolviendo il sistema troviamo:

$$\begin{aligned} c_2 &= \frac{(l_3 c_3 + l_2)A_1 + l_3 s_3 A_2}{l_3^2 + l_2^2 + 2l_3 l_2 c_3} \\ s_2 &= \frac{(l_3 c_3 + l_2)A_2 - l_3 s_3 A_1}{l_3^2 + l_2^2 + 2l_3 l_2 c_3} \end{aligned}$$

Ricaviamo così:  $\theta_2 = \arctan\left(\frac{s_2}{c_2}\right) = \text{atan2}((l_3 c_3 + l_2)A_2 - l_3 s_3 A_1, (l_3 c_3 + l_2)A_1 + l_3 s_3 A_2)$

Risulta immediato calcolare  $\theta_4$  e  $\theta_5$  come segue:

$$\beta_d = \frac{\pi}{2} - \theta_2 - \theta_3 - \theta_4 \rightarrow \theta_4 = \frac{\pi}{2} - \theta_2 - \theta_3 - \beta_d$$

$$\omega_d = \theta_5 \rightarrow \theta_5 = \omega_d$$

Restano da calcolare solo  $A_1$  e  $A_2$ :

$$x_{polso} = x_d - a_4 c_1 c_{234} - d_5 c_1 s_{234}$$

$$y_{polso} = y_d - a_4 s_1 c_{234} - d_5 s_1 s_{234}$$

$$z_{polso} = z_d - a_4 s_{234} + d_5 c_{234}$$

$$\text{Si nota che: } \beta_d = \frac{\Pi}{2} - \theta_2 - \theta_3 - \theta_4 \rightarrow \theta_2 + \theta_3 + \theta_4 = \frac{\Pi}{2} - \beta_d$$

Quindi risulta:

$$c_{234} = \cos(\theta_2 + \theta_3 + \theta_4) = \cos(\frac{\Pi}{2} - \beta_d) = \sin(\beta_d)$$

$$s_{234} = \sin(\theta_2 + \theta_3 + \theta_4) = \sin(\frac{\Pi}{2} - \beta_d) = \cos(\beta_d)$$

Pertanto è possibile riscrivere le equazione come segue:

$$x_{polso} = x_d - a_4 c_1 \sin(\beta_d) - d_5 c_1 \cos(\beta_d)$$

$$y_{polso} = y_d - a_4 s_1 \sin(\beta_d) - d_5 s_1 \cos(\beta_d)$$

$$z_{polso} = z_d - a_4 \cos(\beta_d) + d_5 \sin(\beta_d)$$

Concludiamo calcolando definitivamente  $A_1$  e  $A_2$ :

$$A_1 = x_{polso} c_1 + y_{polso} s_1 = x_d c_1 y_d s_1 - a_4 \sin(\beta_d) - d_5 \cos(\beta_d)$$

$$A_2 = z_{polso} - d_1 = z_d - a_4 \cos(\beta_d) + d_5 \sin(\beta_d) - d_1$$

## 2.4 Formulario

I passi da eseguire per calcolare gli angoli  $\theta_i$  sono i seguenti:

- $\theta_1 = \text{atan2}(y_d, x_d)$
- $A_1 = x_d c_1 + y_d s_1 - a_4 \sin(\beta_d) - d_5 \cos(\beta_d)$
- $A_2 = z_d - a_4 \cos(\beta_d) + d_5 \sin(\beta_d) - d_1$
- $\theta_3 = \arccos(\frac{A_1^2 + A_2^2 - l_3^2 - l_2^2}{2 l_3 l_2})$
- $\theta_2 = \text{atan2}((l_3 c_3 + l_2) A_2 - l_3 s_3 A_1, (l_3 c_3 + l_2) A_1 + l_3 s_3 A_2)$
- $\theta_4 = \frac{\Pi}{2} - \theta_2 - \theta_3 - \beta_d$
- $\theta_5 = \omega_d$

