UNIVERSITÀ DEGLI STUDI DI PALERMO

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA
SEDE DI PALERMO

PROGETTAZIONE E IMPLEMENTAZIONE DI UN LINGUAGGIO BASATO SU COMUNICAZIONE SERIALE PER LA CREAZIONE ED ESECUZIONE DI COMPORTAMENTI COMPLESSI PER UN ROBOT UMANOIDE ROBOVIE-M

Tesi di laurea di:

Allievo Ing. Marisa Saporito

Relatori:

Prof. Ing. Antonio Chella

Dott. Ing. Rosario Sorbello

ANNO ACCADEMICO 2005 - 2006

Prefazione

L'innovazione tecnologica nel campo della robotica ha portato al particolare sviluppo di uno dei suoi rami: la robotica umanoide.
Il robot, per definizione, è antropomorfo e prima ancora di pensare deve saper camminare, non rotolare, e nemmeno strisciare o gironzolare a quattro zampe. Niente ruote, rotelle, cingoli, cinghie, sfere o altri surrogati per muoversi dunque, ma due vere e proprie gambe per muoversi in posizione eretta.
Per l'essere umano camminare è una delle attività più semplici, ma coinvolge il controllo di un numero di variabili talmente elevato che la versione artificiale di due gambe che camminano ha avuto bisogno di quasi venti anni per trovare un'espressione soddisfacente.
In questa tesi è descritto il lavoro di ricerca e sperimentazione che ha portato alla realizzazione di comportamenti, in simulazione e reali, per un robot di tipo umanoide.
Dal punto di vista dell'implementazione, è stato sviluppato un linguaggio che consente la gestione del sistema di controllo del movimento del robot umanoide.
Grazie a questo linguaggio è stato possibile definire comportamenti ad alto livello, come la camminata o la rialzata da terra, e sviluppare comportamenti complessi, come combinazione di comportamenti elementari, che hanno portato alla realizzazione di una libreria di comportamenti.
Questa tesi s'inquadra all'interno di un progetto sperimentale che prevede la realizzazione e l'implementazione di un software per la visione e la movimentazione del robot umanoide *Robovie-M*. Il progetto punta alla realizzazione di un ambiente di sviluppo, che consente di elaborare i dati provenienti dalla telecamera del robot e, conseguentemente di agire da remoto sui motori montati ai giunti in modo tale da far eseguire al robot dei comportamenti in relazione a quello che la telecamera percepisce.

Per la realizzazione di questo progetto si è fatto uso del robot *Robovie-M* versione 3 prodotto dalla ditta giapponese *VStone*, e di una telecamera *wireless* operante alla frequenza di 2.4 Ghz fissata direttamente sulla parte superiore del robot.

La piattaforma hardware utilizzata è caratterizzata da risorse hardware limitate, specialmente per quando concerne la potenza di calcolo e la memoria a disposizione. Come ambiente di test per il robot umanoide è stato scelto *RoboCup*[10], in particolare *l'Humanoid Kid Size League*[9], che ha come obiettivo lo sviluppo di soluzioni per la robotica autonoma.

La tesi è così strutturata:

Capitolo 1: analisi di tutti i sistemi di locomozione bipede presenti in letteratura in modo da poter giustificare l'approccio statico utilizzato.

Capitolo 2: presentazione degli strumenti necessari per la realizzazione di un linguaggio e del relativo traduttore.

Capitolo 3: introduzione ed analisi di tutti gli strumenti software utilizzati per il primo approccio con il robot.

Capitolo 4: analisi dello stato dell'arte nel campo della robotica umanoide nella categoria *Kid Size,* descrizione del robot Robovie-M.

Capitolo 5: descrizione dell'obiettivo della tesi ed il suo contesto applicativo.

Capitolo 6: descrizione della realizzazione applicativa proposta.

Capitolo 7: analisi critica dei risultati raggiunti e valutazione sui possibili sviluppi futuri.

L' **Appendice A** include i moduli relativi al codice realizzato ed i file sorgenti dell'analizzatore sintattico e lessicale, mentre l'**Appendice B** contiene le istruzioni sull'installazione del Software *RobovieMovement*.

Introduzione

La ricerca sui robot in generale, ed in particolare sui robot umanoidi sta avendo negli ultimi anni una forte crescita, dovuta principalmente alle notevoli innovazioni tecnologiche.

Questo tipo di dispositivi è generalmente considerato come lo sviluppo naturale del campo della robotica mobile. Un robot, dotato di gambe ed in particolare di due gambe, può operare in un ambiente conforme all'uomo in modo più efficiente rispetto ai robot su ruote o dotati di sistemi di locomozione diversi, e quindi avere un'area d'applicazione più ampia rispetto ai robot non bipedi.

Un robot bipede dovrebbe essere in grado di camminare con un'andatura stabile e con un'ampia tolleranza ai disturbi, ma anche avere comportamenti più complessi che gli permettano di muoversi su un terreno accidentato o interagire efficientemente con il proprio ambiente, talvolta cooperando con gli stessi esseri umani.

Gli aspetti da migliorare sono principalmente legati alla costruzione meccanica e agli algoritmi di controllo dell'equilibrio: il problema della camminata, infatti, noto per la sua complessità, è reso particolarmente difficile dalla sua non-linearità e dal fatto di essere un processo intrinsecamente caotico e instabile, la cui dinamica è soggetta a fenomeni di discontinuità e a variazioni temporali.

Il robot utilizzato per i miei esperimenti è *Robovie-M* versione 3, un umanoide dotato di 22 gradi di libertà, alto 29 cm e dal peso di circa 1,9 kg.

Ogni grado di libertà è associato ad un servomotore che può essere controllato per mezzo della seriale integrata nella scheda programmabile.

Questa versione di *Robovie-M* non è dotata di un sistema di visione, pertanto è stata applicata una telecamera *wireless* frontale sulla testa del robot in grado di acquisire immagini alla risoluzione di 628x512 pixel.

Il robot é in grado di eseguire svariati compiti, tra i quali camminare, stendersi per poi rialzarsi.

Ad oggi i movimenti compiuti dal robot con questo tipo di configurazione hardware sono ottenuti come una sequenza di configurazione dei giunti. Iniziando da una configurazione di partenza in cui il robot si trova in posizione eretta, sono applicate diverse configurazioni per ottenere i movimenti voluti. Come risultati si hanno movimenti lenti e stabili e per ogni configurazione determinata, il robot si trova in una posizione d'equilibrio statico.

Interrompendo il movimento in un punto qualsiasi della sequenza il robot rimane in quello stato senza cadere. In modo del tutto sperimentale si ottiene che il baricentro del robot si trovi sempre all'interno della base d'appoggio.

Sulla base delle metodologie usate per la realizzazione di movimenti per robot che realizzano comportamenti con stabilità statica, si è pensato di creare un programma che permette di descrivere, mediante un linguaggio a hoc, i singoli stage e come successione di questi definire i comportamenti complessi.

Mediante questo linguaggio sarà possibile programmare i vari movimenti e , secondo il tipo di compilazione selezionata si potrà :

- Eseguire la simulazione del comportamento;
- Testare direttamente sul robot il comportamento.

L'idea che sta alla base di questo software, è quella di sviluppare il movimento off-line (senza la presenza del robot), provarlo nel simulatore e, solo dopo aver accertato che il movimento costruito è corretto, testarlo sul robot inviando il comportamento tramite seriale.

Capitolo 1

La camminata artificiale e umana

1.1 Introduzione

La tecnica di locomozione animale, largamente più diffusa in natura, è quella che prevede l'utilizzo di gambe, piedi, zampe, in generale d'arti.
I robot vogliono quindi prendere spunto dalla più funzionale forma di locomozione che l'evoluzione ha selezionato: l'uomo.
La necessità di realizzare un robot di concezione antropomorfa viene classicamente illustrata con il problema delle scale. Non si possono modificare case e costruzioni per rimuovere le scale così come non si può limitare lo spazio di lavoro di un robot solo a zone perfettamente pavimentate. Sebbene sia possibile studiare particolari attrezzature che permettano al robot di superare scale e gradini, queste normalmente sono una grande limitazione nelle funzionalità della macchina e comunque non permettono di risolvere il problema degli spostamenti sui terreni sconnessi.
Fornendo al robot lo stesso sistema di locomozione di cui è dotato l'uomo, gli si può chiede di operare negli stessi ambienti nei quali l'uomo lavora, e quindi:

- Interagire in modo più naturale con l'uomo;
- Operare in ambienti e manipolare oggetti che sono stati progettati e costruiti in base alla conformazione fisica e alle esigenze dell'uomo;
- Cooperare ed eventualmente assistere l'uomo;
- Non nuocere all'uomo (controllo di rigidità).

La concezione di un robot mobile deve quindi tenere in conto il tipo di spazio nel quale la macchina dovrà agire e se il nostro progetto vuole avere la massima

applicabilità e flessibilità allora il nostro robot dovrà necessariamente avere delle gambe.

1.2 Una panoramica sulla camminata bipede

Una delle principali caratteristiche che si richiedono ad un robot è l'abilità di sapersi muovere nel proprio spazio di lavoro. Per questo, di fianco ai robot manipolatori, sono nati dapprima quelli muniti di ruote e poi quelli provvisti di gambe. Questo ultimo sistema di locomozione è sicuramente più complesso dato che richiede più motori e arti, ma a fronte di questa maggiore complessità si ottengono delle prestazioni che i robot su due ruote non possono fornire. I robot su gambe possono muoversi anche su terreni accidentati, superare ostacoli in maniera più veloce ed elegante, ed eseguire una gamma di movimenti più vasta.Una particolare classe dei robot su gambe è quella rappresentata dai robot bipedi. I vantaggi di questa categoria di robot sono rappresentati:

- Dal numero di gambe e quindi un numero minore di membri meccanici ed attuatori da gestire;
- Dal fatto che molti ambienti in cui i robot mobili devono operare sono realizzati per gli esseri umani.

Un robot bipede, simile all'uomo, può quindi muoversi con disinvoltura in questi luoghi, ma di controparte, non indifferente è il problema del controllo dell'equilibrio durante l'esecuzione dei vari movimenti.
Si distinguono due tecniche diverse per implementare la camminata mediante un sistema artificiale bipede: la camminata attiva e la camminata passiva.
Nella camminata attiva ogni grado di libertà del bipede è equipaggiato con un'attuatore e vengono fissate a priori delle traiettorie per i piedi e per il bacino

in funzione del tipo d'andatura che si vuole eseguire (passeggio, corsa …). Questo approccio è attualmente adottato dai robot giapponesi (*Wabian, Wabian-2, Wabot, Asimo*), americani (*M2, DB*);

Nella camminata passiva si considera il bipede come un doppio pendolo inverso dove non necessariamente tutti i giunti devono essere equipaggiati con degli attuatori; inoltre la sola energia richiesta nel movimento è quella che permette di far iniziare o fermare il movimento. Naturalmente è necessario compensare anche le forze e i momenti d'attrito.

In base alle caratteristiche della camminata è possibile distinguere diverse tipologie di robot bipedi. I robot che camminano rimanendo sempre in equilibrio statico, ovvero mantenendo la proiezione del proprio centro di massa sempre all'interno dell'area d'appoggio, sono compresi nella classe degli *static walkers*.

Una seconda categoria, chiamata *dynamic walkers*, comprende i sistemi che durante il moto sono sempre in equilibrio dinamico, con la particolarità di avere piedi e caviglie attuati. Sia gli *static walkers* che i *dymanic walkers* realizzano una camminata attiva.Un'ulteriore categoria è dedicata a particolari meccanismi puramente passivi definita *passive walkers* che realizzano la camminata passiva.

Allo stato attuale la ricerca sui robot bipedi è in piena evoluzione. Sono stati realizzati molti prototipi che camminano in maniera statica, ma il maggiore interesse è rivolto a quei sistemi che implementano un'andatura di tipo dinamico.

1.2.1 I passive walkers

I *passive walkers*, sono robot in grado di realizzare una camminata bipede sfruttando solamente la forza di gravità, e questo può realizzarsi solo grazie ad un'attenta distribuzione delle masse. I primi studi sono stati condotti da McGeer[6], Collins et al [11], Garcia et al[12]. L'idea fondamentale che sta dietro

ai *passive walkers* è che in un mondo limitatamente piano, un moto fluido è ammesso solo per una ruota, tipo quella rappresentata visibile nella figura 1.1.

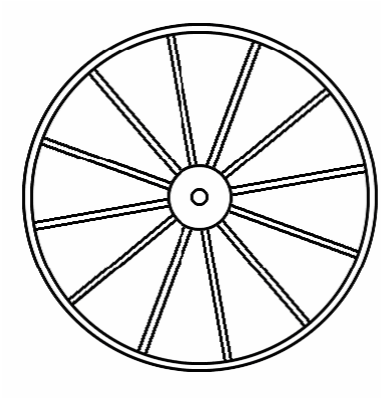


Figura 1.1: La rappresentazione della ruota con cerchione.

Lo studio di McGeer analizza il comportamento di una particolare tipologia di ruota: "Ruota senza cerchione", visibile nella figura 1.2. Si tratta di una ruota costituita di soli raggi che possono essere pensati come delle gambe in serie fissate nell'asse di rotazione. Mentre la normale ruota si muove rotolando su una superficie piana, la ruota senza cerchione, può muoversi anche su terreni irregolari.

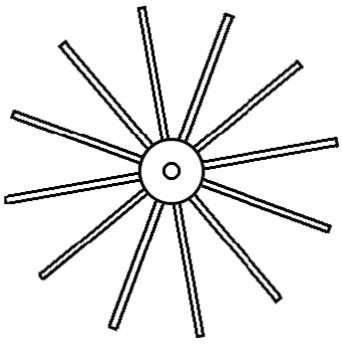


Figura 1. 2: La rappresentazione di una ruota senza cerchione.

Ad ogni contatto con il suolo si collega una perdita d'energia cinetica che porta la ruota a bloccarsi. Tuttavia, se il piano è anche solo leggermente inclinato, l'energia cinetica, persa a causa degli urti, può essere incrementata a scapito dell'energia gravitazionale.

Questa soluzione comporta continui urti e il moto complessivo che ne risulta è tutt'altro che fluido. Una soluzione alternativa all'utilizzo della ruota senza cerchione è proposta dal meccanismo visibile nella figura 1.3 che è caratterizzato da piedi curvi.

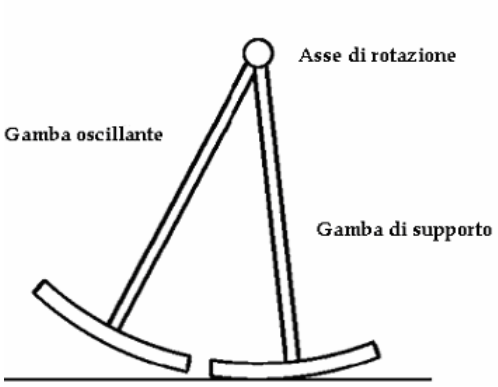


Figura 1. 3: La rappresentazione della soluzione alternativa alla ruota senza cerchione.

A tal proposito si confronti la figura 1.3 con la figura 1.4 estratta dall'originale lavoro di McGeer [6].

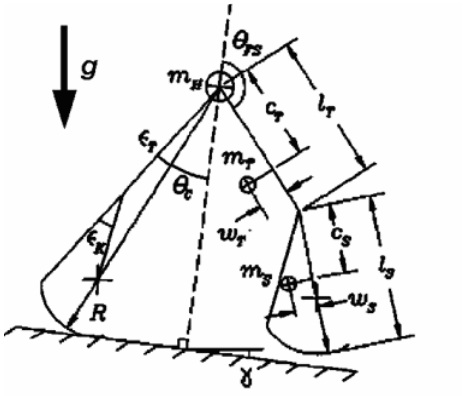


Figura 1. 4: La rappresentazione del primo progetto di Ted McGeer.

Il sistema è ottenuto mantenendo solo un ridotto arco del cerchione esterno, due soli raggi che rendendo mobile il collegamento tra loro. Tale meccanismo ha la stessa cinematica della ruota, quando uno dei due piedi è in contatto con il suolo[15].

Il problema è, quindi di portare in avanti la "gamba oscillante" che non è in contatto con il terreno, in modo da sostituire la "gamba di supporto". Se si realizza questo, si ottiene un movimento del tutto analogo a quello della ruota. Questo problema è stato brillantemente risolto da McGeer, che inserisce una seconda articolazione del ginocchio col compito di spostare il centro di massa della gamba, ottenendo l'avanzamento della gamba oscillante. Al fine di avere un movimento fluido, in un esecutore di camminata passiva si deve prestare attenzione a queste due proprietà:

- E' necessario che il sistema sia caratterizzato da un piede opportunamente sagomato, capace di mantenere costante l'altezza del baricentro rispetto al piano inclinato;
- La massa deve essere concentrata per la maggior parte nel corpo e non nelle gambe, in modo che la dinamica del sistema ne risenta il meno possibile.

Per rendere autonoma questa struttura, è necessario dotarla d'attuatori al fine di fornire un'energia equivalente a quella che fornisce la forza di gravità. Un lavoro che analizza il problema in tal senso, è quello di Collin set al[12], in cui si presentano tre modelli di robot ottenuti da *passive walkers* e dotati di motori di piccola potenza, che possono camminare sul piano. Questa tipologia di robot richiede una potenza ed un controllo nettamente più piccolo delle altre tipologie.
L'utilizzo di due soli parametri per gamba nell'analisi del moto offre comunque soluzioni buone per la locomozione su superfici lisce.
La costruzione di un modello d'esecutore di camminata passiva, ovvero capace di muoversi lungo piani inclinati per effetto gravitazionale, permette di modellare roboticamente un arto inferiore in maniera semplice come si può vedere in figura 1.5 [17].

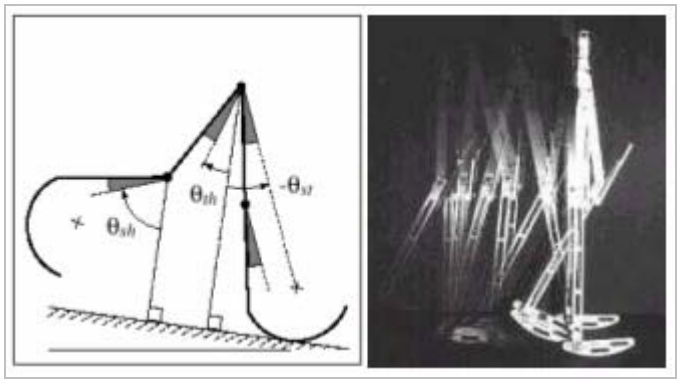


Figura 1.5: La rappresentazione del modello robotico degli arti inferiori.

In questo modo il risultato che si ottiene è comunque qualitativamente sovrapponibile al reale andamento delle variabili di giunto biologiche.

1.2.2 Gli static walkers

Per ottenere una camminata bipede di tipo statico, il primo passo da compiere è quello di mantenere il robot in ogni istante in equilibrio statico, considerando trascurabili gli effetti dinamici. Se queste condizioni sono verificate la strategia di controllo risulta molto semplice.

La categoria degli *static walkers*, comprende tutti quei robot che in ogni istante della camminata si trovano in una posizione d'equilibrio statico. Per ottenere la camminata, quindi, si passa da una posizione d'equilibrio statico alla successiva in maniera lenta, in modo da poter trascurare la dinamica del sistema.

Per mantenere il robot in equilibrio statico è necessario che la proiezione del baricentro del sistema sul piano d'appoggio si trovi in ogni istante all'interno dell'area delimitata dalle piante dei piedi.

Come si può vedere in figura 1.6 sia nel caso di *Supporto Singolo* che nel caso di *Supporto Doppio*, quando la proiezione del baricentro si trova all'interno dell'area delimitata dalle piante dei piedi, ci si trova in una posizione stabile, viceversa ci si trova in una condizione instabile.

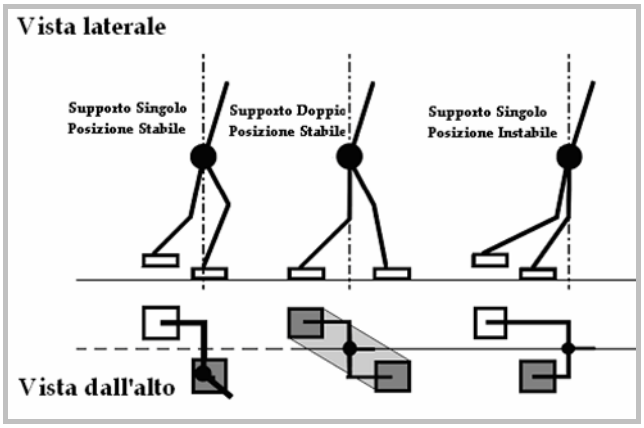


Figura 1. 6 Camminata statica: Vista laterale, Vista dall'alto.

A fronte della semplicità nel controllo, si hanno, quindi, andature con velocità molto ridotte.

Per calcolare il centro di massa siano m_i, $r_i = (x_i; y_i; z_i)$,ω_i I_i rispettivamente il peso, la posizione, la velocità angolare e il momento d'inerzia dell'i-esimo collegamento. Si può calcolare la massa totale del robot m_{all} come nell'equazione (1.1):

$$m_{all} = \sum_{i=0}^{N} m_i \tag{1.1}$$

Le equazioni che descrivono la posizione del centro di massa $r = (r_x; r_y; r_z)$ risultano essere (1.2), (1.3), (1.4):

$$r_x = \frac{\sum_{i=0}^{N} m_i r_{x_i}}{\sum_{i=0}^{N} m_i} = \frac{\sum_{i=0}^{N} m_i r_{x_i}}{m_{all}} \tag{1.2}$$

$$r_y = \frac{\sum_{i=0}^{N} m_i r_{y_i}}{\sum_{i=0}^{N} m_i} = \frac{\sum_{i=0}^{N} m_i r_{y_i}}{m_{all}} \tag{1.3}$$

$$r_z = \frac{\sum_{i=0}^{N} m_i r_{z_i}}{\sum_{i=0}^{N} m_i} = \frac{\sum_{i=0}^{N} m_i r_{z_i}}{m_{all}} \tag{1.4}$$

Di queste, le prime due (1.2), (1.3) rappresentano anche la proiezione sul piano *(x; y)* del baricentro, mentre la terza è una misura del grado di stabilità del sistema.
Riepilogando si ha una camminata statica se fermando il robot in qualunque istante questo si trova in una situazione d'equilibrio. Per avere una camminata statica è necessario avere:

- Piedi grandi, con una estesa base d'appoggio;
- Giunti dell'anca con grossi motori per bilanciare il momento generato dal corpo durante l'alzata di una gamba.

Questo tipo di controllo impone che il centro di massa si proietti all'interno dall'area di supporto dei piedi e che la velocità di rotazione dei giunti siano contenute in modo da poter supporre che le forze inerziali, centrifughe e di Coriolis siano trascurabili.
Quando la dinamica non è più trascurabile, è necessario considerare il sistema di controllo di un *dynamic walker*.

1.2.3 I dynamic walkers

La soluzione per eliminare il vincolo della ridotta velocità degli *static walker* consiste nell'adottare un controllore più complesso, che tenga conto della dinamica del robot. Una classe di robot che opera in questo modo è quella raggruppata sotto il nome di *dyinamic walkers* .
Durante la locomozione questi robot in genere non sono in equilibrio statico, bensì dinamico: questo significa che oltre alle forze gravitazionali bisogna tenere conto anche delle forze d'inerzia e che lo ZMP rimanga sempre all'interno dell'area delimitata dalle piante dei piedi[16].

Il concetto di *Zero Moment Point* fu introdotto formalmente nel 1989 e da allora è diventato d'uso comune nella robotica bipede.
La definizione qui riportata di seguito `e quella d'Arakawa e Fukuda:

Lo ZMP `e quel punto del terreno dove il momento $\mathbf{T}$*: [*$\mathbf{T_x}$*,* $\mathbf{T_y}$*,* $\mathbf{T_z}$ *], generato dalle forze e dalle coppie di reazione vincolare, soddisfa la condizione* $\mathbf{T_x} = 0$*,* $\mathbf{T_y} = 0$*.*

Nel paragrafo precedente si è affermato che per tenere in equilibrio statico un robot è necessario che la proiezione del suo baricentro cada all'interno della sua base d'appoggio, come è visibile nella figura 1.6. Se però si considerano sia le accelerazioni sia le inerzie è possibile che la proiezione del centro di massa esca dell'area delimitata dalle piante dei piedi senza che per questo, il robot cada. Infatti nella teoria dello Zero Moment Point (ZMP), si dimostra che, finche lo ZMP rimane all'interno della superficie di appoggio, la camminata del robot è stabile in senso dinamico.
Analogamente a quanto visto nel paragrafo precedente per calcolare il centro di massa siano m_i, $r_i = (x_i; y_i; z_i)$,ω_i I_i rispettivamente il peso, la posizione, la velocità angolare e il momento di inerzia dell'i-esimo collegamento. Si può calcolare la massa totale del robot m_{all} come nell'equazione (1.5):

$$m_{all} = \sum_{i=0}^{N} m_i \tag{1.5}$$

Le equazioni che descrivono la posizione del centro di massa $r = (r_x; r_y; r_z)$ risultano essere:

$$r_x = \frac{\sum_{i=0}^{N} m_i r_{x_i}}{\sum_{i=0}^{N} m_i} = \frac{\sum_{i=0}^{N} m_i r_{x_i}}{m_{all}} \tag{1.6}$$

$$r_y = \frac{\sum_{i=0}^{N} m_i r_{y_i}}{\sum_{i=0}^{N} m_i} = \frac{\sum_{i=0}^{N} m_i r_{y_i}}{m_{all}} \tag{1.7}$$

$$r_z = \frac{\sum_{i=0}^{N} m_i r_{z_i}}{\sum_{i=0}^{N} m_i} = \frac{\sum_{i=0}^{N} m_i r_{z_i}}{m_{all}} \tag{1.8}$$

A differenza del modello statico, dove si verificava che $(r_x; r_y)$ incluso nella base d'appoggio, in questo caso è necessario considerare anche le grandezze:

$\boldsymbol{M_c}$ il momento attorno al centro di gravità, $\boldsymbol{f} = (f_x; f_y; f_z)$ la reazione vincolare totale che il pavimento esercita sul robot, e $\boldsymbol{T}$ il momento totale che essa esercita attorno ad un punto $\boldsymbol{p} = (p_x; p_y; p_z)$.

Le equazioni dinamiche scritte rispetto al punto p divengono[13], l'equazione d'equilibrio dei momenti:

$$m_{all}(\boldsymbol{r} - \boldsymbol{p}) \times (\ddot{\boldsymbol{r}} + \boldsymbol{g}) + \boldsymbol{M}_c - \boldsymbol{T} = 0 \tag{1.9}$$

e di equilibrio delle forze

$$\boldsymbol{f} = m_{all}(\ddot{\boldsymbol{r}} + \boldsymbol{g}) \tag{1.10}$$

La posizione dello *ZMP* $\boldsymbol{p} = (p_x; p_y)$ rispetto al punto $\boldsymbol{p} = (p_x; p_y; h)$ appartenente al piano

$z = h$, è definito come il punto dove il momento calcolato rispetto al punto p risulti essere $\boldsymbol{T} = (0; 0; \boldsymbol{T}_z)$; di conseguenza le coordinate p_x e p_y possono essere calcolate dalle equazioni 1.9 e 1.10:

$$p_x = r_x - \frac{M_y - m_{all}(r_z - h) \times (\ddot{r}_x + g)}{m_{all}(\ddot{r}_z + g)} \tag{1.11}$$

$$p_y = r_y - \frac{M_x - m_{all}(r_z - h) \times (\ddot{r}_y + g)}{m_{all}(\ddot{r}_z + g)} \tag{1.12}$$

Quando $h = 0$, cioè quando il piano per cui si voglia calcolare lo ZMP sia il pavimento, le equazioni 1.11 e 1.12 possono essere semplificate nel seguente modo:

$$p_x = r_x - \frac{M_y - m_{all} r_z \times (\ddot{r}_x + g)}{f_z} \tag{1.13}$$

$$p_y = r_y - \frac{M_x - m_{all} r_z \times (\ddot{r}_y + g)}{f_z} \tag{1.14}$$

che, riscritte nella notazione utilizzata in Vetturelli [2] esplicitando tutti i termini, divengono:

$$x_{zmp} = \frac{\sum_{i=0}^{N} m_i(\ddot{z} + g)x_i - \sum_{i=0}^{N} \ddot{x} z_i - \sum_{i=0}^{N} I_{iy}\ddot{\theta}_{iy}}{\sum_{i=0}^{N} m_i(\ddot{z} + g)x_i} \tag{1.15}$$

$$y_{zmp} = \frac{\sum_{i=0}^{N} m_i(\ddot{z} + g)y_i - \sum_{i=0}^{N} \ddot{y} z_i - \sum_{i=0}^{N} I_{ix}\ddot{\theta}_{ix}}{\sum_{i=0}^{Nn} m_i(\ddot{z} + g)x_i} \tag{1.16}$$

dove (x_i; y_i; z_i) sono le coordinate del centro di massa di ciascun *link* rispetto al riferimento assoluto, m_i la massa dell'i-esimo *link*, I_{ix} e I_{iy} sono i momenti di inerzia, e θ_{ix} *e* θ_{iy} le velocità angolari lungo gli assi x e y.

Lo *ZMP* esiste sempre, a meno che il denominatore di 1.15 e 1.16 o equivalentemente di 1.13 e 1.14 non sia nullo. Questa situazione risulta essere verificata quando l'accelerazione del robot lungo l'asse z compensa l'accelerazione gravitazionale g, cioè $z = -g$, o equivalentemente $r_z = -g$. In questo caso lo ZMP si trova nel punto all'infinito del piano *(x; y)*.Per garantire la stabilità del sistema è necessario che lo ZMP rimanga sempre all'interno

dell'area definita dai piedi in appoggio. Per stimare a tempo di esecuzione la posizione dello ZMP sono necessari:

- *encoders* ad ogni giunto: servono per misurare la posizione angolare di ciascun giunto;
- sensori di contatto e di forza nei piedi: i primi controllano che il piede sia a contatto con il suolo, i secondi misurano la forza di reazione applicata f necessaria per il calcolo dello *ZMP*; è possibile misurare lo *ZMP* con tre sensori di forza per piede, ma in genere se ne usano almeno quattro;
- accelerometri e giroscopi 3D nel corpo principale: misurano l'orientamento del corpo del bipede rispetto al riferimento fisso del mondo.

Quando lo *ZMP* esce dalla pianta del piede, allora la camminata non è più dinamicamente stabile; in un robot umanoide dotato di arti superiori è possibile, però, riportare la posizione dello *ZMP* all'interno del corpo utilizzando gli arti superiori o il busto.[14][15]
Questa tecnica fornisce un evidente vantaggio nel creare sistemi a catena chiusa, alfine di riprogettare il movimento per rimanere sempre in equilibrio dinamico, una volta calcolato lo *ZMP*. Questo tipo di controllo dinamico rappresenta la modalità più efficace di controllo, al costo di dover gestire un sistema notevolmente complesso e con un elevato numero di ingressi sensoriali.

1.3 La camminata dell'uomo

L'attività di camminare richiede un particolare controllo neuromotorio che artificialmente è molto complesso ricostruire. La dinamica del singolo arto, essendo una struttura a cinematica aperta, diventa simile a quella di un tri-pendolo inverso: i tre segmenti uniti da caviglia, ginocchio e anca formano una costruzione in equilibrio sul piede alla quale il nostro controllo garantisce in ogni momento la stabilità. La stabilità dinamica è realizzata quindi usando le masse e le accelerazioni di tutto quanto il corpo.

Non solo i piedi devono muoversi rispetto al suolo ma anche il bacino, la spina dorsale, le braccia e la testa si devono muovere in sincronismo per mantenere il bilanciamento del sistema. Sebbene complessi questi movimenti si possono analizzare separatamente in modo da evidenziare la meccanica della camminata.

L'uomo rappresenta il risultato finale dell'evoluzione e come tale è il punto di riferimento per la progettazione degli attuali robot bipedi.

Come dedotto dall'analisi appena fatta importante risulta la distribuzione delle masse , che per l'uomo per il 68% del peso corporeo risulta localizzato nelle parti superiori.

I piedi nonostante la loro complessa struttura e delicata funzione rappresentano solo il 10% del peso complessivo. [24]

Durante la camminata i piedi, e quindi le anche le gambe, spingono il corpo in avanti. Una camminata naturale prevede che ogni giunto rimanga leggermente flesso, anche quando la gamba raggiunge la sua massima estensione. Il passo inizia con entrambi i piedi nella posizione di contatto, questo è il punto nel quale il peso si sposta in avanti per generare l'avanzamento e viene sopportato dal piede davanti.

Appena il peso del corpo si trasferisce sul piede davanti, il ginocchio si flette per assorbire lo sforzo. Questa è la posizione nella quale il baricentro del corpo raggiunge il punto più basso della sua traiettoria durante la camminata.

L'energia potenziale persa durante questa fase viene accumulata sotto forma di energia cineteca nelle diverse sezioni del corpo, così da essere restituita durante la successiva fase.

Questo è il primo quarto del ciclo: mentre il corpo si sposta in avanti, il ginocchio si raddrizza ed eleva il corpo fino al punto più alto che il baricentro raggiunge durante la camminata. Questa è chiamata la posizione di passaggio perché è quella nella quale il piede libero oltrepassa la gamba di supporto.

Il piede che sorregge il peso alza il tallone dal suolo mantenendo quindi solo un ristretto contatto sulla punta, insufficiente a generare una pressione di equilibrio anche per via del fatto che l'avanzamento del corpo ha spostato il peso oltre la gamba di appoggio. Il corpo inizia quindi a cadere in avanti e il piede che effettua la corsa di ritorno ruota come un pendolo per prepararsi a colpire il suolo.

La gamba libera entra quindi in contatto esattamente a metà del ciclo di camminamento. La seconda metà sarà poi perfettamente simmetrica alla prima, eventuali differenze risulterebbero in uno zoppicamento durante il passo.

Il centro di massa del corpo può essere molto prossimo ai fianchi e tutto il bilanciamento durante la camminata si basa su questa disposizione. Durante il passo si può osservare un moto assoluto del bacino separabile in due distinti spostamenti.

Durante l'esecuzione del primo passo si osserva una rotazione del bacino lungo l'asse della colonna spinale in avanti e all'indietro insieme alle gambe. Se, ad esempio, la gamba destra è davanti allora la parte destra del bacino ruota rispetto all'asse del corpo in modo da portarsi in avanti. Per questo quando la gamba raggiunge la sua completa estensione anche il bacino è al massimo grado di rotazione. Per mantenere il bilanciamento le spalle devono oscillare nella direzione opposta. In questa fase le spina dorsale rimane relativamente diritta, ovvero non si flette.

Una seconda rotazione avviene invece in corrispondenza della posizione di passaggio: la gamba di appoggio produce un innalzamento relativo della sua

estremità di bacino, forzando così una flessione anche della spina dorsale. Il movimento simmetrico della colonna vertebrale produce nelle spalle una contro rotazione che è speculare a quella del bacino in modo da mantenere in ogni momento l'equilibrio del centro di massa.
Nell'estensione della seconda gamba spalle e bacino ritornano ad essere paralleli. L'avanzamento dell'anca produce però una nuova rotazione del bacino intorno all'asse vertebrale e quindi anche una nuova contro oscillazione nel segmento delle spalle.
Naturalmente nella determinazione e nel mantenimento dell'equilibrio concorrono anche tutte le altri parti del corpo a partire dalle braccia, normalmente utilizzate come appendici di correzione dell'equilibrio dinamico sulla base delle retroazioni sensoriali di accelerazione, fino alla testa, anche se questa, essendo spesso coinvolta nella pianificazione della traiettoria (posizionamento oculare sull'obiettivo), normalmente non ha ruolo rilevante nell'equilibrio umano.[20]

1.4 Conclusioni

In questo capitolo si è cercato di presentare le diverse tipologie di camminata bipede artificiali affrontate in letteratura e di descrivere la camminata umana che con i robot bipedi si cerca di riprodurre. Inoltre si è cercato di mettere in evidenza i pregi che ciascuna classe di robot possiede.
Robovie-M è in grado di eseguire la camminata statica, in quanto possiede grandi piedi per un appoggio maggiore. Il tipo di camminata che ho realizzato per questo robot appartiene alla prima categoria, ossia è di tipo statico. Interrompendo, in un qualsiasi momento, i movimenti che si stanno eseguendo, il robot si trova in una configurazione di equilibrio senza cadere.

Capitolo 2

La creazione di un linguaggio

2.1 Introduzione

Se si decide di realizzare un programma che esegua la compilazione di una sequenza in ingresso, scritta con un particolare linguaggio, ed associ a questa una particolare sequenza d'uscita, si ha la necessità di definire un linguaggio, che permetta di strutturare in maniera opportuna la sequenza d'ingresso ed un traduttore che permetta di avere la sequenza d'uscita corretta.

2.2 La realizzazione di un linguaggio

La sintassi di un linguaggio è l'insieme delle sue regole. Per definire le regole, o produzioni del linguaggio occorre avvalersi di un meta linguaggio diverso da quello di partenza. La descrizione formale di un linguaggio consente di evidenziare le proprietà che possiede e quindi risalire ad una definizione sicura delle caratteristiche e dei limiti del linguaggio ed a un migliore approccio per la messa a punto del programma traduttore. Gli elementi costitutivi del meta linguaggio sono detti elementi terminali mentre gli elementi costitutivi del linguaggio sono detti elementi sintattici o non terminali.
Esempio di descrizione della sintassi di un linguaggio:

<articolo> → *il un*
<sostantivo> → *maschio | femmina*
<soggetto> → *<articolo><sostantivo>*
<oggetto> → *<articolo><sostantivo>*
<gruppo verbale> → *ha <oggetto>*
<frase> → *<oggetto><gruppo verbale>*

La sintassi è accettabile se esiste un solo simbolo non terminale ,detto scopo, che compare unicamente al primo membro di una produzione. Nell'esempio precedente lo scopo è <*frase*>.

L'analisi sintattica procede attraverso trasformazioni di una sequenza fino a raggiungere la configurazione finale <*frase*>.

Date le produzioni del linguaggio ed una frase costruita secondo la sintassi descritta delle produzioni, posso fare un'analisi sintattica della frase fino al raggiungimento della configurazione finale che è lo scopo del linguaggio di cui si sta parlando.

La trasformazione di una sequenza fino al raggiungimento dello scopo può essere rappresentata nella forma di un albero sintattico in cui lo scopo la radice.

Per definire un linguaggio si deve :

- Dire quali sono le parole chiave del linguaggio ;
- Definirne le produzione .

Le produzioni non sono altro che un modo formale di descrivere il diagramma sintattico del linguaggio.

I simboli terminali e le produzioni ci permettono di dire come deve essere fatta la frase tipo del nostro linguaggio; seguendo regole di questo tipo si possono costruire le espressioni sintattiche.

Definito il linguaggio occorre realizzarne il compilatore. Per poter costruire un compilatore occorre associare una semantica alle frasi ben formate del nostro linguaggio e quindi andare a dare un significato alle istruzione del linguaggio.

Per costruire un compilatore si possono utilizzare i linguaggi *Lex* e *Yacc*; il primo ci permette di generare un analizzatore lessicale ed il secondo un analizzatore sintattico[5].

