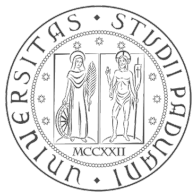

Universita' degli Studi di Padova

Facolta' di Ingegneria

Corso di Laurea in Ingegneria Informatica

Software didattico per Linux per la movimentazione del robot umanoide Robovie-X

Laureando

Luca Gottardo

Relatore

Prof. Emanuale Menegatti

Anno Accademico 2011/2012

Indice

Capitolo 1

Introduzione

Il presente elaborato è la relazione del lavoro svolto nel tirocinio di 500 ore presso l'azienda *IT+Robotics*. Durante il tirocinio è stato sviluppato un software che permette di fare didattica con il robot umanoide Robovie-X, prodotto dalla Vstone, senza che siano necessarie nozioni approfondite di robotica, quindi anche per istituti tecnici, licei e scuole superiori in generale. Per svillupare questo lavoro è stato neccessario approfondire le conoscenze del linguaggio c++ [1] a mia disposizione, nonchè imparare a lavorare in ambiente Linux. Sono state prese come riferimento per la conoscenza del robot Robovie-X alcune tesi già sviluppate su questo robot umanoide, fatte da R. Bonetto [2], A. Casasola [3] e C. Tavian [4]. È stato inoltre usato parte del software sviluppato da Fabio dalla Libera [5] per il robot umanoide Vision 4G, sempre prodotto dalla Vstone.

1.1 L'azienda

IT+Robotics è una spin-off partecipata dell'Universita' degli Studi di Padova, nata nel 2005 dalla collaborazione tra professori di Robotica e giovani del Dipartimento di Ingegneria dell'Informazione dell'Università degli Studi di Padova. Il suo obiettivo è il trasferimento tecnologico dall'Università alle aziende. L'innovazione con l'iniezione di tecnologie fino a pochi anni fa di esclusivo appannaggio della ricerca accademica è la ricetta per crescere e superare la crisi. Il team di scienziati e programmatori di IT+Robotics realizza soluzioni tecnologicamente all'avanguardia grazie ad anni di ricerca e sviluppo industriale nei diversi settori della robotica autonoma: sistemi operativi real-time, sistemi di visione artificiale, coordinamento di agenti software evoluti e simulazioni altamente realistiche. L'azienda si occupa di *simulatori*

3D, *sistemi in real-time*, *controllo qualità* , *videosorveglianza*, *robot umanoidi* e *visione omnidirezionale*.

1.2 Robot umanoidi

I robot sono utilizzati con profitto in molti ambiti: per il sollevamento di grossi pesi, per svolgere azioni ripetitive e per lavorare in ambienti difficilmente raggiungibili o pericolosi per l'uomo. Tuttavia la maggior parte di essi è usato in ambienti ristretti, come celle di lavoro industriali, e non ha mai un vero e proprio contatto con l'uomo. Solo nella società giapponese vi è un uso abbastanza ampio di robot che si interfacciano con gli esseri umani: come per esempio robot che fanno da guida all'interno di edifici. Questo però comporta molte difficoltà, infatti il robot deve essere in grado di muoversi autonomamente in sicurezza per sè, per l'ambiente e soprattutto per le persone. Un altro aspetto molto importante che non va trascurato è il fatto che i robot devono in qualche modo stimolare la familiarità con le persone, proprio per questo vengono costriuiti dei robot detti umanoidi, ovvero a somiglianza dell'essere umano. Sono appunto soprattutto i giapponesi ad essere avanzati in questo campo di sviluppo, non a caso l' IT+Robotics ha una partneship commerciale con l'azienda giapponese Vstone, produttrice tra le altre cose di robot umanoidi, che le garantisce la rivendita ufficiale dei robot Vstone in Europa. L'azienda in più fornisce la possibilità di montare eventuali sensori sui robot, come telecamere USB o mini computer, al fine di migliorare l'autonomia del robot stesso. Uno degli scopi per cui l'azienda si occupa di robot umanoidi è proprio il tentativo di iniziare a fare didattica di robotica educativa nelle scuole, in particolar modo negli istituti tecnici. Da queste considerazione è nata l'idea del mio lavoro. Ecco alcuni dei robot umanoidi commercializzati dall'azienda:

Figura 1.1: Manoi PF01

Figura 1.2: RB2000

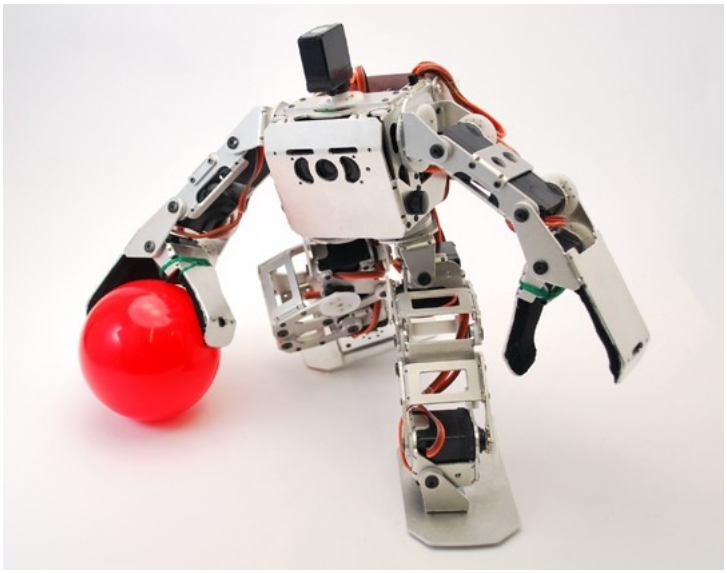

Figura 1.3: Robovie Nano

Figura 1.4: Robovie PC

Capitolo 2

Robovie-X

Il Robovie-X consiste di tre elementi principali: il robot, il controller e il software in dotazione.

2.1 Hardware

Il modello di Robovie utilizzato in questa tesi è lo *Standard Robovie-X*, prodotto dalla V-Stone. Sono disponibili anche altri due modelli: Il Robovie-X Lite e il Robovie-X Pro, che si differenziano dalla versione standard per numero e potenza dei servomotori.
È dotato di 17 gradi di libertà, 1 per la testa, 3 per ogni braccio e 5 per ogni gamba. Ogni grado di libertà è attuato da un servomotore analogico modello VS-S092J capace di sviluppare una coppia di 9.02kg/cm e velocità massima di 9.52rad/s.

La scheda di elaborazione on-board è costituita di un cpu a 60MHz con 64kB di ram e 512kB di ROM dove vengono caricati i movimenti e la mappatura del controller. I servomotori vengono controllati dalla scheda attraverso 17 uscite in PWM.