## 3 MATLAB

### 3.1 Descrizione codice MATLAB

Al fine di verificare la veridicità delle formule ricavate per il calcolo della cinematica inversa, si è reso indispensabile la realizzazione di uno script MATLAB. Nello specifico il codice si occupa di calcolare gli angoli  $\theta_i$  tramite la cinematica inversa e successivamente di calcolare la cinematica diretta utilizzando gli stessi angoli appena ricavati. Siccome la terna prodotta dalla cinematica diretta coincide con quella passata in ingresso alla cinematica inversa, si può affermare che le formule utilizzate sono corrette. Infine come ulteriore verifica viene anche calcolata la matrice  $T_0^5$ , dalla quale si può osservare il vettore  $\vec{p}$  coincidere a sua volate con le altre due terne prima citate.

#### 3.1.1 Codice MATLAB

```
1      %% Calcolo cinematica inversa
2
3      % calcolo theta1
4      theta1 = atan2(y_d, x_d);
5
6      % calcolo A1 e A2
7      A1 = x_d*cos(theta1) + y_d*sin(theta1) - a4*sin(beta_d) - d5*cos(beta_d);
8      A2 = z_d - a4*cos(beta_d) + d5*sin(beta_d) - d1;
9
10     % calcolo theta3
11     num = A1^2 + A2^2 - l2^2 - l3^2;
12     den = 2*l2*l3;
13     theta3 = gomitto*acos(num/den);
14
15     % calcolo theta2
16     S_2 = (l2 + l3*cos(theta3))*A2 - l3*sin(theta3)*A1;
17     C_2 = (l2 + l3*cos(theta3))*A1 + l3*sin(theta3)*A2;
18     theta2 = atan2(S_2, C_2);
19
20     % calcolo theta4
21     theta4 = pi/2 - theta2 - theta3 - beta_d;
22
23     %calcolo theta5
24     theta5 = omega_d;
25
26     %% Calcolo della cinematica diretta per verifica
27
28     Xd = c1*( a4*c234 + l3*c23 + l2*c2 + d5*s234 );
29     Yd = s1*( a4*c234 + l3*c23 + l2*c2 + d5*s234 );
30     Zd = a4*s234 + l3*s23 + l2*s2 + d1 - d5*c234;
31
32     %% Sezione EXTRA
33
34     T01 = [c1 0 s1 0; s1 0 -c1 0; 0 1 0 d1; 0 0 0 1];
35
36     T12 = [c2 -s2 0 l2*c2; s2 c2 0 l2*s2; 0 0 1 0; 0 0 0 1];
37
38     T23 = [c3 -s3 0 l3*c3; s3 c3 0 l3*s3; 0 0 1 0; 0 0 0 1];
39
40     T34 = [c4 0 s4 a4*c4; s4 0 -c4 a4*s4; 0 1 0 0; 0 0 0 1];
41
42     T45 = [c5 -s5 0 0; s5 c5 0 0; 0 0 1 d5; 0 0 0 1];
43
44     % matrice del cambiamento delle coordinate da 0_0 a 0_5
45     T05 = T01*T12*T23*T34*T45
```

Listing 1: Alcune sezioni riassuntive del codice MATLAB

## 4 Processing

Il ruolo del programma scritto in ambiente Processing è quello di calcolare la cinematica del robot, disegnare fedelmente il render 3D dello Scorbobot e comunicare gli angoli calcolati alla scheda Arduino UNO.