2.3 L'analizzatore lessicale

Il Lex è un linguaggio che permette di generare programmi in grado di elaborare, a livello lessicale, sequenze di caratteri in ingresso. Lex accetta specifiche ad alto livello orientate al problema del riconoscimento di stringhe di caratteri e produce un programma in un linguaggio ad uso comune, detto linguaggio ospite, che riconosce linguaggi definiti da espressioni regolari.
Il codice prodotto da Lex segmenta una sequenza di caratteri forniti in ingresso producendo un segmento per ogni sottosequenza.Al riconoscimento di una stringa, viene eseguita una sezione di programma, fornita dal programmatore e associata alla definizione dell'espressione regolare corrispondente.
Il Lex accetta in ingresso una tabella d'espressioni regolari e di corrispondenti frammenti di programma. La tabella è tradotta in un programma che legge la sequenza in ingresso, la copia in una sequenza d'uscita e la partiziona in stringhe che soddisfano le espressioni fornite. Per ogni stringa riconosciuta, è eseguito il corrispondente frammento di codice. Il riconoscimento delle espressioni è eseguito da un automa a stati finiti deterministico.
Lex in genere viene utilizzato come generatore d'analizzatore sintattico per eseguire la fase d'analisi lessicale. Il compito dell'analizzatore lessicale è quello di estrapolare da espressioni predefinite, quelle che sono considerate parole chiave per l'interpretazione: i *token* che l'analizzatore sintattico provvede poi ad interpretare mediante operazioni di *parsing*. Il formato generale di un sorgente Lex è il seguente:

{definizioni}
%%
regole
%%
{sottoprogrammi definiti dal programmatore}

Le regole rappresentano il controllo decisionale del programmatore e costituiscono una tabella dove nella colonna sinistra sono contenute le espressioni regolari e nella colonna di destra i frammenti di codice da eseguire quando le espressioni trovano corrispondenza.
Le espressioni regolari presenti nel sorgente Lex vanno descritte rispettando i formalismi previsti per :

- Gli operatori;
- Le classi di caratteri;
- Le espressioni opzionali;
- Le alternativa e raggruppamento;
- Le sensibilità a contesto(condizioni d'avvio);
- Le Ripetizioni e definizioni.

Si analizzi il seguente programma Lex:

```
#include <stdio.h>
%%
Marisa printf ("trovata marisa");
```

Questo permette di riconoscere la stringa "marisa" nella sequenza d'ingresso e stampare a video il messaggio "trovata marisa" ogni volta che esso appare.
In questo semplice esempio il linguaggio ospite è C.
La sezione delle definizioni contiene una combinazione di:

- Definizione nella forma ' nome spazio traduzione '
- Codice incluso, nella forma 'spazio codice '
- Codice incluso , nella forma

```
%{
 codice
 }%
```

- Tabelle di set di caratteri , dalla forma

  %T

  numero spazio carattere-stringa

  …

  %T

- Dichiarazioni delle Condizioni d'Avvio nella forma

  %S nome1 nome2

  ….

Se dopo aver descritto le regole il programmatore ha bisogno d'opzioni aggiuntive, come definire variabili, queste possono andare sia nelle definizioni sia nelle regole.

2.4 L'analizzatore sintattico

Yacc è un mezzo per imporre strutture all'ingresso di un programma. L'utente *yacc* prepara una specifica del processo di lettura degli ingressi: ciò include regole descriventi la struttura dell'ingresso ed il codice da richiamare a basso livello. *Yacc* genera quindi una funzione per controllare il trattamento dell'ingresso, chiamata analizzatore sintattico, che richiama l'analizzatore lessicale per identificare i token nella sequenza dell'ingresso.

Come descritto nel paragrafo precedente compito dell'analizzatore lessicale è quello di partizionare la sequenza in ingresso ed estrarre i vari *token*; l'analizzatore grammaticale riceve pertanto in ingresso i token ed effettua l'analisi sintattica.

Nella fase d'analisi sintattica i *token* vengono organizzati secondo le regole di strutture dell'ingresso, chiamate regole grammaticali; quando una di queste regole è usata per ridurre l'ingresso allora viene invocato il codice utente relativo alla regola, in pratica un'azione.

Una parte importante del processo d'analisi è svolto dall'analizzatore lessicale che riconoscendo le strutture a più basso livello comunica i token identificati all'analizzatore sintattico. Va puntualizzato che una struttura riconosciuta dall'analizatore lessicale è chiamata simbolo terminale o token, mentre la struttura riconosciuta dall'analizzatore sintattico è chiamata simbolo non terminale. *Yacc* richiede di definire i *token* ed includere l'analizzatore lessicale come parte del file di specifica. Ogni file di specifica di *yacc* è del tipo:

dichiarazioni
%%
regole
%%
programmi.

La sezione delle regole è costruita da una o più regole grammaticali. Una regola grammaticale ha la forma:

A: Body;

A rappresenta il nome di un simbolo non terminale, Body è una sequenza di zero o più nomi o letterali; va precisato che un nome può essere un token o un simbolo non terminale.
Se ci sono diverse regole con la stessa parte sinistra, la regola può essere descritta nel seguente modo:

A: B C D
| E F
|G;

Se il simbolo non terminale indica la stinga vuota la indicheremo in questo modo:

H:;

I nomi che rappresentano token vanno dichiarati nella sezione delle dichiarazioni:

%token nome1 nome2 ….

Ogni nome non definito nella sezione dichiarazione è assunto rappresentante un simbolo non terminale
Ogni simbolo non terminale deve apparire nella parte sinistra di almeno una regola.
Ad ogni regola grammaticale l'utente può associare azioni da eseguire ogni volta che la regola è applicata nel trattamento degli ingressi come nell'esempio seguente:

```
A: '(' B ')'{
        printf("Ecco un'azione associata ad una regola grammaticale\n")
        Flag= 9;
        }
```

Per restituire un valore, l'azione normalmente pone la pseudo variabile $$ a qualche valore.Per esempio un'azione a cui non è associata nessuna regola ma restituisce il valore 1 è cosi descritta:

{$$=1}

Per ottenere e utilizzare i valori restituiti dalle azioni precedenti e dall'analizzatore lessicale, l'azione può usare le pseudo variabili $1 $2… che si riferiscono a valori restituiti dai componenti delle parti destre di una regola , leggendo da sinistra a destra. Quindi , se la regola per esempio è:

A: B C D;

$2 contiene il valore restituito da C e $3 il valore restituito da D.

2.5 Conclusioni

Definito il linguaggio e messo a punto il compilatore non resta che creare l'interfaccia che ha la funzione di far interagire l'utente con il linguaggio.
Per la realizzazione di questo progetto è stato necessario, utilizzo dei *tool Flex e Bison*, versioni per Windows di *lex e yacc* conosciuti sotto ambienti *Linux*, indispensabili per la realizzazione di un traduttore che convertisse una sequenza in ingresso, scritto nel linguaggio definito, in un sequenza comprensibile per i software di gestione del sistema di controllo del movimento del robot umanoide *Robovie-M.*

Capitolo 3
I software utilizzati

3.1 Introduzione

Nella fase iniziale di ricerca e sperimentazione, necessaria per la realizzazione di questo progetto, è stato necessario lo studio e l'utilizzo di strumenti software che hanno permesso di interagire e comunicare con il robot.

3.2. RobovieMaker

Il primo software studiato per la realizzazione di questa tesi è il software proprietario della VStone di cui è corredato il robot. Il software *RobovieMaker*, la cui schermata iniziale è visibile nella figura 3.1, consente la creazione *on line* dei movimenti complessi e la successiva esecuzione degli stessi da parte del robot.

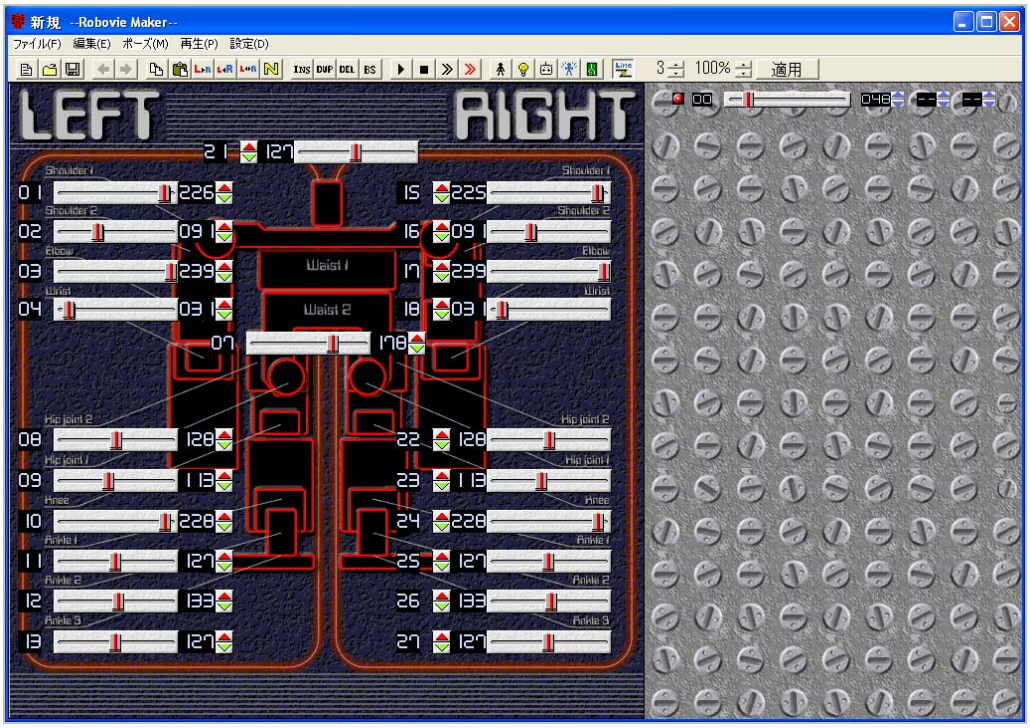


Figura 3.1: La schermata principale di RobovieMovement.

Per un utilizzo efficiente del software, il robot deve essere connesso al PC tramite porta seriale ed alimentato ad una tensione di 6V.
Nella barra degli strumenti, come mostrato in figura 3.2, insieme a pulsanti con funzionalità tipiche per qualsiasi tipo di programma, vi sono tutta una serie di pulsanti necessari per programmare mediante frame i movimenti del robot.

Figura 3.2: Il dettaglio della Barra degli strumenti del programma RobovieMaker.

Il pulsante di *Line* è utilizzato per stabilire una connessione fra il PC, sul qual è istallato *RobovieMaker,* ed il robot.
Il pulsante di *Zero* è utilizzato, dopo aver connesso il PC al robot, per accendere i 22 servomotori e mettere il robot nella configurazione iniziale.
Per inserire un nuovo stage, nella sequenza che definisce un comportamento, è necessario selezionare il pulsante *Ins*.
Inserito uno stage questo sarà visualizzato sulla parte destra del software, come mostrato nella figura 3.1, su questo stage, è possibile cambiare l'angolazione dei servomotori agendo sugli *slider* presenti a sinistra.
Per aggiungere nuovi stage basterà selezionare il tasto *Ins* e programmare gli spostamenti dei vari servomotori.
In ogni stage creato si possono notare 4 sezioni com'è possibile vedere nel dettaglio rappresentato in figura 3.3:

- Il numero dello stage;
- L'angolazione dei motori;
- La velocità d'esecuzione dello stage;
- Il numero del prossimo stage.

Se si vuole eliminare uno stage basta evidenziarlo e selezionare il tasto DEL *Del* e questo sarà rimosso dalla sequenza.

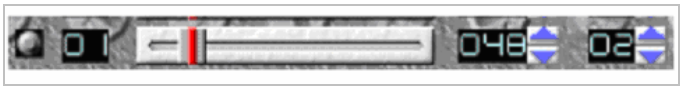


Figura 3.3 Il dettaglio della rappresentazione di un singolo stage nel RobovieMaker.

Creata la sequenza di stage come in figura 3.4 sarà possibile far eseguire il movimento completo al robot selezionando il pulsante ▶ .
Per bloccare la sequenza basta selezionare il pulsante ■ , per eseguire il movimento stage a stage occorrerà selezionare volta per volta il pulsante » .
I pulsanti permettono di visualizzare delle schermate dove è possibile variare le impostazioni di *default* del robot come ad esempio la posizione di zero.

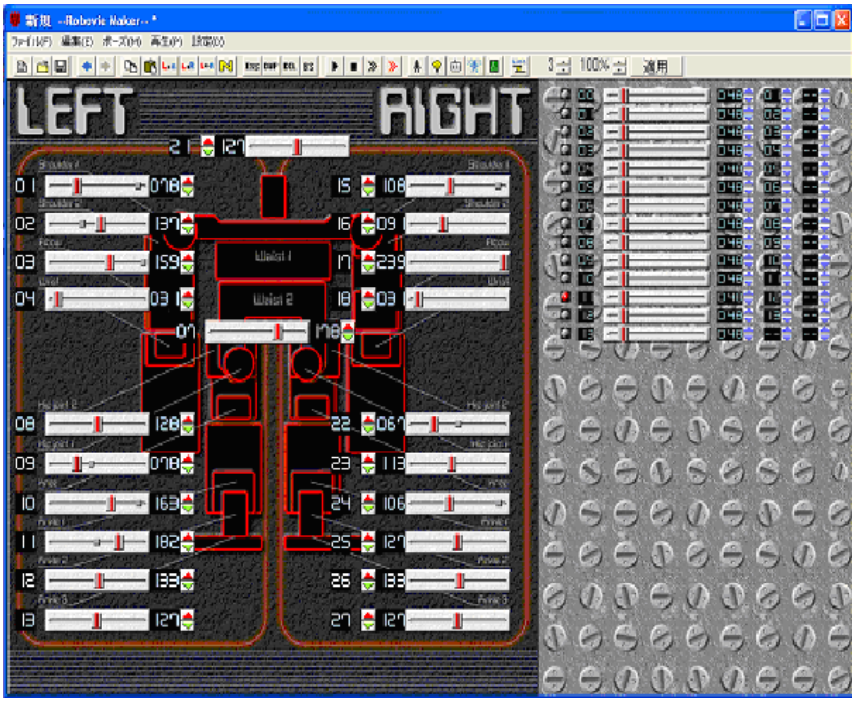


Figura 3.4: La schermata di RobovieMaker con una sequenza di stage inserita.

3.3 Robostage

Il secondo software utilizzato nell'ambito del progetto, come simulatore dei movimenti, è stato un software sviluppato presso l'università degli studi di Padova: *Robostage*.In figura 3.5 è possibile vedere la schermata principale del simulatore dove si distinguono quattro aree di lavoro:

- La Barra degli strumenti;
- Un visualizzazione 3D;
- Un Pannello per il controllo dei giunti;
- Un Pannello Regia.

Il software *Robostage* permette di creare un comportamento come una sequenza di stage e di renderne visibile l'esecuzione nel visualizzatore 3D.
Nel pannello per il controllo dei giunti è possibile muovere uno *slider* ed il modello 3D si muoverà di conseguenza.

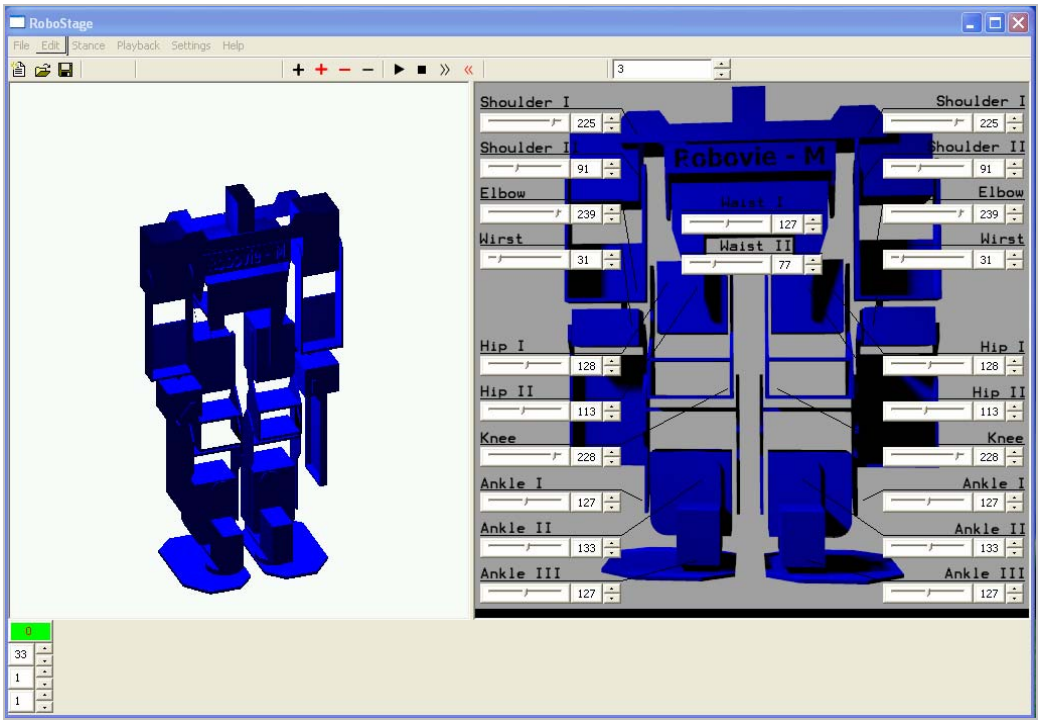


Figura 3.5: La schermata principale di Robostage.

Nella barra degli strumenti visibile in dettaglio nella figura 3.6 insieme a pulsanti con funzionalità tipiche per qualsiasi tipo di programma, vi sono tutta una serie di pulsanti necessari per programmare i comportamenti del robot.

Figura 3.6: Il dettaglio della barra degli strumenti di Robostage.

Selezionando il pulsante + si aggiunge uno stage sul pannello regia.
In ogni stage creato si possono notare 4 sezioni com'è possibile vedere nel dettaglio rappresentato in figura 3.7:

- Il numero dello stage;
- La velocità di transizione dello stage;
- Il prossimo stage quando si è in un ciclo;
- Il prossimo stage.

Figura 3.7: Il dettaglio dello stage in Robostage.

Per la programmazione di uno stage come prima cosa occorre settare la velocità e successivamente agendo sul pannello di controllo dei giunti modificare la posizione del robot.
Se si è programmato un comportamento come una sequenza di stage, selezionando il pulsante ▶ il visualizzatore 3D renderà visibile il movimento.
Agendo sulla barra degli strumenti sarà possibile eliminare uno stage selezionando il pulsante −, bloccare l'esecuzione, con il pulsante ■, visualizzare il movimento stage a stage in avanti o indietro mediante i pulsanti i » «.

3.4 La comunicazione tramite la porta seriale

Per lo sviluppo di questa tesi è stata creata una classe in grado di comunicare con il robot utilizzando l'interfaccia seriale secondo lo standard *POSIX*.
Le caratteristiche tecniche sono una connessione a 38400 bps, 8 bit e nessuna parità, i bit di stop, nessun controllo di *Data Terminal Ready(DTR)* e nessun controllo di *Request To Send(RTs).*
La comunicazione con il robot avviene mediante il passaggio di opportune stringhe codificate secondo un protocollo scelto dalla ditta costruttrice del robot.
La determinazione di tale procedura è difficoltosa, in quanto i manuali forniti con il robot sono in lingua giapponese.
Sono stati utilizzati dei programmi denominati *sniffer,* che permettono di intercettare le comunicazioni che avvengono attraverso la porta seriale.
Tramite questa procedura è stato possibile ricostruire le sequenze di byte da utilizzare per poter costruire un comando da inviare al robot.
Si analizzi nel dettaglio l'insieme di informazioni contenute nelle stringhe inviate tramite porta seriale:

- Txxxx è il comando per muovere i motori, dove xxxx indica un valore compreso nell'intervallo 0x0000-0xffff, anche se usualmente le prime due cifre sono poste a zero e le altre due indicano la posizione del servomotore.
- @xxxx è il comando che imposta la durata di esecuzione dei movimenti;
- Cxxxx è il comando utilizzato per acquisire la posizione dei giunti;
- Px è il comando per spegnere o accendere il servomotore, rispettivamente P0 e P1;
- : è il carattere utilizzato per separare i comandi per i diversi servomotori;

- gxx è il comando per acquisire le informazioni sul sensore di accelerazione;
- \n indica il terminatore di stringa, ogni comando termina con ’nn’;
- Fx è il comando per acquisire la versione del firmware;
- Gxxxxxx, Sxxxxxx sono i comandi per agire direttamente sulla memoria EEPROM;
- Dxx è il comando per utilizzare i canali digitali;
- Mx è utilizzato per impostare il canale in ingresso oppure in uscita, M0 imposta il canale come uscita ed M1 coem ingresso;
- Rxx è il comando per acquisire le informazioni sul giroscopio.

Capitolo 4

L'evoluzione della robotica umanoide

4.1 La robotica umanoide

La realizzazione di robot umanoidi richiede la capacità di comprendere e replicare l'abilità di locomozione bipede tipica dei primati più evoluti.
L'equilibrio in particolare, costituisce un elemento cruciale da considerare, quando si vogliono riprodurre i complessi meccanismi della locomozione bipede: senza il controllo del baricentro, infatti, si cadrebbe al primo passo.
Nei sistemi robotici mobili dotati di zampe si distinguono due categorie di robot: a stabilità statica e a stabilità dinamica.
Nel primo gruppo rientrano quei robot (dotati di almeno quattro zampe) progettati in maniera tale da mantenere costantemente il centro di gravità all'interno della superficie d'appoggio durante la marcia; il secondo gruppo è costituito invece da quei robot (ad esempio i bipedi), il cui centro di gravità non è mantenuto all'interno di tale superficie e che raggiungono l'equilibrio solo grazie alla continuità del movimento.
Nel caso particolare dell'uomo, l'abilità di camminare sugli arti inferiori si è sviluppata grazie anche alle numerose terminazioni nervose di cui è dotato.
Queste permettono al cervello di essere sempre informato sugli sforzi a cui sono sottoposte le fibre muscolari e sulla posizione assunta da ogni articolazione, adeguando quindi la postura (spostando il dorso e la testa o facendo oscillare le braccia) per mantenere il baricentro del corpo in equilibrio. In modo analogo, i vari sensori integrati sui robot mobili devono comunicare ai sistemi di controllo i dati relativi alle forze che agiscono sui giunti e sulle strutture.
Gli studi relativi alla robotica su zampe si sono intensificati negli ultimi anni, in parte per il sogno di molti ricercatori di realizzare macchine sempre più simili all'uomo (anche per facilitarne l'inserimento in ambienti umani) e in parte per i

vantaggi che le zampe offrono rispetto ad altri sistemi locomotori. Infatti, se da una parte è vero che il movimento tramite arti mobili richiede meccaniche e algoritmi di controllo molto complessi ed evoluti rispetto a quelli usati per ruote e cingoli (specialmente nel caso della deambulazione bipede, nella quale ci si trova in costante equilibrio precario), d'altra parte è indubbio che tale movimento permette di ottenere una versatilità superiore. I robot su zampe, infatti, possono muoversi anche su terreni sconnessi e irregolari, affrontare pendenze e superare macerie, senza correre il rischio di perdere l'appoggio di una ruota motrice e perciò la possibilità di muoversi[21].

Non c'è quindi da stupirsi se molte équipe internazionali stiano lavorando sodo per far camminare autonomamente le proprie macchine. Tra i progetti più noti vi sono sicuramente ASIMO e QRIO prodotti rispettivamente dai due colossi giapponesi Honda e Sony rispettivamente mostrati nella figura 4.1 e 4.2

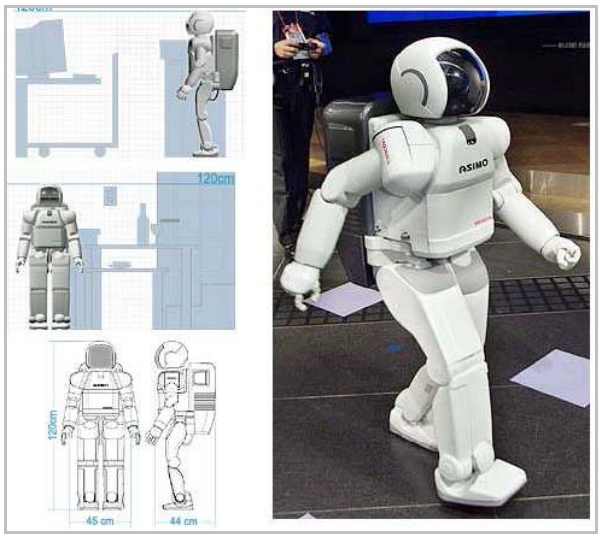


Figura 4.1: Il robot Asimo prodotto dalla Honda 2000.

Asimo è il più avanzato dei bipedi Honda. E' un robot umanoide realizzato per simulare la camminata umana sia statica che dinamica.

Il numero di gradi di liberta delle sue gambe è ridotto al minimo indispensabile. Infatti ha 6 gradi di liberta per gamba. I 14 gradi di liberta rimanenti permettono ad Asimo di mantenere l'equilibrio in ogni situazione. I piedi di

Asimo hanno un sistema di assorbimento delle forze di impatto che consente di ammortizzare l'andatura e di avere una minore sollecitazione dei giunti. Asimo ha dimensioni ridotte principalmente per due motivi: innanzitutto i motori degli attuatori peserebbero troppo e occuperebbero uno spazio eccessivo, inoltre la batteria necessaria per il funzionamento di questi motori avrebbe dimensioni e peso rilevanti. Poiché Asimo deve essere in grado di muoversi in maniera naturale, e risultato necessario dare al robot delle dimensioni convenienti.

Figura 4.2: Il robot Qrio prodotto dalla Sony.

Qrio ha una mobilità paragonabile a quella di Asimo, con stabilità e un bilanciamento maggiore ,dovuto principalmente alla minore altezza e peso. Qrio è dotato di un sistema di riconoscimento vocale e un sistema di riconoscimento dei volti che gli permettono di memorizzare le persone incontrate, e di associare ad esse dei comportamenti.

Ma c'è anche una piccola novità di nome PINO, visibile in figura 4.3 che sta facendosi strada sul web. La sua peculiarità sta nel fatto di essere un robot *open source*, vale a dire i suoi progetti sono liberamente disponibili al pubblico: chiunque, quindi, può contribuire al suo sviluppo.

Le dimensioni di Pino sono create ad immagine di un bambino di un anno e mezzo circa e risulta alto circa 70 cm. Il numero di gradi di libertà è pari a ventisei così distribuiti:

- Cinque per ciascun arto superiore;
- Due per la testa;
- Due per il busto.

Figura 4.3: Il Robot Pino Progetto Open Source.

La locomozione bipede autonoma è una premessa necessaria alla realizzazione dei robot umanoidi che da sempre popolano l'immaginario collettivo.
Le applicazioni di questa tecnologia potranno però rivestire un ruolo fondamentale anche in ambiti come la medicina e la fisioterapia.
I principi e le tecniche usati per realizzare le gambe dei robot bipedi, infatti, potranno essere impiegati nella costruzione di protesi robotiche sempre più sofisticate che, interfacciate al sistema nervoso, saranno utili all'uomo per rimpiazzare arti amputati e donare funzionalità motorie a coloro che ne sono privi.

Allo stesso tempo, c'è chi sta pensando di impiegare i risultati degli studi sull'andatura robotica bipede per comprendere meglio il funzionamento della biomeccanica umana[1].

4.2 I robot umanoidi Kid Size della VStone

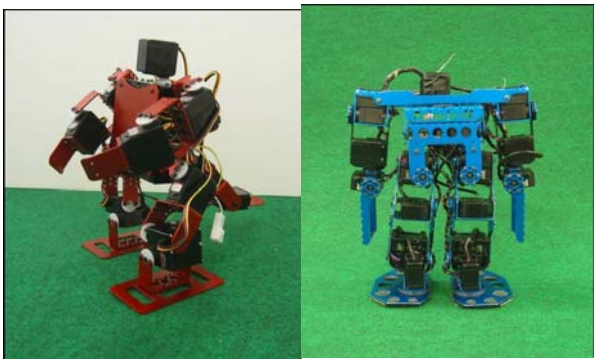
La ditta giapponese VStone[23] produce i seguenti robot umanoidi:

- Robovie-MS, con 17 gradi di libertà;
- Robovie-M vers. 2, con 22 gradi di libertà;
- Robovie_V, con 23 gradi di libertà, e telecamera;
- VisiON, come Robovie_V con proporzioni degli arti differenti;
- VisiON NEXTA, modello di dimensioni maggiori, due processori

Robovie-MS, si veda la figura 4.4 (a), è il primo robot della serie in ordine di tempo e rappresenta anche la piattaforma più povera, sia dal punto di vista meccanico, sia elettronico. Si tratta di un robot con un totale di 17 gradi di libertà così distribuiti:

- 5 per ogni arto inferiore;
- 3 per ogni artosuperiore;
- 1 per il busto.

Le sue dimensioni sono di 280x185x50mm per un peso totale di 860g. La scheda di controllo è dotata di una CPU Renesas H8/3687 funzionante a 16MHz, il sistema operativo è proprietario.



(a) Robovie-MS (b)Robovie-M vers 2

Figura 4. 4 : La piattaforma Robovie-Ms e Robovie-M vers. 2

Robovie-M versione 2, visibile in figura 4.4 a , nella versione tre sarà la piattaforma meccanica utilizzata per questo lavoro di tesi, pertanto sarà descritta più approfonditamente nei paragrafi successivi. Robovie-M versione 2 è un robot con un totale d i22 gradi di libertà così distribuiti:

- 6 per ogni arto inferiore,;
- 4 per ogni arto superiore;
- 2 per il busto.

Le sue dimensioni sono di 290x240x65mm per un peso totale di 1.9kg .La scheda di controllo è dotata di una CPU Renesas H8/3684 funzionante a16MHz, e tre Microchip PIC16F877 per il controllo dei motori, un accelerometro su due assi, una porta seriale RS232C. Il sistema operativo è proprietario.

(a)Robovie-V. (b)VisiON.

Figura 4.5: La piattaforma Robovie-V e VisiON della Vistone.

Robovie-V, si veda la figura 4.5a, è un robot con un totale di 23 gradi di libertà così distribuiti:

- 6 per ogni arto inferiore;
- 4 per ogni arto superiore;
- 2 per il busto
- 1 per la testa.

Le sue dimensioni sono di 390x280x85mm per un peso totale di 2.2kg. La scheda di controllo è dotata di una CPU Renesas SH2/7054 funzionante a 40MHz, una ram da 262144 word da 16-bit, ed una EEPROM esterna da 65536 byte, un accelerometro su due assi, una porta seriale RS232C.
E' fornito completo di una telecamera digitale CMOS direttamente collegabile alla scheda. La struttura metallica è ricoperta con superficie in ABS. Non è fornito il sistema operativo.
VisiON, visibile in figura 4.5b, è analogo per dotazione hardware ed elettronica a Robovie-V e differisce da quest'ultimo per un diverso rapporto tra gli arti e le dimensioni d'alcuni componenti al fine di adattarlo alle regole imposte dall'Humanoid League della RoboCup.
VisiON NEXTA, si veda la figura 4.6, è un robot con un totale di 23 gradi di libertà così distribuiti:

- 6 per ogni arto inferiore;
- 4 per ogni arto superiore;
- 2 per il busto
- 1 per la testa.

Le sue dimensioni sono di 465x260x160mm per un peso totale di 3.2kg.La scheda di controllo è dotata di una coppia di processori: il processore principale, GEODE funzionante a 400 MHz è dotato di 4 GB di memoria FLASH

e 256 MB di RAM, si occupa dell'image processing e del controllo dei comportamenti del robot; il processore secondario, SH2/7054 funzionante a 40 MHz si occupa del controllo dei motori.

Figura 4.6: La piattaforma VisiON NEXTA della Vistone.

E' dotato d'interface USB1.1 ed RS232, di un accelerometro su tre assi e di un giroscopio su tre assi e di una telecamera digitale CMOS direttamente collegabile alla scheda primaria che si occupa di acquisire ed elaborare le immagini. La struttura metallica è ricoperta con superficie in ABS.

4.3 Il robot Robovie-M versione 3

La piattaforma hardware adoperato in fase di sperimentazione, è Robovie-M versione 3 prodotto dalla Vistone . Il robot, visibile in figura 4.7, è un sistema bipede autonomo.

Il robot può operare in maniera autonoma, nel senso che è possibile caricare dei movimenti predefiniti sulla flash del microcontrollore e farli eseguire senza che un uomo lo controlli; oppure può essere comandato per mezzo di una comunicazione seriale mediante il protocollo RS232C.

E' alto circa 29 cm per un peso di 1.9 kg. E 'dotato di un processore H8-3687 prodotto dalla ditta Renesas[7] con una potenza di calcolo di 20 Mhz.

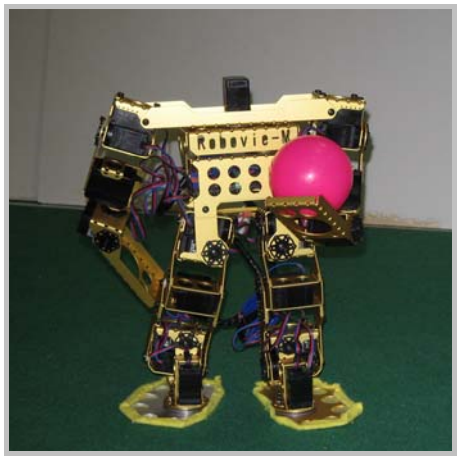


Figura 4.7 : La piattaforma Robovie-M versione 3 della Vistone.

Il microcontrollore `e costituito da quattro timer, tre a 8 bit e uno a 16 bit, e tre Microchip PIC16F877 per il controllo dei motor Presenta un'interfaccia di tipo seriale e un convertitore A/D a 10 bit, e di una telecamera wireless operante alla frequenza di 2.4 Ghz fissata direttamente sulla parte superiore del robot.
E' dotato di 22 gradi di libertà cosı distribuiti:

- 6 per ciascuna gamba
- 4 per ciascun braccio
- 2 per il busto

Il robot `e dotato di due tipi di sensori: un *encoder* su ciascun giunto e un accelerometro lungo due assi per un totale di ventitre sensori.
Sulla scheda incorporata (VS-H8PWM24), oltre al microcontrollore, sono presenti due supporti di memorizzazione: una Flash di 2 kilobyte e una RAM di 32 kilobyte.

Il sistema operativo `e proprietario della Vstone. I servomotori adottati sono di due tipi:

- SPEC-APZ per le braccia;
- Hyper ERG-VB per il busto e le gambe.

Entrambi sono prodotti dalla stessa ditta Sanwa[22]. L'associazione di un motore ad un giunto, si veda a tal proposito la tabella 4.1 dipende principalmente dalla massima coppia che potenzialmente può erogare: i motori dotati di coppia maggiore sono quelli degli arti inferiori, al fine di garantire una maggiore solidità al robot stesso.

Motore	Coppia	Velocità	Dimensione
Hyper ERG-VB	13 kg x cm (6V)	60±/0.10s (6V)	39 x 20 x 37.4 mm
SPEC-APZ	4 kg x cm (4.8V)	60±/0.20s (4.8V)	39 x 20 x 35.5 mm

Tabella 1: Le caratteristiche tecniche dei motori Sanwa.

4.3.2 I servocomandi

I motori di cui `e dotato Robovie-M sono dei servocomandi o anche detti soltanto "servo". Il loro compito `e di convertire un segnale elettrico in un movimento meccanico. Per ottenere questo all'interno `e presente un micromotore e una circuiteria elettronica che realizza la logica di controllo. In figura 4.8 si possono vedere quali sono i componenti fondamentali di un servo.
Il servocomando `e connesso alla scheda del microcontrollore mediante un cavo tripolare, in cui due fili servono per l'alimentazione (pari a 6V) e sul terzo `e presente il segnale di comando generato dal microcontrollore. In generale il segnale di comando `e un treno di impulsi ciascuno della durata compresa tra 1 e 2 millisecondi. Il micromotore `e collegato ad un potenziometro connesso a un circuito monostabile.

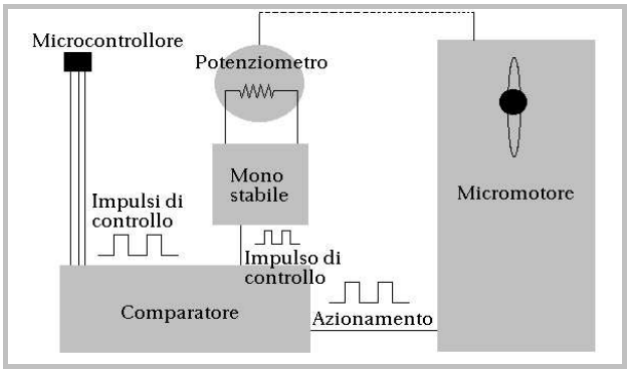


Figura 4.8: La schema di controllo dei servo motori.

Secondo come si muove l'albero del motore, il potenziometro genera un altro segnale,chiamato "segnale di controllo". I due segnali vengono confrontati con un circuito comparatore, che ne raffronta la durata temporale; se il "segnale di comando" differisce da quello "di controllo", il comparatore aziona il motore, fornendo allo stesso la polarità corretta per farlo girare nel giusto verso, finché la durata dei due impulsi risulta la stessa. In questo modo il servo si muove esattamente quanto previsto dall'impulso fornitogli dal microcontrollore.

4.3.3 Il software VStone

All'accensione di Robovie-M, sulla scheda viene eseguito un software proprietario della VStone.

Tale software è in grado di disporre in una posizione di default il robot. Sulla scheda sono presenti degli switch, che, a seconda della configurazione, permettono al robot di eseguire dei movimenti pre memorizzati nei dispositivi di memorizzazione, oppure impostano il microcontrollore a ricevere dei comandi dalla porta seriale di cui la scheda è dotata.

La connessione con il robot può avvenire mediante un software proprietario,oppure sfruttando quello fornito dalla VStone, che è chiamato *RobovieMaker*. Quest'ultimo `e in grado di comandare tutti i motori del robot e gestire la Flash e la RAM del robot.

Le caratteristiche tecniche sono una connessione a 38400 bps, 8 bit, nessuna parità,1 bit di stop. La comunicazione con il robot avviene mediante il passaggio di opportune stringhe codificate secondo un protocollo scelto dalla ditta costruttrice del robot.
Le stringhe sono costituite da una sequenza di char del tipo quelli rappresentati in figura 4.9 dove ogni sottostringa compresa tra : identifica il giunto e il valore della sua orientazione.

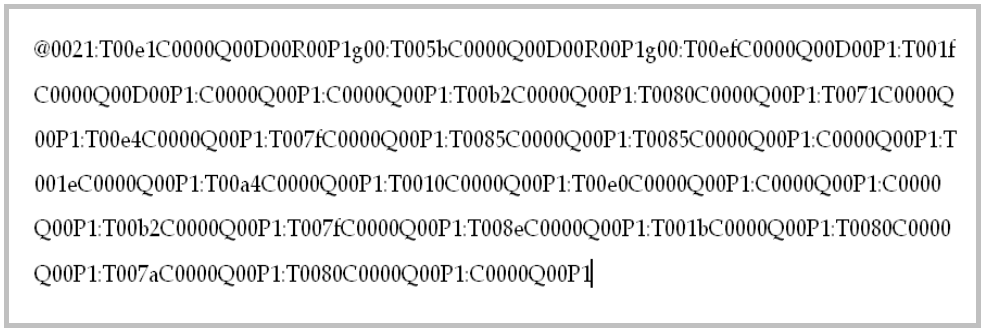


Figura 4.9: La stringa da inviare tramite la seriale in corrispondenza allo stage zero.

Il valore è un numero in esadecimale compreso nell'intervallo [0, FF]. Due stringhe particolari servono a definire l'accensione e lo spegnimento dei motori del robot: F0 e F7 rispettivamente. Nel caso in questione queste due stringhe sono inviate all'avvio del programma e allo spegnimento dello stesso.
Quando si invia un messaggio è importante ricordarsi che oltre ai valori dei giunti è necessario terminare il comando con il carattere '\n'.
Inoltre per ogni stringa inviata è necessario eseguire una lettura, altrimenti una volta riempito il buffer della seriale, i messaggi in uscita potrebbero essere ignorati.