La scheda può comunicare con un controller wireless, come per esempio il controller PlayStation2 (o compatibili); e può essere programmata mediante il software *RobovieMaker2* attraverso un'interfaccia USB2.0. Possiede inoltre un selettore a nove posizioni che permette di scegliere tra nove diversi set di movimenti e configurazioni di controller senza dover riprogrammare il robot. Completa l'equipaggiamento del Robovie la scheda opzionale VS-IX001 contente il sensore inerziale, formato da un accelerometro a tre assi e un giroscopio a due assi (manca l'asse del giroscopio corrispondente all'asse longitudinale del robot). Il robot è inoltre fornito di un altoparlante per riprodurre dei suoni wav anch'essi caricati nella ROM della scheda.

Figura 2.1: Robovie-X, l'umanoide usato in questo lavoro.

Figura 2.2: Scheda di elaborazione del Robovie-X

Il robot è alimentato da un pacco batterie ricaricabile NiMh 6V composto da 5 celle con capacità di 1600mAh, che garatisce un'autonomia di venti minuti con il robot in continuo movimento.
Le altre due versioni del Robovie, Robovie-X Lite e Robovie-X Pro si differenziano per queste caratteristiche:

- Il Robivie-X Lite ha 13 servomotori ed è più piccolo e leggero della versione standard, caratteristiche che gli permettono una varietà di movimenti leggermente inferiore al Robovie-X, movimenti che però sono più veloci (grazie al peso ridotto) e performanti.

- Il Robovie-X Pro ha 19 servomotori, 16 dei quali più potenti della versione standard che gli permettono una camminata più veloce e stabile e inoltre gli consentono di portare alcuni pesi (come per esempio un Robovie-X (1.3kg)).

Figura 2.3: Robovie-X Lite

Figura 2.4: Robovie-X Pro

	Robovie-X	Robovie-X Lite	Robovie-X Pro
Dimensioni	343x180x71mm	340x180x73mm	383x180x73mm
Peso	1.30kg (con batteria)	1.20kg (con batteria)	1.96kg (con batteria)
Gradi di libertà	17 gradi di libertà	13 gradi di libertà	19 gradi di libertà
Servo Motori	VS-S092J x 17	VS-S092J x 13	VS-S281J x 16, VS-S092J x 3
CPU	VS-RC003HV		
Alimentazione	Batterie 6V NiMH		
OS supportato	Windows2000/XP/Vista/7		
Interfaccia PC	USB		
Altro	Altoparlante e occhi LED (VS-LED1 x 2)		

Tabella 2.1: Caratteristiche tecniche Robovie

2.2 Software

Il software fornito per l'utilizzo del Robovie è chiamato *RobovieMaker 2*. È disponibile unicamente per Windows, ma si può usare su Linux con l'ausilio di una *virtual machine*, come *VirtualBox*. *RobovieMaker 2* si divide principalmente in due pannelli: nel pannello di sinistra (*Pose Area*) sono presenti delle *slidebar* che consentono di controllare il valore di ogni variabile di giunto, oltre che la luminosità dei led ed altri parametri meno importanti. Nel pannello di destra (*Motion Area*) è possibile programmare dei movimenti attraverso un intuitivo linguaggio a blocchi, che permette di memorizzare le pose costruite nella *Pose Area* e di impostare le transizioni tra le varie pose. È possibile utilizzare anche strutture condizionali (if-then-else) ed effettuare semplici operazioni tra un numero limitato di variabili. È importante sottolineare come le pose all'interno della *Pose Area* non possano essere definite in funzione di alcun parametro: l'unico modo di variare la posizione di un giunto è agire attraverso la relativa slidebar. Inoltre non è possibile in alcun modo interrompere, accelerare, variare l'esecuzione di una posa in corso, poiché tutte le operazioni sulle variabili vengono effettuate tra il termine di una posa e la successiva, e dunque non è neppure possibile iniziare un'altra posa o elaborare variabili nel contempo. L'unico modo per variare la posizione del motore al di fuori del programma creato nella *motion area* è rappresentato dalla possibilità di sommare, al comando ciascun motore, una variabile. Questa variabile però sarà la stessa per ciacuna posa, e non è possibile modificarla durante l'esecuzione di una posa ma solo attraverso i blocchi di calcolo nell'intervallo tra il termine di una posa e l'inizio della posa successiva.

Una volta creato il programma di movimento del robot nella *Motion Area* è possibile caricarlo nella memoria del robot ed associare un evento per l'esecuzione di tale programma. L'evento può essere la pressione di un tasto

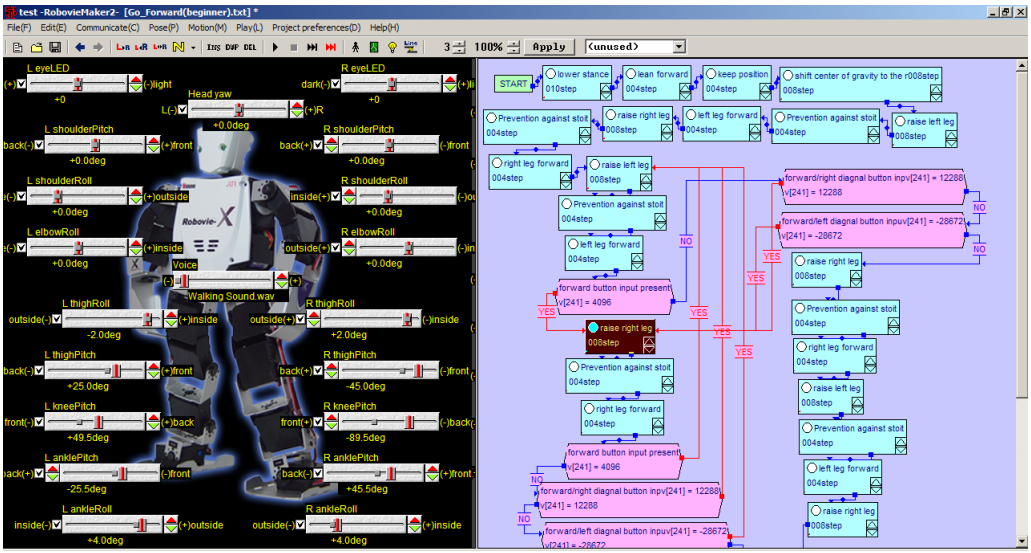

Figura 2.5: Schermata principale del programma *Roboviemaker2*

o di una combinazione di tasti sul controller, o la variazione di una variabile interna del robot (ad esempio la variabile che rappresenta il valore letto dall'accelerometro).