### 4.1 Comunicazione seriale

Il protocollo di comunicazione è stato progettato con l'intento di prevenire connessioni indesiderate con periferiche diverse dalla scheda Arduino UNO, che si occupa di controllare il robot. Nello specifico ogni comunicazione tra Processing e Arduino avviene con mutuo scambio di codici identificativi, rispettivamente **PRO04IDE** è l'identificativo di processing, mentre **ARO22ARL** è l'identificativo della scheda Arduino UNO. Quando il programma viene avviato per la prima volta esegue il tentativo di connessione con Arduino inviando il proprio id e aspettando la risposta della scheda. Il numero di tentativi, per ogni porta seriale individuata, è limitato da un tempo massimo per evitare che il programma resti perennemente alla ricerca della scheda Arduino. Questo significa che per utilizzare il programma è necessario collegare la scheda Arduino UNO al PC con il corretto firmware caricato in memoria. Nel caso in cui la comunicazione viene avviata correttamente Processing mostra all'utente una finestra di dialogo in cui si chiede di premere il tasto CTRL per avviare il disegno del render 3D. La comunicazione degli angoli avviene sempre sotto forma di caratteri e non nel loro valore intero decimale, questo per ovviare a possibili problemi di compatibilità tra i formati delle variabili di Processing e quelli di Arduino. Ci sono due possibili tipi di comunicazioni, la prima serve per inviare un normale pacchetto di sei angoli, la seconda è una leggera variante della prima che serve ad inviare un pacchetto indicizzato di 6 angoli. Il secondo metodo è definito indicizzato perché i 6 angoli fanno parte di un frames i-esimo di un movimento complesso.

```
1      /*
2      * Invia gli angoli da far attuare ai servomotori:
3      * prima comunica il codice identificativo di questo programma (PROCESSING_ID)
4      * successivamente comunica i 6 angoli nella loro rappresentazione a caratteri
5      * Se un numero ha meno di tre cifre intere (come 7) viene inviata sempre
6      * una stringa da tre caratteri (come "007") per motivi di compatibilità
7      * e semplicità nella ricezione dei dati.
8      */
9      void serialSendPositions(float[] angle) {
10         if (ack_flag == true)
11         {
12             ack_flag = false;
13             port.write(PROCESSING_ID);
14
15             for (j=0; j<MOTORS_NUM; j++)
16             {
17                 point_ascii = Integer.toString((int)deg(angle[j]));
18
19                 if (point_ascii.length() == 2)
20                 {
21                     point_ascii = "0" + point_ascii;
22                 }
23
24                 if (point_ascii.length() == 1)
25                 {
26                     point_ascii = "00" + point_ascii;
27                 }
28
29                 port.write(point_ascii);
30                 positionFile.println("angle[" + j + "]: " + (int)deg(angle[j]));
31                 println("angle[" + j + "]: " + (int)deg(angle[j]) + ", in ascii: " +
32                     point_ascii);
33             }
34         }
35     }
36     /*
```