4.4 La Storia dell'arte di Robovie-M

I primi in Italia a lavorare con robot umanoidi di questo tipo sono stati gli "*Artisti Veneti*",un gruppo formato da studenti e ricercatori dell'università degli

studi di Padova che si è occupato di sviluppare i robot della categoria *Humanoid Kid Size League*. per la competizione *RoboCup* raggiungendo ottimi traguardi sia dal punto di vista tecnico-scientifico sia nelle competizioni internazionali.
Da un'analisi generale dell'elettronica disponibile a bordo della scheda fornita col robot Robovie-M versione 2 e da una prima analisi comparativa fra gli obiettivi dei vari progetti e la potenza di calcolo disponibile, hanno realizzato che il raggiungimento di tali obiettivi non era possibile con quanto fornito in quanto:

- non è previsto l'uso di una telecamera, le cui informazioni siano elaborate on-board;
- la potenza di calcolo è appena sufficiente per il controllo dei 22 motori;
- non è presente una memoria non volatile a parte quella del microcontrollore.

Hanno quindi optato per la scelta aggiuntiva di una VS-7054 prodotta sempre dalla VStone che viene fornita come scheda base per il robot Robovie-V, corredata di una telecamera digitale 0V7620, di una CPU SH 7054.
Dal punto di vista implementativo è stato realizzato il firmware per il microprocessore del robot, che rende l'umanoide autonomo controllando l'acquisizione delle immagini dal mondo, la loro elaborazione, la scelta dell'azione da compiere ed infine i motori e la movimentazione. L'architettura realizzata è di tipo gerarchico, caratterizzata dalla successione delle funzioni di sense, reasoning e act; in questo modo si riescono a gestire meglio le problematiche relative alle scarse risorse disponibili.
Sono stati implementati alcuni movimenti tipici del calcio, come ad esempio il calcio della palla. Il robot è capace di individuare la palla, raggiungerla e calciarla.

I movimenti compiuti dal robot, Robovie-M versione 2, sono stati ottenuti come una sequenza di configurazione dei giunti al fine di eseguire un comportamento più o meno complesso.
Per comandare il robot hanno integrato la parte di visione con quella relativa al controllo del movimento ed hanno scelto di operare secondo lo schema di figura 4.10.

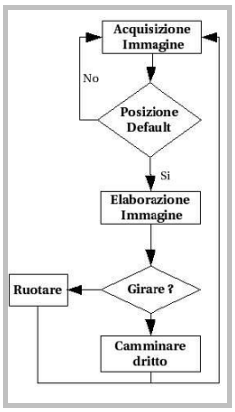


Figura 4.10 : Lo schema di controllo del movimento di Robovie-M versione 2.

Lo schema può essere riassunto nei tre passi:

- acquisizione immagine;
- verifica posizione robot;
- effettiva esecuzione movimento.

La scelta di questa sequenza è stata dettata dal fatto che la telecamera, essendo collocata sopra il busto del robot, non rimane sempre nella stessa posizione. Perciò, fintanto che il robot sta eseguendo un movimento, sia esso camminare dritto o ruotare, un è ventuale elaborazione dell'immagine determinerebbe una posizione del blob differente ad ogni elaborazione. Potrebbe accadere che durante il movimento il robot non esegua quello voluto, ma ne inizi uno nuovo, generando un comportamento del tutto instabile. Si è scelto, allora, di evitare l'elaborazione di immagini fintanto che Robovie-M è in movimento. Questa scelta ha comportato una riduzione notevole del numero di frame elaborati al

secondo, però ha come conseguenza che il movimento eseguito dal robot sia stabile e non comporti la caduta dello stesso. Se il robot si trova in posizione "di riposo" l'immagine acquisita viene elaborata.

Determinato il target, il controllo di come muovere Robovie-M viene eseguito sulla base delle coordinate del centro del target. Il sistema di riferimento ha l'origine in alto a sinistra dell'immagine, l'ordinata è orientata verso destra mentre l'ascissa è orientata verso il basso (considerando il verso positivo in questa direzione). Per decidere quale movimento azionare sul robot si è scelto di suddividere l'immagine in tre regioni verticali e un'orizzontale. In particolare se il target si trova nella regione d'intersezione fra quella verticale-centrale e quella orizzontale il target è considerato raggiunto. Altrimenti il robot si muove nella direzione della regione in cui è contenuto il target.

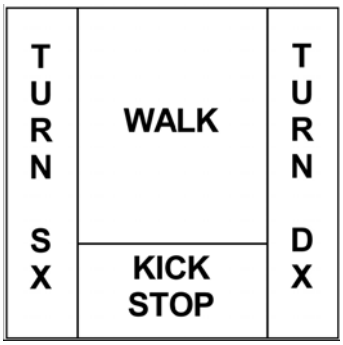


Figura 4.11: La suddivisione dell'immagine in regioni.

Ad esempio si osservi la figura 4.11 : se la palla viene rilevata nella regione a sinistra il movimento che viene eseguito è quello di rotazione antioraria, mentre nel caso della regione a destra quello di rotazione oraria. La regione centrale è suddivisa in due sottoregioni: quella superiore impone il movimento della camminata mentre quella inferiore l'azione di calcio, seguita dal mantenimento della posizione verticale.

Esiste anche un comportamento non evidenziato nel diagramma che viene assunto quando il robot non vede la palla. Il robot rimane in posizione verticale ruotando solo il busto per individuare il target.[1][2]

Capitolo 5

La soluzione Proposta

5.1 Introduzione

Un robot senza l'aspetto umanoide non è un robot. Discendendo direttamente dal paradigma "creato a immagine e somiglianza", che tecnologicamente fa innalzare l'uomo al rango di un dio dispensatore di vita, l'immagine che noi esseri umani abbiamo del robot è qualcosa di radicato in maniera talmente istintiva, che anche il progresso tecnologico è costretto a farci i conti.
Per l'essere umano camminare è una delle attività più semplici, poiché acquisita fin dalla primissima infanzia a livello istintivo, ma coinvolge il controllo di un numero di variabili talmente elevate che la versione artificiale di due gambe che camminano ha avuto bisogno di quasi venti anni per trovare un'espressione soddisfacente.

5.2 L'obiettivo del progetto

L'obiettivo di questa tesi è quello di sviluppare un linguaggio che permetta la creazione di comportamenti elementari a partire dai quali determinare i comportamenti complessi.
Grazie a questo linguaggio sarà possibile realizzare comportamenti ad alto livello, nonché tutta una libreria di comportamenti,sviluppando movimenti complessi come combinazione di movimenti elementari, quale lo spostamento del singolo giunto.
Mediante l'utilizzo di questo linguaggio sarà possibile realizzare un movimento, testarlo, e decidere di inserirlo o meno nella libreria dei movimenti.

In definitiva l'obiettivo è quello di realizzare tutta una libreria di comportamenti che possono essere richiamati da qualsiasi altro modulo deliberativo.

5.3 Le specifiche del progetto

L'interpretazione di un linguaggio richiede l'analisi di tre livelli di comprensione: il livello lessicale e quindi l'identificazione delle corrispondenze tra unità significative fondamentali morfologiche, quello sintattico e quindi l'analisi delle strutture grammaticali delle parole e del linguaggio stesso ed infine quello semantico, relativo all'attribuzione di significati concettuali alle strutture identificate.
La traduzione è eseguita sfruttando un analizzatore lessicale che individua i *token*, cioè i termini del linguaggio considerati parole chiave, e un analizzatore sintattico, che riceve i token, ed è in grado di interpretarne una sequenza attraverso l'uso d'opportune regole sintattiche, ed attribuire significati concettuali alle strutture identificate.
Il software sviluppato si basa su un traduttore che esegue una conversione del testo in ingresso in una sequenza di caratteri.
Tali sequenze di caratteri vengono interpretate da *RoboStage* (il simulatore) o *RobovieMaker* come comportamenti complessi, da simulare o da eseguire.
Per compiere la traduzione dal linguaggio definito a stringhe idonee alla comunicazione tramite seriale dei comportamenti, si esportano i dati in ingresso su di un file; tale file viene tradotto in stringhe significative, mediante gli analizzatori in precedenza descritti, generando un file d'uscita che viene opportunamente manipolato per ottenere stringhe inviabili direttamente tramite la porta RS232.
Per la realizzazione di questo progetto è stato necessario l'utilizzo delle applicazioni *Flex* e *Bison*, versioni per Windows di *lex* e *yacc* conosciuti sotto

ambiente Linux e djgpp v 2.0, indispensabili per la realizzazione di un traduttore che convertisse una sequenza in ingresso, scritta nel linguaggio definito, in una sequenza d'uscita comprensibile per i software di gestione del sistema di controllo del movimento del robot.

In un secondo tempo, sopra questo traduttore è stato necessario costruire ilinterfaccia di "RobovieMovement", che riuscisse a fare interagire l'utente sviluppatore di comportamenti con il linguaggio nella maniera più semplice ed intuitiva possibile.

Per lo sviluppo dell'interfaccia si è fatto uso del *Borland Developer Studio 2006*, ottimo compilatore *RAD* che ha permesso di realizzare l'interfaccia del software in maniera visuale riducendo notevolmentte i tempi di sviluppo

Il software *RobovieMovement* è considerato un applicativo di un progetto più grande sviluppato presso l'Università degli Studi di Palermo, che è *RobovieTotalControl*.

Figura 6.1: Un'istantanea del programma RobovieM TotalControl.

Su *RobovieTotalControl*, figura 6.1, convogliano i lavori di coloro i quali si sono occupati di visione (riconoscimento di oggetti come palla, marker, porte) il nostro, che è stato un lavoro prettamente rivolto alla movimentazione ed il lavoro di coloro che si sono occupati dei moduli deliberativi del robot.

Uno degli obiettivi finali del progetto *RobovieTotalControl* è quello di fornire al robot un modello decisionale, una sorta di cervello elettronico che gli consenta di decidere qual è la migliore azione da compiere in relazione a quello che viene osservato dalla telecamera.

Tale modello inizia con il processamento delle immagini; in relazione a quello che viene riconosciuto, i moduli decisionali a più alto livello scelgono all'interno di una libreria di movimenti già creata, qual è l'azione migliore da far eseguire al robot. Tale azione viene trasferita al robot mediante il modulo che gestisce la comunicazione seriale ed eseguita immediatamente.

Mediante *RobovieMovement* è possibile programmare dei comportamenti che possono essere passati al simulatore o al software proprietario della VStone

Inoltre è possibile far eseguire direttamente al robot i comportamenti programmati, senza dovere necessariamente passare per il software proprietario per l'esecuzione dei movimenti, mediante la comunicazione seriale diretta.

I comportamenti creati sono stati sviluppati utilizzando il suddetto applicativo con buoni risultati.

Oltre ai comportamenti base quali per esempio la movimentazione d'ogni singolo giunto, particolare importanza si è data allo sviluppo di comportamenti che potessero essere utili in ambiente *Robocup*.

Altri comportamenti sviluppati sono stati movimenti emozionali e movimenti coreografici.

Capitolo 6

La creazione di movimenti con RobovieMovement

6.1 Il linguaggio per la programmazione dei movimenti.

Per potere programmare un comportamento complesso bisogna descriverlo come sequenza di stage. Al primo stage, i motori, associati ai singoli giunti hanno delle posizioni dette posizioni di zero; se si desidera che nello stage successivo un motore si sposti realizzando un movimento elementare questo va descritto indicando la variazione relativa d'ogni giunto che si desidera spostare rispetto allo stage precedente utilizzando una delle primitive del linguaggio realizzato, e così via in ogni stage successivo. La struttura delle primitive del linguaggio è:

<azione > < identificativo giunto> < variazione relativa>;

Queste caratterizzano i movimenti elementari e sono suddivise in base ai giunti in :

1. Movimenti shoulder 2;
2. Movimenti shoulder 1;
3. Movimenti elbow;
4. Movimenti wirst;
5. Movimenti hip 1;
6. Movimenti hip 2;
7. Movimenti knee;
8. Movimenti ankle 1;
9. Movimenti ankle 2;
10. Movimenti ankle 3;
11. Movimenti waist 1;
12. Movimenti waist 2.

Nelle figure 6.2 - 6.13 sono mostrate, divise per giunti, le primitive del linguaggio:

muovi shoulder 1 <destro/sinistro> di <intero> gradi verso l'interno
muovi shoulder 1 <destro/sinistro> di <intero> gradi verso l'esterno
muovi shoulder 1 <destro/sinistro> di <intero> gradi all'interno
muovi shoulder 1 <destro/sinistro> di <intero> gradi all'esterno
muovi shoulder 1 <destro/sinistro> di <intero> gradi verso dentro
muovi shoulder 1 <destro/sinistro> di <intero> gradi verso fuori
muovi shoulder 1 <destro/sinistro> di <intero> gradi in dentro
muovi shoulder 1 <destro/sinistro> di <intero> gradi in fuori
ruota shoulder 1 <destro/sinistro> di <intero> gradi in senso orario
ruota shoulder 1 <destro/sinistro> di <intero> gradi in senso antiorario

Figura 6.2 : Primitive Shoulder 1 .

alza shoulder 2 <destro/sinistro> di <intero> gradi
abbassa shoulder <destro/sinistro> 2 di <intero> gradi
alzare shoulder 2 <destro/sinistro> di <intero> gradi
abbassare shoulder 2 <destro/sinistro> di <intero> gradi
muovi shoulder 2 <destro/sinistro> di <intero> gradi verso l'alto
muovi shoulder 2 <destro/sinistro> di <intero> gradi in alto
muovi shoulder 2 <destro/sinistro> di <intero> gradi verso il basso
muovi shoulder 2 <destro/sinistro> di <intero> gradi in basso

Figura 6.3 : Primitive *Shoulder 2* .

muovi wirst <destro/sinistro> di <intero> gradi verso l'interno
muovi wirst <destro/sinistro> di <intero> gradi verso l'esterno
muovi wirst <destro/sinistro> di <intero> gradi all'interno
muovi wirst <destro/sinistro> di <intero> gradi all'esterno
muovi wirst <destro/sinistro> di <intero> gradi verso dentro
muovi wirst <destro/sinistro> di <intero> gradi verso fuori
muovi wirst <destro/sinistro> di <intero> gradi in dentro
muovi wirst <destro/sinistro> di <intero> gradi in fuori
ruota wirst <destro/sinistro> di <intero> gradi in senso orario
ruota wirst <destro/sinistro> di <intero> gradi in senso antiorario
ruotare wirst <destro/sinistro> di <intero> gradi in senso orario
ruotare wirst <destro/sinistro> di <intero> gradi in senso antiorario

Figura 6.5 : Primitive *Wirst* .

ruota elbow <destro/sinistro> di <intero> gradi in senso orario
ruota elbow <destro/sinistro> di <intero> gradi in senso antiorario
ruotare elbow <destro/sinistro> di <intero> gradi in senso orario
ruotare elbow <destro/sinistro> di <intero> gradi in senso antiorario

Figura 6.4 : Primitive *Elbow.*

muovi hip 1 <destro/sinistro> di <intero> gradi verso l'interno
muovi hip 1 <destro/sinistro> di <intero> gradi verso l'esterno
muovi hip 1 <destro/sinistro> di <intero> gradi all'interno
muovi hip 1 <destro/sinistro> di <intero> gradi all'esterno
muovi hip 1 <destro/sinistro> di <intero> gradi verso dentro
muovi hip 1 <destro/sinistro> di <intero> gradi verso fuori
muovi hip 1 <destro/sinistro> di <intero> gradi in dentro
muovi hip 1 <destro/sinistro> di <intero> gradi in fuori
ruota hip 1 <destro/sinistro> di <intero> gradi in senso orario
ruota hip 1 <destro/sinistro> di <intero> gradi in senso antiorario
ruotare hip 1 <destro/sinistro> di <intero> gradi in senso orario

Figura 6.6 : Primitive *Hip 1.*

alza hip 2 <destro/sinistro> di <intero> gradi
abbassa hip 2 <destro/sinistro> di <intero> gradi
sposta hip 2 <destro/sinistro> di <intero> gradi in avanti
sposta hip 2 <destro/sinistro> di <intero> gradi avanti
sposta hip 2 <destro/sinistro> di <intero> gradi indietro
muovi hip 2 <destro/sinistro> di <intero> gradi in avanti
muovi hip 2 <destro/sinistro> di <intero> gradi avanti
muovi hip 2 <destro/sinistro> di <intero> gradi indietro
muovi hip 2 <destro/sinistro> di <intero> gradi in alto
muovi hip 2 <destro/sinistro> di <intero> gradi verso l'alto
muovi hip 2 <destro/sinistro> di <intero> gradi in basso
muovi hip 2 <destro/sinistro> di <intero> gradi verso il basso

Figura 6.7: Primitive *Hip 2*.

alza knee <destro/sinistro> di <intero> gradi
abbassa knee <destro/sinistro> di <intero> gradi
sposta knee <destro/sinistro> di <intero> gradi in avanti
sposta knee <destro/sinistro> di <intero> gradi avanti
sposta knee <destro/sinistro> di <intero> gradi indietro
muovi knee <destro/sinistro> di <intero> gradi in avanti
muovi knee <destro/sinistro> di <intero> gradi avanti
muovi knee <destro/sinistro> di <intero> gradi indietro
muovi knee <destro/sinistro> di <intero> gradi in alto
muovi knee <destro/sinistro> di <intero> gradi verso l'alto
muovi knee <destro/sinistro> di <intero> gradi in basso
muovi knee <destro/sinistro> di <intero> gradi verso il basso

Figura 6.8 : Primitive *knee*.

alza ankle 1 <destro/sinistro> di <intero> gradi
abbassa ankle 1 <destro/sinistro> di <intero> gradi
sposta ankle 1 <destro/sinistro> di <intero> gradi in avanti
sposta ankle 1 <destro/sinistro> di <intero> gradi avanti
sposta ankle 1 <destro/sinistro> di <intero> gradi indietro
muovi ankle 1 <destro/sinistro> di <intero> gradi in avanti
muovi ankle 1 <destro/sinistro> di <intero> gradi avanti
muovi ankle 1 <destro/sinistro> di <intero> gradi indietro
muovi ankle 1 <destro/sinistro> di <intero> gradi in alto
muovi ankle 1 <destro/sinistro> di <intero> gradi verso l'alto
muovi ankle 1 <destro/sinistro> di <intero> gradi in basso
muovi ankle 1 <destro/sinistro> di <intero> gradi verso il basso

Figura 6.9 : Primitive ankle1.

ruota ankle 2 <destro/sinistro> di <intero> gradi in senso orario
ruota ankle 2 <destro/sinistro> di <intero> gradi in senso antiorario
ruotare ankle 2 <destro/sinistro> di <intero> gradi in senso orario
ruotare ankle 2 <destro/sinistro> di <intero> gradi in senso antiorario

Figura 6.10 : Primitive *ankle 2*.

ruota ankle 3 <destro/sinistro> di <intero> gradi in senso orario
ruota ankle 3 <destro/sinistro> di <intero> gradi in senso antiorario

Figura 6.11 : Primitive *ankle 3*.

ruota waist 1 di <intero> gradi in senso orario
ruota waist 1 di <intero> gradi in senso antiorario
ruota waist 1 di <intero> gradi a destra
ruota waist 1 di <intero> gradi verso destra
ruota waist 1 di <intero> gradi a sinistra
ruota waist 1 di <intero> gradi verso sinistra

Figura 6.12: Primitive *Waist 1*.

alza waist 2 di <intero> gradi
abbassa waist 2 di <intero> gradi
muovi waist 2 di <intero> gradi in avanti
muovi waist 2 di <intero> gradi avanti
muovi waist 2 di <intero> gradi indietro

Figura 6.13: Primitive *Waist 2*.

I comportamenti caratterizzati da movimenti simultanei si realizzano richiamando sullo stesso stage più primitive relative a giunti diversi; comportamenti caratterizzati da movimenti consecutivi si effettuano richiamando le primitive su stage successivi. Si analizzino i seguenti movimenti:

alza shoulder 2 dx di 50 gradi e alza shoulder 2 sx di 50 gradi

alza shoulder 2 dx di 50 gradi
alza shoulder 2 sx di 50 gradi.

Apparentemente possono sembrare il medesimo movimento, ma in realtà non lo sono; il primo richiama sullo stesso stage il movimento elementare di due giunti andando a realizzare un movimento simultaneo, mentre il secondo realizza un movimento consecutivo degli stessi giunti.
Ad ogni stage può essere associata una velocità d'esecuzione; ciò è possibile facendo concludere la descrizione di uno stage dalla frase:

alla velocità di <intero>.

E' inoltre possibile utilizzare delle strutture che permetto di definire quante volte un comportamento deve essere ripetuto nel seguente modo. La struttura tipo di un ciclo è la seguente:

ripeti n
muovi hip 1 dx di 3 gradi verso l'interno con velocità
stop.

Dalla combinazione di movimenti simultanei e consecutivi a velocità costanti o variabili sarà possibile realizzare i comportamenti desiderati.
Tutti i comportamenti realizzati potranno essere salvati in una libreria di comportamenti ed essere riutilizzati per creare movimenti complessi.
Descritto il linguaggio realizzato va giustificato il perché di certe scelte progettuali.
Si è scelto di utilizzare espressioni simil parlate che richiamassero i nomi dei giunti in maniera analoga alle terminologie usate sia nel simulatore che nel RobovieMaker, in quanto è risultata più facile ed accessibile una programmazione dei comportamenti di questo tipo, dato che in una fase iniziale delle sperimentazioni è stato indispensabile lo studio di questi programmi ed è diventato intuitivo e facile per tutto il gruppo di lavoro comunicare utilizzando espressioni di questo tipo.

6.2 La traduzione dei comportamenti

Come in precedenza accennato il software RobovieMovement si basa su di un traduttore che converte i comportamenti descritti mediante il linguaggio analizzato nel paragrafo precedente, in stringhe significative.
Realizzata la programmazione del comportamento complesso come combinazione delle varie primitive del linguaggio, è eseguita la compilazione e

quindi la traduzione in stringhe particolari che contengono le varie informazioni relative ai giunti.
Per chiarire meglio quando detto saranno analizzate le varie fasi della creazione e traduzione di un comportamento complesso.
Si supponga di aver programmato un comportamento come combinazione delle seguenti primitive:

alza shoulder 2 destro di 50 gradi
alza shoulder 2 sinistro di 50 gradi

Eseguita la traduzione del comportamento complesso si otterrà l'uscita mostrata in figura 6.14.

```
@21A7fB8eC1bD80E7aF80G1eHa4I10Je0K80L71Me4N7fO85P7fQe1R5bSefT1fUb2V80,+1!00
@21A7fB8eC1bD80E7aF80G1eHa4I10Je0K80L71Me4N7fO85P7fQe1R8dSefT1fUb2V80,+1!00
@21A7fB8eC1bD80E7aF80G1eH72I10Je0K80L71Me4N7fO85P7fQe1R8dSefT1fUb2V80,+1!00
@21A7fB8eC1bD80E7aF80G1eHa4I10Je0K80L71Me4N7fO85P7fQe1R5bSefT1fUb2V80,+0!00
```

Figura 6.14 Le stringhe d'uscita del comportamento programmato.

Come si può vedere l'uscita corrispondetene al comportamento complesso è composta da 4 stringhe: la prima e l'ultima rappresentano rispettivamente la posizione iniziale e finale, la seconda è la terza contengono informazioni sulle angolazioni dei vari motori nei singoli stage da attraversare al fine di realizzare il comportamento programmato.
Si analizzi nel dettaglio la stringa iniziale corrispondente allo stage zero per individuare e spiegare l'insieme d'informazioni in essa contenute. La stringa in questione è rappresentata nella figura 6.15.

```
@21A7fB8eC1bD80E7aF80G1eHa4I10Je0K80L71Me4N7fO85P7fQe1R5bSefT1fUb2V8,+1!00
```

Figura 6.15: La stringa di zero per RobovieMaker.

Si può subito notare come ogni gruppo di tre caratteri è composto da un carattere identificativo (@,A,B,..,V,",",!)rappresentato dalla lettera maiuscola dell'alfabeto americano e da due caratteri significativi.

La sottostringa @XX porta informazioni sulla velocità con la quale deve essere eseguita l'azione di passaggio allo stage: la @ è l'identificativo relativo alla velocità ed i due caratteri successivi (21) rappresentano in base esadecimale un intero, compreso nell'intervallo [0, 255].

Per tutte quelle sottostringhe in cui il primo carattere è l'identificativo A, B, …, V i due caratteri successivi, invece, rappresentano un intero, compreso nell'intervallo [0,255],rappresentato in base esadecimale [00, ff].

Per i gruppi di tre caratteri in cui il primo carattere è "," o "!", sono presenti informazioni riguardante i cicli.

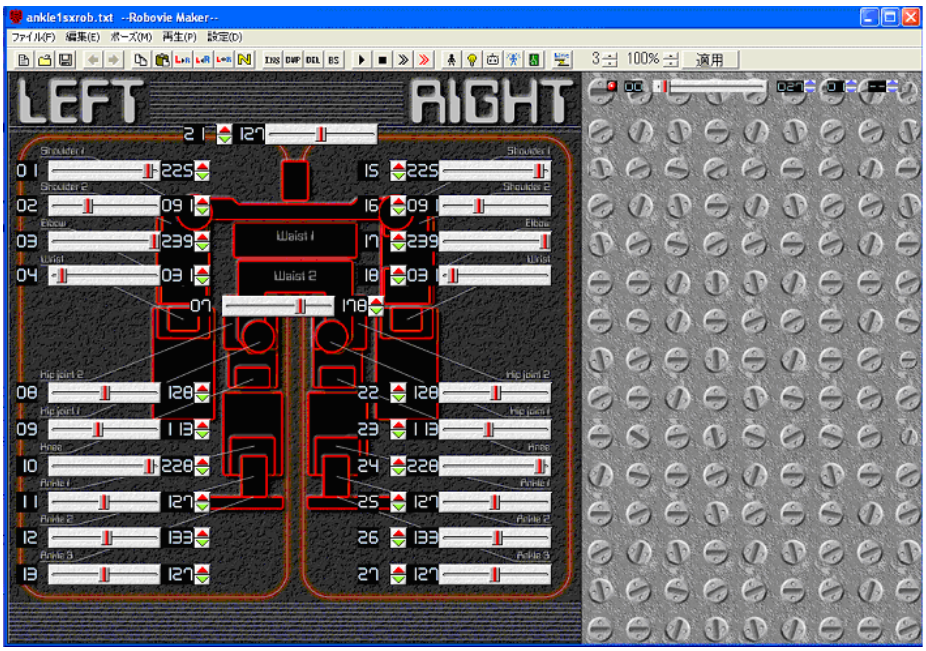


Figura 6.16 : La schermata principale di RobovieMaker della Vstone.

Ad ogni identificativo A, B, …, V è associato un giunto ed a ogni giunto è associata una posizione iniziale come si può vedere nella schermata principale di RobovieMaker .

Per comprendere meglio le associazioni tra l'identificativo del giunto, i caratteri significativi presenti nella stringa e la posizione iniziale di ogni giunto si consideri la tabella 2

Come si può notare per i giunti di destra, l'informazione, corrispondente all'identificativo, contenuta nella stringa, indica la posizione iniziale e coincide con il valore dello *slider* corrispondente al giunto come mostrato nella figura 6.16.

Per i giunti di sinistra, la posizione iniziale è indicata come il complementare a 255 del valore che si ricava dalla stringa.

Ad esempio in corrispondenza all'Hip 2 sinistro il cui identificativo è A l'informazione contenuta nella stringa in base esadecimale è 7f che in decimale corrisponde a 127; la posizione di zero sarà uguale a 255-127 = 128;

Ginto	Lettera	Informazione (Base esadecimale)	Informazione (Base decimale)	Posizione motore
Hip 2 sinistro	A	7f	127	128
Hip 1 sinistro	B	8e	142	113
Knee sinistro	C	1b	27	228
Ankle 1 sinistro	D	80	128	127
Ankle 2 sinistro	E	7a	122	133
Ankle 3 sinistro	F	80	128	127
Shoulder 1 sinistro	G	1e	30	225
Shoulder 2 sinistro	H	a4	164	91
Elbow sinistro	I	10	16	239
Wirst sinistro	J	e0	224	31
Hip 2 destro	K	80	128	128
Hip 1 destro	L	71	113	113
Knee destro	M	e4	228	228
Ankle 1 destro	N	7f	127	127
Ankle 2 destro	O	85	133	133
Ankle 3 destro	P	7f	127	127
Shoulder 1 destro	Q	e1	255	255
Shoulder 2 destro	R	5b	91	91
Elbow destro	S	ef	239	239
Wirst destro	T	if	31	31
Waist 2	U	b2	178	178
Waist 1	V	80	127	127

Tabella 2 : La corrispondenze tra gunti ,identificativo giunto(lettera) e la posizione di zero.

6.3 L'analisi delle stringhe da passare tramite porta seriale

Per rendere possibile l'esecuzione diretta dei movimenti è stato necessario studiare il modo con cui i valori dell'angolazione dei vari motori negli stage sono trasmessi.
Si è notato che non esiste una corrispondenza diretta tra le stringhe significative per "RobovieMaker" e quelle comunicate tramite RS232.
Per maggiore chiarezza, pertanto, si analizzi nel dettaglio la generica stringa creata per essere inviata al robot tramite seriale in relazione alla stringa creata per *RobovieMaker*, considerando la stringa relativa alla posizione iniziale di zero.
La stringa generata per RobovieMaker è rappresentata in figura 6.17:

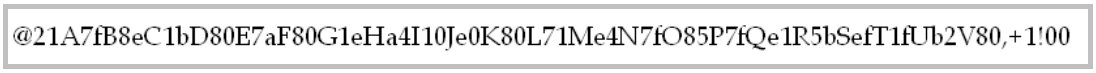


Figura 6.17: La stringa di zero per RobovieMaker.

Il cui significato è stato ampiamente discusso nel paragrafo precedente.
La generica stringa inviata al robot tramite RS232 relativa alla stringa suddetta è indicata nella figura 6.18:

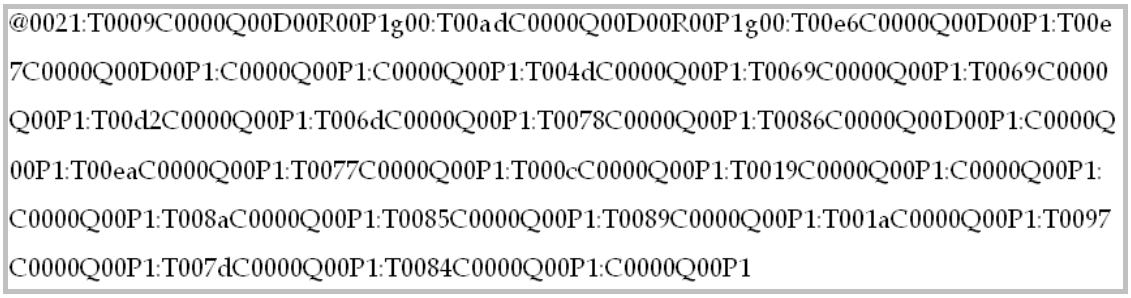


Figura 6.18: La stringa da inviare tramite porta seriale al robot.

La sottostringa @0021 porta informazioni sulla velocità con la quale deve essere eseguita l'azione di passaggio allo stage successivo.

Le informazioni contenute nelle sottostringhe successive sono invece delle informazioni che sono associate ai diversi motori del robot. Nella tabella 3 sono riportate le corrispondenze tra i giunto, l'identificativo del giunto, il motore e le stringhe che contengono dati riguardati le angolazioni relative a quel particolare giunto nello stage che si sta inviando.

Giunto	Identificativo Giunto	Identificativo motore	Sottostringa relativa al motore
Shoulder 1 destro	Q	1	:T0009C0000Q00D00R00P1g00
Shoulder 2 destro	R	2	:T00adC0000Q00D00R00P1g00
Elbow destro	S	3	:T00e6C0000Q00D00P1
Wirst destro	T	4	:T00e7C0000Q00D00P1
		5	C0000Q00P1
		6	C0000Q00P1
Waist 2	U	7	:T004dC0000Q00P1
Hip 2 destro	K	8	:T0069C0000Q00P1
Hip 1 destro	L	9	:T0069C0000Q00P1
Knee destro	M	10	:T00d2C0000Q00P1
Ankle 1 destro	N	11	:T006dC0000Q00P1
Ankle 2 destro	O	12	:T0078C0000Q00P1
Ankle 3 destro	P	13	:T0086C0000Q00P1
		14	:C0000Q00P1
Shoulder 1sinistro	G	15	:T00eaC0000Q00P1
Shoulder 2sinistro	H	16	:T0077C0000Q00P1
Elbow sinistro	I	17	:T000cC0000Q00P1
Wirst sinistro	J	18	:T0019C0000Q00P1
		19	:C0000Q00P1
		20	:C0000Q00P1
Waist 1	V	21	:T008aC0000Q00P1
Hip 2 sinistro	A	22	:T0085C0000Q00P1
Hip 1 sinistro	B	23	:T0089C0000Q00P1
Knee sinistro	C	24	:T001aC0000Q00P1
Ankle 1 sinistro	D	25	:T0097C0000Q00P1
Ankle 2 sinistro	E	26	:T007dC0000Q00P1
Ankle 3 sinistro	F	27	:T0084C0000Q00P1
		28	:C0000Q00P1

Tabella 3: La corrispondenza tra il giunto, il suo identificativo, il motore e la stringa da inviare.

Nella stringa inviata tramite porta seriale, le informazioni relative alle posizioni dei motori sono comunicate rispettando un ordine crescente, dal motore 1 al motore 28.

La sottostringa corrispondente alle informazioni sul motore 1 è riporta in figura 6.18.

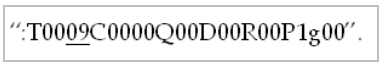


Figura 6.19: La sottostringa corrispondente al motore 1.

I caratteri sottolineati rappresentano in esadecimale un'informazione sull'angolazione del giunto. Il valore associato alla posizione di zero dello Shoulder 1 destro è differente da quella indicata per lo stesso giunto nella stringa per *RobovieMaker* visibile in figura 6.20:

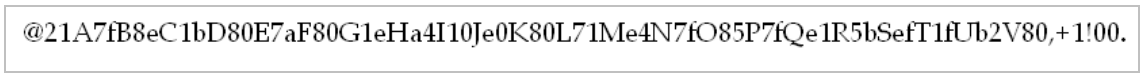


Figura 6.20: La stringa di zero per RobovieMaker.

Questo indica che i valori relativi alle angolazioni dei motori che devono essere passati tramite porta seriale, devono prima essere modificati secondo un particolare algoritmo. Ad ogni motore è associato un intero relativo particolare che è stato scoperto mediante numerose prove sperimentali. Le associazioni tra giunti e l'intero relativo introdotto sono riportate nella tabella 4.

Giunto	Identificativo Motore	Numero relativo associato
Shoulder 1 destro	1	-21
Shoulder 2 destro	2	9
Elbow destro	3	-9
Wirst destro	4	7
Waist 2	7	-101
Hip 2 destro	8	-23
Hip 1 destro	9	-8
Knee destro	10	-18
Ankle 1 destro	11	-18
Ankle 2 destro	12	-13
Ankle 3 destro	13	7
Shoulder 1 sinistro	15	9
Shoulder 2 sinistro	16	20
Elbow sinistro	17	-4
Wirst sinistro	18	-6
Waist 1	21	10
Hip 2 sinistro	22	6
Hip 1 sinistro	23	-5
Knee sinistro	24	-1
Ankle 1 sinistro	25	23
Ankle 2 sinistro	26	3
Ankle 3 sinistro	27	4

Tabella 4: La corrispondenza tra il giunto, l'identificativo del motore e l'identificativo del range.

- **VALORE SLIDER**: il valore letto sugli slider del RobovieMaker;
- **VALORE RANGE** : l'intero relativo caratterizzante il range di valori ammissibili per i servomotori;
- **RISULTATO** il valore che, convertito in esadecimale, sarà trasmesso tramite seriale.

L'algoritmo di conversione che è applicato alla stringa generata per *RobovieMaker* per ottenere la stringa significativa da comunicare al robot tramite seriale, per i giunti corrispondenti ai motori 1, 2, 4, 17,21,22,23,24,25,26,27, è il seguente:

RISULTATO =255- VALORE SLIDER + (VALORE RANGE);

Per tutti gli altri giunti la formula di conversione è:

RISULTATO = VALORE SLIDER +(VALORE RANGE)

Calcolato il valore della variabile RISULTATO, prima bisogna verificare se esso è compreso all'interno di un particolare intervallo, successivamente sarà convertito in cifra esadecimale e poi scritto sulla stringa da inviare tramite seriale. Tali intervalli variano da giunto a giunto ed in relazione a questo che viene fatta l'associazione tra il motore e l'intero relativo indicato nella tabella 4.Questi intervalli sono verificati sperimentalmente e riportati nelle tabelle 5/26.

Shoulder 1 destro/motore 1/Q

Valore RISULTATO	Valore da trasmettere (decimale)	Valore da trasmettere (esadecimale)
0	0	0
X	RISULTATO	(RISULTATO)hex
da 234 a 255	234	ea

Tabella 5: Il range di Shoulder 1 destro.

Shouder 2 destro/motore 2/R

Valore RISULTATO	Valore da trasmettere (decimale)	Valore da trasmettere (esadecimale)
255	255	ff
X	RISULTATO	(RISULTATO)hex
da 0 a 9	9	09

Tabella 6: Il range di Shoulder 2 destro.

Elbow destro/motore 3/S

Valore RISULTATO	Valore da trasmettere (decimale)	Valore da trasmettere (esadecimale)
255	255	ff
X	RISULTATO	(RISULTATO)hex
da 0 a 9	9	09

Tabella 7: Il range di Elbow destro.

Wirst destro/motore 4/T

Valore RISULTATO	Valore da trasmettere (decimale)	Valore da trasmettere (esadecimale)
255	255	ff
X	RISULTATO	(RISULTATO)hex
da 0 a 7	7	07

Tabella 8: Il range di Wirst destro**.**

Waist 2/motore 7/U

Valore RISULTATO	Valore da trasmettere (decimale)	Valore da trasmettere (esadecimale)
255	101	9a
X	RISULTATO	(RISULTATO)hex
da 0 a 101	0	0

Tabella 9: Il range di Waist 2**.**

Hip 2 destro/motore 8/K

Valore RISULTATO	Valore da trasmettere (decimale)	Valore da trasmettere (esadecimale)
da 232 a 255	232	e8
X	RISULTATO	(RISULTATO)hex
da 0 a 23	0	0

Tabella 10 : Il range di Hip 2 destro**.**

Hip 1 destro/motore 9/L

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere (esadecimale)
da 247 a 255	247	f7
X	RISULTATO	(RISULTATO)hex
da 0 a 8	0	0

Tabella 11: Il range di Hip 1 destro.

Knee destro/motore 10/M

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere (esadecimale)
da 237 a 255	237	ed
X	RISULTATO	(RISULTATO)hex
da 0 a 18	0	0

Tabella 12: Il range di Knee destro.

Ankle1 destro/motore 11/N

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere (esadecimale)
da 237 a 255	237	ed
X	RISULTATO	(RISULTATO)hex
da 0 a 18	0	0

Tabella 13 : lL range di Ankle 1 destro**.**

Ankle 2 destro/motore 12/O

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere (esadecimale)
da 242 a 255	242	f2
X	RISULTATO	(RISULTATO)hex
da 0 a 13	0	0

Tabella 14: Il range di Ankle 2 destro.

Ankle 3 destro/motore 13/P

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere (esadecimale)
da 0 a 7	7	07
X	RISULTATO	(RISULTATO)hex
da 248 a 255	255	ff

Tabella 15: Il range di Ankle 3 destro.

Shoulder 1 sinistro/motore 15/G

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere (esadecimale)
da 0 a 9	9	09
X	RISULTATO	(RISULTATO)hex
da 246 a 255	255	ff

Tabella 16: Il range di Shoulder 1 sinistro.

Shoulder 2 sinistro/motore 16/H

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere(esadecimale)
da 0 a 20	20	14
X	RISULTATO	(RISULTATO)hex
da 235 a 255	255	ff

Tabella 17 :Il range di Shoulder 2 sinistro.