Durante l'esecuzione di un programma, se il robot è collegato via USB al computer è possibile accedere alla lettura delle variabili interne attraverso il programma.

Capitolo 3

Protocollo USB

La comunicazione con il Robovie-X e il sotfware *Roboviemaker2* avviene tramite protocollo USB in ambiente Windows. Per poter comunicare anche in ambiente Linux è stato fatto un lavoro di *reverse engineering* da N. Carlon e C. Tavian [4] che partendo dal lavoro svolto da R. Bonetto [2], che ha compreso la struttura base del pacchetto USB e i metodi per il comando dei motori del RB2000, anch'esso prodotto dalla Vstone, hanno creato una infrastuttura di comunicazione con il robot. Tuttavia per lo scopo di questo lavoro è stato neccessario continuare il lavoro di *reverse engineering*, per aggiungere nuove funzioni a quelle già conosciute che sono:

- Funzioni per comunicare in lettura/scrittura col robot
- Funzioni per modificare l'angolo di ogni singolo motore del robot
- Funzioni per leggere i valori dei giroscopi e accelerometri interni al robot

È quindi stato fatto un ulteriore lavoro di *reverse engineering* per approfondire la conoscienza del protocollo del robot, che l'azienda non mette a disposizione, e in particolare per conoscere le variabili per lo spegnimento dei singoli motori del robot. Per la cattura dei pacchetti USB è stato usato il software USBTrace in ambiente Windows. L'implementazione in C++ del software per Linux è stata sviluppata a partire da un'implementazione di Fabio Della Libera e ne eredita alcune funzioni.

3.1 Struttura del pacchetto USB

Viene ora descritta la struttura comune ad ogni pacchetto, determinata da R. Bonetto. Ogni pacchetto USB, sia in ingresso che in uscita, è formato da

64 byte, organizzati in 32 Word. Ogni coppia rappresenta un valore, con la convenzione *Little Endian*, cioè con il byte più significativo trasmesso dopo il byte meno significativo. In risposta ad un singolo pacchetto inviato dal PC si possono ricevere uno o tre pacchetti. Tutti i pacchetti hanno la seguente struttura:

- byte 00: **0x55**
- byte 01: rappresenta la quantità di pacchetti inviati/ricevuti consecutivamente. La prima cifra è l'indice del pacchetto corrente, la seconda è la quantità totale di pacchetti. Ad esempio, se il byte 01 vale "03", vuol dire che il pacchetto corrente è il primo di tre pacchetti (i successivi avrebbero i byte 01 posti a "13" e "23")
- byte 02, 03, 05: rappresentano un codice che identifica il tipo di trasmissione.
- byte 04: solitamente usato quando il pacchetto tratta di un indice.

3.1.1 Lettura e scrittura variabili

La scheda di elaborazione del robot ha la memoria strutturata in 256 variabili, ciascuna indicata da un indice da 1 a 256. Molte di queste sono riservate per l'utilizzo esclusivo del robot, altre invece sono disponibili esclusivamente in lettura o in scrittura. Sono stati determinati dei range di alcune variabili:

- Le variabili da 66 a 128 sono disponibili in lettura e scrittura per l'utente
- Le variabili 129, 130 e 131 rappresentano il valore degli accelerometri (nell'ordine X, Y, Z) e sono di sola lettura
- Le variabili 132 e 133 rappresentano il valore dei giroscopi X e Y e sono di sola lettura
- Le variabili immediatamente successive sono di sola lettura, probabilmente per schede diverse dalla VS-IX001 che ospitino più sensori

Altri range di variabili non sono stati indagati.

3.2 Funzioni conosciute

Carlon e Tavian hanno poi proseguito il lavoro di *reverse engineering*, per implementare delle funzioni basilari per la comunicazione in ambiente Linux

fra il robot e il pc. In particolare hanno implementato le funzioni per scrivere e leggere alcune variabili rispettivamente mandate al robot e ricevute dal robot.

3.2.1 Comunicazione col robot

Sono state implementate due funzioni, una per scrivere variabili, *rcWrite*, una per leggere le variabili, *rcRead*. Entrambe le funzioni memorizzano le variabili di interesse in più array di 65 elementi (64 con Windows), che poi vengono mandati al robot tramite le librerie *USB* e le librerie *HID* di Linux.

3.2.2 Inizializzazione robot

Sono stati implementati i metodi per inizializzare la comunicazione e l'accensione del robot nella posizione standard di accensione data dal software *Roboviemaker2* e il metodo per terminare correttamente la comunicazione col robot.

3.2.3 Movimento motori

Per ogni servomotore è stata fatta la funzione che permette dato un certo angolo di muovere il motore di conseguenza; di seguito si riporta l'esempio della funzione che permette di controllare il giunto *pitchAnkleSx*:

```
void UsbRc003::pitchAnkleSx(int degree)
{
  WORD offset;
  offset = degToOffset(degree);
  actualPosition.anPitchL = degree;
  offset += degToOffset(initial.anPitchL)+ANKLE_PITCH_L;
  toRobot1[21] = (BYTE)offset;
  toRobot1[22] = offset>>8;
}
```

3.2.4 Lettura giroscopi e accelerometri

Sono stati implementati i metodi per leggere i valori dei giroscopi e degli accelerometri interni al robot, al fine di avere valori di feedback per migliorare la stabilità della camminata del robot, come è stato fatto da Tavian.

3.3 Aggiunte apportate

Tuttavia per lo scopo di questa tesi è stato neccessario continuare il lavoro di *reverse engineering*, per indagare nuove funzioni delle variabili inviate al robot. In particolare, si è cercato di capire come funziona lo spegnimento del singolo servomotore. Dopo la fase di *sniffing* della comunicazione fra robot e il software *RobovieMaker2*, si è capito come sono codificate le variabili:

- byte 10:
  - bit 0: rollAnlkeDx
  - bit 1: pitchAnkleSx
  - bit 2: pitchKneeDx
  - bit 3: pitchThighDx
  - bit 4: rollThighDx
  - bit 6: rollAnkleSx
  - bit 7: pitchAnkleSx
- byte 11:
  - bit 0: pitchKneeSx
  - bit 1: pitchThighSx
  - bit 2: rollThighSx
  - bit 4: eyeSx
  - bit 7: rollElbowDx
- byte 12:
  - bit 0: rollShoulderDx
  - bit 1: pitchShoulderDx
  - bit 2: eyeDx
  - bit 5: rollElbowSx
  - bit 6: rollShoulderSx
  - bit 7: pitchShoulderSx
- byte 13:
  - bit 2: head