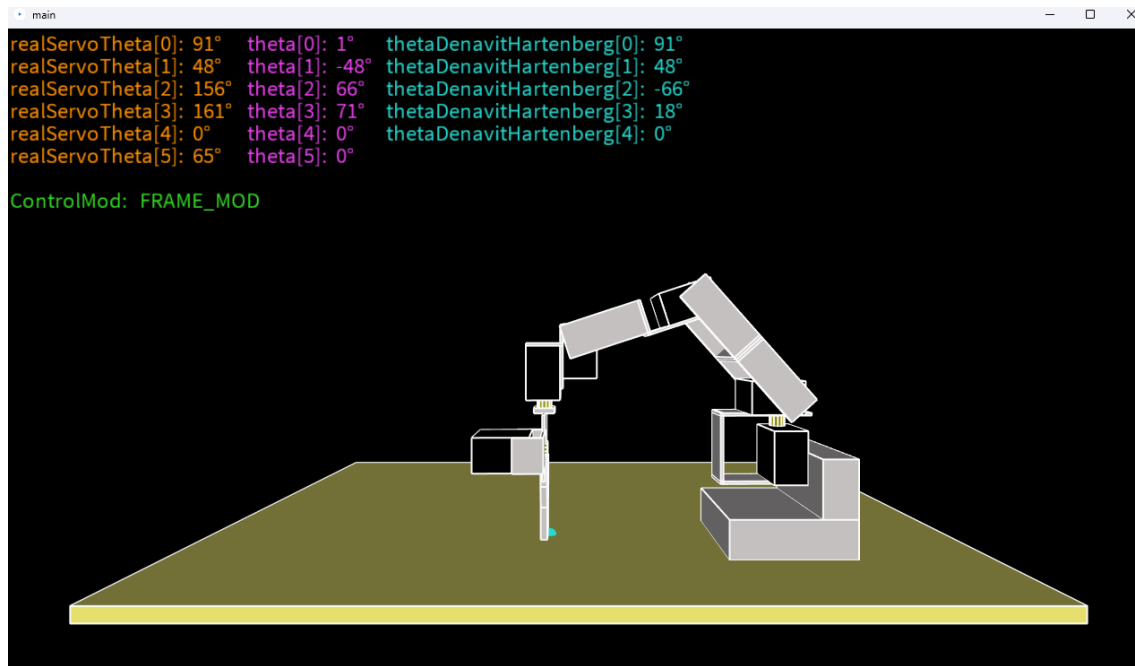


```

37      * Invia gli angoli da far attuare ai servomotori memorizzati nella matrice
        frames[][]
38      * prima comunica il codice identificativo di questo programma (PROCESSING_ID)
39      * successivamente comunica i 6 angoli nella loro rappresentazione a caratteri
40      * Se un numero ha meno di tre cifre intere (come 7) viene inviata sempre
41      * una stringa da tre caratteri (come "007") per motivi di compatibilit 
42      * e semplicit  nella ricezione dei dati.
43      * La matrice frames[FRAMES_NUM][MOTORS_NUM] per ogni indice di FRAMES_NUM
44      * contiene 6 angoli da attuare, l'indice usato per FRAMES_NUM e' framesCount e
        viene
45      * incrementato ad ogni chiamata di questa funzione.
46      */
47      void serialSendFrame() {
48          if (ack_flag == true)
49          {
50              ack_flag = false;
51              port.write(PROCESSING_ID);
52              positionFile.println("[message] --> coordinate frame: " + framesCount);
53              println("[message] --> coordinate frame: " + framesCount);
54
55              for (j=0; j<MOTORS_NUM; j++)
56              {
57                  realServoTheta[j] = rad(frames[framesCount][j]);
58                  point_ascii = Integer.toString(int(frames[framesCount][j]));
59
60                  if (point_ascii.length() == 2)
61                  {
62                      point_ascii = "0" + point_ascii;
63                  }
64
65                  if (point_ascii.length() == 1)
66                  {
67                      point_ascii = "00" + point_ascii;
68                  }
69
70                  port.write(point_ascii);
71                  positionFile.println("angle[" + j + "]: " + frames[framesCount][j]);
72                  println("angle[" + j + "]: " + frames[framesCount][j] + ", in ascii: " +
                        point_ascii);
73              }
74
75              framesCount++;
76              if (framesCount >= FRAMES_NUM) framesCount = 0;
77          }
78      }

```

Listing 2: Funzioni per la comunicazione degli angoli



## 4.2 Render 3D

Il file scorbot3D contiene le funzioni utilizzate per ricostruire il render del robot nello spazio 3D di processing (attenendosi agli assi standard di Processing). Il disegno dello Scorbot è riprodotto in scala ed è il più fedele possibile all'originale. Gli assi di rotazione vengono attuati dalle variabili **theta[i]**, che non sono altro che angoli relativi al sistema di riferimento dello spazio 3D di Processing. Questi angoli hanno il loro corrispettivo nello spazio cartesiano secondo Denavit-Hartenberg con il nome di **thetaDenavitHartenberg[i]** e nella loro rappresentazione fisica, cioè relativa al sistema di riferimento dei singoli servomotori. Questi ultimi sono indicati con il nome di **realServoTheta[i]**.

## 4.3 Modalità e comandi

Il programma presenta tre modalità di funzionamento e ognuna di queste dispone di un set di comandi propri. Sono anche presenti comandi validi in tutte le modalità.

### 4.3.1 Comandi base

In questa sezione vengono descritti i comandi comuni a tutte le tre modalità. Oltre ai comandi da tastiera è anche possibile premere con il mouse su un punto qualsiasi della finestra grafica per riposizionare il centro di riferimento del render 3D all'interno dello spazio di Processing.