Elbow sinistro/motore 17/I

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere(esadecimale)
0	0	0
X	RISULTATO	(RISULTATO)hex
da 251 a 255	251	fb

Tabella 18: Il range di Elbow sinistro.

Wairst sinistro/motore 18/J

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere (esadecimale)
255	249	f9
X	RISULTATO	(RISULTATO)hex
da 0 a 6	6	0

Tabella 19: Il range di Wairst sinistro.

Waist 1/motore 21/V

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere (esadecimale)
255	255	ff
X	RISULTATO	(RISULTATO)hex
da 0 a 10	10	0a

Tabella 20 : Il range di Waist 1 sinistro.

Hip 2 sinistro/motore 22/A

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere(esadecimale)
255	255	ff
X	RISULTATO	(RISULTATO)hex
da 0 a 6	6	06

Tabella 21: Il range di Hip2 sinistro**.**

Hip 1 sinistro/motore 23/B

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere (esadecimale)
0	0	0
X	RISULTATO	(RISULTATO)hex
da 250 a 255	250	fa

Tabella 22 :Il range di Hip 1sinistro.

Knee sinistro/motore 24/C

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere(esadecimale)
0	0	0
X	RISULTATO	(RISULTATO)hex
da 254 a 255	254	fe

Tabella 23: Il range di knee sinistro.

Ankle1 sinistro/motore 25/D

Valore RISULTATO	Valore da trasmettere (decimale)	Valore da trasmettere (esadecimale)
255	255	ff
X	RISULTATO	(RISULTATO)hex
da 0 a 23	23	17

Tabella 24 : Il range di Ankle 1 sinistro**.**

Ankle 2 sinistro/motore 26/E

Valore RISULTATO	Valore da trasmettere decimale)	Valore da trasmettere (esadecimale)
255	255	03
X	RISULTATO	(RISULTATO)hex
da 0 a 3	3	3

Tabella 25: Il range di Ankle 2 sinistra.

Ankle 3 sinistro/motore 27/F

Valore RISULTATO	Valore da trasmettere(decimale)	Valore da trasmettere (esadecimale)
255	255	ff
X	RISULTATO	(RISULTATO)hex
da 0 a 4	4	04

Tabella 26: Il range di Ankle 3 sinistro.

Il codice del metodo che permette di ottenere le stringhe che vengono comunicate tramite seriale è contenuto nell'appendice A.3

6.4 La descrizione del software RobovieMovement

Il software *RobovieMovement* esegue la traduzione dei comportamenti programmati con un linguaggio creato a hoc, in espressioni interpretabili o dal simulatore *Robostage* o dal software RobovieMaker o dal robot Robovie-M.
Per eseguire la traduzione dal linguaggio definito a stringhe significative, si esportano le sequenze di stage programmati su un file di testo; tale file viene

poi tradotto mediante gli analizzatori lessicali e sintattici, generando un file d'uscita il cui contenuto viene infine visualizzato.

Nell'appendice A sono contenuti i sorgenti relativi all'analizzatore lessicale ed all'analizzatore sintattico. Nella figura 6.21 è mostrata l'interfaccia principale del software. Nel programma è presente un componente di testo, *Programming window*, dove è possibile inserire i comportamenti relativi ad ogni singolo giunto mediante un linguaggio "simil-parlato" descritto nel paragrafo precedente.Per facilitare l'uso del programma, è presente un immagine dove sono indicati tutti i giunti dell'umanoide ed una lista ad albero, *Primitive ai giunti*, dove divise per giunti, vi sono le varie primitive che rappresentano esempi di corretta programmazione di movimenti elementari.

E' presente inoltre una casella di testo, *Comportamenti in libreria*, dove sono elencati tutti i comportamenti presenti in libreria.

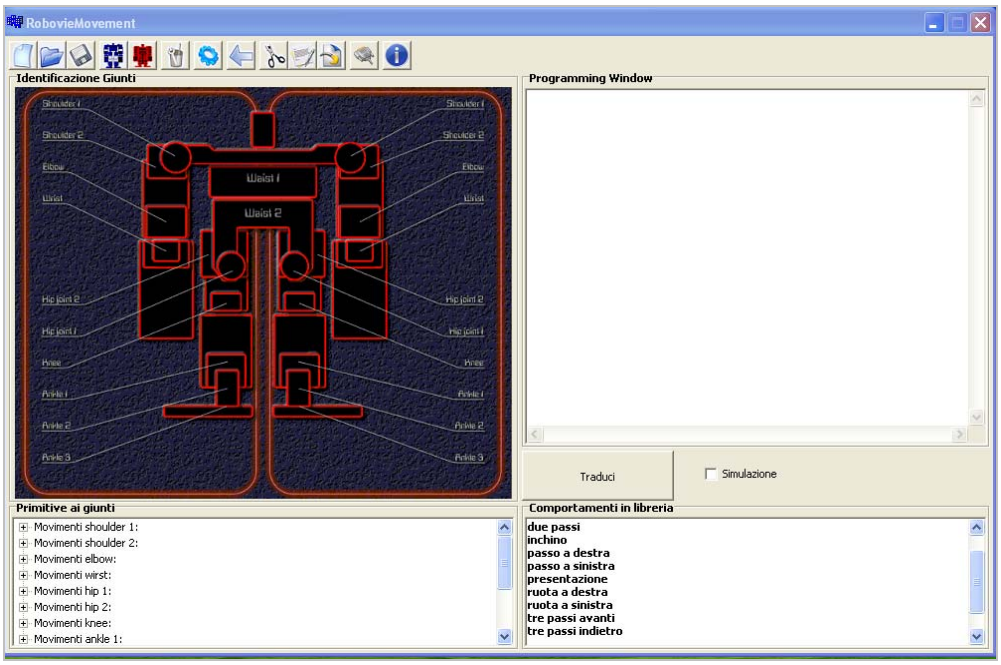


Figura 6.21: La schermata principale del RobovieMovement.

Prima di procedere all'analisi del linguaggio, va puntualizzato che Robovie-M com'è stato ampiamente descritto nei paragrafi precedenti, è un esecutore di comportamenti statici e quindi nel momento in cui si progetta un suo

comportamento si deve pensare a quest'ultimo come ad una sequenza di stati d'equilibrio stabile. In questo contesto più che di stati si parla di stage; ogni stage rappresenta una particolare configurazione dei motori associati ai vari giunti del robot.

Una successione di stage permette di definire un comportamento più o meno complesso. Si supponga di programmare la seguente successione di stage, come mostrato nella figura 6.22 e figura 6.23:

muovi hipr 1 dx di 50 gradi verso l'esterno
muovi hipr 1 sx di 50 gradi verso l'esterno
muovi hip 1 dx di 50 gradi verso l'interno e muovi hip 1 sx di 50 gradi verso l'interno.

Figura 6.22: Un esempio di comportamento programmato**.**

Ad ogni riga corrisponde uno stage ed in ogni stage è possibile modificare il valore relativo all'angolazione di qualsiasi giunto, rispetto al valore che lo stesso aveva nello stage precedente.

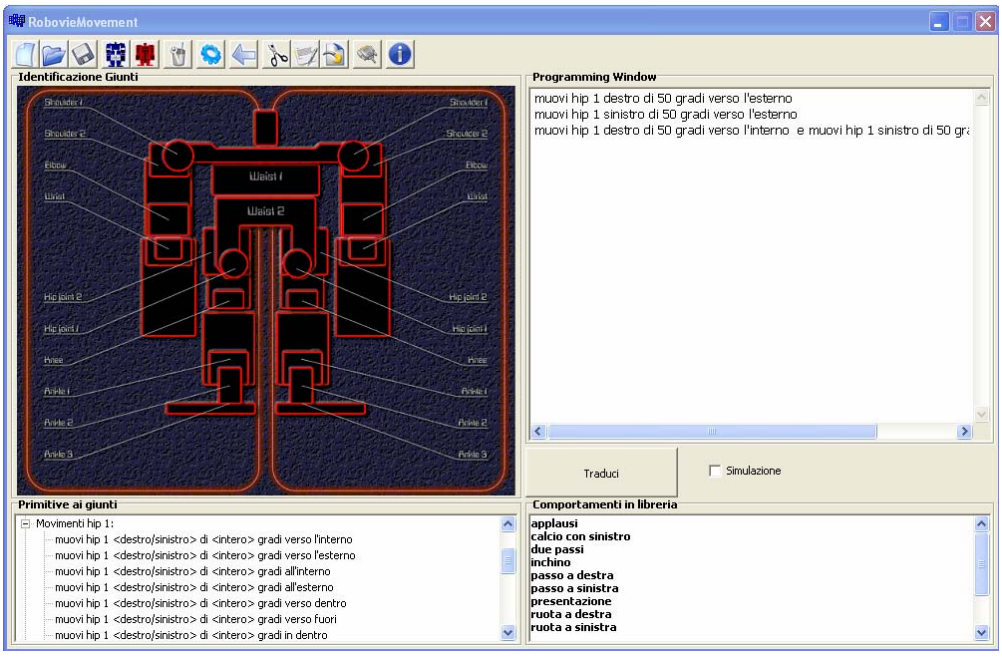


Figura 6.23: La programmazione di un movimento complesso su RobovieMovement.

La compilazione di questo programma dà in uscita un movimento complesso composto da 5 stage così composti:

Lo *Stage 0* rappresenta posizione di zero. In questo stage Il robot assume la posizione iniziale, è questa la prima configurazione che assume il robot appena viene acceso.

Lo *Stage 1* è creato in corrispondenza alla traduzione del comportamento: muovi hip 1 dx di 50 gradi verso l'esterno.

In questo stage il valore di tutti i giunti rimane invariato rispetto allo stage precedente tranne il valore relativo al giunto hip 1 destro che varia di 50 gradi.

Lo *Stage 2* è creato in corrispondenza al comportamento: muovi hip 1 sx di 50 gradi verso l'esterno.

In questo stage il valore di tutti i giunti rimane invariato rispetto allo stage precedente tranne il valore relativo al giunto hip 1 sinistro che varia di 50 gradi.

Lo *Stage 3* è creato in corrispondenza a muovi hip 1 dx di 50 gradi verso l'interno e muovi hip 1 sx di 50 gradi verso l'interno.

In questo stage il valore di tutti i giunti rimane invariato rispetto allo stage precedente tranne il valore relativo al giunto shoulder1 sinistro che varia di 50 gradi e quello relativo al giunto shoulder1 destro che varia anch'esso di 50 gradi.

Nello *Stage 4* il robot torna nella posizione di zero. Il robot assume la posizione iniziale, i valori ai giunti sono gli stessi dello stage 0.

Definito un comportamento si può decidere di tradurre il file per la simulazione o meno ed ottenere a seconda della scelta file d'uscita differenti.

Come si può vedere dalla figura 6.3 accanto al pulsante che avvia la traduzione c'è un *CheckBox* che permette di decidere che tipo di traduzione fare.

Nel caso si decida di compilare per la simulazione, si otterrà in uscita un file che consente la visualizzazione del movimento tramite il simulatore sviluppato presso l'Università degli Studi di Padova.

Nell'altra ipotesi si otterranno due file: uno può essere aperto direttamente dal software *RobovieMaker* in modo tale da trasferirlo al robot e farlo eseguire, e l'altro può essere inviato direttamente al robot tramite porta seriale.

Figura 6.24: Il dettaglio della barra degli strumenti di RobovieMovement.

Selezionando il pulsante *Traduci*  , se non si sono verificati errori dapprima comparirà un messaggio con la scritta "Compilazione eseguita con successo" come mostra nella figura 6.25 e contemporaneamente sulla schermata principale verrà visualizzato il pulsante *Visualizza Uscita*, mentre il tasto *Invia al Robot*  verrà abilitato.

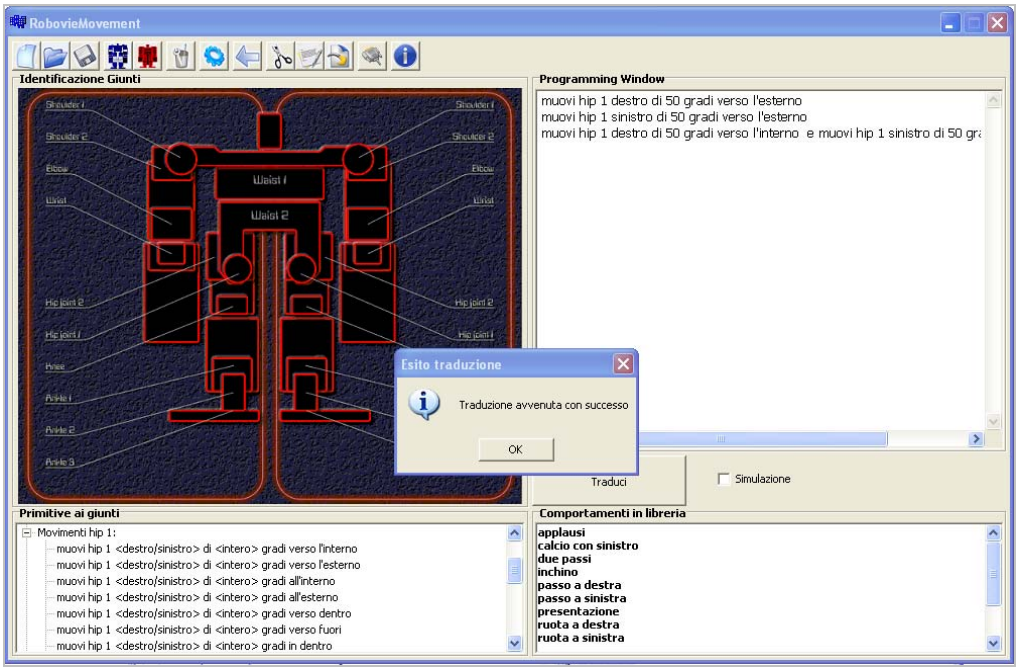


Figura 6.25: Il messaggio che comunica l'esito della traduzione**.**

Selezionato il tasto *ok* e cliccato il pulsante *Visualizza Uscita* sarà possibile visionare la sequenza d'uscita com'è mostrata nella figura 6.26.

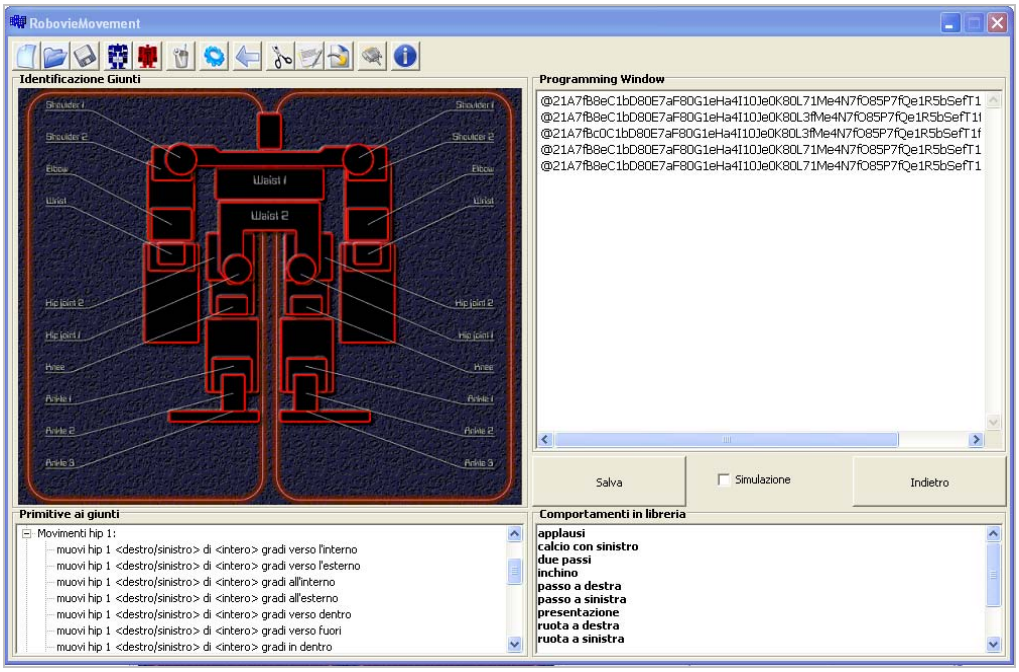


Figura 6. 26: La visualizzazione della sequenza d'uscita.

Selezionando il tasto *Salva*  saranno salvati la sequenza d'output ed il sorgente del comportamento e sarà chiesto se s'intende inserire il comportamento in libreria come mostrato dalla figura 6.27.

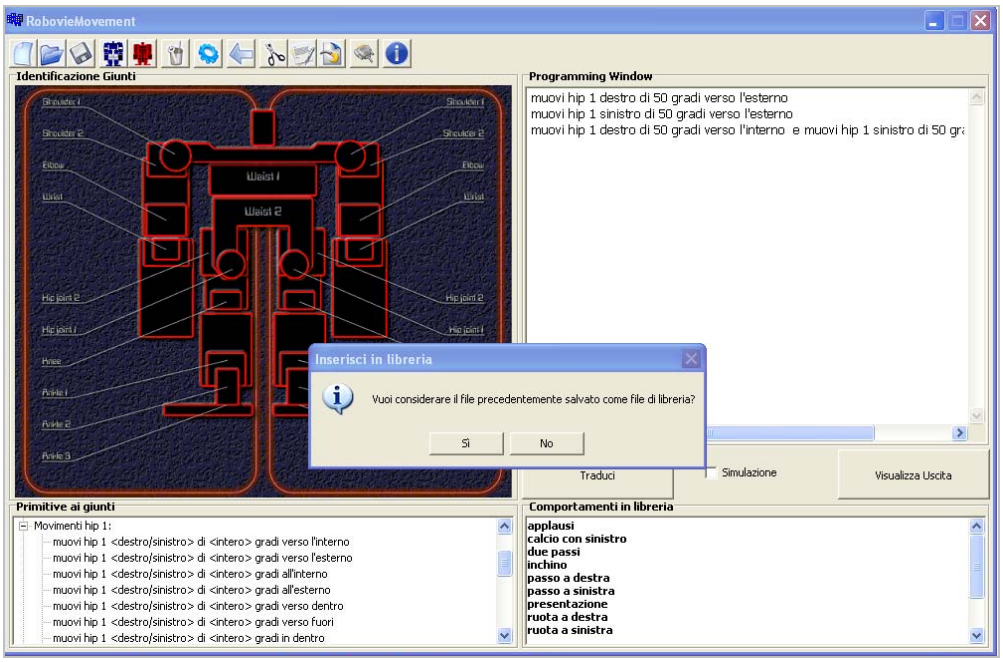


Figura 6.27: Il messaggio per il salvataggio del comportamento in libreria.

A questo punto se la compilazione non è stata eseguita per il simulatore, e si vuole caricare il file su RobovieMaker basta selezionare il tasto *RobovieMaker*  nella barra degli strumenti; viceversa se la compilazione è stata fatta per il simulatore basta selezionare il tasto $-3concernente il simulatore 

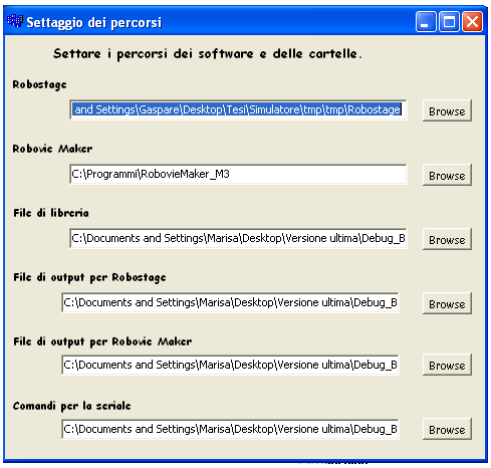


Figura 6.28: La schermata di RobovieMovement per impostare i percorsi.

Per aprire *Robostage* o *RobovieMaker* occorre prima settare il percorso in cui questo si trova. Per fare ciò basta selezionare il pulsante  e comparirà una schermata, visibile in figura 6.28, in cui impostare i percorsi richiesti.

Dopo aver creato e salvato in libreria due comportamenti distinti sarà possibile riutilizzarli per la programmazione di un altro comportamento. Si supponga di avere in libreria due comportamenti distinti il primo chiamato *tre passi* che permette di ottenere l'esecuzione di tre passi, ed uno chiamato *presentazione* che permette di ottenere un movimento di tipo coreografico.

Se nella programming window tali comportamenti sono inseriti modo rappresentato in figura 2.29:

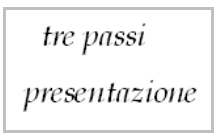


Figura 6.29: Un esempio di programmazione di comportamenti.

si otterrà un nuovo comportamento complesso ottenuto dalla combinazione dei due comportamenti di libreria in precedenza creati.

A sua volta questo comportamento potrà essere salvato in libreria e successivamente riutilizzato.

Selezionando il tasto *Invia al Robot*  sarà possibile inviare direttamente il comportamento al robot.

Mediante il software *RobovieMovement* è possibile, come descritto nei paragrafi precedenti comunicare direttamente al robot un comportamento tramite seriale, utilizzando l'applicazione presente nel *RobovieM TotalControl*, *Serial Tool.*

Nella figura 6.30 e 6.31 sono mostrati tutti i moduli presenti nel programma *RobovieM Total Control* tra cui *il Serial Tool* e *RobovieMovement*.

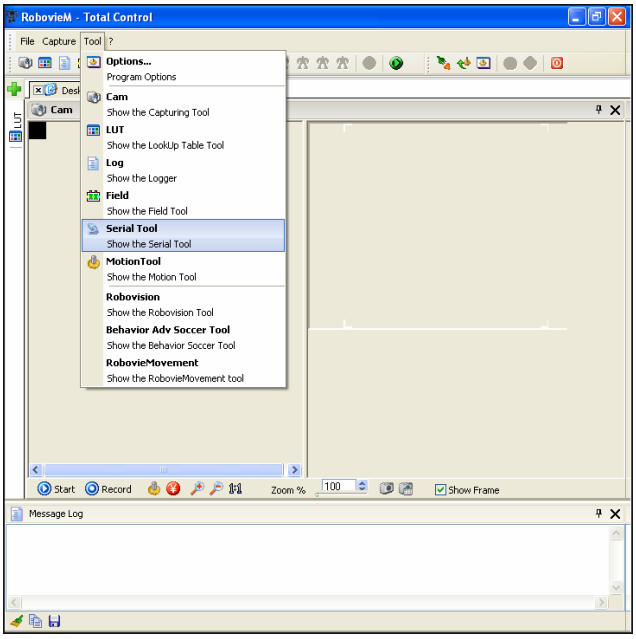


Figura 6.30: La schermata di RobovieM Total Control e dell'avvio dell'applicazione Serial Tool.

Questa applicazione, aperta la comunicazione con il robot, permette l'invio delle stringhe che costituiscono il movimento complesso programmato.

Ogni singola stringa inviata rappresenta uno stage nel quale il robot deve andare, mentre una sequenza di stage definisce un movimento.

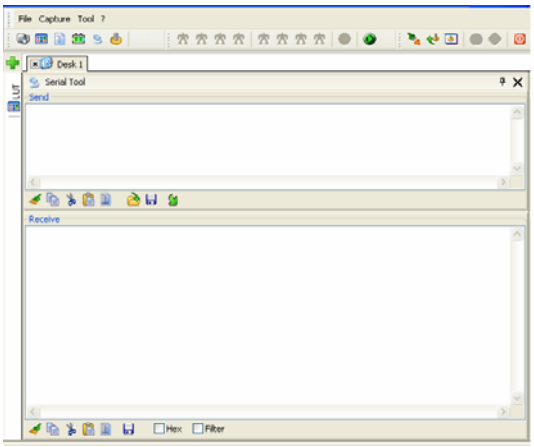


Figura 6.31: La schermata principale dell'applicazione Serial Tool.

Selezionato *Invia al Robot*  , le sequenze di messaggi create secondo l'algoritmo descritto nei paragrafi precedenti, vengono trasferite a *Serial Tool* come mostrato nella figura 6.31 e 6.32.

Se il Robot è gia connesso al PC ed i motori posti nella posizione iniziale basterà selezionare il pulsante *send* per eseguire il comportamento.

In fine, con il pulsante  si pulisce la *Programming window*, con il pulsante  , è possibile caricare file gia esistenti, con il pulsante  , si può ritornare indietro, mentre ai pulsanti  , sono associate rispettivamente le funzionalità taglia, copia ,incolla.

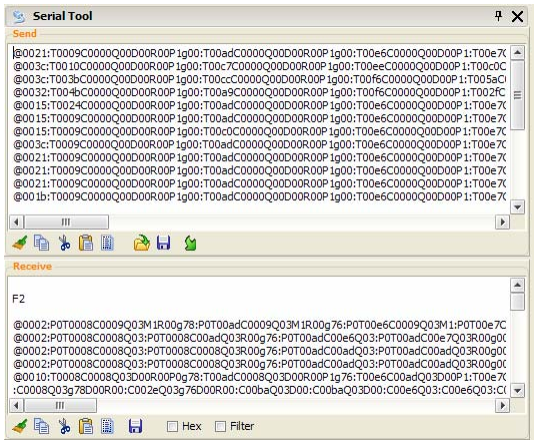


Figura 6.11:La schermata principale dell'applicazione Serial Tool con le stringhe da inviare tramite seriale.

Capitolo 7

I dati sperimentali

7.1 I risultati raggiunti.

I comportamenti elementari e complessi compiuti dal robot Robovie-M versione 3 sono stati ottenuti come sequenza di stage.
Ogni stage rappresenta una particolare configurazione del giunto ed è stato realizzato in modo da mantenere comunque la proiezione del baricentro del robot all'interno dell'area delimitata dalla base d'appoggio pertanto il robot si trova in una posizione d'equilibrio stabile in ogni stage. Interrompendo in un punto qualsiasi la sequenza, il robot rimane in quello stato senza cadere .Iniziando dalla configurazione di partenza in cui Robovie-M si trova in posizione eretta(detta posizione di zero) sono state sperimentate varie configurazioni per ottenere i movimenti voluti. Considerando i comportamenti umani si sono determinate in modo sperimentale le posizioni dei giunti nelle varie configurazioni. I comportamenti creati sono stati sviluppati utilizzando RobovieMovement con buoni risultati. Oltre ai comportamenti base, quali per esempio la movimentazione d'ogni singolo giunto, particolare importanza si è data allo sviluppo di comportamenti che potessero essere utili in ambiente Robocup. I principali comportamenti realizzati sono:

- La camminata frontale;
- La camminata laterale (sinistra e a destra);
- Le rotazioni a destra e sinistra di 30, 60,90, ….gradi;
- Il calcio della palla da destra e da sinistra;
- La distesa e il sollevamento da terra (a faccia in su e a faccia in giù);
- Le parate centrali, a destra ed a sinistra;
- Movimenti coreografici.

Nei paragrafi successivi sono riportati alcuni dei risultati ottenuti. Per ciascuno dei comportamenti portati ad esempio sarà fornito un set d'immagini che ne mostrano l'esecuzione eseguita dal robot e dal visualizzatore 3D.

7.2 La camminata in avanti

Come ampiamente descritto nella parte iniziale di questa tesi la realizzazione di una camminata bipede è un problema non indifferente.Nel realizzare questo particolare comportamento oltre a determinare la sequenza di stage corretta si è dovuto prestare molta attenzione alle velocità di transizioni degli stage al fine di ottenere una camminata stabile.

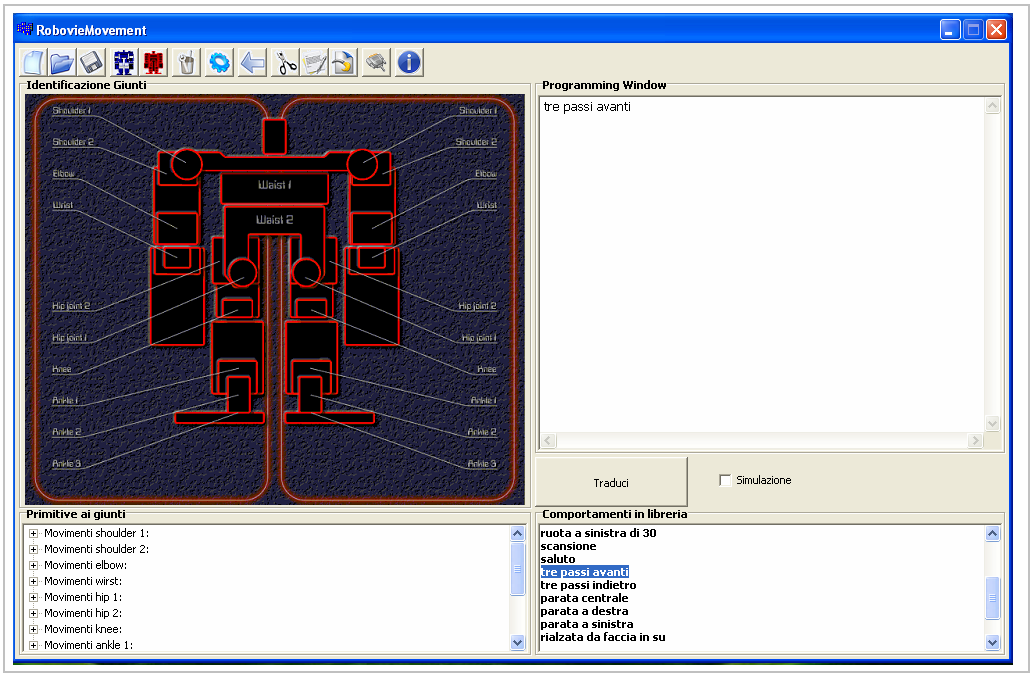


Figura 7.1: Il comportamento di libreria descritto: tre passi avanti.

Il comportamento è presente nella libreria realizzata come mostrato nella figura 7.1.Nelle Figure 7.2 e 7.3 è mostrata la sequenza di stage principali che caratterizzano il comportamento: la prima immagine mostra la sequenza in simulazione la seconda ne mostra l'esecuzione reale.

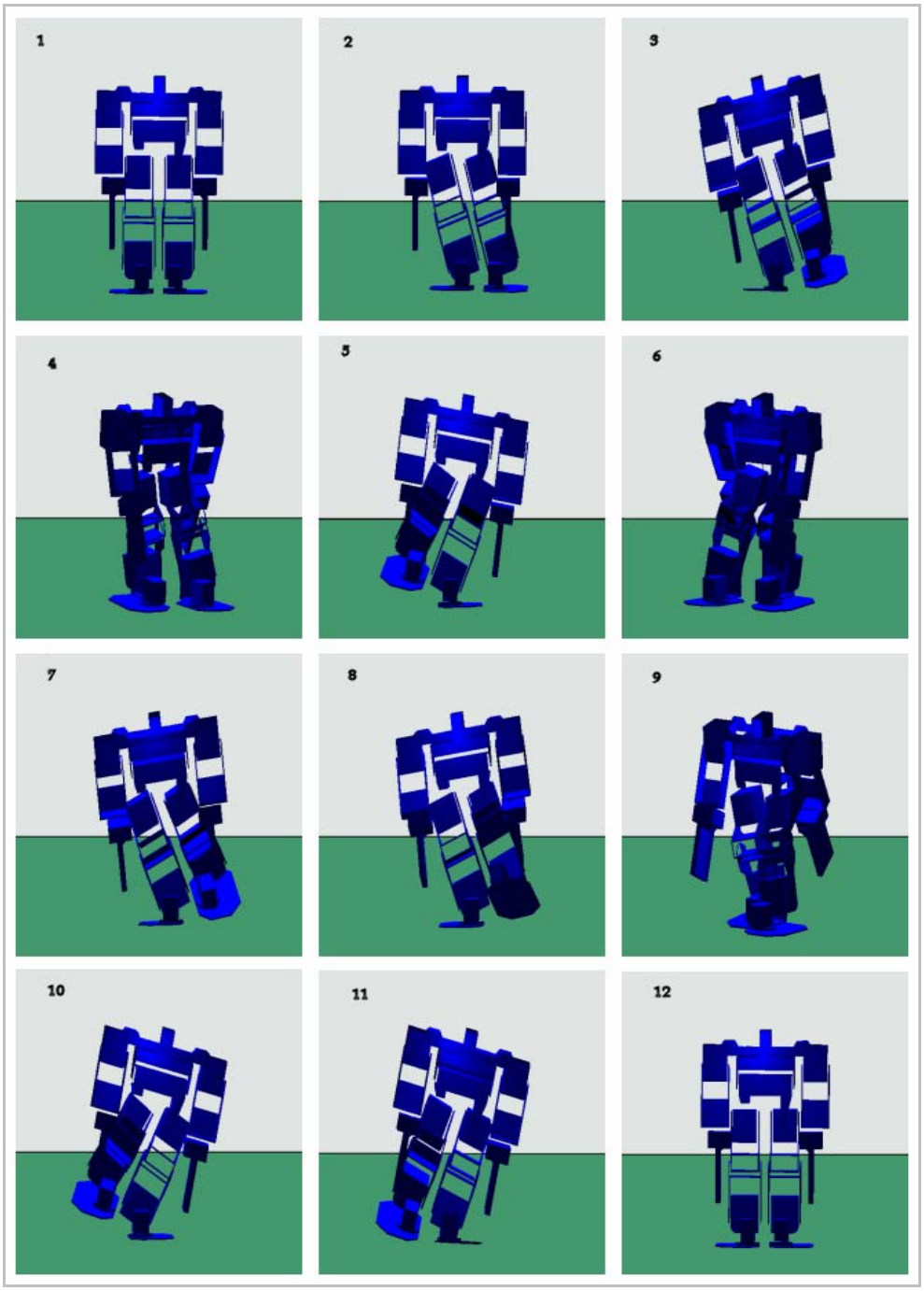


Figura 7.2: La successione di stage caratterizzante i tre passi avanti in Robostage.

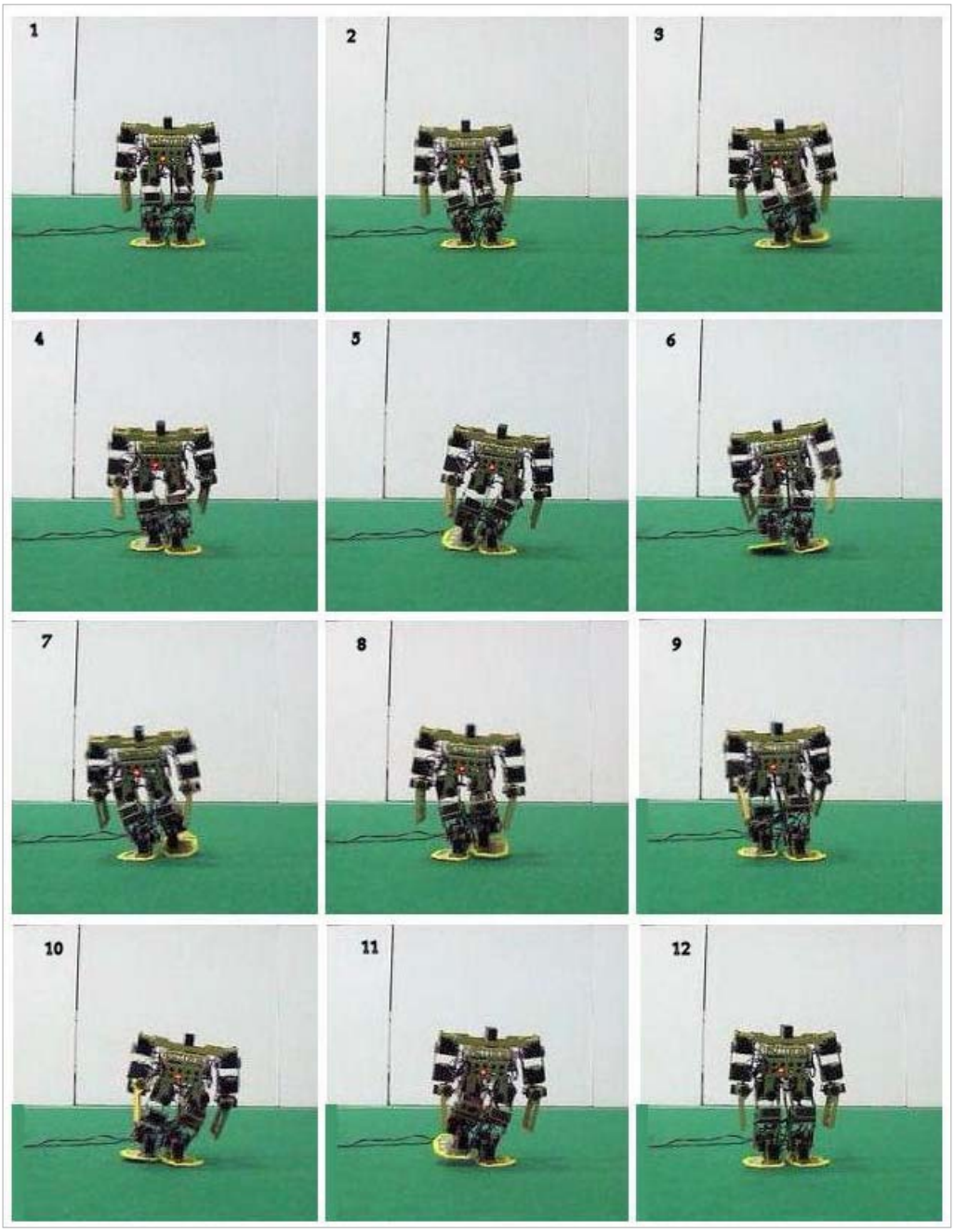


Figura 7.3: La successione di Stage caratterizzante l'esecuzione di tre passi in avanti.

7.3 La camminata indietro

Procedimento del tutto analogo a quello usato per la camminata in avanti è stato adottato per la programmazione della camminata indietro.
Anche in questo caso indispensabile è stato in ogni istante monitore l'equilibrio del robot e le velocità d'esecuzione dei vari stage.
Indispensabile in entrambi i casi è stata la gestione dei movimenti non solo delle gambe ma anche di tutti gli altri giunti; grazie alle variazione relative dei giunti, delle braccia e del busto, si è gestito meglio l'equilibrio del robot .

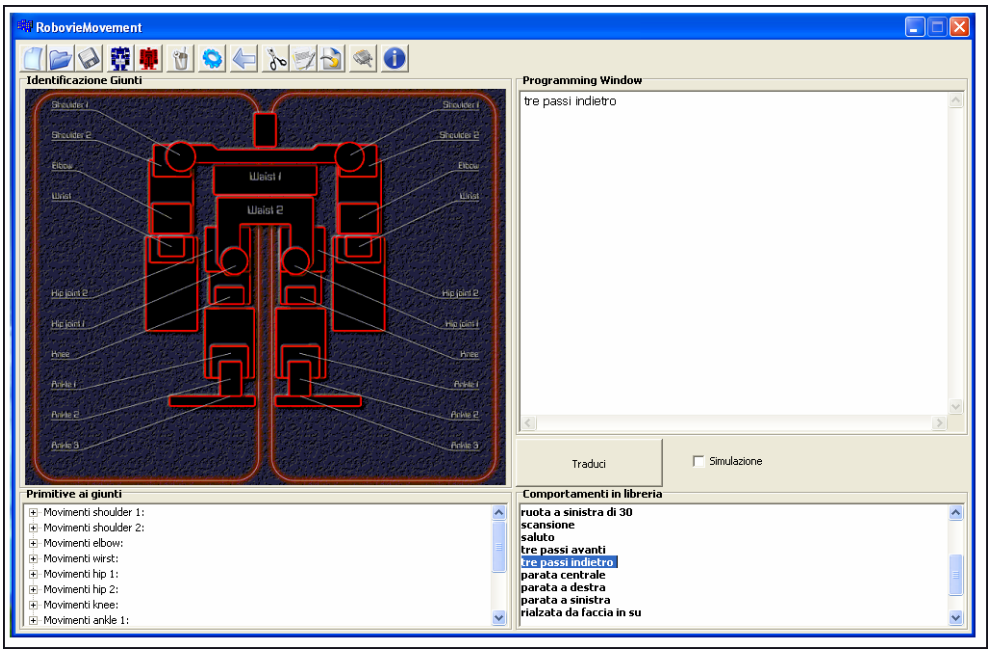


Figura 7.4: Il comportamento di libreria descritto: tre passi indietro.