Sono quindi state create le funzioni per lo spegnimento e l'accensione dei singoli motori, e per leggere lo stato di ogni motore del robot.

```
void UsbRc003::setPower( int joint, bool state ){
switch (joint) {
  case (0) :
if (state==false){
          if(pwstate.head==1)
toRobot3[10]+=0x04;
        pwstate.head=0;}
else
          if (pwstate.head==0){
          toRobot3[10]-=0x04;
          pwstate.head=1;}
break;

  case (1) :
if (state==false){
           if(pwstate.shPitchR==1)
toRobot3[9]+=0x02;
        pwstate.shPitchR=0;}
else
           if (pwstate.shPitchR==0){
               toRobot3[9]-=0x02;
               pwstate.shPitchR=1;}
break;

  case (2) :
        if (state==false){
           if(pwstate.shPitchL==1)
toRobot3[9]+=0x80;
        pwstate.shPitchL=0;}
else
           if (pwstate.shPitchL==0){
               toRobot3[9]-=0x80;
               pwstate.shPitchL=1;}
break;

  case (3) :
        if (state==false){
           if(pwstate.shRollR==1)
```

```
toRobot3[9]+=0x01;
        pwstate.shRollR=0;}
else
           if (pwstate.shRollR==0){
               toRobot3[9]-=0x01;
               pwstate.shRollR=1;}
break;

  case (4) :
        if (state==false){
           if(pwstate.shRollL==1)
toRobot3[9]+=0x40;
        pwstate.shRollL=0;}
else
           if (pwstate.shRollL==0){
               toRobot3[9]-=0x40;
               pwstate.shRollL=1;}
break;

  case (5) :
        if (state==false){
           if(pwstate.elRollR==1)
toRobot3[8]+=0x80;
        pwstate.elRollR=0;}
else
           if (pwstate.elRollR==0){
               toRobot3[8]-=0x80;
               pwstate.elRollR=1;}
break;

  case (6) :
        if (state==false){
           if(pwstate.elRollL==1)
toRobot3[9]+=0x20;
        pwstate.elRollL=0;}
else
           if (pwstate.elRollL==0){
               toRobot3[9]-=0x20;
               pwstate.elRollL=1;}
break;
```

```
  case (7) :
        if (state==false){
           if(pwstate.thRollR==1)
toRobot3[7]+=0x10;
        pwstate.thRollR=0;}
else
           if (pwstate.thRollR==0){
               toRobot3[7]-=0x10;
               pwstate.thRollR=1;}
break;

  case (8) :
        if (state==false){
           if(pwstate.thRollL==1)
toRobot3[8]+=0x04;
        pwstate.thRollL=0;}
else
           if (pwstate.thRollL==0){
               toRobot3[8]-=0x04;
               pwstate.thRollL=1;}
break;

  case (9) :
        if (state==false){
           if(pwstate.thPitchR==1)
toRobot3[7]+=0x08;
        pwstate.thPitchR=0;}
else
           if (pwstate.thPitchR==0){
               toRobot3[7]-=0x08;
               pwstate.thPitchR=1;}
break;

  case (10) :
        if (state==false){
           if(pwstate.thPitchL==1)
toRobot3[8]+=0x02;
        pwstate.thPitchL=0;}
else
           if (pwstate.thPitchL==0){
               toRobot3[8]-=0x02;
```

```
               pwstate.thPitchL=1;}
break;

  case (11) :
        if (state==false){
           if(pwstate.knPitchR==1)
toRobot3[7]+=0x04;
        pwstate.knPitchR=0;}
else
           if (pwstate.knPitchR==0){
               toRobot3[7]-=0x04;
               pwstate.knPitchR=1;}
break;

  case (12) :
        if (state==false){
           if(pwstate.knPitchL==1)
toRobot3[8]+=0x01;
        pwstate.knPitchL=0;}
else
           if (pwstate.knPitchL==0){
               toRobot3[8]-=0x01;
               pwstate.knPitchL=1;}
break;

  case (13) :
        if (state==false){
           if(pwstate.anPitchR==1)
toRobot3[7]+=0x02;
        pwstate.anPitchR=0;}
else
           if (pwstate.anPitchR==0){
               toRobot3[7]-=0x02;
               pwstate.anPitchR=1;}
break;

  case (14) :
        if (state==false){
           if(pwstate.anPitchL==1)
toRobot3[7]+=0x80;
        pwstate.anPitchL=0;}
```

```
else
           if (pwstate.anPitchL==0){
               toRobot3[7]-=0x80;
               pwstate.anPitchL=1;}
break;

  case (15) :
        if (state==false){
           if(pwstate.anRollR==1)
toRobot3[7]+=0x01;
        pwstate.anRollR=0;}
else
           if (pwstate.anRollR==0){
               toRobot3[7]-=0x01;
               pwstate.anRollR=1;}
break;

  case (16) :
        if (state==false){
           if(pwstate.anRollL==1){
toRobot3[7]+=0x40;
           }
        pwstate.anRollL=0;}
else
           if (pwstate.anRollL==0){
               toRobot3[7]-=0x40;
               pwstate.anRollL=1;}
break;
   default :
     cout<< "errore funzione setPower" <<endl;
     break;
    }

}
```

Capitolo 4

Struttura del Software

Il software prodotto in questo lavoro è stato svilluppato in linguaggio c++ in ambiente Linux partendo da un lavoro di Fabio dalla Libera [5] e consiste in due parti:

- editazione libera di ogni giunto
- controllo arti inferiori

La prima parte verrà affrontata in questo capitolo, mentre per la seconda si rimanda al capitolo cinque.

4.1 Editazione libera di ogni giunto

Per svillupare questa parte del software si è creata un'interfaccia grafica utilizzando la libreria *Gtkmm* [6] con degli *slider*, uno per ogni giunto del Robovie, e due pulsanti, *Power On* e *Power Off*, che permettono di spegnere e accendere tutti i servomotori del robot, senza però inizializzare o terminare la connessione col pc.