I comandi base(da tastiera) sono:

- **tasto f/F:** arresta l'esecuzione del programma salvando su memoria di massa il contenuto dei file di log chiamati **connection\_data.txt** e **position\_data.txt**
- **tasto FRECCIA DESTRA:** ruota lo spazio 3D in senso antiorario rispetto all'asse verticale
- **tasto FRECCIA SINISTRA:** ruota lo spazio 3D in senso orario rispetto all'asse verticale
- **tasto FRECCIA ALTA:** sposta il render 3D in avanti verso l'utente
- **tasto FRECCIA BASSA:** sposta il render 3D in dietro verso l'utente
- **tasto 0:** ruota lo spazio 3D in senso orario rispetto all'asse orizzontale
- **tasto m/M:** permette di cambiare la modalità corrente, alternando fra le tre possibili

### 4.3.2 Manual Mod

Questa è stata la prima modalità ad essere stata sviluppata, poiché la sua funzione era indispensabile per tarare correttamente le relazioni che intercorrono tra le diverse rappresentazioni degli angoli theta precedentemente descritti.

I comandi in **Manual Mod** sono:

- **tasto q/Q**: resetta gli angoli theta al loro valore di default
- **tasto s/S**: cambia il verso di rotazione di tutti i motori
- **tasto 1**: azione la rotazione del motore 1
- **tasto 2**: azione la rotazione del motore 2
- **tasto 3**: azione la rotazione del motore 3
- **tasto 4**: azione la rotazione del motore 4
- **tasto 5**: azione la rotazione del motore 5
- **tasto 6**: azione la rotazione del motore 6

### 4.3.3 Inverse Kinematic Mod

Questa modalità è quella che implementa il calcolo della cinematica inversa e che a sua volta si è resa indispensabile per poter successivamente realizzare la terza modalità.

I comandi in **Inverse Kinematic Mod** sono:

- **tasto g/G**: imposta il gomito alto o basso per il calcolo della cinematica inversa.
- **tasto q/Q**: decrementa il valore della coordinata X desiderata
- **tasto w/W**: incrementa il valore della coordinata X desiderata
- **tasto e/E**: decrementa il valore della coordinata Y desiderata
- **tasto r/R**: incrementa il valore della coordinata Y desiderata
- **tasto t/T**: decrementa il valore della coordinata Z desiderata
- **tasto y/Y**: incrementa il valore della coordinata Z desiderata
- **tasto u/U**: decrementa il valore dell'angolo  $\beta$  desiderato
- **tasto i/I**: incrementa il valore dell'angolo  $\beta$  desiderato
- **tasto o/O**: decrementa il valore dell'angolo  $\omega$  desiderato
- **tasto p/P**: incrementa il valore dell'angolo  $\omega$  desiderato
- **tasto a/A**: apre la pinza
- **tasto s/S**: chiude la pinza

### 4.3.4 Frame Mod

Questa modalità è la più particolare nonché il risultato dello sviluppo delle precedenti modalità. Si occupa di comunicare alla scheda Arduino, in ogni istante di tempo, i valori dei singoli angoli che compomgono un frame. L'insieme dei frame costituisce un movimento complesso che il robot compie ripetutamente, esattamente come se fosse impiegato in un vero processo industriale. I frame sono memorizzati nel file **frames\_data.txt** che il programma legge al suo avvio. Il file è di grandi dimensioni ed è stato realizzato sfruttando l'Inverse Kinematic Mod per calcolare gli angoli corrispondenti al **movimento** studiato prendendo in considerazione la terna  $[x_d, y_d, z_d]$ , gli angoli  $\beta_d, \omega_d$  e la rotazione del motore addetto alla pinza.

Questa modalità non presenta comandi specifici associati ad essa, sono comunque sempre validi i comandi base del programma.

## 5 Arduino

La scheda Arduino UNO svolge la funzione di driver per i servomotori, che movimentano i giunti e la pinza del robot. Inoltre pilota un display LCD 16x2 che mostra all'utente i valori correnti degli angoli attutati dai servomotori rispetto al loro proprio sistema di riferimento (chiamati **realServoTheta[i]** su Processing).