Il comportamento è presente nella libreria realizzata come mostrato nella figura 7.4 mentre nelle Figure 7.5 e 7.6 è mostrata la sequenza di stage principali che caratterizzano il comportamento: la prima immagine mostra la sequenza in simulazione la secondo ne mostra l'esecuzione reale.

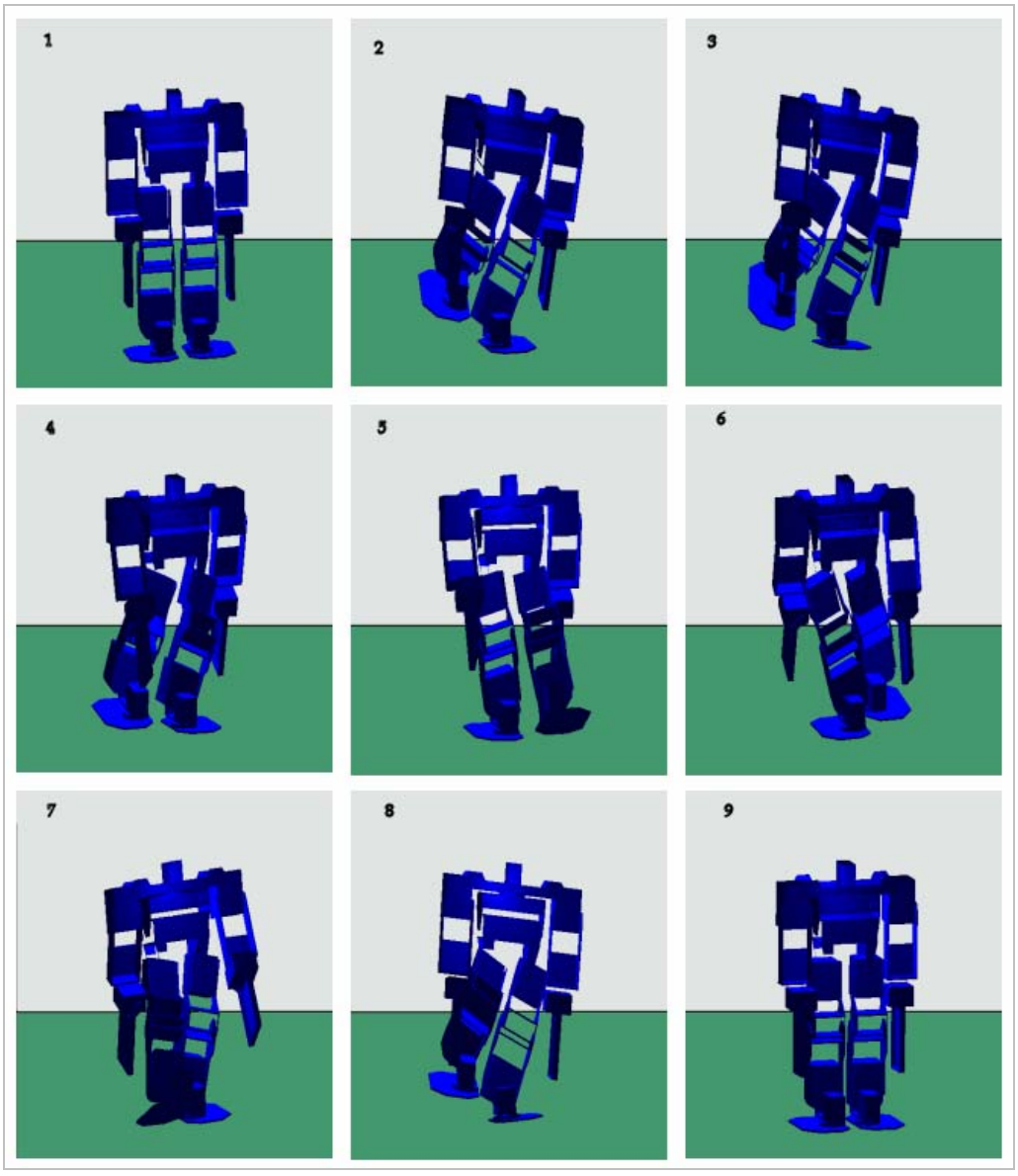


Figura 7.5: Successione di stage caratterizzante i tre passi indietro in Robostage.

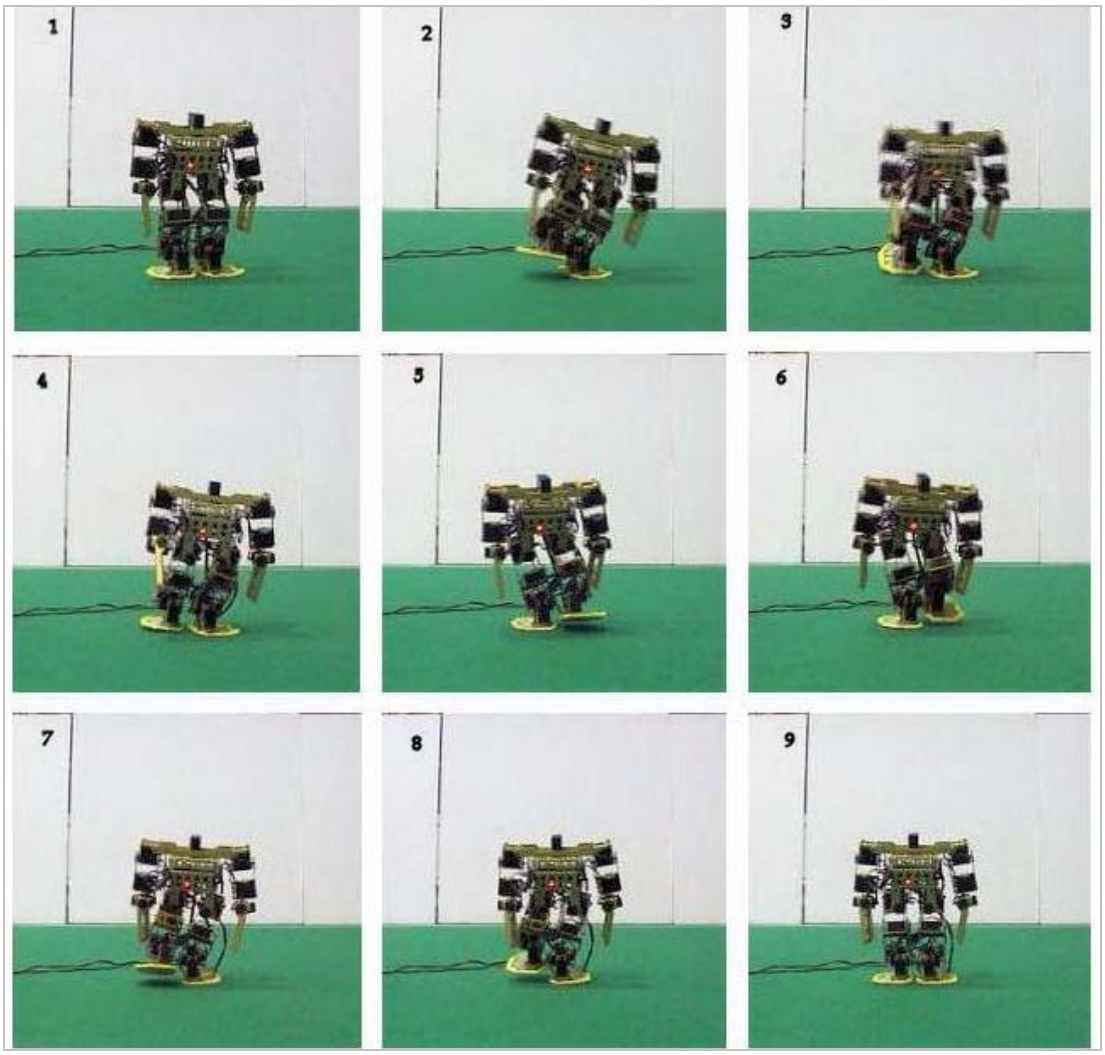


Figura 7.6: La successione di Stage caratterizzanti l'esecuzione di tre passi indietro

7.4 La camminata laterale a destra

Dopo la realizzazione di comportamenti che permettessero di ottenere la camminata in avanti e la camminata indietro si sono sviluppati quelli relativi alle camminate laterali verso una determinata direzione: la camminata verso destra e la camminata laterale verso sinistra.

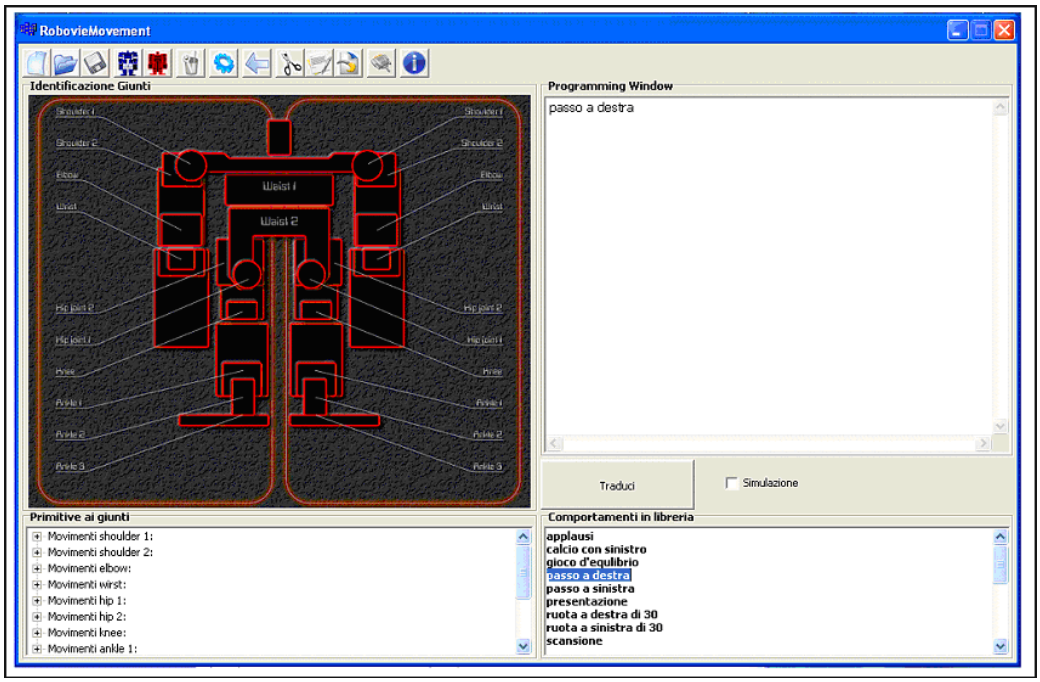


Figura 7.7: La schermata che mostra l'utilizzo del comportamento di libreria descritto: passo a destra.

Il comportamento è presente nella libreria realizzata come mostrato nella figura 7.4 mentre nelle 7.8 e 7.9 sono mostrate le successioni di stage eseguite dal visualizzatore 3D e dal robot.

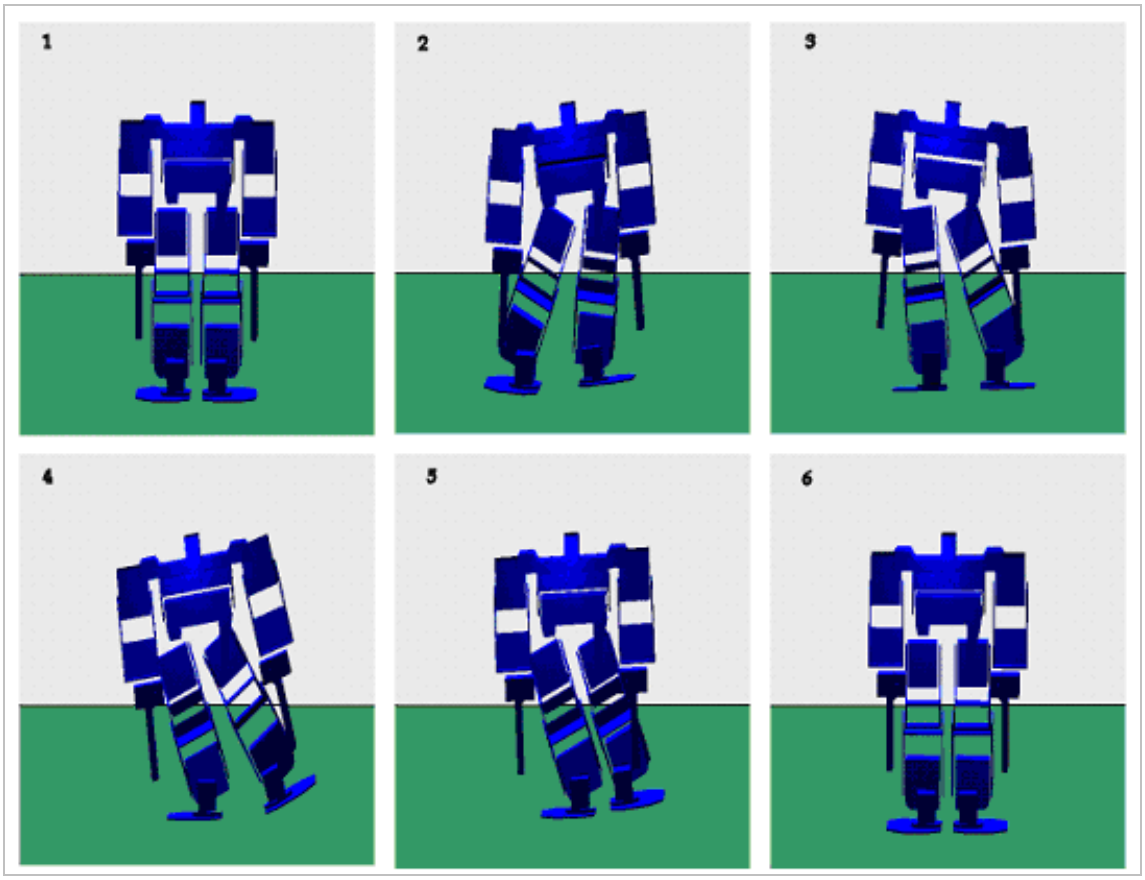


Figura 7.8: La successione di stage caratterizzante il passo a destra in Robostage.

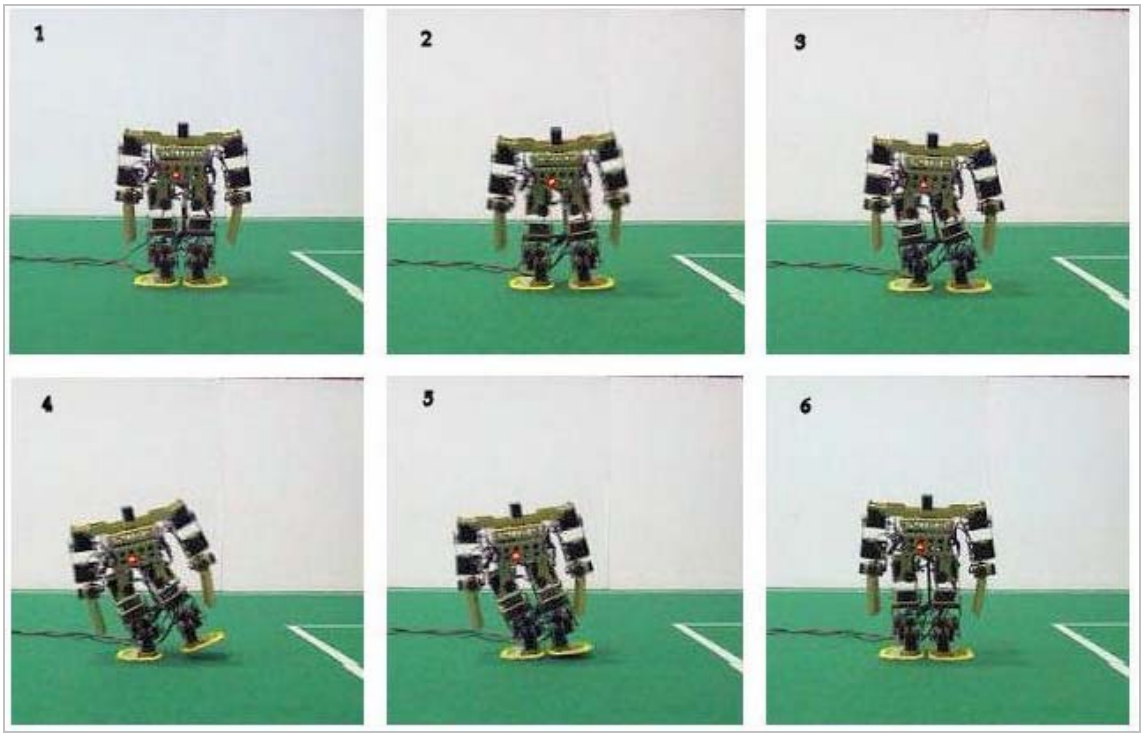


Figura 7.9: La successione di Stage caratterizzante l'esecuzione del passo a destra.

7.5 La camminata laterale a sinistra

Simmetricamente alla camminata verso destra si è realizzato un comportamento che ci permettesse di ottenere lo spostamento verso sinistra del robot.

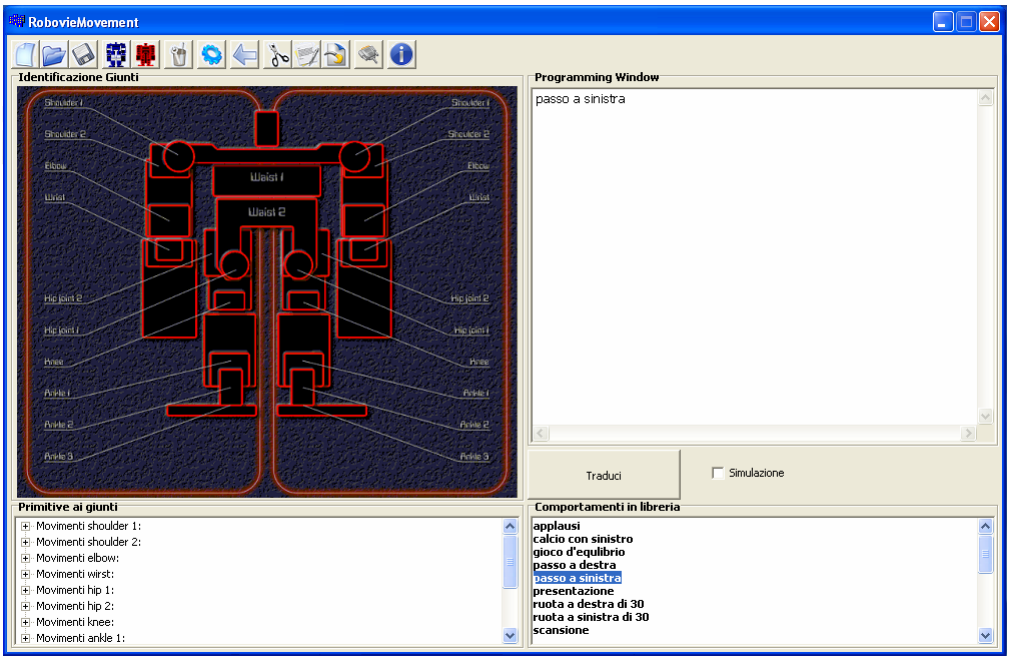


Figura 7.10: Il comportamento di libreria descritto: passo a sinistra.

Entrambi i comportamenti fanno parte della libreria realizzata come mostrato nelle figura 7. 7 e 7.10 . Le figure riportate mostrano l'esecuzione di un solo passo verso la direzione definita, ma risulta ovvio, che come combinazioni di questi passi si possono ottenere le camminate laterali

Nelle Figure 7.11 e 7.12 sono mostrate le successioni di stage eseguite dal visualizzatore 3D e dal robot

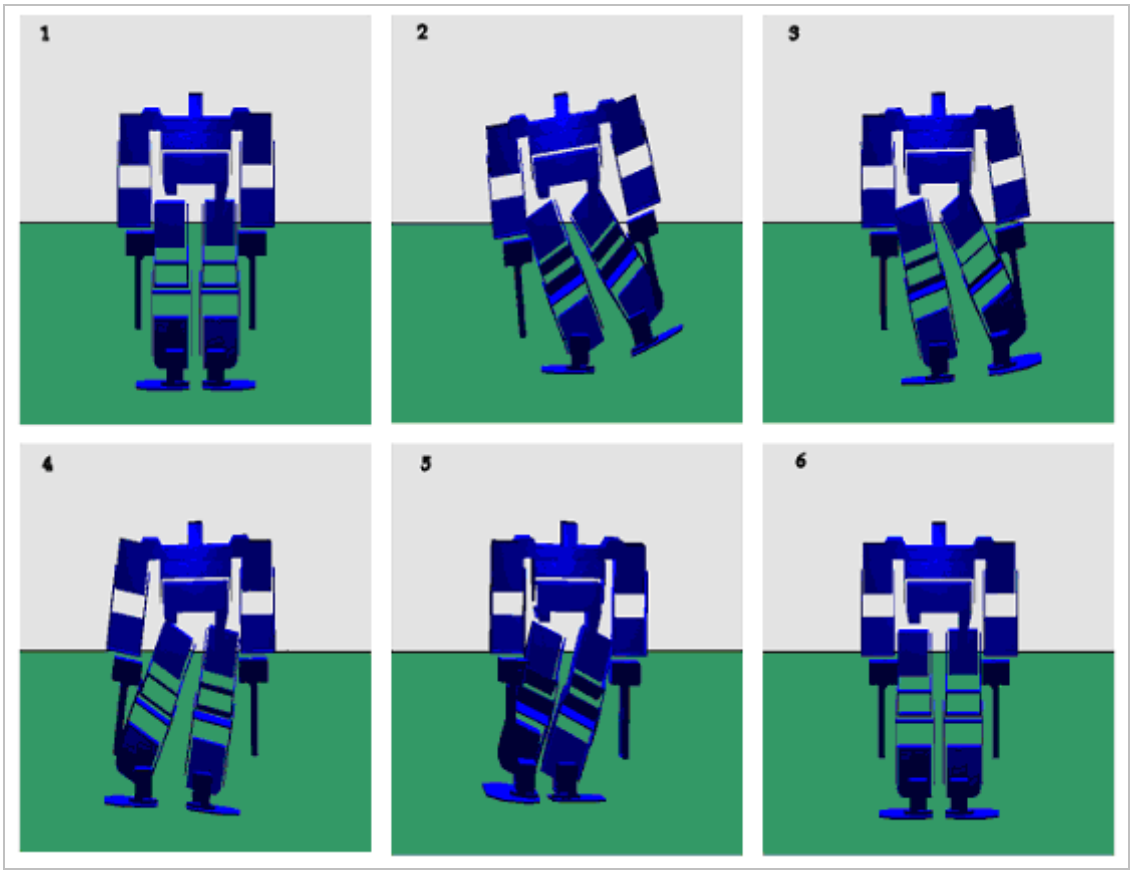


Figura 7.11 : La successione di stage caratterizzante il passo a sinistra in Robostage.

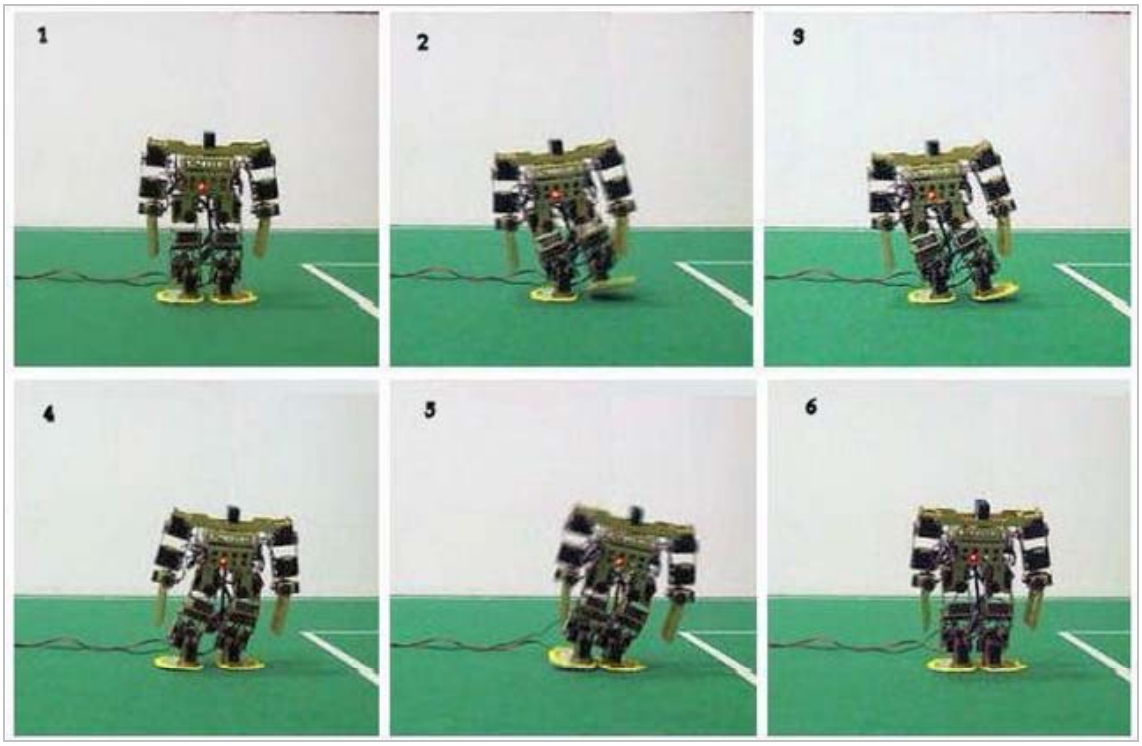


Figura 7.12: La successione di stage caratterizzante l'esecuzione del passo a destra.

7.6 Le rotazione a destra

Tra i comportamenti realizzati ci sono quelli che permettono al robot di ruotare tutto il corpo verso una direzione stabilita: la rotazione verso destra e la rotazione verso sinistra

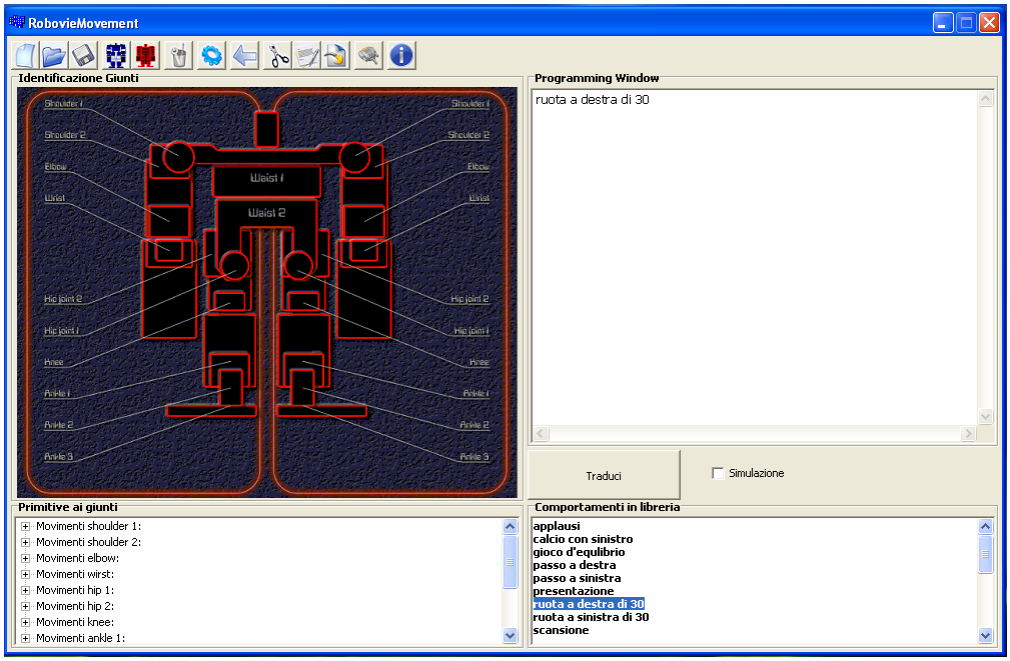


Figura 7.13 : Il comportamento di libreria descritto: rotazione a destra di 30 gradi.

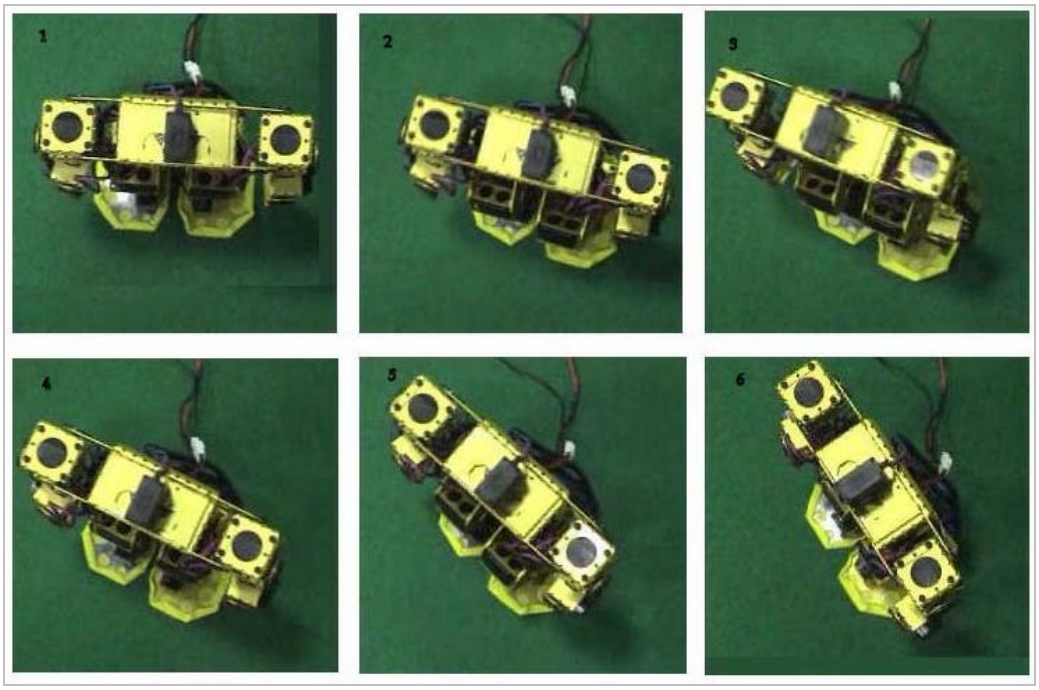


Figura 7.14 : La successione di stage caratterizzanti l'esecuzione della rotazione a destra vista dall'alto.

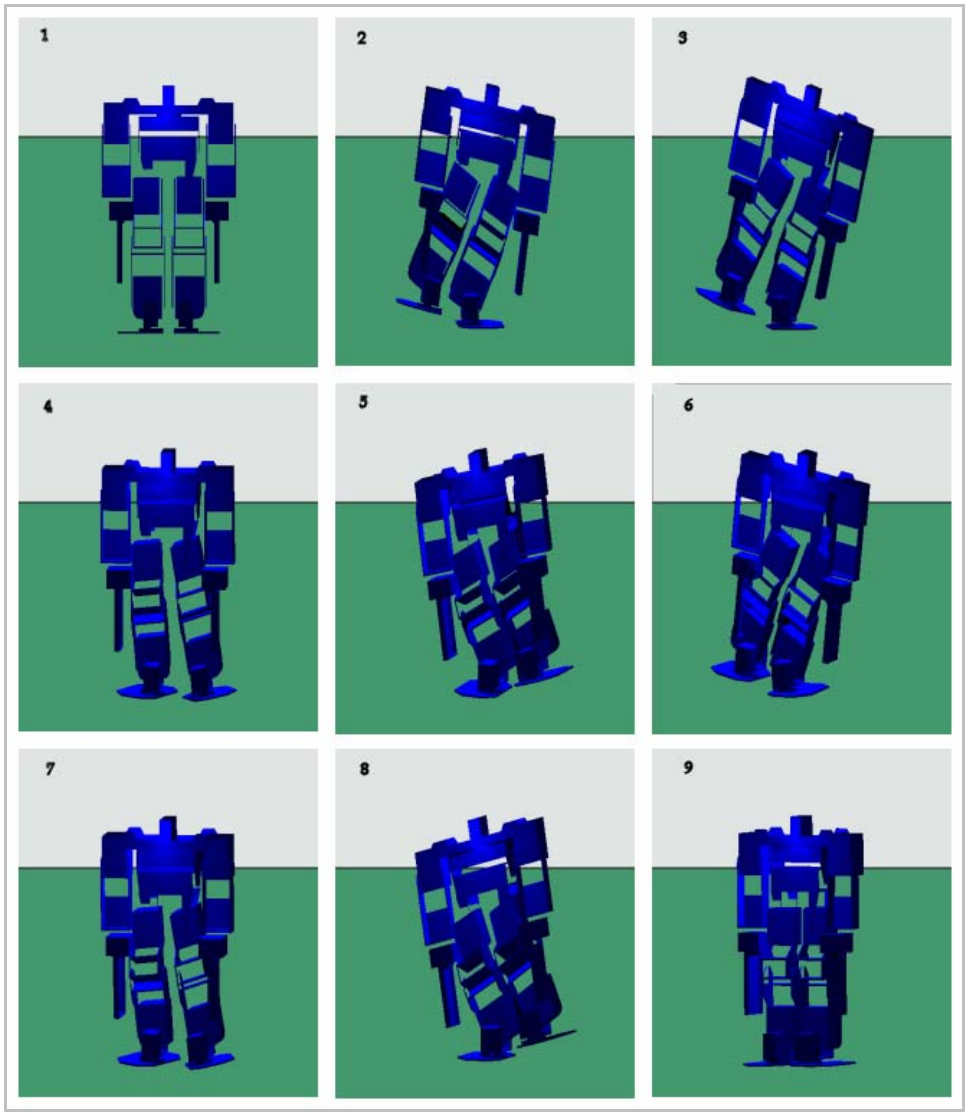


Figura 7.15 : La successione di stage caratterizzanti la rotazione a destra in Robostage.

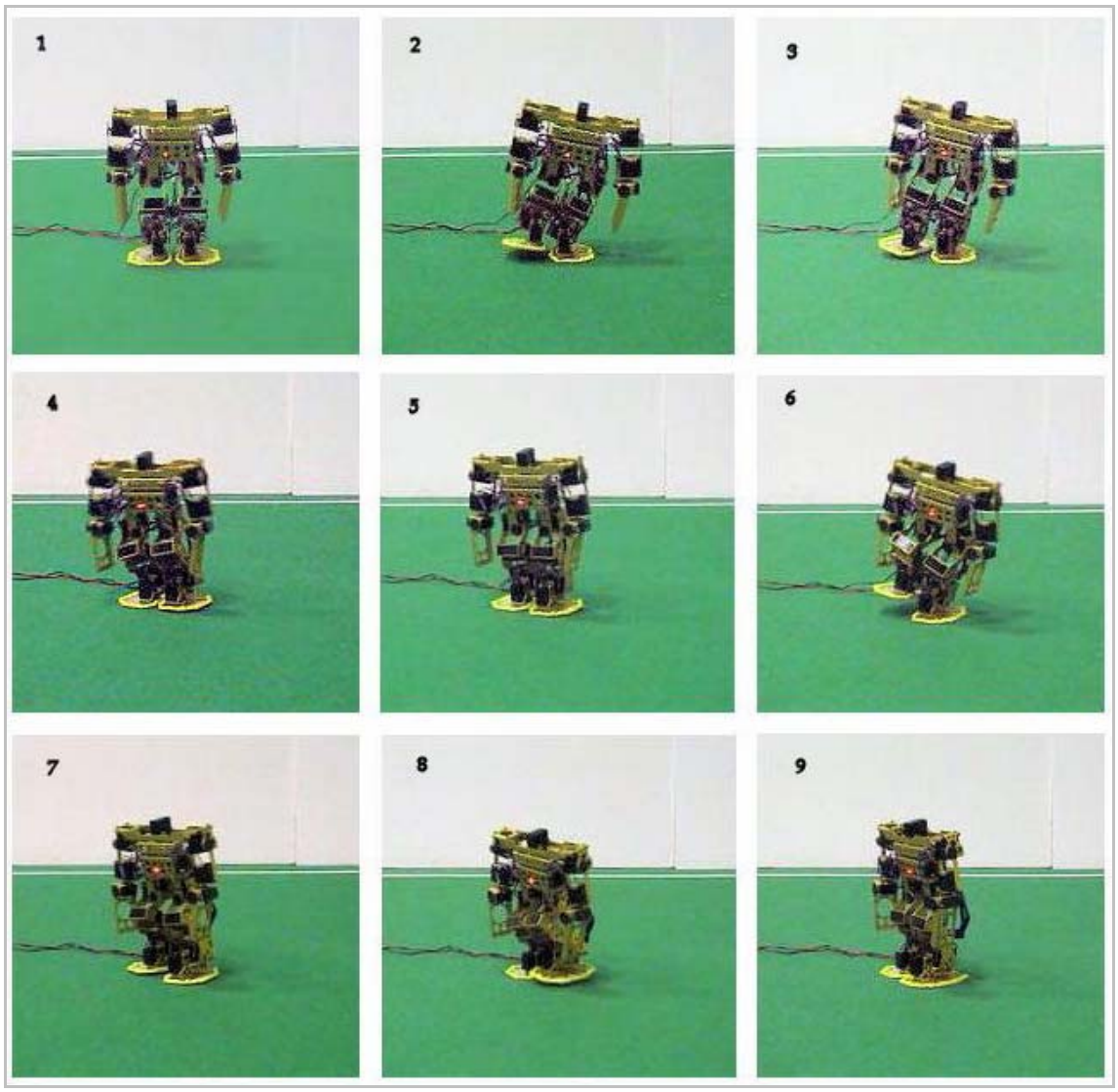


Figura 7.16: La successione di Stage caratterizzante l'esecuzione della rotazione a destra.

Nelle figure 7.14, 7.15 e 7.16 analogamente a quanto fatto per gli alti comportamenti sono raffigurate una successione di stage che mostrano come il comportamento è eseguito dal robot nel primo caso dal *Visualizzatore 3D* nel secondo.

7.7 La rotazione a sinistra

Nella figura 7.17 è mostrato il primo passo da compiere per l'esecuzione del comportamento predefinito

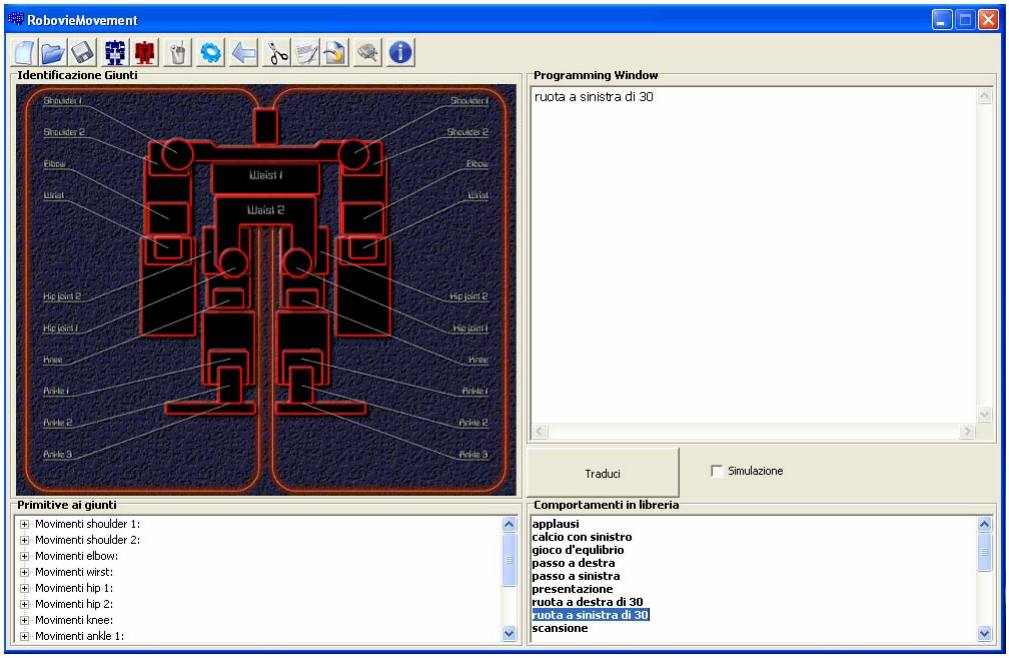


Figura 7.17 : Il comportamento di libreria descritto: rotazione a sinistra .