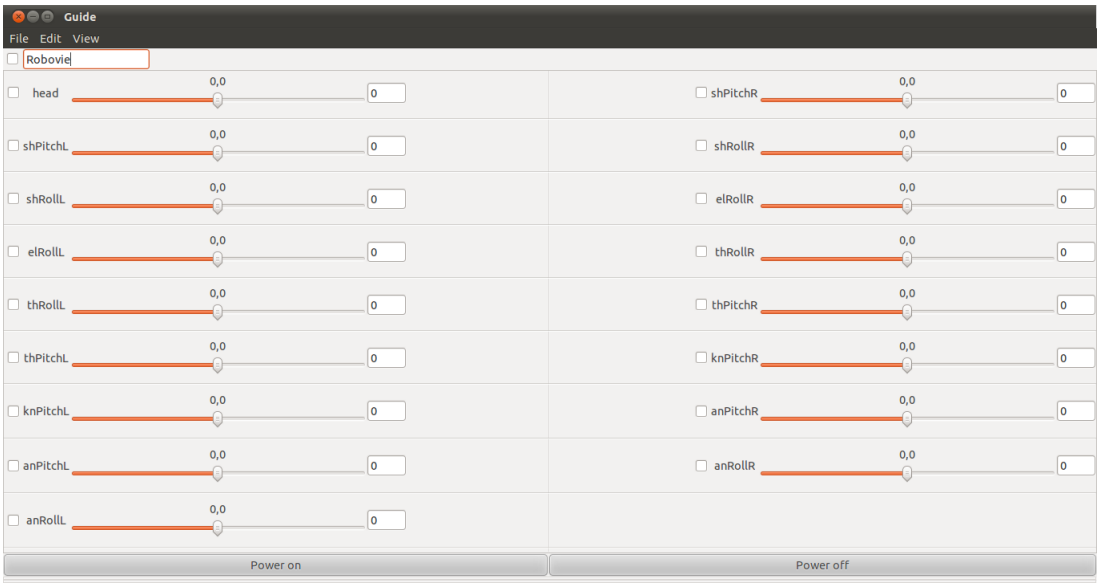

Figura 4.1: Come appare l'interfaccia grafica

Per implementare più facilmente le varie funzioni è stata creata una sovrastruttura per accedere al protocollo del robot, ovvero la classe *remoteMotors*, che permette tramite funzione più generali di comandare agevolmente il robot. Ad esempio la funzione *rotate* permette di interagire direttamente con tutti i giunti del Robovie, senza bisogno di usare le funzioni giunto per giunto.

```
void RemoteMotors::rotate(int j,double v){
if (j!=-1){

        switch ( j){
    case (0) :
       robovie->yawHead(v);
        break;
    case (1) :
        robovie->pitchShoulderDx(v);
         break;
    case (2) :
        robovie->pitchShoulderSx(v);
          break;
    case (3) :
        robovie->rollShoulderDx(v);
         break;
```

```
case (4) :
    robovie->rollShoulderSx(v);
     break;
case (5) :
    robovie->rollElbowDx(v);
     break;
case (6) :
    robovie->rollElbowSx(v);
     break;
case (7) :
    robovie->rollThighDx(v);
     break;
case (8) :
    robovie->rollThighSx(v);
     break;
case (9) :
    robovie->pitchThighDx(v);
     break;
case (10) :
    robovie->pitchThighSx(v);
     break;
case (11) :
    robovie->pitchKneeDx(v);
     break;
case (12) :
    robovie->pitchKneeSx(v);
     break;
case (13) :
    robovie->pitchAnkleDx(v);
     break;
case (14) :
    robovie->pitchAnkleSx(v);
     break;
case (15) :
    robovie->rollAnkleDx(v);
     break;
case (16) :
    robovie->rollAnkleSx(v);
     break;
default :
 cout<< "inizializzazione giunti non riuscita" <<endl;
```

```
   }
  }else{
for (int i=0;i<getMotorNum();i++){
    rotate(i,v);
    }
  }
robovie->doMovements();
```

Si può notare che per fare questo sono stati numerati i giunti del Robovie, secondo la tabella 4.1.

I giunti posso essere editati o tramite lo slider o scrivendo direttamente il valore dell'angolo che si vuole far assumere al servomotore. Il range di inserimento varia da -180° a + 180°, tuttavia sono valori troppo grandi per la maggior parte dei motori del robot, che si devono scontrare con la struttura e la meccanica del robot. Questo però non è un problema, perchè anche in caso di valore troppo elevato, il robot esegue comunque il movimento, shiftando il valore immesso nell'intervallo in cui il motore agisce. Nel caso l'utente scriva un valore fuori dal limite consentito dagli slider il valore viene automaticamente trasformato nel valore massimo possibile più vicino al numero inserito. Il *checkBox* a sinistra di ogni slider serve per spegnere e accendere il singolo servomotore. Questa interfaccia grafica simula le principali funzioni dell'interfaccia grafica del *RobovieMaker2*, creando cosi una solida base per eventuali applicazioni del Robovie-X in ambiente Linux.

Nome giunto	Numero assegnato
head	0
shPitchR	1
shPitchL	2
shRollR	3
shRollL	4
elRollR	5
elRollL	6
thRollR	7
thRollL	8
thPitchR	9
thPitchL	10
knPitchR	11
knPitchL	12
anPitchR	13
anPitchL	14
anRollR	15
anRollL	16

Tabella 4.1: Corrispondenza giunti-numeri

Capitolo 5

Controllo degli arti inferiori

Partendo da uno script *MatLab* di Tavian, si è svilluppata una semplice applicazione per usare il Robovie-X in ambiente didattico. Questa applicazione permette, tramite funzioni di trigonometria, di calcolare i valori degli angoli dei giunti, data la posizione nello spazio della gamba e l'inclinazione del piede. Si viene quindi a creare un'interessante applicazione per effettuare didattica nelle scuole: si può infatti usare il Robovie-X come uno strumento simpatico, divertente e stimolante per dare agli studenti delle scuole superiori un esempio di come quello studiato possa trovare applicazione in ambiti a loro poco conosciuti ma molti interessanti.

5.1 Calcolo degli angoli

Questa parte del software svilluppato si basa su semplici calcoli trigonometrici per ottenere la posizione dei giunti della gamba, dati alcuni dati decisi dall'utente. Per prima cosa è neccessario conoscere la meccanica della gamba del Robovie: essa è costituita da 5 assi di rotazione, due sul piano sagittale e tre sul piano frontale.Non è quindi possibile ruotare il piede sull'asse Z del robot, non avendo assi di rotazione nel piano trasversale, perciò il piede perde un grado di libertà. Date le caratteristiche della gamba (cinque assi, di cui gli ultimi due non sono concorrenti) non vale la condizione sufficiente enunciata dal teorema di Piper, cioè che quando un robot presenta 6 assi, di cui 3 concorrenti un un punto si ha una separazione delle variabili di giunto: l'orientazione è data dai valori dei giunti dei 3 assi concorrenti (che formano così un polso sferico), mentre il posizionamento è ottenuto mediante gli altri 3 giunti. Ma in realtà è possibile applicare un procedimento molto simile, separando le ultime due variabili, che determineranno l'orientamento del piede, dalle tre precedenti, che ne determineranno la posizione nello spazio.