### 5.1 schema elettrico

Il circuito elettrico è costituito da un alimentatore da laboratorio in grado di erogare fino a 300W in corrente continua, 6 servomotori, 1 display LCD, cavi per i collegamenti, il microcontrollore Arduino UNO e 2 breadboard per la prototipazione. I servomotori sono controllati tramite segnali PWM, che la scheda Arduino UNO genera in base ai valori ricevuti dal programma su Processing tramite la comunicazione seriale.

### 5.2 codice

```
1      void setup() {
2
3          Serial.begin(9600);
4          lcd.begin(16, 2);
5          lcd.print("Hello Scorbot!");
6          myRobot.setupRobot();
7
8          pinMode(PIN_LED, OUTPUT);
9          digitalWrite(PIN_LED, LOW);
10
11         if (Serial.available() > 0) { // pulisco il buffer della porta seriale
12             while (Serial.read() != -1);
13         }
14
15         if (Serial.availableForWrite()) {
16             Serial.write(ARDUINO_ID);
17         }
18
19         while (loop_flag) {
20             if (Serial.available() >= strlen(PROCESSING_ID)) {
21                 id = Serial.readString();
22
23                 if (id.equals(PROCESSING_ID)) {
24                     digitalWrite(PIN_LED, HIGH);
25                     loop_flag = false;
26                 }
27             }
28         }
29
30         point_ascii[0][3] = '\0';
31         point_ascii[1][3] = '\0';
32         point_ascii[2][3] = '\0';
33         point_ascii[3][3] = '\0';
34         point_ascii[4][3] = '\0';
35         point_ascii[5][3] = '\0';
36     }
37
38     void loop() {
39
40         if (Serial.available() >= strlen(PROCESSING_ID)+3*MOTORS_NUM) {
41             Serial.readBytes(buffer, strlen(PROCESSING_ID));
42             id = buffer;
43
44             if (id.equals(PROCESSING_ID)) {
45
```

```

46     for(i=0; i<MOTORS_NUM; i++) {
47         Serial.readBytes(&point_ascii[i][0], 3);
48     }
49
50     pos.x1 = atoi(&point_ascii[0][0]);
51     pos.x2 = atoi(&point_ascii[1][0]);
52     pos.x3 = atoi(&point_ascii[2][0]);
53     pos.x4 = atoi(&point_ascii[3][0]);
54     pos.x5 = atoi(&point_ascii[4][0]);
55     pos.x6 = atoi(&point_ascii[5][0]);
56
57     lcd.clear();
58     lcd.home();
59
60     lcd.print(&point_ascii[0][0]);
61     lcd.print("-");
62     lcd.print(&point_ascii[1][0]);
63     lcd.print("-");
64     lcd.print(&point_ascii[2][0]);
65
66     lcd.setCursor(0,1);
67
68     lcd.print(&point_ascii[3][0]);
69     lcd.print("-");
70     lcd.print(&point_ascii[4][0]);
71     lcd.print("-");
72     lcd.print(&point_ascii[5][0]);
73
74     myRobot.setPosition(&pos);
75
76     if (Serial.availableForWrite()) {
77         Serial.write(ARDUINO_ID);
78     }
79 }
80 }
81 }

```

Listing 3: Parti principali del firmware

## Riferimenti bibliografici

- [1] F. Martinelli, *Denavit-Hartenberg*, 11/10/2022, dispense del corso di Robotica con Laboratorio, <http://robot2.disp.uniroma2.it/~fmartine/RobLab/appuntiLezioni/DenHart.pdf>
- [2] F. Martinelli, *Cinematica diretta e inversa del robot SCORBOT*, 14 e 18/10/2022, dispense del corso di Robotica con Laboratorio, <http://robot2.disp.uniroma2.it/~fmartine/RobLab/appuntiLezioni/SCORBOT.pdf>
- [3] G. Ortolani, E. Venturi, *Manuale di elettrotecnica, elettronica e automazione*, 2017, Hoepli.
- [4] Documentazione ambiente di sviluppo Processing, <https://processing.org/reference>
- [5] Documentazione ambiente di sviluppo Arduino, <https://www.arduino.cc/reference/en/>