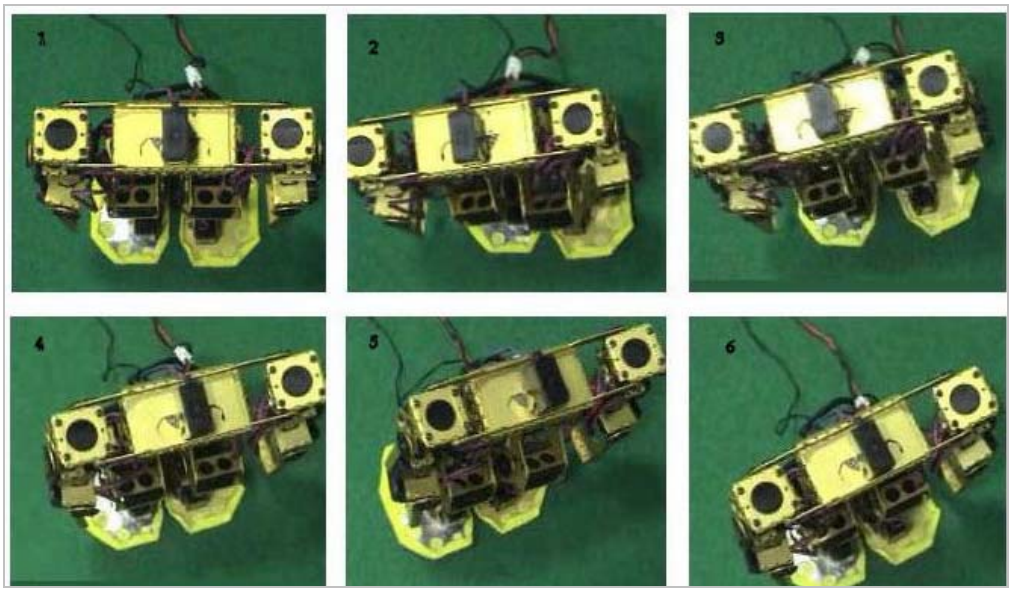


Figura 7.18 : La successione di Stage caratterizzante l'esecuzione della rotazione a destra vista dall'alto.

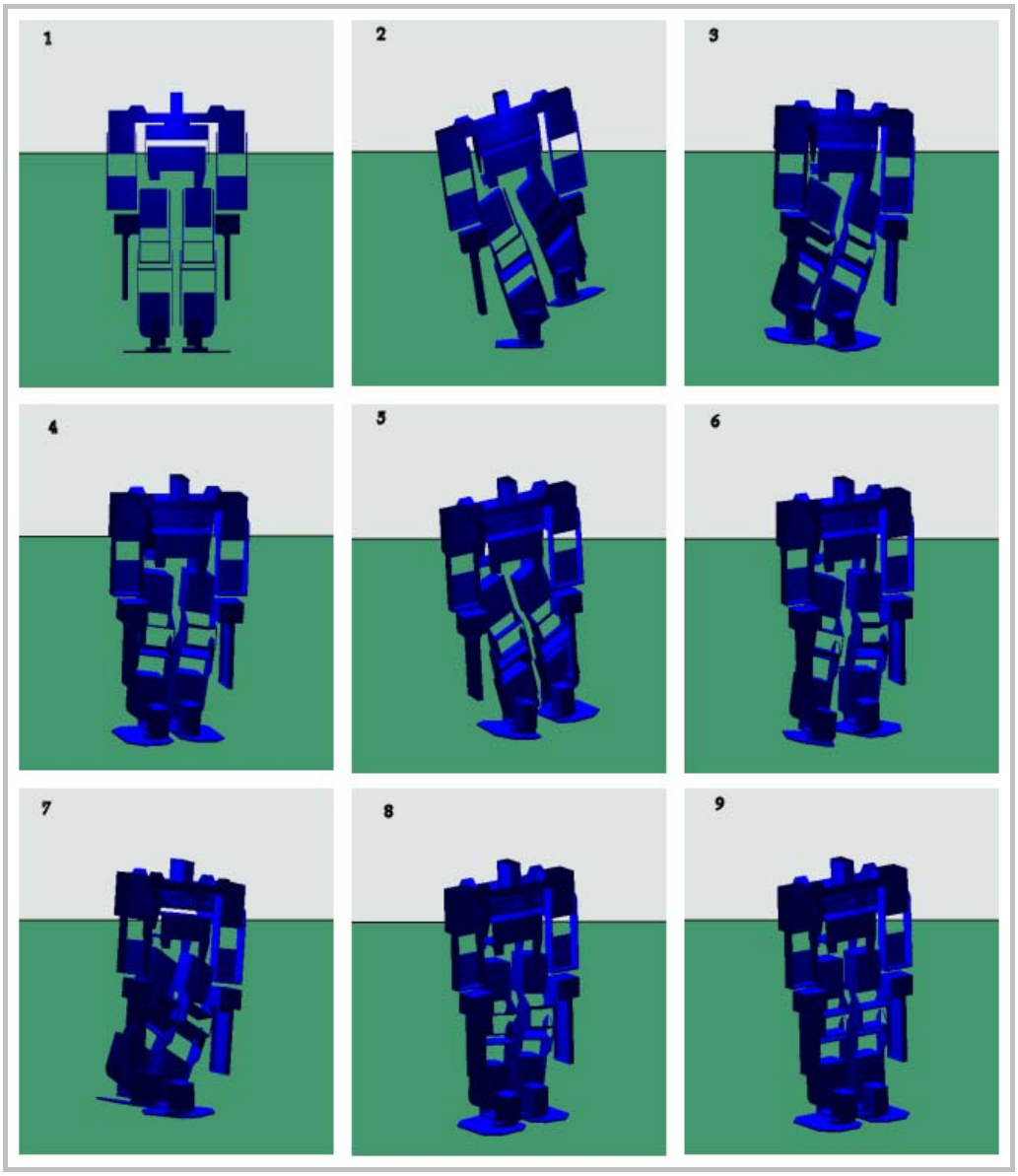


Figura 7.19: La successione di stage caratterizzante la rotazione a sinistra in Robostage.

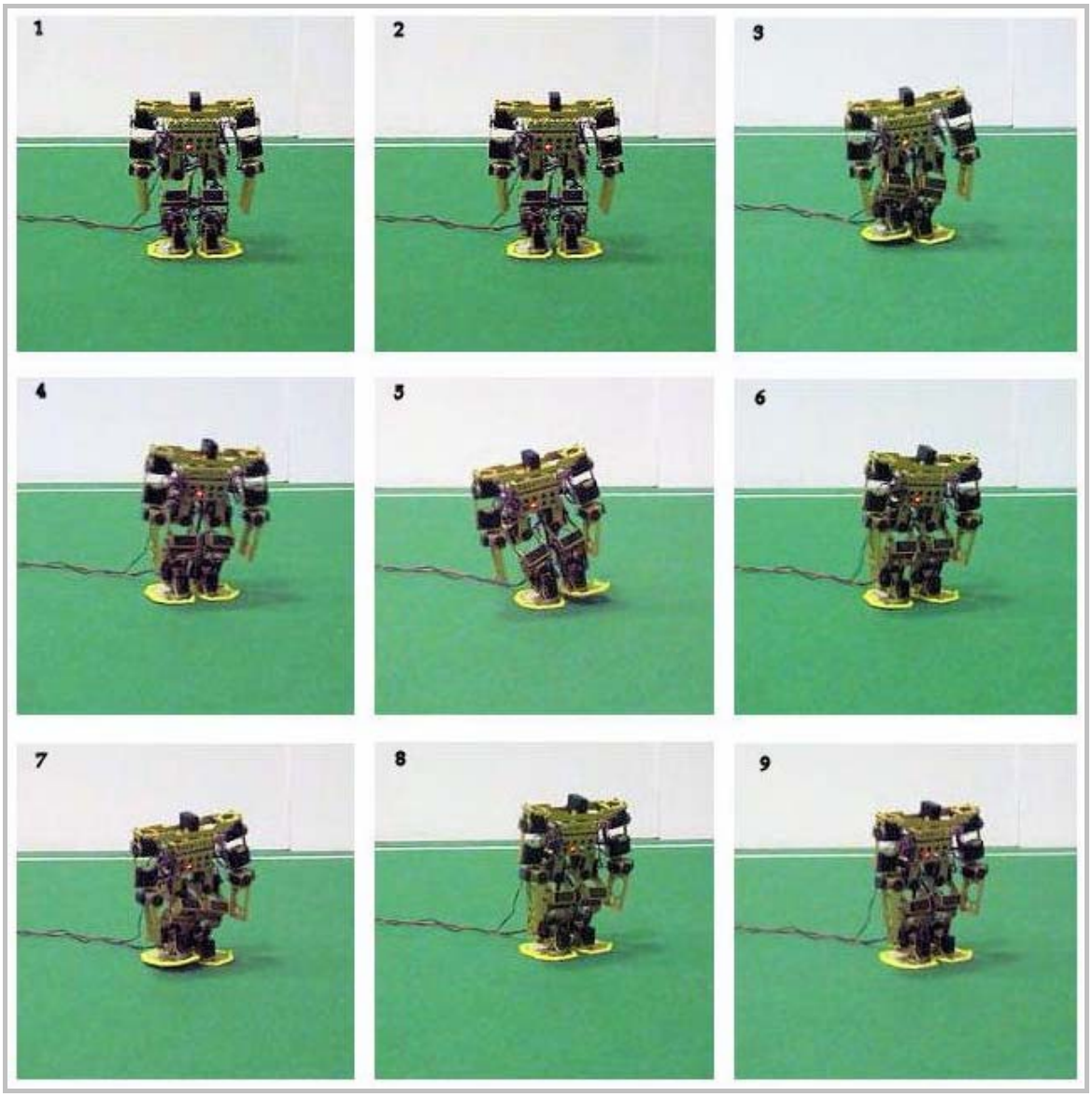


Figura 7.20: La successione di stage caratterizzante l'esecuzione della rotazione a sinistra.

Nella figura 7.18 è mostrata vista dall'alto la rotazione verso sinistra di 30 gradi mentre nelle figure 7.19 e 7.20 sono raffigurate la sequenza di stage principale che realizzano questo comportamento. Nella figura 7.18 si ha la sequenza realmente eseguita del robot, mentre nella figura 7.17 è visibile la medesima sequenza ma realizzata dal visualizzatore 3d

7.8 La scansione

Accanto a tutti quei comportamenti che permettono al robot di muoversi, è stato realizzato un comportamento complesso indispensabile al robot per potere iniziare le varie operazioni di localizzazione e riconoscimento: la scansione

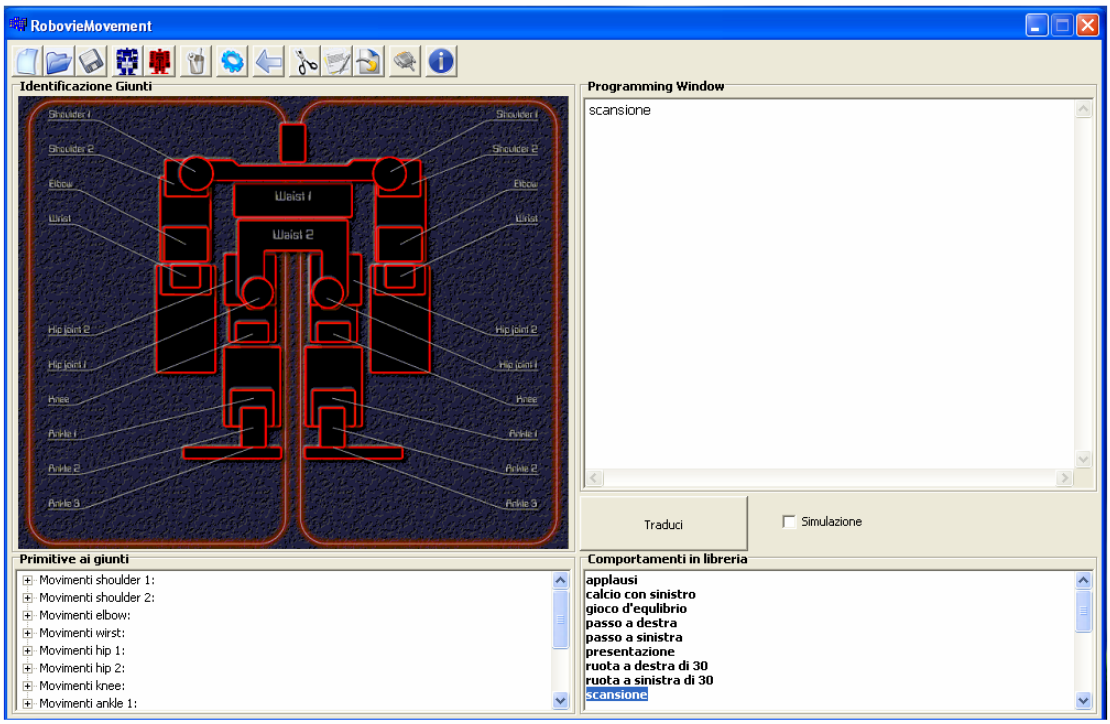


Figura 7.21: Il comportamento di libreria descritto: scansione.

Il comportamento è presente nella libreria realizzata come mostrato nella figura 7.4 mentre nelle 7.8 e 7.9 sono mostrate le successioni di stage eseguite dal visualizzatore 3D e dal robot.

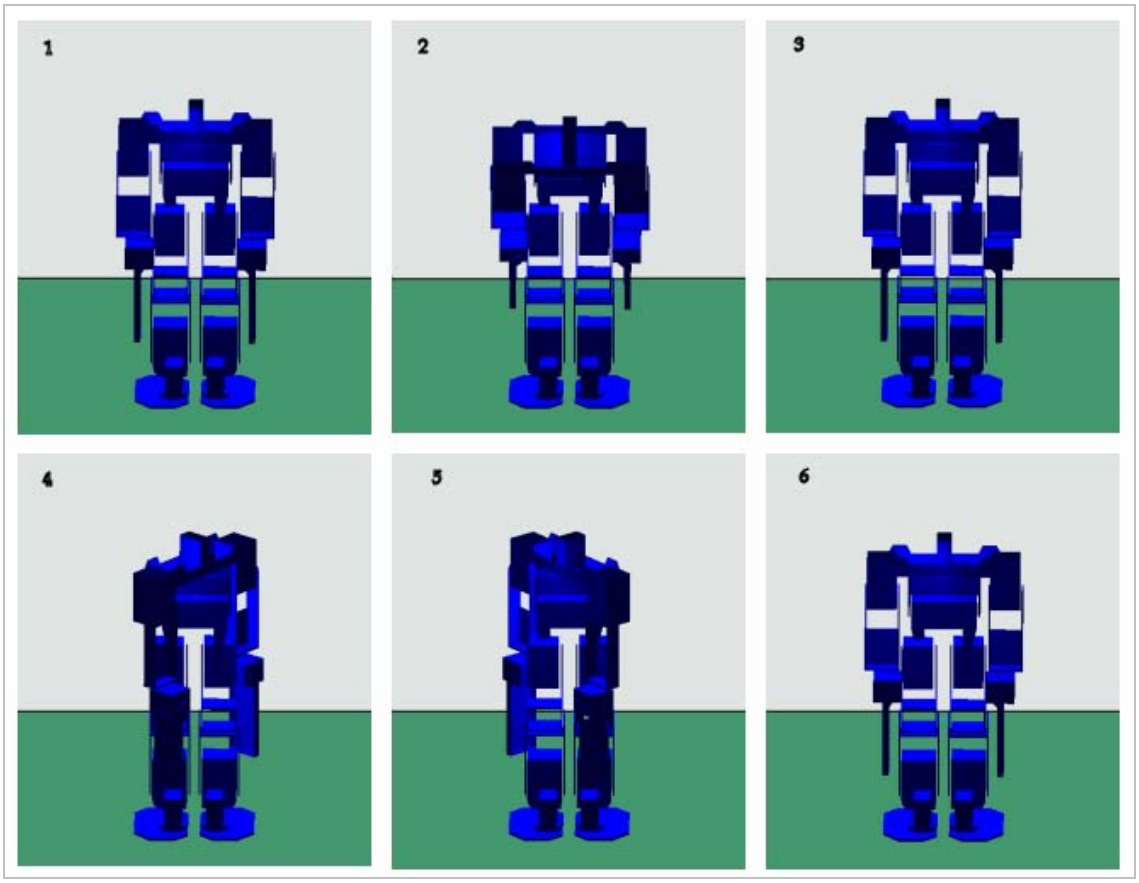


Figura 7.22: La successione di stage caratterizzante la scansione dell'ambiente Robostage.

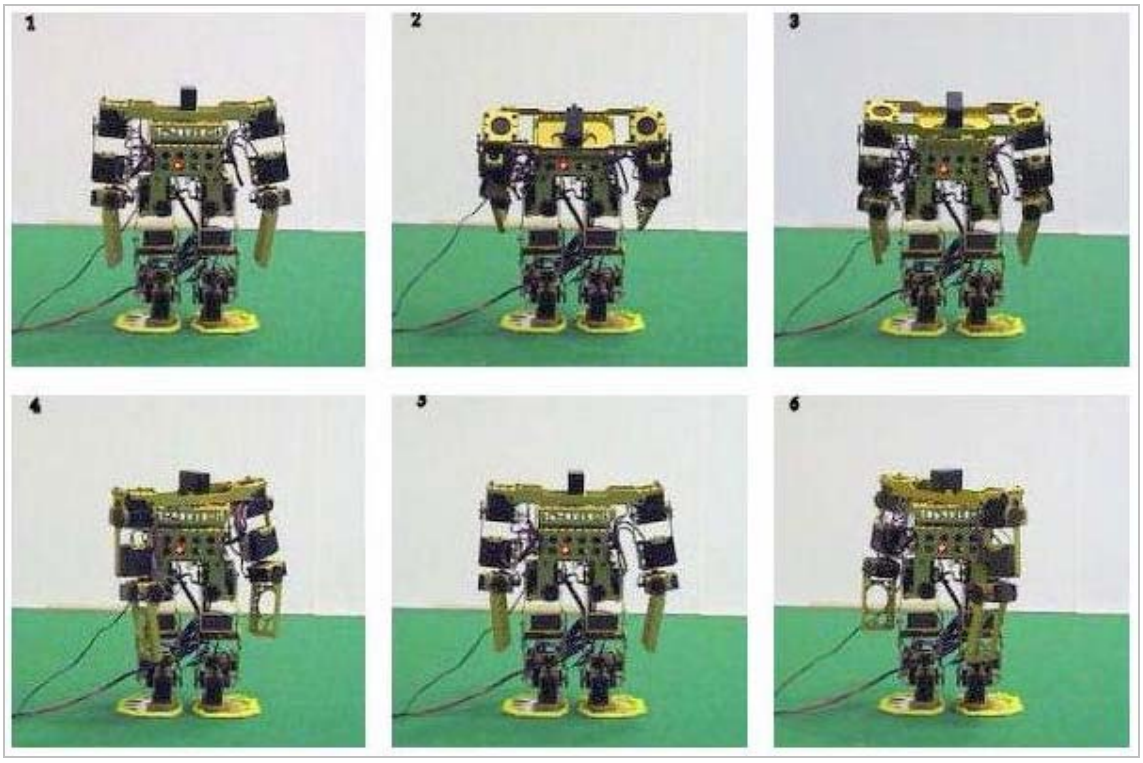


Figura 7.32 : La successione di Stage caratterizzante l'esecuzione della scansione.

7.9 Le conclusioni.

In questa tesi è stato sviluppato un linguaggio che consente la gestione del sistema di controllo del movimento del robot umanoide in modo da poter definire movimenti elementari e a partire da questi determinare movimenti complessi, come la camminata o la rialzata da terra.

Questa tesi s'inquadra all'interno di un progetto sperimentale, che prevede la realizzazione e l'implementazione di un software proprietario per la visione e la movimentazione del robot umanoide "Robovie-M". Il progetto punta alla realizzazione di un ambiente di sviluppo che consente di elaborare i dati provenienti dalla telecamera del robot, e conseguentemente agire da remoto sui motori montati ai giunti in modo tale da far eseguire al robot certi comportamenti, in relazione a quello che la telecamera percepisce.

La piattaforma hardware utilizzata è caratterizzata da risorse hardware limitate, specialmente per quanto riguarda la potenza di calcolo e la memoria a disposizione.

Come ambiente di test si è scelto *RoboCup*, in particolare l'*Humanoid Kid Size League*.

A tale scopo sono stati implementati movimenti utili in una partita di calcio, come la camminata o il calcio stesso: il robot è capace di individuare la palla, raggiungerla e calciarla.

La movimentazione statica del robot e il set di movimenti impostati sono sufficientemente flessibili per i compiti proposti anche se i vari comportamenti realizzati non raggiungono velocità elevate; questo ultimo fattore risulta strettamente connesso al fatto che si sta lavorando su uno *static walzer*.

7.10 Gli sviluppi futuri

Interessante sarebbe rendere il linguaggio di programmazione più sintetico e meno prolisso in modo da avere una rappresentazione a stage più sintetica; importante sarebbe l'arricchimento della libreria dei movimenti di Robovie-M con l'eventuale realizzazione di movimenti complessi che includono anche comportamenti di tipo emozionale.

Un aspetto da sviluppare è rappresentato dalla creazione di movimenti più fluidi e veloci rispetto alla camminata statica, al fine di ottenere prestazioni superiori.

Per far questo una strada percorribile è dotare il robot dell'apparato sensoriale necessario per misurare lo Zero Moment Point, e quindi poter effettuare c dinamiche.

Interessante sarebbe sviluppare un modello del robot che tenesse conto della cinematica inversa rendendo un po' meno ostico il problema relativo al controllo del moto e dell'equilibrio dell'umanoide.

BIBLIOGRAFIA

[1] Guseo T(2005) Architettura Software per un robot umanoide autonomo, Tesi di laurea, Università degli studi di Padova, Padova.

[2] Vetturelli V(2005), Sviluppo di software di visione e movimentazione per robot umanoide, Tesi di laurea, Università degli studi di Padova, Padova.

[3] Marcassa A,(2005) Realizzazione di un interfaccia grafica per la gestione del movimento del robot umanoide, Tesi di laurea, Università degli studi di Padova, Padova.

[4] Zonfrilli F(2001) ,Progetto ed implementazione di un'andatura bipede per il robot quadrupede Sony ERS,210,Tesi di laurea, Università degli studi di Roma "La Sapienza", Roma.

[5] Sorbello F., Dispense del corso di Linguaggi e Traduttori, Università degli studi di Palermo, Palermo.

[6] Tad McGeer, Passive dynamic walking, The International Journal of Robotic esearch 9 (1990), no. 2, 62+.

[7] Renesas Technology Corp., (2003). Hitachi SuperHTMRISC engine, SH-2, SH7052 F-ZTATTM, SH7053 F-ZTATTM, SH7054 F-ZTATTM, Hardware Manual, Electronic Devices Sales & Marketing Group Semiconductor & Integrated Circuits Hitachi, Ltd., Chiyoda-ku, Tokyo, Japan, URL: http://www.renesas.com/.

[8]Sanwa (2005). Speci¯che dei servocomandi Hyper ERG-VB e SPEC-APZ, URL: http://www.sanwa-denshi.co.jp/, Japan.

[9]RoboCup, compiled by Sven Behnke (2005). Humanoid Kid Size League, Humanoid Medium Size League. The Rules & Setup for Osaka 2005, URL: http://www.robocup.org/.

[10]RoboCup, compiled by Sven Behnke (2006). RoboCupSoccer Humanoid League Rules and Setup for the 2006 competition in Bremen, Germany, URL: http://www.robocup.org/.

[11] Collins, S., Ruina, A., Tedrake, R. e Wisse, M. (2005). E±cient Bipedal Robots
Based on Passive-Dynamic Walkers, *SCIENCE*, vol. 307, pp. 1082-1085, February, URL: http://www.sciencemag.org.
[12]Garcia, M., Ruina, A. e Coleman, M. (1997). Some Results In Passive-Dynamic Walking. A research proposal submitted to the National Science Foundation.
[13] Kagamia,S., Mochimarua, M., Eharac, Y., Miyataa, N., Nishiwakid, K., Kanadea, T. e Inoued, H. (2004) Measurement and comparison of humanoid H7 walking with human being, Robotics and Autonomous Systems, Elsevier, n. 48, pp. 177-187.
[14] Sugihara, T., Nakamura, Y. (2003). \Whole-body Cooperative COG Contr through ZMP Manipulation for Humanoid Robots".
[15]]Popovic, M.B., Herr, H. (2005). \Global Motion Control and Support Base Planning", Mit Press, Cambridge, MA, U.S.A.+.
[16] D. Juricic M. Vukobratovic, Contribution to the synthesis of biped gait, IEEE Transactions on Bio-Medical Engineering (1969), no. 1, 1{6.
[17] Tad McGeer, Passive walking with knees, 1990 IEEE Robotics & Automation Conference (1990) 1640+.
[18] Kazuo Tani Shuuji Kajita, Study of dynamic biped locomotion on rugged terrain, Proceedings of the 1991 IEEE International Conference on Robotics & Automation (1991), 1405+.
[19] Experimental study of biped dynamic walking in the linear inverted pendulum mode, IEEE International Conference on Robotics & Automation (1995),2885+.
[20] http://www.valentiniweb.com/Piermo
[21]http://en.wikipedia.org/wiki/Humanoid_robot
[22] http://www.sanwa.co.jp/
[23] http://www.vstone.co.jp/

[24]Gini G(2005) , Dispense del corso di Robotica2, Robotica umanoide(approfondimento sui bipedi), Università degli studi di Bologna Bologna

Appendice A

A.1 Il sorgente Lex

```
%{
/*************************************************************************
mylexer.l
                                    LEX file.

Progetto: Linguaggio x la gestione dei movimenti di "Robovie-M"
Date: 01/08/2006
*************************************************************************/

#include "myparser_tab.h"
#include <string.h>


char* crea_stringa(char* s, int n);

%}

A [aAà]
B [bB]
C [cC]
D [dD]
E [eEèé]
F [fF]
G [gG]
H [hH]
I [iI]
J [jJ]
K [kK]
L [lL]
M [mM]
N [nN]
O [oO]
P [pP]
Q [qQ]
R [rR]
S [sS]
T [tT]
U [uUù]
V [vV]
W [wW]
X [xX]
Y [yY]
Z [zZ]

%%

[ \t\n]*{R}{I}{P}{E}{T}{I}[ \t\n]*
```

```
                                |
[ \t\n]*{R}{I}{P}{E}{T}{I}[ \t\n]*{A}{Z}{I}{O}{N}({E}|{I})[ \t\n]*
                                |
[ \t\n]*{R}{E}{P}{E}{A}{T}[ \t\n]*
                                                { return(RIPETI); }

[ \n\t]*{S}{T}{O}{P}[ \n\t]*           |
[ \n\t]*{E}{N}{D}[ \n\t]*        |
[ \n\t]*{F}{I}{N}{E}[ \n\t]*           { return(STOP);}

[ \n\t]*{B}{E}{G}{I}{N}[ \n\t]*  |
[ \n\t]*{I}{N}{I}{Z}{I}{O}[ \n\t]*     |
[ \n\t]*{S}{T}{A}{R}{T}[ \n\t]*       {return(INIZIO);}

{A}{L}{Z}{A}(({R}{E})?)                   { return(ALZA);}
{A}{B}{B}{A}{S}{S}{A}(({R}{E})?)          {return(ABBASSA);}
{M}{U}{O}{V}{I}                           |
{R}{U}{O}{T}{A}(({R}{E})?)                {return(RUOTA);}
{S}{P}{A}{L}{L}{A}                          |
{S}{H}{O}{U}{L}{D}{E}{R}[ \t]*({I}|"1")[ \t]* { return(SHOULDER_I); }
{B}{R}{A}{C}{C}{I}{O}                       |
{S}{H}{O}{U}{L}{D}{E}{R}[ \t]*{I}{I}[ \t]*    |
{S}{H}{O}{U}{L}{D}{E}{R}[ \t]*"2"[ \t]*       { return(SHOULDER_II); }
{P}{O}{L}{S}{O}                             |
{E}{L}{B}{O}{W}                             {return(ELBOW);}
{M}{A}{N}{O}                                |
{W}{I}{R}{S}{T}                             {return(WIRST);}

{V}{E}{R}{S}{O}[ \t]*{L}[ \t]*(("'"[ \t]*)?){A}{L}{T}{O}   |
{I}{N}[ \t]*{A}{L}{T}{O}                    {return(VERSO_ALTO);}

{V}{E}{R}{S}{O}[ \t]*{I}{L}[ \t]*{B}{A}{S}{S}{O}|
{I}{N}[ \t]*{B}{A}{S}{S}{O}                   {return(VERSO_BASSO);}

{V}{E}{R}{S}{O}[ \t]*{L}[ \t]*(("'"[ \t]*)?){I}{N}{T}{E}{R}{N}{O}       |
{A}{L}{L}[ \t]*(("'"[ \t]*)?){I}{N}{T}{E}{R}{N}{O}                      |
(({I}{N})|({V}{E}{R}{S}{O}))[ \t]*{D}{E}{N}{T}{R}{O} {return(VERSO_DENTRO);}

{V}{E}{R}{S}{O}[ \t]*{L}[ \t]*(("'"[ \t]*)?){E}{S}{T}{E}{R}{N}{O}       |
{A}{L}{L}[ \t]*(("'"[ \t]*)?){E}{S}{T}{E}{R}{N}{O}                      |
(({I}{N})|({V}{E}{R}{S}{O}))[ \t]*{F}{U}{O}{R}{I}    {return(VERSO_FUORI);}

{D}{E}{S}{T}{R}({A}|{O})  |
{D}{X}                    {yylval.str=crea_stringa("Destra",6);
                           return(DX_O_SX);}

(({A})|({V}{E}{R}{S}{O}))[ \t]*{D}{E}{S}{T}{R}{A}     {return(VERSO_DESTRA); }

(({A})|({V}{E}{R}{S}{O}))[ \t]*{S}{I}{N}{I}{S}{T}{R}{A}
                                                 { return(VERSO_SINISTRA);}

{S}{I}{N}{I}{S}{T}{R}({O}|{A})   |
{S}{X}                           {yylval.str=crea_stringa("Sinistra",8);
                                  return(DX_O_SX);   }

{I}{N}[ \t*]{S}{E}{N}{S}{O}[ \t]*{O}{R}{A}{R}{I}{O}                      |
[ \t]*(({A})|({V}{E}{R}{S}{O}))[ \t]*(({S}{I}{N}{I}{S}{T}{R}{A})|({S}{X}))
[ \t]*                                    {return (IN_SENSO_ORARIO);}

{I}{N}[ \t*]{S}{E}{N}{S}{O}[ \t]*{A}{N}{T}{I}{O}{R}{A}{R}{I}{O}         |
[ \t]*(({A})|({V}{E}{R}{S}{O}))[ \t]*(({D}{E}{S}{T}{R}{A})|({D}{X}))[ \t]*
{return (IN_SENSO_ANTIORARIO);}

{D}{I}                                  {return(DI); }
{G}{R}{A}{D}({O}|{I})                    {return(GRADI);}
```

```
{S}{P}{O}{S}{T}{A}(({R}{E})?)           {return (SPOSTARE);}
{T}{O}{R}{A}{C}{E}                      |
{W}{A}{I}{S}{T}[ \t]*{I}{I}[ \t]*       |
{W}{A}{I}{S}{T}[ \t]*"2"[ \t]*          {return (WAIST_II);}
{V}{I}{T}{A}                            |
{W}{A}{I}{S}{T}[ \t]*{I}[ \t]*          |
{W}{A}{I}{S}{T}[ \t]*"1"[ \t]*              {return (WAIST_I);}
{A}{N}{C}{A}                                |
{H}{I}{P}[ \t]*{I}[ \t]*                    |
{H}{I}{P}[ \t]*"1"[ \t]*                    {return (HIP_I);}
{C}{O}{S}{C}{I}{A}                          |
{H}{I}{P}[ \t]*{I}{I}[ \t]*                 |
{H}{I}{P}[ \t]*"2"[ \t]*                    {return (HIP_II);}
{K}{N}{E}{E}                                |
{G}{I}{N}{O}{C}{C}{H}{I}{O}                 {return(KNEE);}
{G}{A}{M}{B}{A}                             |
{A}{N}({C}|{K}){L}{E}[ \t]*{I}[ \t]*        |
{A}{N}({C}|{K}){L}{E}[ \t]*"1"[ \t]*        {return (ANKLE_I);}
{C}{A}{V}{I}{G}{L}{I}{A}                     |
{A}{N}({C}|{K}){L}{E}[ \t]*{I}{I}[ \t]*      |
{A}{N}({C}|{K}){L}{E}[ \t]*"2"[ \t]*        {return (ANKLE_II);}
{P}{I}{E}{D}{E}                              |
{A}{N}({C}|{K}){L}{E}[ \t]*{I}{I}{I}[ \t]*   |
{A}{N}({C}|{K}){L}{E}[ \t]*"3"[ \t]*        {return (ANKLE_III);}
{I}{N}[\t* ]*{A}{V}{A}{N}{T}{I}             {return (AVANTI);}
{I}{N}{D}{I}{E}{T}{R}{O}                    {return (INDIETRO);}
{V}{O}{L}{T}{E}[ \t]*                        {return(VOLTE);}

","                         |
{V}{I}{R}{G}{O}{L}{A}       |
"\n"                        |
{N}{U}{O}{V}{A}[ \t]*(({A}{Z}{I}{O}{N}{E})?)  {return(NUOVA_AZIONE);}
{E} {return(E);}
{C}{O}{N}[ \t]*{V}{E}{L}{O}{C}{I}{T}{A}[ \t]*(({D}{I})?)   |
{A}{L}{L}{A}[ \t]*{V}{E}{L}{O}{C}{I}{T}{A}[ \t]*{D}{I} {return(SET_VELOCITA);}

[0-9]+                      {yyval.str=crea_stringa(yytext,yyleng);
                                    return(INTEGER);      }
{U}{N}[ \t]* |
{U}{N}{O}[ \t]*                         { yyval.num=1;return (NUMEROU); }

{D}{U}{E}[ \t]*                         {yyval.num=2;return (NUMEROU);}
{T}{R}{E}[ \t]*                         { yyval.num=3;return (NUMEROU); }
{Q}{U}{A}{T}{T}{R}{O}[ \t]*             |
{Q}{U}{A}{T}{T}{R}[ \t]*                { yyval.num=4;return (NUMEROU);}
{C}{I}{N}{Q}{U}{E}[ \t]*                |
{C}{I}{N}{Q}{U}[ \t]*                    { yyval.num=5;return (NUMEROU);}
{S}{E}{I}[ \t]*                         { yyval.num=6;return (NUMEROU);}
{S}{E}{T}{T}{E}[ \t]*                   |
{S}{E}{T}{T}[ \t]*                      { yyval.num=7;return (NUMEROU); }
{O}{T}{T}{O}[ \t]*                      |
{O}{T}{T}[ \t]*                         { yyval.num=8;return (NUMEROU); }


{N}{O}{V}{E}[ \t]*                       |
{N}{O}{V}[ \t]*                         {yyval.num=9;return (NUMEROU); }
{D}{I}{E}{C}{I}[ \t]*                   {yyval.num=10;return (NUMEROD);}
{U}{N}{D}{I}{C}[ \t]*                   |
{U}{N}{D}{I}{C}{I}[ \t]*                { yyval.num=11;return (NUMEROD); }
{D}{O}{D}{I}{C}[ \t]*                   |
{D}{O}{D}{I}{C}{I}[ \t]*                {yyval.num=12;return (NUMEROD); }
{T}{R}{E}{D}{I}{C}[ \t]*                |
{T}{R}{E}{D}{I}{C}{I}[ \t]*             { yyval.num=13;return (NUMEROD); }
{Q}{U}{A}{T}{T}{O}{R}{D}{I}{C}[ \t]*    |
{Q}{U}{A}{T}{T}{O}{R}{D}{I}{C}{I}[ \t]*    {yyval.num=14;return(NUMEROD);}
```

```
{Q}{U}{I}{N}{D}{I}{C}[ \t]*              |
{Q}{U}{I}{N}{D}{I}{C}{I}[ \t]*           {yylval.num=15;return (NUMEROD);}
{S}{E}{D}{I}{C}[ \t]*                    |
{S}{E}{D}{I}{C}{I}[ \t]*                 {yylval.num=16;return (NUMEROD);}
{D}{I}{C}{I}{A}{S}{S}{E}{T}{T}{E}[ \t]*  {yylval.num=17;return (NUMEROD);}
{D}{I}{C}{I}{O}{T}{T}[ \t]*              |
{D}{I}{C}{I}{O}{T}{T}{O}[ \t]*           { yylval.num=18; return (NUMEROD); }
{D}{I}{C}{I}{A}{N}{N}{O}{V}{E}[ \t]*     {yylval.num=19;return (NUMEROD); }
{V}{E}{N}{T}[ \t]*                       |
{V}{E}{N}{T}{I}[ \t]*                    {yylval.num=20;return (NUMEROD); }
{T}{R}{E}{N}{T}[ \t]*                    |
{T}{R}{E}{N}{T}{A}[ \t]*                 {yylval.num=30;return (NUMEROD); }
{Q}{U}{A}{R}{A}{N}{T}[ \t]*              |
{Q}{U}{A}{R}{A}{N}{T}{A}[ \t]*           {yylval.num=40;return (NUMEROD); }
{C}{I}{N}{Q}{U}{A}{N}{T}[ \t]*           |
{C}{I}{N}{Q}{U}{A}{N}{T}{A}[ \t]*        {yylval.num=50;return (NUMEROD); }
{S}{E}{S}{S}{A}{N}{T}[ \t]*              |
{S}{E}{S}{S}{A}{N}{T}{A}[ \t]*           {yylval.num=60;return (NUMEROD); }
{S}{E}{T}{T}{A}{N}{T}[ \t]*              |
{S}{E}{T}{T}{A}{N}{T}{A}[ \t]*           {yylval.num=70;return (NUMEROD); }
{O}{T}{T}{A}{N}{T}[ \t]*                 |
{O}{T}{T}{A}{N}{T}{A}[ \t]*              {yylval.num=80;return (NUMEROD); }
{N}{O}{V}{A}{N}{T}[ \t]*                 |
{N}{O}{V}{A}{N}{T}{A}[ \t]*              {yylval.num=90;return (NUMEROD); }
{C}{E}{N}{T}[ \t]*                       |
{C}{E}{N}{T}{O}[ \t]*                    {yylval.num=100;return (NUMEROC); }
{M}{I}{L}{L}{E}[ \t]*                    |
{M}{I}{L}{A}[ \t]*                       {yylval.num=1000;return (NUMEROML);}
{M}{I}{L}{I}{O}{N}{I}[ \t]*              |
{M}{I}{L}{I}{O}{N}{E}[ \t]*              {yylval.num=1000000;return
(NUMEROMN);}
{M}{I}{L}{I}{A}{R}{D}[ \t]*              |
{M}{I}{L}{I}{A}{R}{D}{I}[ \t]*           |
{M}{I}{L}{I}{A}{R}{D}{O}[ \t]*      {yylval.num=1000000000;return  NUMEROMD);}


%%

char* crea_stringa(char* s, int n)
         {
            char* t =malloc(n+1);
            strncpy(t, s, n);
            t[n]='\0';
            return t;

}
```