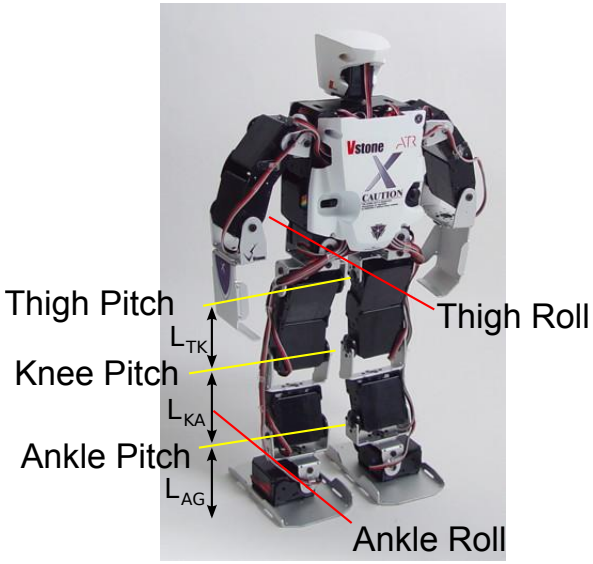

Figura 5.1: Assi di rotazione della gamba del Robovie-X

La posizione e l'orientazione del piede sono descritte quidi da 5 variabili, che sono:

- *dx*: scostamento del piede rispetto al giunto thigh roll lungo l'asse X;
- *dy*: scostamento del piede rispetto al giunto thigh roll lungo l'asse Y;
- *dz*: scostamento del piede rispetto al giunto thigh roll lungo l'asse Z;
- *alpha*: rotazione del piede lungo l'asse X del robot;
- *beta*: rotazione del piede lungo l'asse Y del robot.

Altro dato fondamentale per poter calcolare la posizione dei giunti è la distanza fra i vari assi del robot (con riferimento alla figura 5.1):

- $\overline{T_{roll}T} = 34mm$
- $\overline{TK} = 62mm$
- $\overline{KA} = 60mm$
- $\overline{AG} = 48mm$

Non viene considerata la distanza dell'asse ankle roll a terra, facendo quindi la supposizione che il robot poggi con un segmento dell'asse a terra.

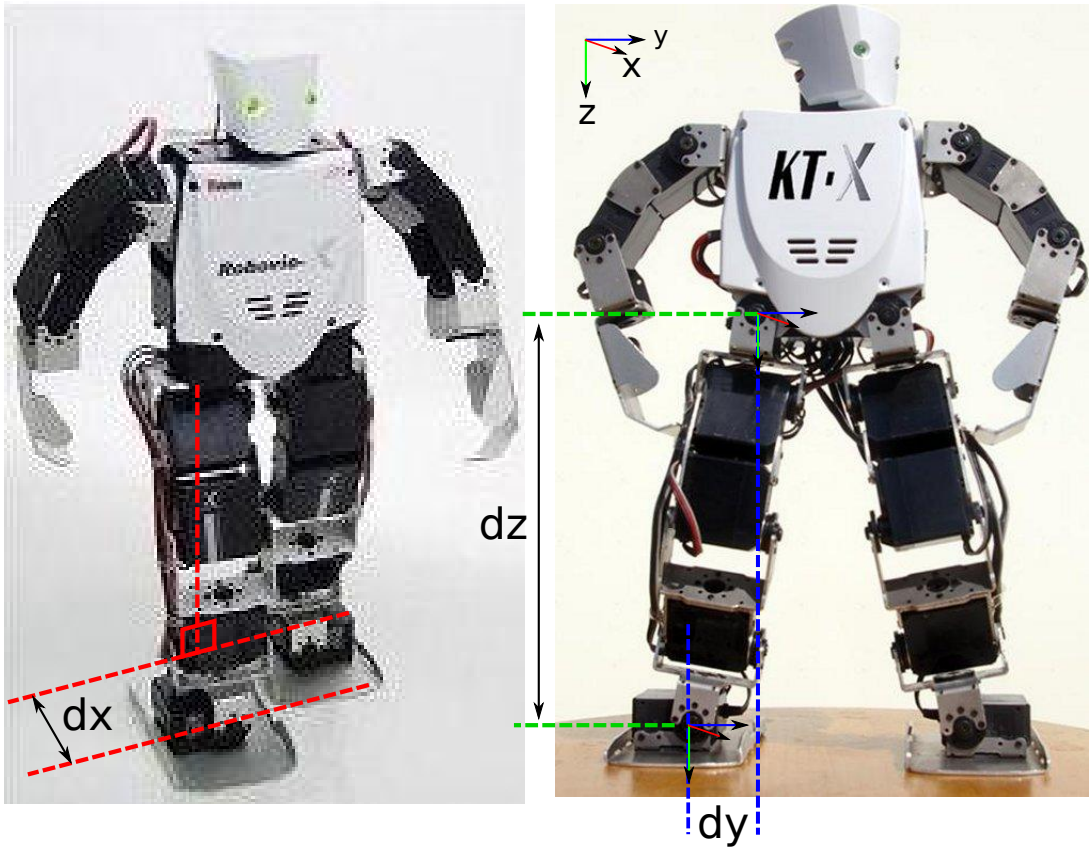

Figura 5.2: Traslazione del piede

5.1.1 Piano frontale

Partendo dal piano frontale, si vogliono calcolare i giunti thigh roll e ankle roll. Per traslare il piede lungo l'asse Y bisogna muovere il giunto thigh roll. È quindi necessario calcolare il valore del giunto per traslare il piede lungo l'asse Y della quantità dy. La lunghezza della proiezione della gamba nel piano frontale si può ottenere facilmente con il *teorema di Pitagora*:

$$L_{front} = \sqrt{dy^2 + dz^2}$$

Per ottenere dunque uno spostamento dy è necessario ruotare il giunto thigh roll di un angolo q_1 tale che: $dy = L_{front} \cdot sin(q_1)$, dunque:

$$q_1 = arcsin(dy/L_{front})$$

Una volta calcolato q_1, sappiamo che la terna di riferimento del piede è ruotata lungo l'asse X di un angolo q_1. Se si vuole portare la terna ad un

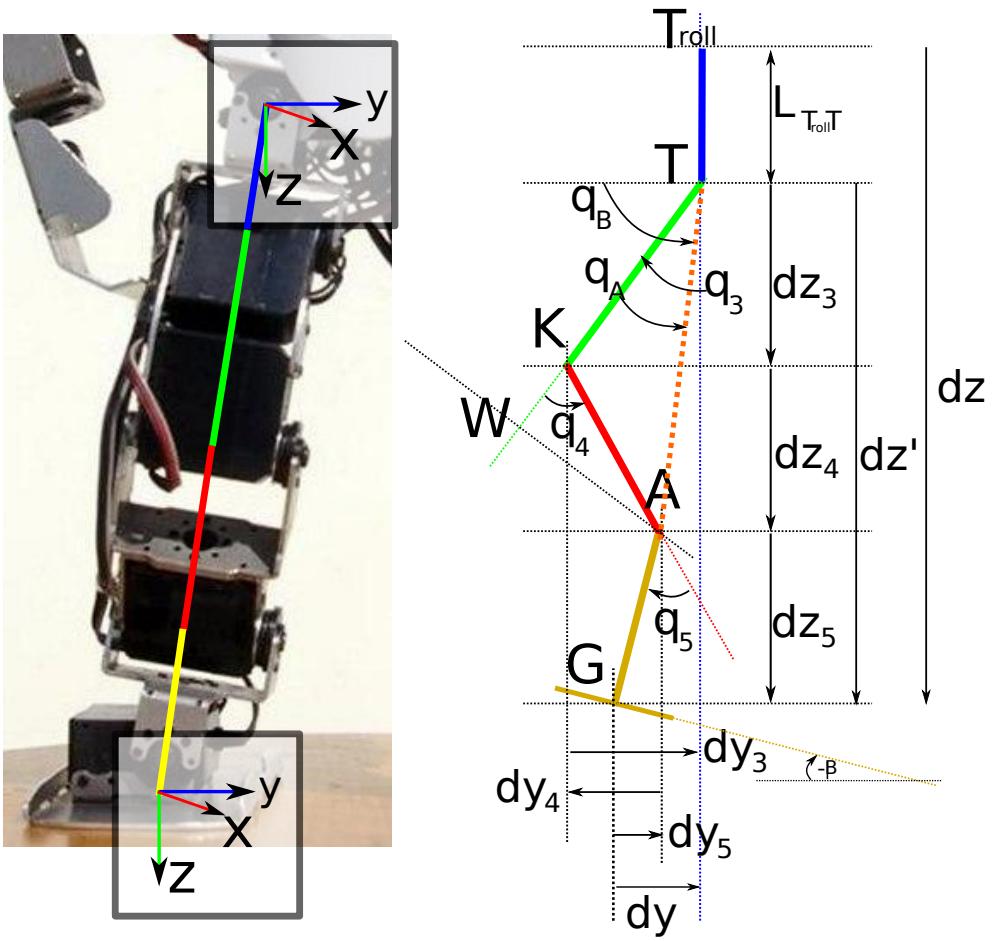

Figura 5.3: a sinistra schema sul piano frontale; a destra sul piano sagittale

angolo α rispetto alla terna di riferimento del busto è necessario ruotare il piede lungo l'asse X di un angolo:

$$q_2 = \alpha - q_1$$

5.1.2 Piano sagittale

Gli angoli dei tre giunti pitch della gamba rimanenti, thigh pitch, knee pitch, ankle pitch si calcolano cosi: bisogna imporre la posizione e la rotazione del piede lungo l'asse Y e poi calcolare la posizione del punto A.

Ruotando il piede di un angolo β il punto A viene a trovarsi nelle coordinate:

$$\begin{cases} A_x = & dx \\ A_y = & -dy + dy_5 & = -dy + L_{AG} \cdot sin(\beta) \\ A_z = & -dz + dz_5 & = -dz + L_{AG} \cdot cos(\beta) \end{cases}$$

Determinato il punto A, dobbiamo trovare q_3 e q_4 in modo che l'estremo del secondo collegamento del robot vada a coincidere con A. Passiamo in coordinate polari sul piano su cui giace la gamba: A si trova ad una distanza $\overline{TA}$, e ad un angolo ϕ.

Determiniamo la distanza $\overline{TA}$:

$$\overline{TA} = \sqrt{(-dy - dy_5)^2 + (dz' - dz_5)^2 + dx^2}$$

con $dz' = -dz + \overline{T_{roll}T}$

Avendo precedentemente fissato q_1, questa distanza dipende esclusivamente dal valore dell'angolo q_4; che può dunque essere determinato attraverso il teorema del coseno:

$$\overline{TA} = \overline{TK}^2 + \overline{KA}^2 - 2\,\overline{TA}\,\overline{TK}\;cos(180 - q_4) \Rightarrow$$

$$\Rightarrow q_4 = \pm acos\left(\frac{\overline{TA}^2 - \overline{TK}^2 - \overline{KA}^2}{2\,\overline{TK}\,\overline{KA}}\right) \qquad [cos(q_4) = -cos(180 - q_4)]$$

E si ottengo due soluzioni; viene presa in considerazione solo la soluzione positiva, che è quella corrispondente alla gamba umana, perché la soluzione negativa avrebbe il ginocchio piegato all'indietro, cosa possibile nel Robovie-X, ma non per una gamba umana.

Calcolata la lunghezza $\overline{TA}$ attraverso q_4, l'estremo del secondo collegamento si può muovere su una circonferenza di raggio $\overline{TA}$ passante per A. bisogna calcolare il valore dell'angolo q_B per il quale il secondo estremo si sovrappone ad A:

$$q_B = atan2(\,-TA_z,\; -TA_y\,)$$

Dove atan2 è un funzione a due argomenti che restituisce l'angolo il cui seno è il primo argomento e il coseno il secondo. L'angolo q_A si calcola nello stesso modo:

$$q_A = atan2(\ \overline{KA}sin(q_4),\ \overline{TK} + \overline{KA}cos(q_4)\ )$$

Facendo attenzione ai segni, si determina il valore dell'angolo q_3:

$$90 - q_3 = q_B - q_a \Rightarrow q_3 = 90 - (q_B - q_A)$$

Infine, si determina l'angolo q_5:

$$-\beta = q_3 - q_4 + q_5 \Rightarrow q_5 = -\beta - q_3 + q_4$$

Per la gamba destra, essendo i motori di "pitch" montati specularmente sul Robovie-X, i valori di q_3,q_4,q_5, vanno presi di segno opposto.