A.2 Il sorgente Yacc

```
%{
/*************************************************************************
myparser.y
                                  YACC file.

Progetto: Linguaggio x la gestione dei movimenti di "Robovie-M"
Date: 01/08/2006
*************************************************************************/

#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <math.h>

#define TOP_DANCER 100

int VELOCITA=1;
int SHOULDER_I_SX=52;
int SHOULDER_I_DX=22;
int SHOULDER_II_SX=55;
int SHOULDER_II_DX=25;
int ELBOW_SX=58;
int ELBOW_DX=28;
int WIRST_SX=61;
int WIRST_DX=31;

int WAIST_I_C=67;
int WAIST_II_C=64;
int HIP_I_DX=4;
int HIP_I_SX=34;
int HIP_II_DX=7;
int HIP_II_SX=37;
int KNEE_DX=10;
int KNEE_SX=40;
int ANKLE_I_DX=13;
int ANKLE_I_SX=43;
int ANKLE_II_DX=16;
int ANKLE_II_SX=46;
int ANKLE_III_DX=19;
int ANKLE_III_SX=49;
int LOOP=70;

char FILE_OUTPUT[]="tradotta.txt";
// Contiene il nome del File di output
char FILE_INPUT[]="start.txt";
// File di Input
char FILE_FINETRADUZIONE[]="semaforo.txt";
// File che segnala la fine della traduzione
char FILE_TIPOTRADUZIONE[]="translation";
// File contenente le informazioni necessarie alla determinazione del tipo di
traduzione
 //da effettuare

char
posizione_topdancer[]="@21A7fB8eC1bD80E7aF80G1eHa4I10Je0K80L71Me4N7fO85P7fQe1R5bSe
fT1fUb2V80,+1!00";

char
posizione_iniziale_topdancer[]="@21A7fB8eC1bD80E7aF80G1eHa4I10Je0K80L71Me4N7fO85P
7fQe1R5bSefT1fUb2V80,+1!00";

char
posizione_finale_topdancer[]="@21A7fB8eC1bD80E7aF80G1eHa4I10Je0K80L71Me4N7fO85P7fQ
e1R5bSefT1fUb2V80,+0!00";

char
posizione_iniziale[]="@21A7fB8eC1bD80E7aF80G1eHa4I10Je0K80L71Me4N7fO85P7fQe1R5bSe
fT1fUb2V80,+1!00";

char posizione_finale[]=  "@21A7fB8eC1bD80E7aF80G1eHa4I10Je0K80L71Me4N7fO85P7fQe1R5
bSefT1fUb2V80,+0!00";

char
posizione_attuale[]="@21A7fB8eC1bD80E7aF80G1eHa4I10Je0K80L71Me4N7fO85P7fQe1R5bSefT1
```

```
fUb2V80,+1!00";
int  posizione_motori[75]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0};

char Nexa[2]={'0','0'};

int numero_ripetizioni=0;
int PrimaAzioneDaRipetere=0;
int NumeroAzioni=0;
int flag_ripeti_n_volte=0;
int flag_ripeti=0;
//flag che quando posto a 1 indica che sono all'interno di un loop
int flag_dopo_loop=0;
int NumeroAzioniDaRipetere=0;
//Per la determinazione del numero si passi di cui si deve
//tornare indietro al fine di creare dei loop
int TipoTraduzione=0;
// Per la determinazione del tipo di traduzione ke devo effettuare
//(Per il Software Robostage e RoboveiMaker)

//Struttura utilizzata per la creazione di una lista. Ogni elemento di tale lista
//contiene una riga del FILE_OUTPUT
struct riga {
       char informazione[105];
// Di conseguenza la lista contiene le righe del file: FILE_OUTPUT
       struct riga *next;
};
struct riga *testa=NULL;
//Puntatore alla testa della lista


//************************************************
//Prototipo delle funzioni utilizzate
//************************************************
void Dec2Esa(int,char*);
//Conversione da Decimale a Esadecimale relativamente ai motori della parte destra
void Dec2Esa_SX(int,char*);
//Conversione da Decimale a Esadecimale relativamente ai motori della parte
sinistra
int Esa2Dec(char*);
//Conversione da Esadecimale a Decimale relativamente ai motori della parte destra
char ValoreEsadecimale(int);
char ValoreEsadecimaleParteSinistra(int);
int ValoreDecimale(char);
void MemorizzaMovimento(void);
//Memorizza il movimento da eseguire all'interno di una lista
void CreaRipetizioni(int);
//funzione utilizzata per la creazione dei cicli di azioni
void inizializza(void);
//Inizializza la posizione dei motori
void AggiornaMotori(char*,int);
//aggiorna la posizione dei motori in seguito ad un azione

void CreaNRipetizioni(int,int);
struct riga* CreaListaDaRipetere(int);
struct riga * ConcatenaListe(struct riga *,struct riga *);
void VisualizzaLista(struct riga*);

int Arrotonda(float f);

void FineTraduzione(void);
void yyerror(char*);
//************************************************

%}
```

```
%token ALZA, ABBASSA, RUOTA, IN_SENSO_ORARIO, IN_SENSO_ANTIORARIO, SPOSTARE,
AVANTI, INDIETRO
%token VERSO_ALTO, VERSO_BASSO, VERSO_DENTRO, VERSO_FUORI, VERSO_DESTRA,
VERSO_SINISTRA
%token SHOULDER_I, SHOULDER_II, ELBOW, WIRST, WAIST_I, WAIST_II, HIP_I, HIP_II,
KNEE, ANKLE_I,ANKLE_II,ANKLE_III
%token SET_VELOCITA, NUOVA_AZIONE, E, DI, GRADI
%token RIPETI, VOLTE, STOP, INIZIO
%token <num> NUMEROU,NUMEROD,NUMEROC,NUMEROML,NUMEROMN,NUMEROMD
%token <str> DX_O_SX
%token <str> INTEGER

%type <str> intero
%type <num> numero_letterale,centinaia,migliaia,miliardi,milioni

%union {
       int num;
       char* str;
}

%%

statement: azione{ FILE *fp;
//memorizzo tutti gli elementi della lista nel file FILE_OUTPUT
                    fp=fopen(FILE_OUTPUT,"a");
                           while(testa!=NULL) {
                                        fprintf(fp,"%s\n",testa->informazione);
                                          testa=testa->next;
                                            }
                                  if(TipoTraduzione==TOP_DANCER)
                                     fprintf(fp,"%s\n",posizione_finale_topdancer);
                                  //deve essere  diversificata asseconda dell'uscita
                                  else
                                     fprintf(fp,"%s\n",posizione_finale);
                                            fclose(fp);
                           }
                           ;

azione: azione NUOVA_AZIONE movimenti {NumeroAzioni++;
                                       if(flag_dopo_loop==1) {
             //Nel caso si prosegua con l'esecuzione di azioni dopo un loop
                                                              struct riga* pointer;
                                                    int index,position;
                                                    pointer=testa;

       for(index=0;index<(PrimaAzioneDaRipetere-1);index++) pointer=pointer->next;
                                              Dec2Esa_SX(NumeroAzioni,Nexa);

       position=strlen(pointer>informazione);
       pointer->informazione[position-2]=Nexa[0];
       pointer->informazione[position-1]=Nexa[1]
              posizione_motori[74]=NumeroAzioni+1;
       flag_dopo_loop=0;
                     }
       if(flag_ripeti==1) NumeroAzioniDaRipetere++;
       MemorizzaMovimento();     }


       |azione RIPETI movimenti{flag_ripeti=1; NumeroAzioniDaRipetere++;
                                              NumeroAzioni++;
                                              PrimaAzioneDaRipetere=NumeroAzioni;
                                              MemorizzaMovimento();     }

       |azione RIPETI INIZIO movimenti{flag_ripeti=1; NumeroAzioniDaRipetere++;
```

```
                                    NumeroAzioni++;
                                    PrimaAzioneDaRipetere=NumeroAzioni;
                                    MemorizzaMovimento();}

      |azione RIPETI movimenti E

{flag_ripeti=1;PrimaAzioneDaRipetere=NumeroAzioni+1;}

      |azione RIPETI INIZIO movimenti E
                          {flag_ripeti=1;PrimaAzioneDaRipetere=NumeroAzioni+1; }

      |RIPETI movimenti{flag_ripeti=1; NumeroAzioniDaRipetere++;
                                    NumeroAzioni++;
                                    PrimaAzioneDaRipetere=NumeroAzioni;
                                    MemorizzaMovimento();     }

      |RIPETI INIZIO movimenti{flag_ripeti=1; NumeroAzioniDaRipetere++;
                                          NumeroAzioni++;
                                          PrimaAzioneDaRipetere=NumeroAzioni;
                                          MemorizzaMovimento();     }

      |RIPETI movimenti E{ flag_ripeti=1;PrimaAzioneDaRipetere=NumeroAzioni+1;}

      |RIPETI INIZIO movimenti E{
flag_ripeti=1;PrimaAzioneDaRipetere=NumeroAzioni+1;}

      |azione RIPETI intero VOLTE movimenti {flag_ripeti_n_volte=1;
                                          numero_ripetizioni=atoi($3);
                                          NumeroAzioni++;
                                          PrimaAzioneDaRipetere=NumeroAzioni;
                                          MemorizzaMovimento();     }

      |azione RIPETI intero VOLTE movimenti E {     flag_ripeti_n_volte=1;
                                          numero_ripetizioni=atoi($3);

PrimaAzioneDaRipetere=NumeroAzioni+1;}

      |azione RIPETI intero VOLTE INIZIO movimenti {flag_ripeti_n_volte=1;
                                                numero_ripetizioni=atoi($3);
                                          NumeroAzioni++;
                                          PrimaAzioneDaRipetere=NumeroAzioni;
                                          MemorizzaMovimento();     }

      |azione RIPETI intero VOLTE INIZIO movimenti E {flag_ripeti_n_volte=1;
                                          numero_ripetizioni=atoi($3);

PrimaAzioneDaRipetere=NumeroAzioni+1;}

      |RIPETI intero VOLTE movimenti { flag_ripeti_n_volte=1;
                                    numero_ripetizioni=atoi($2);
                                    NumeroAzioni++;
                                    PrimaAzioneDaRipetere=NumeroAzioni;
                                                MemorizzaMovimento();     }

      |RIPETI intero VOLTE movimenti E {     flag_ripeti_n_volte=1;
                                    numero_ripetizioni=atoi($2);
                                    PrimaAzioneDaRipetere=NumeroAzioni+1; }

      |RIPETI intero VOLTE INIZIO movimenti {flag_ripeti_n_volte=1;
                                      numero_ripetizioni=atoi($2);
                                      NumeroAzioni++;
                                    PrimaAzioneDaRipetere=NumeroAzioni;
                                    MemorizzaMovimento();     }

      |RIPETI intero VOLTE INIZIO movimenti E {flag_ripeti_n_volte=1;
```

```
                                        numero_ripetizioni=atoi($2);
                                        PrimaAzioneDaRipetere=NumeroAzioni+1; }

      |azione STOP
                  {if (flag_ripeti_n_volte==1)
                                 {
CreaNRipetizioni(PrimaAzioneDaRipetere,numero_ripetizioni);
                           flag_ripeti_n_volte=0; }
            else { CreaRipetizioni(NumeroAzioniDaRipetere); flag_dopo_loop=1; }
                                 }

      |azione NUOVA_AZIONE movimenti E {}

      |azione movimenti   {      if(flag_ripeti==1) NumeroAzioniDaRipetere++;
                                        NumeroAzioni++;
                                        MemorizzaMovimento();      }

      |azione movimenti E {}
      |movimenti E {}
      |movimenti   {      if(flag_ripeti==1) NumeroAzioniDaRipetere++;
                                 NumeroAzioni++;
                                 MemorizzaMovimento();      }
      ;

movimenti:
      ALZA SHOULDER_II DX_O_SX DI intero GRADI       {
                                 if (!strcmp($3,"Destra")) {
                                        posizione_motori[SHOULDER_II_DX]+=atoi($5);
                                        if (posizione_motori[SHOULDER_II_DX]>255)
                                           posizione_motori[SHOULDER_II_DX]=255;
               //Controllo ke il valore non sia fuori dal IL range di
                                        AggiornaMotori($3,SHOULDER_II_DX);}
                     //Aggiorno la posizione dei motori
                                 else {
                                        posizione_motori[SHOULDER_II_SX]+=atoi($5);
                                        if (posizione_motori[SHOULDER_II_SX]>255)
                                             posizione_motori[SHOULDER_II_SX]=255;
                                        AggiornaMotori($3,SHOULDER_II_SX); }
                                       }
      |RUOTA SHOULDER_II DX_O_SX DI intero GRADI VERSO_ALTO       {
                                 if (!strcmp($3,"Destra")) {
                                        posizione_motori[SHOULDER_II_DX]+=atoi($5);

                                        if (posizione_motori[SHOULDER_II_DX]>255)
                                              posizione_motori[SHOULDER_II_DX]=255;
                        //Controllo ke il valore non sia fuori dal IL range di
                                        AggiornaMotori($3,SHOULDER_II_DX);}
                           //Aggiorno la posizione dei motori
                                 else {
                                        posizione_motori[SHOULDER_II_SX]+=atoi($5);
                                        if (posizione_motori[SHOULDER_II_SX]>255)
                                             posizione_motori[SHOULDER_II_SX]=255;
                                        AggiornaMotori($3,SHOULDER_II_SX); }
                                 }
      |ABBASSA SHOULDER_II DX_O_SX DI intero GRADI  {
                                 if (!strcmp($3,"Destra")) {
                                 posizione_motori[SHOULDER_II_DX]-=atoi($5);
                                           if(posizione_motori[SHOULDER_II_DX]<0)
                                             posizione_motori[SHOULDER_II_DX]=0;
                                        AggiornaMotori($3,SHOULDER_II_DX);}
                                 else {
                                        posizione_motori[SHOULDER_II_SX]-=atoi($5);
                                        if (posizione_motori[SHOULDER_II_SX]<0)
                                             posizione_motori[SHOULDER_II_SX]=0;
                                        AggiornaMotori($3,SHOULDER_II_SX); }
```

```
                                }
      |RUOTA SHOULDER_II DX_O_SX DI intero GRADI VERSO_BASSO     {
                                if (!strcmp($3,"Destra")) {
                                       posizione_motori[SHOULDER_II_DX]-=atoi($5);
                                       if (posizione_motori[SHOULDER_II_DX]<0)
                                            posizione_motori[SHOULDER_II_DX]=0;
                                       AggiornaMotori($3,SHOULDER_II_DX);}
                                else {
                                       posizione_motori[SHOULDER_II_SX]-=atoi($5);
                                       if (posizione_motori[SHOULDER_II_SX]<0)
                                             posizione_motori[SHOULDER_II_SX]=0;
                                        AggiornaMotori($3,SHOULDER_II_SX); }
                                }
      |RUOTA SHOULDER_I DX_O_SX DI intero GRADI IN_SENSO_ORARIO  {
                                if (!strcmp($3,"Destra")) {
                                       posizione_motori[SHOULDER_I_DX]-=atoi($5);
                                       if (posizione_motori[SHOULDER_I_DX]<0)
                                            posizione_motori[SHOULDER_I_DX]=0;
                                       AggiornaMotori($3,SHOULDER_I_DX); }
                                else {
                                       posizione_motori[SHOULDER_I_SX]+=atoi($5);
                                       if (posizione_motori[SHOULDER_I_SX]>255)
                                            posizione_motori[SHOULDER_I_SX]=255;
                                       AggiornaMotori($3,SHOULDER_I_SX);}
                                }
      |RUOTA SHOULDER_I DX_O_SX DI intero GRADI VERSO_DENTRO           {
                                if (!strcmp($3,"Destra")) {
                                       posizione_motori[SHOULDER_I_DX]+=atoi($5);

                                       if (posizione_motori[SHOULDER_I_DX]>255)
                                             posizione_motori[SHOULDER_I_DX]=255;
//Controllo ke il valore non sia fuori dal IL range di
                                       AggiornaMotori($3,SHOULDER_I_DX);}
                    //Aggiorno la posizione dei motori
                                else {
                                       posizione_motori[SHOULDER_I_SX]+=atoi($5);
                                       if (posizione_motori[SHOULDER_I_SX]>255)
                                            posizione_motori[SHOULDER_I_SX]=255;
                                       AggiornaMotori($3,SHOULDER_I_SX); }
                                }
      |RUOTA SHOULDER_I DX_O_SX DI intero GRADI VERSO_FUORI            {
                                if (!strcmp($3,"Destra")) {
                                       posizione_motori[SHOULDER_I_DX]-=atoi($5);

                                       if (posizione_motori[SHOULDER_I_DX]>255);
                                            posizione_motori[SHOULDER_I_DX]=255
//Controllo ke il valore non sia fuori dal IL range di
                                       AggiornaMotori($3,SHOULDER_I_DX);
//Aggiorno la posizione dei motori
                                else {
                                       posizione_motori[SHOULDER_I_SX]-=atoi($5);
                                       if (posizione_motori[SHOULDER_I_SX]>255)
                                            posizione_motori[SHOULDER_I_SX]=255;
                                       AggiornaMotori($3,SHOULDER_I_SX); }
                                }
      |RUOTA SHOULDER_I DX_O_SX DI intero GRADI IN_SENSO_ANTIORARIO    {
                                if (!strcmp($3,"Destra")) {
                                       posizione_motori[SHOULDER_I_DX]+=atoi($5);
                                       if (posizione_motori[SHOULDER_I_DX]>255)
                                             posizione_motori[SHOULDER_I_DX]=255;
                                       AggiornaMotori($3,SHOULDER_I_DX);}
                                else {
                                       posizione_motori[SHOULDER_I_SX]-=atoi($5);
                                       if (posizione_motori[SHOULDER_I_SX]<0)
                                            posizione_motori[SHOULDER_I_SX]=0;
```

```
                                    AggiornaMotori($3,SHOULDER_I_SX);}
                              }
      |RUOTA ELBOW DX_O_SX DI intero GRADI IN_SENSO_ANTIORARIO   {
                              if (!strcmp($3,"Destra")) {
                                    posizione_motori[ELBOW_DX]-=atoi($5);
                                    if (posizione_motori[ELBOW_DX]<0)
                                         posizione_motori[ELBOW_DX]=0;
                                    AggiornaMotori($3,ELBOW_DX);}
                              else {
                                    posizione_motori[ELBOW_SX]+=atoi($5);
                                    if (posizione_motori[ELBOW_SX]>255)
                                         posizione_motori[ELBOW_SX]=255;
                                    AggiornaMotori($3,ELBOW_SX);}

                              }
      |RUOTA ELBOW DX_O_SX DI intero GRADI IN_SENSO_ORARIO{
                              if (!strcmp($3,"Destra")) {
                                    posizione_motori[ELBOW_DX]+=atoi($5);
                                    if (posizione_motori[ELBOW_DX]>255)
                                          posizione_motori[ELBOW_DX]=255;
                                    AggiornaMotori($3,ELBOW_DX);}
                              else {
                                    posizione_motori[ELBOW_SX]-=atoi($5);
                                    if (posizione_motori[ELBOW_SX]<0)
                                           posizione_motori[ELBOW_SX]=0;
                                    AggiornaMotori($3,ELBOW_SX);}

                              }
      |RUOTA WIRST DX_O_SX DI intero GRADI IN_SENSO_ORARIO {
                              if (!strcmp($3,"Destra")) {
                                    posizione_motori[WIRST_DX]-=atoi($5);
                                    if (posizione_motori[WIRST_DX]<0)
                                          posizione_motori[WIRST_DX]=0;
                                    AggiornaMotori($3,WIRST_DX);}
                              else {
                                    posizione_motori[WIRST_SX]+=atoi($5);
                                    if (posizione_motori[WIRST_SX]>255)
                                         posizione_motori[WIRST_SX]=255;
                                    AggiornaMotori($3,WIRST_SX);}
                              }
      |RUOTA WIRST DX_O_SX DI intero GRADI IN_SENSO_ANTIORARIO   {
                              if (!strcmp($3,"Destra")) {
                                    posizione_motori[WIRST_DX]+=atoi($5);
                                    if (posizione_motori[WIRST_DX]>255)
                                         posizione_motori[WIRST_DX]=255;
                                    AggiornaMotori($3,WIRST_DX);}
                              else {
                                    posizione_motori[WIRST_SX]-=atoi($5);
                                    if (posizione_motori[WIRST_SX]<0)
                                          posizione_motori[WIRST_SX]=0;
                                    AggiornaMotori($3,WIRST_SX);}
                                    }
      |RUOTA WIRST DX_O_SX DI intero GRADI VERSO_DENTRO    {
                              if (!strcmp($3,"Destra")) {
                                    posizione_motori[WIRST_DX]+=atoi($5);
                                    if (posizione_motori[WIRST_DX]>255);
                                         posizione_motori[WIRST_DX]=255
                                    AggiornaMotori($3,WIRST_DX);}
                              else {
                                    posizione_motori[WIRST_SX]+=atoi($5);
                                    if (posizione_motori[WIRST_SX]>255)
                                        posizione_motori[WIRST_SX]=255;
                                     AggiornaMotori($3,WIRST_SX);}
                              }
      |RUOTA WIRST DX_O_SX DI intero GRADI VERSO_FUORI     {
```

```
                                if (!strcmp($3,"Destra")) {
                                       posizione_motori[WIRST_DX]-=atoi($5);
                                       if (posizione_motori[WIRST_DX]<0)
                                            posizione_motori[WIRST_DX]=0;
                                       AggiornaMotori($3,WIRST_DX);}
                                else {
                                       posizione_motori[WIRST_SX]-=atoi($5);
                                       if (posizione_motori[WIRST_SX]<0)
                                            posizione_motori[WIRST_SX]=0;
                                       AggiornaMotori($3,WIRST_SX);}
                                }

      |RUOTA HIP_I DX_O_SX DI intero GRADI IN_SENSO_ORARIO {
                                if (!strcmp($3,"Destra")) {
                                       posizione_motori[HIP_I_DX]-=atoi($5);
                                       if (posizione_motori[HIP_I_DX]<0)
                                            posizione_motori[HIP_I_DX]=0;
                                       AggiornaMotori($3,HIP_I_DX); }
                                else {
                                       posizione_motori[HIP_I_SX]+=atoi($5);
                                       if (posizione_motori[HIP_I_SX]>255)
                                             posizione_motori[HIP_I_SX]=255;
                                       AggiornaMotori($3,HIP_I_SX);}
                                }
      | RUOTA HIP_I DX_O_SX DI intero GRADI IN_SENSO_ANTIORARIO  {
                                if (!strcmp($3,"Destra")) {
                                       posizione_motori[HIP_I_DX]+=atoi($5);
                                       if (posizione_motori[HIP_I_DX]>255)
                                            posizione_motori[HIP_I_DX]=255;
                                       AggiornaMotori($3,HIP_I_DX);}
                                else {
                                       posizione_motori[HIP_I_SX]-=atoi($5);
                                       if (posizione_motori[HIP_I_SX]<0)
                                            posizione_motori[HIP_I_SX]=0;
                                       AggiornaMotori($3,HIP_I_SX);     }
                                }
      |RUOTA HIP_I DX_O_SX DI intero GRADI VERSO_FUORI           {
                                if (!strcmp($3,"Destra")) {
                                       posizione_motori[HIP_I_DX]-=atoi($5);
                                       if (posizione_motori[HIP_I_DX]<0)
                                            posizione_motori[HIP_I_DX]=0;
                                       AggiornaMotori($3,HIP_I_DX); }
                                else {
                                       posizione_motori[HIP_I_SX]-=atoi($5);
                                       if (posizione_motori[HIP_I_SX]<0)
                                            posizione_motori[HIP_I_SX]=0;
                                       AggiornaMotori($3,HIP_I_SX);}
                                }
      |RUOTA HIP_I DX_O_SX DI intero GRADI VERSO_DENTRO    {
                                if (!strcmp($3,"Destra")) {
                                       posizione_motori[HIP_I_DX]+=atoi($5);
                                       if (posizione_motori[HIP_I_DX]>255)
                                             posizione_motori[HIP_I_DX]=255;
                                        AggiornaMotori($3,HIP_I_DX);}
                                else {
                                       posizione_motori[HIP_I_SX]+=atoi($5);
                                       if (posizione_motori[HIP_I_SX]>255)
                                            posizione_motori[HIP_I_SX]=255;
                                       AggiornaMotori($3,HIP_I_SX);     }
                                }
.
.
.
.
.
```

```
.
.
.
.


//-------------------------------------------------------------------------

/*     Settaggio della VELOCITA con cui eseguire i movimenti  */
velocita: SET_VELOCITA intero    {
                  if(TipoTraduzione==TOP_DANCER){
                                int vel1;
                                int vel2;
                                int vel;
                                float kkk;
                                vel1=atoi($2);
                                printf("\nVel1= %d",vel1);
                                vel2= vel1 * 255 ;
                                printf("\nVel2 = %d",vel2);
                                kkk= (float)(vel2)/211;
                                printf("\nkkk vale %f",kkk);
                                getchar();
                                vel= Arrotonda(kkk);
                                printf("\nvel vale %d",vel);
                                getchar();
                                if (vel<0 || vel>100)
                                       vel=27;
                                Dec2Esa_SX(vel,Nexa);
                                posizione_topdancer[VELOCITA]=Nexa[0];
                                posizione_topdancer[VELOCITA+1]=Nexa[1];
                                                                        }
                           else{
                                int vel;
                                vel=atoi($2);
                                if (vel<0 || vel>100)
      //Controllo ke il valore della velocita sia nel IL range di : 0 < velocita <
100
                                       vel=33;
                                Dec2Esa_SX(vel,Nexa);
                                posizione_attuale[VELOCITA]=Nexa[0];
                                posizione_attuale[VELOCITA+1]=Nexa[1];
                           }
                           }
      ;

intero: numero_letterale  {     itoa($1,$$,10);     }
      |      INTEGER              {      $$=$1;       }
      ;

numero_letterale:      miliardi                {  $$=$1;  }
                    |   centinaia               {  $$=$1;  }
                    |   migliaia                {  $$=$1;  }
                    |   milioni                 {  $$=$1;}
                    ;

centinaia: NUMEROU                             {$$=$1; }
          |NUMEROD                             {$$=$1; }
          |NUMEROD NUMEROU                      {$$=$1+$2;}
          |NUMEROC                             {$$=$1; }
          |NUMEROC NUMEROD                      {$$=$1+$2;}
          |NUMEROC NUMEROU                      {$$=$1+$2;}
          |NUMEROC NUMEROD NUMEROU              {$$=$1+$2+$3;}
          |NUMEROU NUMEROC                     {$$=((100*$1)-100)+$2;}
          |NUMEROU NUMEROC NUMEROD              {$$=((100*$1)-100)+$2+$3;}
          |NUMEROU NUMEROC NUMEROU              {$$=((100*$1)-100)+$2+$3;}
```

```
           |NUMEROU NUMEROC NUMEROD NUMEROU{$$=((100*$1)100)+$2+$3+$4;}
                  ;

migliaia:    NUMEROML                          {$$=$1;}
            |centinaia NUMEROML                {$$=$1*$2;}
            |NUMEROML centinaia                 {$$=$1+$2;}
            |centinaia NUMEROML centinaia       {$$=$1*$2+$3;}
             ;

milioni:     NUMEROMN                                  {$$=$1;}
             |      migliaia NUMEROMN                 {$$=$1*$2;}
             |      NUMEROMN migliaia                 {$$=$1+$2;}
             |      migliaia NUMEROMN migliaia        {$$=$1*$2+$3;}
             |      centinaia NUMEROMN                {$$=$1*$2;}
             |      NUMEROMN centinaia                {$$=$1+$2;}
             |      centinaia NUMEROMN centinaia      {$$=$1*$2+$3;}
             |      centinaia NUMEROMN migliaia       {$$=$1*$2+$3;}
             |      migliaia NUMEROMN centinaia       {$$=$1*$2+$3;}
            ;

miliardi:    NUMEROMD                                 {$$=$1;}
             |     milioni NUMEROMD             {$$=$1*$2;}
             |      NUMEROMD milioni                  {$$=$1+$2;}
             |      milioni NUMEROMD milioni  {$$=$1*$2+$3;}
             |      milioni NUMEROMD centinaia        {$$=$1*$2+$3;}
             |      milioni NUMEROMD migliaia         {$$=$1*$2+$3;}
             |      migliaia NUMEROMD                 {$$=$1*$2;}
       |     NUMEROMD migliaia          {$$=$1+$2;}
       |      migliaia NUMEROMD migliaia        {$$=$1*$2+$3;}
       |      centinaia NUMEROMD                {$$=$1*$2;}
       |     NUMEROMD centinaia         {$$=$1+$2;}
       |      centinaia NUMEROMD centinaia  {$$=$1*$2+$3;}
       |      centinaia NUMEROMD migliaia       {$$=$1*$2+$3;}
       |      centinaia NUMEROMD milioni        {$$=$1*$2+$3;}
       |      migliaia NUMEROMD milioni         {$$=$1*$2+$3;}
       |      migliaia NUMEROMD centinaia       {$$=$1*$2+$3;}
       ;

%%

extern FILE *yyin;

//---------------------------------------------------------------------

int main(void)
{
FILE *fp;
       yyin=fopen(FILE_INPUT,"r");
       if (yyin==NULL) {
              printf("ERRORE NON APRE IL FILE %s \n",FILE_INPUT);
             exit(0); }

       fp=fopen(FILE_TIPOTRADUZIONE,"r");
       //Leggo dal FILE_TIPOTRADUZIONE il valore ke mi permette di
       fscanf(fp,"%d",&TipoTraduzione);
       // capire se effettuare una traduzione per TOP DANCER o SLIDER
       fclose(fp);

       inizializza();

       while (!feof(yyin)){
              yyparse();   }
FineTraduzione();
return 0;
}
```

```
//---------------------------------------------------------------------

/*  Funzione x l'aggiornamento della posizione dei motori.Riceve in ingresso:
TipoArto: il tipo di arto ke devo muovere (Elbow, wirst, etc...)PosizioneArto: la
posizione dell'arto (ovvero se è un arto destro o sinistro)*/

void AggiornaMotori(char* PosizioneArto,int TipoArto)
{

if(!strcmp(PosizioneArto,"Destra")) {
//Effettuo la conversione in esadecimale relativamente agli arti della //parte
destra
             if(TipoTraduzione==TOP_DANCER)
Dec2Esa_SX(posizione_motori[TipoArto],Nexa);
             else Dec2Esa(posizione_motori[TipoArto],Nexa);
             }
else {
//Effettuo la conversione in esadecimale relativamente agli arti della //parte
sinistra
             if(TipoTraduzione==TOP_DANCER)
Dec2Esa(posizione_motori[TipoArto],Nexa);
             else Dec2Esa_SX(posizione_motori[TipoArto],Nexa);
       }
//Chiudi else



// A questo punto non mi resta ke aggiornare la posizione dei motori
if(TipoTraduzione==TOP_DANCER){               //Se il file d'uscita deve essere
utilizzato dal Software TOP DANCER
       posizione_topdancer[TipoArto]=Nexa[0];
//allora aggiorno il vettore contente la forma leggibile x TOP DANCER
       posizione_topdancer[TipoArto+1]=Nexa[1];}
else {
 // Se deve essere utilizzato con il Simulatore Slider
       posizione_attuale[TipoArto]=Nexa[0];
//allora aggiorno il vettore contenente la riga leggibile x SLIDER
       posizione_attuale[TipoArto+1]=Nexa[1];}
}
//---------------------------------------------------------------------

/*Funzione che memorizza in una lista la riga contenente la posizione dei motori
corrispondente ad un movimento
In questo modo viene creata una lista contenente le righe del file d'output!
Tale lista verrà memorizzata in un file solo alla fine
Non memorizzo direttamente le righe perchè queste devono essere modificate quando
ci Loop
*/
void MemorizzaMovimento()
{
int lungh;
struct riga *p, *temp;

p=(struct riga *)malloc(sizeof(struct riga));
//alloco la quantita di spazio necessaria x creare una cella contenente
l'informazione
if(TipoTraduzione==TOP_DANCER) strcpy(p->informazione,posizione_topdancer);
// Se la traduzione deve essere x TOP DANCER allora 'informazione' conterr  Euna
riga leggibile a TOP DANCER
else strcpy(p->informazione,posizione_attuale);

// altrimenti la riga contenuta in informazione sar  Eleggibile al Simulatore
SLIDER
p->next=NULL;
```

```
if(posizione_motori[74]!=0) {
//questa condizione risulta vera quando si esce da un LOOP
       Dec2Esa_SX(posizione_motori[74],Nexa);
       //Permette di proseguire l'esecuzione delle azioni subito dopo un LOOP
       lungh=strlen(p->informazione);
       p->informazione[lungh-2]=Nexa[0];
       p->informazione[lungh-1]=Nexa[1];
       posizione_motori[74]++;
       }

// Adesso devo inserire la cella appena creata in coda alla lista

if (testa==NULL) testa=p;
// Se la lista è vuota allora inserisco la cella appena creata in testa alla lista
else {                          //altrimenti
       temp=testa;
       while(temp->next!=NULL){
       //scorro la lista fino a determinare l'ultimo elemento
              temp=temp->next;
              //e inserisco la cella in coda alla lista
              }
       temp->next=p;
       }
}

//---------------------------------------------------------------------

/*
Funzione di Inizializzazione:
Si occupa di memorizzare nella lista dei movimenti da effettuare la posizione
iniziale assunta dal Robot e di memorizzare il valore iniziale dei motori nel
vettore posizione_motori*/

void inizializza()
{
FILE *fp;                       //Creo il FILE_OUTPUT
fp=fopen(FILE_OUTPUT,"w");// se già esiste ne cancello il contenuto
fclose(fp);

MemorizzaMovimento();
//In questo modo inserisco in testa alla lista dei movimenti
//la posizione iniziale ke il robot deve assumere

if(TipoTraduzione==TOP_DANCER) {
//Se la traduzione deve essere relativa a TOP DANCER modifico
//opportunamente i valori assunti dai vari arti
       printf("\n Traduzione x Top Dancer\n");
       VELOCITA=1;
       SHOULDER_I_SX=22;
       SHOULDER_I_DX=52;
       SHOULDER_II_SX=25;
       SHOULDER_II_DX=55;
       ELBOW_SX=28;
       ELBOW_DX=58;
       WIRST_SX=31;
       WIRST_DX=61;
       WAIST_I_C=67;
       WAIST_II_C=64;
       HIP_I_DX=37;
       HIP_I_SX=7;
       HIP_II_DX=34;
       HIP_II_SX=4;
       KNEE_DX=40;
       KNEE_SX=10;
       ANKLE_I_DX=43;
```

```
       ANKLE_I_SX=13;
       ANKLE_II_DX=46;
       ANKLE_II_SX=16;
       ANKLE_III_DX=49;
       ANKLE_III_SX=19;
       LOOP=70;
       posizione_motori[SHOULDER_I_DX]=225;
       posizione_motori[SHOULDER_I_SX]=225;
       posizione_motori[SHOULDER_II_DX]=91;
       posizione_motori[SHOULDER_II_SX]=91;
       posizione_motori[ELBOW_DX]=239;
       posizione_motori[ELBOW_SX]=239;
       posizione_motori[WIRST_DX]=31;
       posizione_motori[WIRST_SX]=31;
       posizione_motori[WAIST_I_C]=128;
       posizione_motori[WAIST_II_C]=178;
       posizione_motori[HIP_I_DX]=113;
       posizione_motori[HIP_I_SX]=113;
       posizione_motori[HIP_II_DX]=128;
       posizione_motori[HIP_II_SX]=128;
       posizione_motori[KNEE_DX]=228;
       posizione_motori[KNEE_SX]=228;
       posizione_motori[ANKLE_I_DX]=127;
       posizione_motori[ANKLE_I_SX]=127;
       posizione_motori[ANKLE_II_DX]=133;
       posizione_motori[ANKLE_II_SX]=133;
       posizione_motori[ANKLE_III_DX]=127;
       posizione_motori[ANKLE_III_SX]=127;

       }

       //Inizializzo la posizione iniziale dei motori
       else{
       posizione_motori[SHOULDER_I_DX]=225;
       posizione_motori[SHOULDER_I_SX]=225;
       posizione_motori[SHOULDER_II_DX]=91;
       posizione_motori[SHOULDER_II_SX]=91;
       posizione_motori[ELBOW_DX]=239;
       posizione_motori[ELBOW_SX]=239;
       posizione_motori[WIRST_DX]=31;
       posizione_motori[WIRST_SX]=31;

       posizione_motori[WAIST_I_C]=127;
       posizione_motori[WAIST_II_C]=77;
       posizione_motori[HIP_I_DX]=128;
       posizione_motori[HIP_I_SX]=128;
       posizione_motori[HIP_II_DX]=113;
       posizione_motori[HIP_II_SX]=113;
       posizione_motori[KNEE_DX]=228;
       posizione_motori[KNEE_SX]=228;
       posizione_motori[ANKLE_I_DX]=127;
       posizione_motori[ANKLE_I_SX]=127;
       posizione_motori[ANKLE_II_DX]=133;
       posizione_motori[ANKLE_II_SX]=133;
       posizione_motori[ANKLE_III_DX]=127;
       posizione_motori[ANKLE_III_SX]=127;
       }

}  //Chiudi Funzione Inizializza

//---------------------------------------------------------------------

/*Funzione che converte il numero Decimale in esadecimale
 relativamente ai motori della parte destra ; Riceve in ingresso:    ndec:numero
decimale da convertire;   nesad: è un vettore di caratteri contenente il
```

```
n°esadecimale convertito*/

void Dec2Esa(int ndec, char* nesad)
{
int temp;
char n_esad[2];

       temp=ndec%16;              //resto della divisione
       nesad[1]=ValoreEsadecimale(temp);
       temp=ndec/16;              //quoziente della divisione
       nesad[0]=ValoreEsadecimale(temp);
}

//---------------------------------------------------------------------

/* Funzione che converte il numero Decimale in esadecimale relativamente ai motori
della parte Sinistra
 Riceve in ingresso:      ndec:numero decimale da convertire;
nesad: è un vettore di caratteri contenente il n°esadecimale convertito */

void Dec2Esa_SX(int ndec, char* nesad)
{
int temp;
char n_esad[2];

       temp=ndec%16;              //resto della divisione
       nesad[1]=ValoreEsadecimaleParteSinistra(temp);
       temp=ndec/16;              //quoziente della divisione
       nesad[0]=ValoreEsadecimaleParteSinistra(temp);
}

//---------------------------------------------------------------------

/*
 Funzione che converte il numero esadecimale in in decimale
 Ingresso: ne: numero esadecimale da convertire
 Uscita:     ritorna il numero intero decimale corrispondente
*/
int Esa2Dec(char* ne)
{
int risultato=0;
int temp;

       temp=ValoreDecimale(ne[0]);
       risultato=risultato + temp*16;
       temp=ValoreDecimale(ne[1]);
       risultato=risultato + temp;

return (risultato);
}

//---------------------------------------------------------------------

/*Funzioni d'appoggio alla conversione*/
char ValoreEsadecimaleParteSinistra(int index)
{
char vettore_valori[16]={'0','1','2','3','4','5','6','7'
                         ,'8','9','a','b','c','d','e','f'};

return (vettore_valori[index]);

}

char ValoreEsadecimale(int index)
{
```

```
char vettore_valori[16]={'f','e','d','c','b','a','9','8','7',
                         '6','5','4','3','2','1','0'};

return (vettore_valori[index]);
}


int ValoreDecimale(char caratt)
{
int h;
char vettore_valori[16]={'f','e','d','c','b','a','9','8','7',
                         '6','5','4','3','2','1','0'};
int index;

for(h=0;h<16;h++)
{
      if(vettore_valori[h]==caratt) {
                  index=h;
                  h=16;
                  }
}

return (index);
}

//---------------------------------------------------------------------

/*
Funzione x la creazione di cicli ke permettono la ripetizione di movimenti Riceve
in Ingresso: NumRipetizioni - Numero di azioni da ripetere */

void CreaRipetizioni(int NumRipetizioni)
{
char Int2Char[31]={'0','1','2','3','4','5','6','7','8','9'};
struct riga *pointer;

pointer=testa;
while(pointer->next!=NULL) pointer=pointer->next;
//scorro la lista delle azioni fino a determinare l'ultima
pointer->informazione[LOOP]='-';
//aggiorno l'ultima azione in modo ke terminata l'esecuzione della stessa
//torno indietro x ripetere il ciclo voluto
if (NumRipetizioni<10){
      pointer->informazione[LOOP+1]=Int2Char[NumRipetizioni];}

else if (NumRipetizioni==10) {//Nel caso in cui il numero di azioni da ripetere sia
10
      pointer->informazione[LOOP+1]='1';
      pointer->informazione[LOOP+2]='0';
      pointer->informazione[LOOP+3]='!';
      pointer->informazione[LOOP+4]='0';
      pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==11) {//Nel caso in cui il numero di azioni da ripetere sia
11
      pointer->informazione[LOOP+1]='1';
      pointer->informazione[LOOP+2]='1';
      pointer->informazione[LOOP+3]='!';
      pointer->informazione[LOOP+4]='0';
      pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==12) {//Nel caso in cui il numero di azioni da ripetere sia
12
      pointer->informazione[LOOP+1]='1';
      pointer->informazione[LOOP+2]='2';
      pointer->informazione[LOOP+3]='!';
      pointer->informazione[LOOP+4]='0';
```

```
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==13) {//Nel caso in cui il numero di azioni da ripetere sia
13
       pointer->informazione[LOOP+1]='1';
       pointer->informazione[LOOP+2]='3';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==14) {//Nel caso in cui il numero di azioni da ripetere sia
14
       pointer->informazione[LOOP+1]='1';
       pointer->informazione[LOOP+2]='4';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==15) {
       pointer->informazione[LOOP+1]='1';
       pointer->informazione[LOOP+2]='5';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==16) {
       pointer->informazione[LOOP+1]='1';
       pointer->informazione[LOOP+2]='6';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==17) {
       pointer->informazione[LOOP+1]='1';
       pointer->informazione[LOOP+2]='7';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==18) {
       pointer->informazione[LOOP+1]='1';
       pointer->informazione[LOOP+2]='8';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==19) {
       pointer->informazione[LOOP+1]='1';
       pointer->informazione[LOOP+2]='9';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==20) {
       pointer->informazione[LOOP+1]='2';
       pointer->informazione[LOOP+2]='0';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==21) {
       pointer->informazione[LOOP+1]='2';
       pointer->informazione[LOOP+2]='1';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==22) {
       pointer->informazione[LOOP+1]='2';
       pointer->informazione[LOOP+2]='2';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==23) {
       pointer->informazione[LOOP+1]='2';
```

```
       pointer->informazione[LOOP+2]='3';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==24) {
       pointer->informazione[LOOP+1]='2';
       pointer->informazione[LOOP+2]='4';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==25) {
       pointer->informazione[LOOP+1]='2';
       pointer->informazione[LOOP+2]='5';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==26) {
       pointer->informazione[LOOP+1]='2';
       pointer->informazione[LOOP+2]='6';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==27) {
       pointer->informazione[LOOP+1]='2';
       pointer->informazione[LOOP+2]='7';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==28) {
       pointer->informazione[LOOP+1]='2';
       pointer->informazione[LOOP+2]='8';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==29) {
       pointer->informazione[LOOP+1]='2';
       pointer->informazione[LOOP+2]='9';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}
else if (NumRipetizioni==30) {
       pointer->informazione[LOOP+1]='3';
       pointer->informazione[LOOP+2]='0';
       pointer->informazione[LOOP+3]='!';
       pointer->informazione[LOOP+4]='0';
       pointer->informazione[LOOP+5]='0';}

}


//---------------------------------------------------------------------

/* Funzione x la creazione di un numero prestabilito di azioni da         ripetere
;*Riceve in Ingresso: Num(Numero di azioni da ripetere) Prim(Primo elemento da
ripetere) */

void CreaNRipetizioni(int Prim,int Num)
{
int indice;
//struct riga *primo;
struct riga *temp;
struct riga *p;
//struct riga *Lista1;
struct riga *ScorriLista;
struct riga *nuova_lista;
```

```
nuova_lista=CreaListaDaRipetere(Prim);
//crea una lista contenente le azioni ke voglio siano ripetute

//Aggiungo alla lista principale le righe ke mi permettono la ripetizione delle
azioni
for(indice=0;indice<Num;indice++) {
                    temp=nuova_lista;
                    while (temp!=NULL) {
                                p=(struct riga *)malloc(sizeof(struct riga));
                                //alloco la quantita di spazio necessaria x creare
una cella contenente l'informazione
                                strcpy(p->informazione,temp->informazione);
                                p->next=NULL;

                                ScorriLista=testa;
                                while(ScorriLista->next!=NULL)
ScorriLista=ScorriLista->next;
                                //scorro la lista fino a determinare l'ultimo
elemento
                                ScorriLista->next=p;

                                //e inserisco la cella in coda alla lista
                                NumeroAzioni++;

                                //Incremento il numero di azioni eseguite
                                temp=temp->next;
                                }// Chiude il while
                    } //Chiude il for

}



//---------------------------------------------------------------------

/* Crea la lista contenente gli elementi da ripetere*/

struct riga* CreaListaDaRipetere(int PrimoElementoDaRipetere)
{
int i;
struct riga *primo;
struct riga *Lista=NULL;
struct riga *p;
struct riga *temp;

primo=testa;
for (i=0;i<PrimoElementoDaRipetere;i++) //Determino il I elemento da ripetere
                          primo=primo->next;

//Creo una lista i cui elementi sono lo righe da ripetere
while (primo!=NULL) {
      p=(struct riga *)malloc(sizeof(struct riga));
      //alloco la quantita di spazio necessaria x creare una cella contenente
l'informazione
      strcpy(p->informazione,primo->informazione);
      p->next=NULL;

      if (Lista==NULL) Lista=p;
      // Se la lista  Evuota allora inserisco la cella appena creata in testa alla
lista
      else {                          //altrimenti
            temp=Lista;
            while(temp->next!=NULL) temp=temp->next;
      //scorro la lista fino a determinare l'ultimo elemento
```

```
       temp->next=p;             //e inserisco la cella in coda alla lista
              }
       primo=primo->next;
       }//chiudi while

return(Lista);
}

//---------------------------------------------------------------------

/*Funzione ke visualizza gli elementi di una lista*/
void VisualizzaLista(struct riga* Lista)
{
while(Lista!=NULL) {
              printf("\n %s",Lista->informazione);
              Lista=Lista->next;

       }
}
//-----------------------------------------------------------------
/* Funzione ke concatena due liste */
struct riga* ConcatenaListe(struct riga *PrimaLista,struct riga *SecondaLista)
{
struct riga *temporan;
temporan=PrimaLista;
while(temporan->next!=NULL) temporan=temporan->next;
temporan->next=SecondaLista;
return(PrimaLista);
}
//-----------------------------------------------------------------------
void FineTraduzione()
{
FILE *fp;
fp=fopen(FILE_FINETRADUZIONE,"w");
fclose(fp);
}
//---------------------------------------------------------------------
void yyerror(s)
       char* s;
{
printf("Errore");
//FineTraduzione();
}
//---------------------------------------------------------------------
int yywrap(void)
{
return 1;
}
//-----------------------------------------------------------------------
int Arrotonda(float f)
{
int in;
float res;
in= (int)(f);
res= f-in;
if(res>0.5){
return(in+1);
}
else{
return(in);
}

}
```

A.3 Medoto per inviare il comportamento al robot

```
void __fastcall TMainFormRobovieMovement::PassaAlRobotButtonClick(
        TObject *Sender)
{
MemoSend->Clear();
MemoSend->Lines->LoadFromFile("serialeoutput.txt");
if (MemoSend->Lines->Count>0)
            {
            for (int i = 0; i <MemoSend->Lines->Count; i++) {
                  message = "";
                  message = message + MemoSend->Lines->Strings[i] +
'\n';
                  MainForm->SendToRobovie(message);
            }
//          message = '\n';
//          message = message + MemoSend->Lines->Strings[0] + '\n';
//          MainForm->SendToRobovie(message);
      }
      else
            Application->MessageBox("Nothing to send.",
                        "Serial Information",
                        MB_ICONINFORMATION);
                        //if (CheckBox1->Checked==true) ;


}
```

A.4 Metodo per la conversione delle stringhe da inviare tramite seriale

```
void __fastcall
TMainFormRobovieMovement::ConvertiDaRobovieMakerASeriale(void)
 {
FILE *fp;
FILE *fq;
//struttura dati che contiene la stinga di riferimento per la seriale
//ed il numero del motore
typedef struct elemento
            {
            char mss[26];
            int motore;
            }Telemento ;
//vettore di strutture deve contenere uno stage
Telemento ele[29];
//implica che ogni stage deve contenere informazioni relativi a 28
motori
//messaggi standard
```

```
char mss1[12]=":C0000Q00P1";//5 6 14 19 20 28

char mss4[17]=":T0000C0000Q00P1";//altri

char mss2[26]=":T0000C0000Q00D00R00P1g00";//1 e 2

char mss3[20]=":T0000C0000Q00D00P1";//3 e 4


//struttura dati che contiene contiene una stringa del Tipo
@21AnnBmm..........
typedef struct elem
        {
            char sms[78];//stringa che rappresenta gli stage

            }Telem ;
//vettore di strutture che rappresenta tutti gli stage possibili in
una azione
Telem elem[MAX];
int x;
int n=0;
int y;
int i;
int j;
int num;
int num1;
char c;
int bin[7];
int dec;
char hex;
char Nexa[2]={'0','0'};
char esadecimale_255[1];
char numero_esadecimale[1];
int numero=0;
int numero1=0;
int numero2=0;
int k; float num2;
int val;

/*
int fun1(char n);
int fun2(char n);
void Dec2Esa(int ndec, char* nesad);
char ValoreEsadecimale(int index);
void Dec2Esa_SX(int ndec, char* nesad);
char ValoreEsadecimale(int index);
int ValoreDecimale(char caratt);
char ValoreEsadecimaleParteSinistra(int index);
*/



//sto creando la struttura dati di defoult

for(i=1;i<=28;i++)
              {
              ele[i].motore=i;
```

```
              if((i==5)||(i==6)||(i==14)||(i==19)||(i==20)||(i==28))
                                  {
                   strcpy(ele[i].mss , mss1);
                          }
              else{
                          if((i==1)||(i==2)){

strcpy(ele[i].mss , mss2);
                                          }
                            else{
                                   if((i==3)||(i==4))
                                        {

strcpy(ele[i].mss , mss3);
                                         }
                                   else{
                                       strcpy(ele[i].mss , mss4 );
                                      }
                                           }
                    }


              }

//apro in lettura il file che deve contenere i vari stage
fp= fopen("tradotta.txt", "r");
     if(fp == NULL )
           {
                        printf("Errore nell'apertura del file");
                 fclose(fp);
                 }
       else{
              i=0;
                   j=0;
            do{
                c=fgetc(fp);
                if(c!='\n')
                    {
                                elem[j].sms[i]=c;
                    i++;
                    }
                else{
                     if(c=='\n'){
                                 elem[j].sms[i]='\0';
                          i=0;
                          j++;
                          }
                     }
                    }while(c!=EOF);
              elem[j].sms[0]='\0';
              n=(j);
             //getchar();
              fclose(fp);
               }


j=0;
fq=fopen("output.txt", "w");
```

```
while(elem[j].sms[0]!='\0'){
                            i=0;
                            fprintf(fq, "%c",'@');
                            fprintf(fq, "%c",'0');
                                        fprintf(fq, "%c",'0');
//convesione che viene fatta sulla velocita per ottenere in uscita
l'esadecimale
//corrispondende alla  velocita inserita nella programmazione del
movimento
                                        numero1= fun1
(elem[j].sms[1]);
                                        numero2=
fun2(elem[j].sms[2]);
                                        num1=(numero1+numero2);
                                        num2= (float)num1*0.82;

                                        num1= SimpleRoundTo(num2,0);
                                        Dec2Esa(num1, Nexa);
//funzione che riceve in ingresso un numero in base decimale elo
restituisce
//in base esadecimale
                            fprintf(fq, "%c",Nexa[0]);
                            fprintf(fq, "%c",Nexa[1]);
                            while(elem[j].sms[i]!='\0')
                                                    {
                                  switch(i) {
                case 3: {
                           //Hip 2 sx -->lettera A--->motore 22

numero_esadecimale[0]=elem[j].sms[4];
numero_esadecimale[1]=elem[j].sms[5];
numero1= fun1 (numero_esadecimale[0]);
numero2= fun2(numero_esadecimale[1]);
                                    num1=(numero1+numero2);
                                    num=num1+6;
                                                    numero=num;
                                    if(numero>= 255)
                                       numero=6;
                                    if(numero<=0)
                                       numero=255;
                                   Dec2Esa(numero,Nexa);
                                   for(k=1;k<=28;k++) {
                                     if(ele[k].motore==22){val=k;
                                      ele[k].mss[4]=Nexa[0];
                                          ele[k].mss[5]=Nexa[1];
                                             }
                                                            }
                                      break;
                                    }
                case 6:{
                                 //Hip 1 sx---->lettera B--->motore 23

numero_esadecimale[0]=elem[j].sms[7];
numero_esadecimale[1]=elem[j].sms[8];
numero1= fun1 (numero_esadecimale[0]);
numero2= fun2(numero_esadecimale[1]);
                          num1 =(numero1+numero2);
                                    num=num1-5;
                                    numero=num;
```

```
                                        if(numero<=0)
                                              numero=250;
                                       if(numero>=250)
                                              numero=0;
                                        Dec2Esa(numero,Nexa);
                                                for(k=1;k<=28;k++) {
                              if(ele[k].motore==23){val=k;
                                 ele[k].mss[4]=Nexa[0];
                                  ele[k].mss[5]=Nexa[1];

                                       }
                                         }
                                            break;
                                       }

                                       case 9:{
                              //knee sx--->lettera c----->motore 24

numero_esadecimale[0]=elem[j].sms[10];
numero_esadecimale[1]=elem[j].sms[11];
numero1= fun1 (numero_esadecimale[0]);
numero2= fun2(numero_esadecimale[1]);
                                        num1=(numero1+numero2);
                                        num=num1-1;
                                                        numero=num;
                                        if(numero<=0)
                                            numero=0;
                                       if(numero>254)
                                              numero=254;
                                             Dec2Esa(numero,Nexa);
                                                for(k=1;k<=28;k++) {
                                                   if(ele[k].motore==24)
                                                              {val=k;
                                                    ele[k].mss[4]=Nexa[0];
                                                     ele[k].mss[5]=Nexa[1];
                                                                        }
                                         }
                                              break;
                                       }

                     case 12:{
                              //ankle1--->lettera d--->motore 25

numero_esadecimale[0]=elem[j].sms[13];
numero_esadecimale[1]=elem[j].sms[14];
 numero1= fun1 (numero_esadecimale[0]);
 numero2= fun2(numero_esadecimale[1]);
                                        num1=(numero1+numero2);
                                        num=num1+23;
                                        numero=num;
                                      if(numero>= 255)
                                          numero= 23;
                                          if(numero<=0)
                                            numero=255;
                                        Dec2Esa(numero,Nexa);
                         for(k=1;k<=28;k++) {
                                       if(ele[k].motore==25){val=k;

                                        ele[k].mss[4]=Nexa[0];
```

```
                                        ele[k].mss[5]=Nexa[1];
                                                                }

              }  break; }
                    case 15:{
           //ankle 2--->lettera E----->motore 26

numero_esadecimale[0]=elem[j].sms[16]
numero_esadecimale[1]=elem[j].sms[17];
numero1= fun1 (numero_esadecimale[0]);
numero2= fun2(numero_esadecimale[1]);
                                       num1=(numero1+numero2);
                                      num=num1+3;
                                       numero=num;
                                       if(numero>255)
                                        numero= 3;
                                        if(numero<= 0)
                                     numero=255;
                                       Dec2Esa(numero,Nexa);
                                         for(k=1;k<=28;k++) {
                                            if(ele[k].motore==26){
                                                     val=k;
                                              ele[k].mss[4]=Nexa[0];
                                               ele[k].mss[5]=Nexa[1];
                                                                    }
                                                        }
                                      break;
                                                        }

                  case 18:{
                    //ankle 3---->lettera F----->motore 27

numero_esadecimale[0]=elem[j].sms[19];
numero_esadecimale[1]=elem[j].sms[20];
numero1= fun1 (numero_esadecimale[0]);
  numero2= fun2(numero_esadecimale[1]);
                                       num1=(numero1+numero2);
                                       num=num1+4;
                                        numero=num;
                                       if(numero>255)
                                       numero=4;
                                       if(numero<=0)
                                       numero=255;
                                      Dec2Esa(numero,Nexa);
                                            for(k=1;k<=28;k++) {
                                                if(ele[k].motore==27)
                                                    {val=k;
                                                     ele[k].mss[4]=Nexa[0]
                                                     ele[k].mss[5]=Nexa[1];
                                                         }
                                                        }
                                      break;
                                       }

                      case 21:{
                  //shoulder 1 sx---->lettera G----->motore 15

numero_esadecimale[0]=elem[j].sms[22]
numero_esadecimale[1]=elem[j].sms[23];
```

```
numero1= fun1 (numero_esadecimale[0]);
numero2= fun2(numero_esadecimale[1]);
                                    num1 = numero1+numero2;
                                    num= 255-num1+9;
                                    numero=num;
                                  if(numero<=9)
                                      numero=9;
                                    if(numero>=246)
                                        numero=255;
                                    Dec2Esa(numero,Nexa);
                                for(k=1;k<=28;k++) {
                                       if(ele[k].motore==15){val=k;
                                            ele[k].mss[4]=Nexa[0];
                                             ele[k].mss[5]=Nexa[1];
                                                                  }
                                                }
                                   break;
                                    }

                        case 24:{
                    //shoulder 2 sx---->lettera h----> motore 16

numero_esadecimale[0]=elem[j].sms[25];
numero_esadecimale[1]=elem[j].sms[26];
numero1= fun1 (numero_esadecimale[0]);
 numero2= fun2(numero_esadecimale[1]);
                            num=(numero1+numero2);
                                    numero=255-num+20;
                                    if(numero<=0)
                                       numero=20;
                                    if(numero>=235)
                                   numero=255;
                                    Dec2Esa(numero,Nexa);
                                            for(k=1;k<=28;k++){
                                              if(ele[k].motore==16)
                                                      {val=k;
                                          ele[k].mss[4]=Nexa[0];
                                           ele[k].mss[5]=Nexa[1];
                                                      }
                                                      }
                                   break;
                                                      }

                           case 27:{
                       //Elbow sx----->lettera I----->motore 17

numero_esadecimale[0]=elem[j].sms[28];
numero_esadecimale[1]=elem[j].sms[29];
numero1= fun1 (numero_esadecimale[0]);
 numero2= fun2(numero_esadecimale[1]);
                                    num1=(numero1+numero2);
                                    num=num1-4;
                                   numero=num;
                                     if(numero<=0)
                                       numero=251;
                                    if(numero>=251)
                                        numero=0;
                                   Dec2Esa(numero,Nexa);
                                        for(k=1;k<=28;k++) {
```

```
                                        if(ele[k].motore==17){val=k;
                                              ele[k].mss[4]=Nexa[0];
                                              ele[k].mss[5]=Nexa[1];

                           }
                                                        }
                                     break;
                                             }


                         case 30:{
                       //Wirst sx---->lettera j----->motore 18

numero_esadecimale[0]=elem[j].sms[31];
numero_esadecimale[1]=elem[j].sms[32];
numero1= fun1 (numero_esadecimale[0]);
numero2= fun2(numero_esadecimale[1]);
                                      num =numero1+numero2;
                                      numero=255-num-6;//
                                       if(numero<=0)
                                          numero=0;
                                      if(numero>=255)
                                          numero=255;
                                      Dec2Esa(numero,Nexa);
                                               for(k=1;k<=28;k++)
                                          if(ele[k].motore==18){
                                             val=k;
                                             ele[k].mss[4]=Nexa[0];
                                              ele[k].mss[5]=Nexa[1];
}}
                  break;
                                      }

                    case 33:{
                   //hip 2 dx---->lettera k---->motore 8

numero_esadecimale[0]=elem[j].sms[34];
numero_esadecimale[1]=elem[j].sms[35];
numero1= fun1 (numero_esadecimale[0]);
 numero2= fun2(numero_esadecimale[1]);
                                      num= numero1+numero2;
                                      numero=num-23;
                                       if(numero<=0)
                                         numero=0;
                                    if(numero>=255)
                                         numero=232;
                                      Dec2Esa(numero,Nexa);
                                               for(k=1;k<=28;k++) {
                                                   if(ele[k].motore==8)
                                                               {val=k;

                                                  ele[k].mss[4]=Nexa[0];
                                                  ele[k].mss[5]=Nexa[1];

                                                                      }
                                                                      }
                                                                 break;
                                                                      }
                        case 36:{
```

```
                      //hip 1 dx---->lettera l---->motore 9

numero_esadecimale[0]=elem[j].sms[37];
numero_esadecimale[1]=elem[j].sms[38];
numero1= fun1 (numero_esadecimale[0]);
numero2= fun2(numero_esadecimale[1]);
                                    num= numero1+numero2;
                                    numero=num-8;
                                                  if(numero<=0)
                                       numero=0;
                                    if(numero>=255)
                                        numero=255;
                                    Dec2Esa(numero,Nexa);
                                                for(k=1;k<=28;k++) {
                                               if(ele[k].motore==9){
                                                              val=k;

                                                     ele[k].mss[4]=Nexa[0];

                                                     ele[k].mss[5]=Nexa[1];

                                                                   }
                                                                   }

                                                              break;
                                                                   }
                                 case 39:{
                       //knee dx---->lettera M----->motore 10

numero_esadecimale[0]=elem[j].sms[40];
numero_esadecimale[1]=elem[j].sms[41];
 numero1= fun1 (numero_esadecimale[0]);
numero2= fun2(numero_esadecimale[1]);
                                    num= numero1+numero2;
                                    numero=num-18;
                                     if(numero<=0)
                                       numero=0;
                                                  if(numero>=255)
                                        numero=255;
                                    Dec2Esa(numero,Nexa);
                                            for(k=1;k<=28;k++) {
                                             if(ele[k].motore==10){

                                                              val=k;
                                       ele[k].mss[4]=Nexa[0];
                                       ele[k].mss[5]=Nexa[1];
                                                                   }
                                                                   }

                                   break;
                                   }

                          case 42:{
                      //ankle1 dx---->lettera N----->motore 11

numero_esadecimale[0]=elem[j].sms[43];
numero_esadecimale[1]=elem[j].sms[44];
numero1= fun1 (numero_esadecimale[0]);
 numero2= fun2(numero_esadecimale[1]);
```

```
num= numero1+numero2;
                                    numero=num-18;
                                         if(numero<=0)
                                           numero=0;
                                        if(numero>=255)
                                       numero=255;
                                        Dec2Esa(numero,Nexa);
                                                for(k=1;k<=28;k++) {

                                             if(ele[k].motore==11){val=k;
                                                   ele[k].mss[4]=Nexa[0];

                                                   ele[k].mss[5]=Nexa[1];

                                                                          }
                                                                          }
                                       break;
                                       }

                     case 45:{
                   //ankle2 ---->lettera o----->motore 12

numero_esadecimale[0]=elem[j].sms[46];
numero_esadecimale[1]=elem[j].sms[47];
numero1= fun1 (numero_esadecimale[0]);
numero2= fun2(numero_esadecimale[1]);
                                        num= numero1+numero2;
                                        numero=num-13;
                                         if(numero<=0)
                                                                numero=0;
                                        if(numero>=255)
                                            numero=255;
                                        Dec2Esa(numero,Nexa);
                                        for(k=1;k<=28;k++) {

                                             if(ele[k].motore==12){val=k;
                                              ele[k].mss[4]=Nexa[0];
                                              ele[k].mss[5]=Nexa[1];
                                                                   }
                                                        }

                                        break;
                                       }

                        case 48: {
                      //ankle 3----->lettera P----->motore 13

numero_esadecimale[0]=elem[j].sms[49];
numero_esadecimale[1]=elem[j].sms[50];
 numero1= fun1 (numero_esadecimale[0]);
 numero2= fun2(numero_esadecimale[1]);
                           num= numero1+numero2;
                                    numero=num+7;
                                     if(numero<=7)
                                           numero=7;
                                        if(numero>=248)
                                            numero=255;
                                        Dec2Esa(numero,Nexa);
                                                     for(k=1;k<=28;k++) {
```

```
                                                    if(ele[k].motore==13)
                                                                  {val=k;

                                                   ele[k].mss[4]=Nexa[0];

                                                   ele[k].mss[5]=Nexa[1];

                                                                       }
                                                                       }
                                        break;
                                                        }

                     case 51:{
                       //shoulder 1 dx----->lettera Q------->motore 1

numero_esadecimale[0]=elem[j].sms[52];
numero_esadecimale[1]=elem[j].sms[53];
numero1= fun1 (numero_esadecimale[0]);
 numero2= fun2(numero_esadecimale[1]);
                                        num1= numero1+numero2;
                                         num= 255-num1-21;
                                        numero=num;
                                         if(numero<=0)
                                           numero=234;
                                        if(numero>=234)
                                                                numero=0;
                                        Dec2Esa(numero,Nexa);
                                           for(k=1;k<=28;k++) {
                                           if(ele[k].motore==1){val=k;

                                                   ele[k].mss[4]=Nexa[0];

                                                   ele[k].mss[5]=Nexa[1];
                                                                       }
                                                                       }
                                        break;
                                        }

                        case 54:{
                           //shoulder 2 dx---->lettera R------>motore 2

numero_esadecimale[0]=elem[j].sms[55];
numero_esadecimale[1]=elem[j].sms[56];
 numero1= fun1 (numero_esadecimale[0]);
 numero2= fun2(numero_esadecimale[1]);
                                        num1=numero1+numero2;
                                        num = 255-num1+9;
                                        numero=num;
                                                          if(numero>= 255)
                                              numero= 255;
                                      if(numero<=0)
                                             numero= 9;
                                        Dec2Esa(numero,Nexa);

                                                    for(k=1;k<=28;k++) {
                                              if(ele[k].motore==2){val=k;

                                                   ele[k].mss[4]=Nexa[0];
```

```
                                                    ele[k].mss[5]=Nexa[1];
                                                                        }

                                                                        }
                                        break;
                                        }

                                case 57:{
                        //Elbow dx---->lettera S----->motore 3

numero_esadecimale[0]=elem[j].sms[58];
 numero_esadecimale[1]=elem[j].sms[59];
 numero1= fun1 (numero_esadecimale[0]);
 numero2= fun2(numero_esadecimale[1]);
num= numero1+numero2;
                                     numero=num-9;
                                      if(numero<=9)
                                            numero=255;
                                         if(numero>=255)
                                                                  numero=9;
                                         Dec2Esa(numero,Nexa);
                                         for(k=1;k<=28;k++) {
                                         if(ele[k].motore==3){val=k;
                                             ele[k].mss[4]=Nexa[0];
                                                    ele[k].mss[5]=Nexa[1];
                                                                        }
                                                                        }

                                          break;
                                                        }

                                 case 60:{
                                //Wirst dx----->lettera T----->motore 4

numero_esadecimale[0]=elem[j].sms[61];
numero_esadecimale[1]=elem[j].sms[62];
numero1= fun1 (numero_esadecimale[0]);
numero2= fun2(numero_esadecimale[1]);
                                        num1= numero1+numero2;
                                        num= 255-num1;
                                                        numero=num+7;
                                         if(numero<=7)
                                            numero=7;
                                         if(numero>=255)
                                             numero=255;
                                                     Dec2Esa(numero,Nexa);
                                                      for(k=1;k<=28;k++) {
                                                     if(ele[k].motore==4){
                                                                  val=k;
                        ele[k].mss[4]=Nexa[0];
                                                    ele[k].mss[5]=Nexa[1];
                                                                        }
                                                                        }
                                                                  break;
                                                                        }
                           case 63:{
                                //Waist 2--->lettera U----->motore 7
 numero_esadecimale[0]=elem[j].sms[64];
```

```
 numero_esadecimale[1]=elem[j].sms[65];
  umero1= fun1 (numero_esadecimale[0]);
   numero2= fun2(numero_esadecimale[1]);
                           num= numero1+numero2;
                            numero=num-101;
                             if(numero>= 255)
                            numero= 101;
                          if(numero<=0)
                                  numero=0;
                                Dec2Esa(numero,Nexa);
                                for(k=1;k<=28;k++) {
                                           if(ele[k].motore==7)
                                                                {val=k;
                                                ele[k].mss[4]=Nexa[0];
                                                ele[k].mss[5]=Nexa[1];

                                                                    }

                                                 }
                                      break;
                                      }
                     case 66:{
    //Waist 1--->lettera V---->motore 21
numero_esadecimale[0]=elem[j].sms[67];
 numero_esadecimale[1]=elem[j].sms[68];
 numero1= fun1 (numero_esadecimale[0]);
 numero2= fun2(numero_esadecimale[1]);
                           num1=(numero1+numero2);
                           num=num1+10;
                           numero=num;
                           if(numero<=10)
                             numero=255;
                                if(numero>=255)
                                          numero=10;
                                     Dec2Esa(numero,Nexa);
                                       for(k=1;k<=28;k++) {
                                              if(ele[k].motore==21){
                                                      val=k;
                                             ele[k].mss[4]=Nexa[0];
                                              ele[k].mss[5]=Nexa[1];
                                                                  }
                                                       }
                                        break;
                                                      }
                                          }

                                      i++;
                                       }for(i=1; i<= 28;i++)
                                        fprintf(fq, "%s",ele[i].mss);
                                        fprintf(fq, "\n");

                                    j++;
                                                     }
                                fclose(fq);
            MemoSend->Lines->LoadFromFile("serialeoutput.txt");
}
```

Appendice B

B.1 L' installazione "RobovieMovement"

Prima di procedere con l'istallazione chiudere tutti i programmi in esecuzione. La presenza di altri programmi attivi, potrebbe interferire con l'istallazione stessa.

L'installazione di "RobovieMovement" viene effettuata eseguendo il file *RobovieMovement_Setup.exe,* presente all'interno del CD in dotazione alla presente documentazione. In seguito, sarà sufficiente seguire le semplici istruzioni che compariranno nelle varie schermate.

Sarà possibile, inoltre, creare automaticamente un collegamento sul desktop al file *RobovieMovement.exe* al fine di eseguire direttamente da desktop il programma.

Per il corretto funzionamento del programma, sono necessari i seguenti requisiti minimi di sistema:

- Sistema operativo: Windows 95 o superiori;
- Processore: Pentium a 166 Mhz o equivalente;
- Lettore cd-rom;
- Hard Disk: 20 Mb liberi;
- Memoria RAM: 64 Mb;
- Installazione di "RoboStage.
- Istallazione di "RobovieMaker"

Il programma viene di default installato nella cartella *C:\Programmi\RobovieMovement* e può essere eseguito anche facendo doppio click sul file *RobovieMovement.exe* all'interno della suddetta cartella.

RobovieMovement viene anche inserito nella barra di avvio veloce presente in basso al desktop e seguendo il percorso *Start->Programmi -> RobovieMovement*.

La disinstallazione può essere effettuata da pannello di controllo o mediante l'apposito unistall in dotazione al programma

A.2 Sorgente Yacc

A.3 Metodo che realizza la conversione delle stringhe da inviare tramite seriale

Appendice B

Appendice B.1: Istallazione "RobovieMovement"

Prima di procedere con l'istallazione chiudere tutti i programmi in esecuzione. La presenza di altri programmi attivi, potrebbe interferire con l'istallazione stessa.

L'installazione di "RobovieMovement" viene effettuata eseguendo il file RobovieMovement_Setup.exe, presente all'interno del CD in dotazione alla presente documentazione. In seguito, sarà sufficiente seguire le semplici istruzioni che compariranno nelle varie schermate.

Sarà possibile, inoltre, creare automaticamente un collegamento sul desktop al file RobovieMovement.exe al fine di eseguire direttamente da desktop il programma.

Per il corretto funzionamento del programma, sono necessari i seguenti requisiti minimi di sistema:

Sistema operativo: Windows 95 o superiori;

Processore: Pentium a 166 Mhz o equivalente;

Lettore cd-rom;

Hard Disk: 20 Mb liberi;

Memoria RAM: 64 Mb;

Installazione di "RoboStage.

Istallazione di "RobovieMaker"

Il programma viene di default installato nella cartella C:\Programmi\RobovieMovement e può essere eseguito anche facendo doppio click sul file RobovieMovement.exe all'interno della suddetta cartella. RobovieMovement viene anche inserito nella barra di avvio veloce presente in basso al desktop e seguendo il percorso Start->Programmi -> RobovieMovement.
La disinstallazione può essere effettuata da pannello di controllo o mediante l'apposito unistall in dotazione al programma

B.2 Come aggiungere il tool

Verrà mostrato come aggiungere un tool. Per semplificare verrà mostrata la creazione del form ToolX. Verranno evidenziate in **rosso** le parti del codice che dovranno essere riadattate per la creazione di un form di nome diverso. In **grassetto** verranno date indicazioni ulle parti che verranno create in seguito.

1. Creare un Form (File → New→Form)
    a. oppure aggiungere uno esistente (Project->Add to …)
2. Nell'Object Inspector modificare le seguenti proprietà

    a. Name = **ToolXForm** **(NON modificare se il form esisteva)**
    b. Caption = **ToolX**
    c. DragKind = dkDock
    d. DragMode = dmAutomatic
    e. FormStyle = fsStayOnTop
    f. ScreenSnap = true
    g. BorderStyle = bsSizeToolWin
    h. Icon (opzionale, verrebbe visualizzata nelle schede e nella toolbar)
    i. Visibile = false

3. Nell'Object Inspector nella scheda Event fare doppio clic accanto ai seguenti

    a. OnStartDock ed aggiungere le seguenti righe
    ```
    DragObject = new TDragDockObjectEx(this);
    DragObject->Brush->Color = clAqua; // Per il riquadro rosso
    ```

b. OnClose ed aggiungere le seguenti righe

```
//Stacco la finestra dall'applicazione e creo un rettangolo nullo
ManualFloat(Rect(0, 0, 0, 0));
//In chiusura libero automaticamente le risorse
Action = caFree;
ToolXForm = NULL;
MainForm->ResetToolXOnDesk();    //verrà creata in seguito
```

4. Salvare il file aggiungendo il suffisso Source (es.: ToolXFormSource.cpp)
5. Dal Menù Project→Option alla voce Form

   a. Selezionare il form dalla lista di sinistra (auto-create forms)
   b. Premere la freccia > per spostarla nella lista di destra (Available forms)
   c. Premere Ok per accettare le modifiche

6. Salvare tutto
7. Nel file .CPP commentare la variabile globale in alto

   a. Es.: **//TToolXForm *ToolXForm;**

8. Nel file .H aggiungere la seguente riga tra le altre #include

   ```
   #include "MainFormSource.h"
   ```

9. Dal file .CPP copiare la direttiva es.: **#include "ToolXFormSource.h"** tra le altre #include nel file MainFormSource.h
10. Nel file MainFormSource.cpp ggiungere la variabile globale in alto

    TToolXForm *ToolXForm;

11. Nel file MainFormSource.h nella sezione private aggiungere

    int **ToolXOnDesk**;

12. Nella sezione public

    void __fastcall **ResetToolXOnDesk**(void);

13. Nel file MainFormSource.cpp aggiungere il metodo

```
void __fastcall TMainForm::ResetToolXOnDesk(void)
{
        ToolXOnDesk=0;
}
```

14. Dal Menù View→Toggle Unit/Form per visualizzare il Form dell'applicazione
15. Fare doppio clic sull'oggetto ActionManager
16. Selezionare la categoria Tool e premere il pulsante per aggiungere una nuova Action
17. Nell'Object Inspector modificare le seguenti proprietà

    Name = **ToolX**
    Caption = **ToolX**
    Hint = Show the ToolX Tool
    ImageIndex = selezionare un icona opp lasciare -1

Nella scheda Event fare doppio clic accanto OnExecute ed aggiungere il seguente codice

```
bool closeOK=false;
if (ToolXOnDesk==(JvPageList1->ActivePageIndex+1))
        closeOK = true; //Voglio chiuderla
else
        closeOK = false; //Era in un altro Desk o la voglio aprire
//Provo a chiudere la finestra
if (ToolXForm!=NULL)
        ToolXForm->Close();
if ((!closeOK)&&(ToolXOnDesk!=(JvPageList1->ActivePageIndex+1))) {
        ToolXForm = new TToolXForm(this);
        ToolXOnDesk=JvPageList1->ActivePageIndex+1;
        TPanel *pnlA;
        //Per la creazione in runtime sul Desk Attivo
        //Cerco tra i controlli della pagina attiva, il pannello principale
        for (int i = 0; i < JvPageList1->ActivePage->ControlCount; i++) {
                if (JvPageList1->ActivePage->Controls[i]-
>InheritsFrom(__classid(TPanel)))
                {
                        if (JvPageList1->ActivePage->Controls[i]-
>Name.SubString(1,3)=="Pan") {
                                pnlA =(TPanel *)(JvPageList1->ActivePage-
>Controls[i]);
                        break;
                        }
                }
        }
        //All'interno del pannello Principale cerco il pannello
denominato dsk...
        for (int i = 0; i < pnlA->ControlCount; i++) {
```

```
                if (pnlA->Controls[i]->InheritsFrom(__classid(TPanel)))
                {
                        if (pnlA->Controls[i]->Name.SubString(1,3)=="dsk")
{
                                TPanel *pnlB =(TPanel *)(pnlA->Controls[i]);
                                //Visualizzo il Tool nel pannello selezionato
                                        ToolXForm->ManualDock(pnlB,
NULL, alNone);
                                break;
                        }
                }
        }
        ToolXForm->Show();
}
```

18. Dal Menù View→Toggle Unit/Form per visualizzare il Form dell'applicazione
19. In Structure (sopra l'Object Inspector) (per aggiungere il tool nel menu)

    a. selezionare ActionManager→ActionBars→ (0 - ActionBar→ActionMainMenuBar)→Items→Tools→Items
    b. Fare clic con il tasto destro e scegliere Add Item

20. Nell'Object Inspector impostare la proprietà Action a **ToolX**
21. Nel metodo CloseTool aggiungere le seguenti righe

```
if (ToolXOnDesk==(activeIdx+1))
        ToolXForm->Close();
```

22. Nel metodo dskPnlDockDrop aggiungere le seguenti righe

```
if (Source->Control->InheritsFrom(__classid(TToolXForm))) {
        ToolXOnDesk = JvPageList1->ActivePageIndex+1;
}
```

23. Per aggiungere un pulsante alla toolbar premere F12 per visualizzare il form se non è già visualizzato

    a. selezionare la toolbar ToolBar1 (quella con i pulsanti degli altri tool per intenderci)
    b. fare clic con il tasto destro e selezionare New Button
    c. Nell'Object Inspector impostare la proprietà Action a **ToolX**
    d. Se non era già stata inserita un'icona nel form provare a selezionare un icona nella proprietà ImageIndex altrimenti impostare le proprietà

   i. Style = btsTextButton
   ii. Selezionate la ToolBar1 e nell'Object Inspector e impostare la proprietà
   iii. AllowTextButtons = true

24. Salvare tutto
25. Compilare e provare.