5.2 Interfaccia grafica

Sfruttanto quanto detto nei paragrafi precedenti, è stata creata un'interfaccia grafica per permettere il calcolo degli angoli dei giunti della gamba nascondendo i calcoli descritti precedentemente. Come per l'editazione libera dei giunti del Robovie-X, ci sono cinque slider che permettono di selezionare i valori dx, dy, dz, rx e ry su cui poi verrà effettuato il calcolo della posizione della gamba. Non tutte le combinazioni di valori sono permesse, quindi in caso di posizione della gamba impossibile viene segnalato all'utente che con i dati inseriti non è possibile effettuare il calcolo. I dati possono anche essere inseriti dall'utente digitando direttamente il valore voluto per la determinata variabile. Si nota inoltre che per dz sono accettati anche valori negati, cosa impossibile nelle realtà, per questo prima di iniziare i calcoli viene calcolato il suo valore assoluto, da cui poi si calcola la posizione della gamba del robot. Quindi l'unica variabile che deve per forza essere inizializzata è proprio dz, in quanto il valore 0 non è accettabile come lunghezza della gamba. I pulsanti *legDx* e *legSx* servono per selezionare su quale delle due gambe del robovie-X si vuole effettuare il processo e vanno a modificare una variabile interna al programma. Il pulsante *muovi* serve per far effettuare i movimenti dei giunti, mentre il pulsante *reset* riporta il robot alla posizione di partenza. I pulsanti *Power On* e *Power Off* hanno la stessa funzione che avevano nell'editazione libera dei giunti. Per esempio: se come utente digitiamo come dati in ingresso i seguenti valori per la gamba destra

- dx= -50;

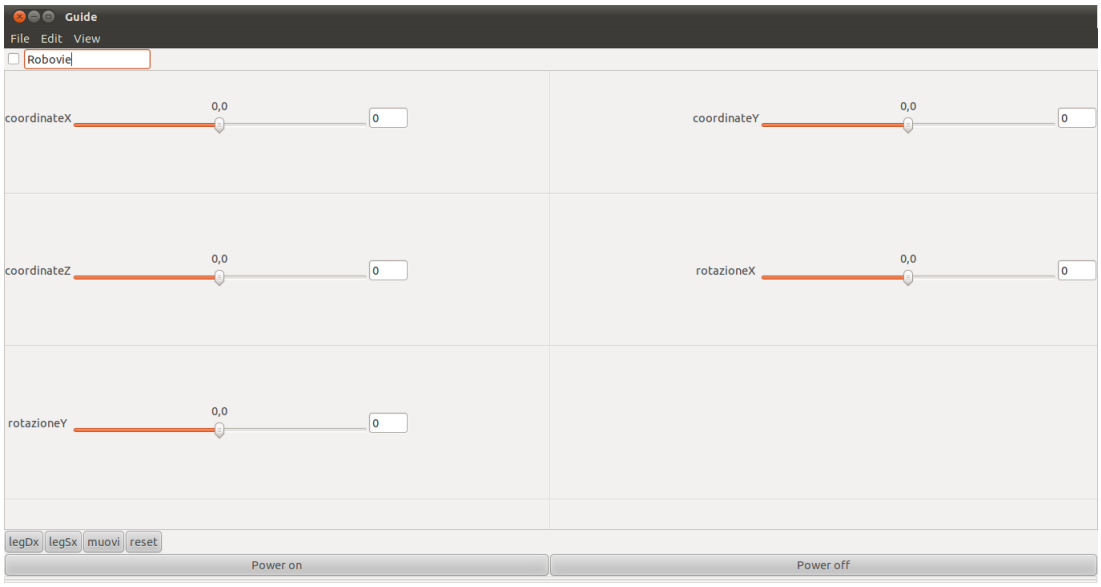

Figura 5.4: Come appare l'interfaccia grafica per il calcolo dei giunti della gamba

- dy= -20;
- dz= 150;
- rx= -5;
- ry= 20;

avremo come valori degli angoli:

- $Thighpitch$= 65;
- $Kneepitch$= 102;
- $Anklepitch$= -17;
- $thighroll$= 9;
- $Ankleroll$= 296;

col il Robovie che si ritrova con la gamba nella posizione voluta:

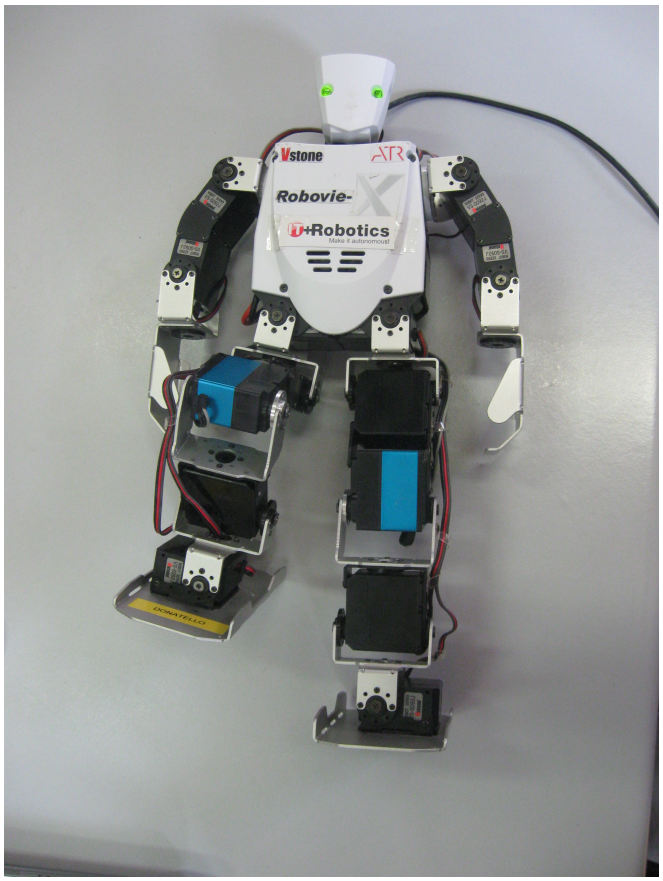

Figura 5.5: Come appare la gamba destra del RobovieX coi dati impostati

Capitolo 6

Conclusioni e Sviluppi Futuri

Partendo dal lavoro svolto nelle tesi precedenti riguardanti il Robovie-X, durante il tirocinio è stata implementata un'interfaccia grafica con l'aggiunta di alcune caratteristiche del protocollo USB usato per la comunicazione tra robot e pc, per simulare il software ufficiale per il Robovie-X *RobovieMaker2*. Questo software permette l'uso del Robovie-X in ambiente Linux, e in particolare per la movimentazione dei singoli giunti, compreso il loro spegnimento e accensione senza bisogno di terminare o inizializzare la comunicazione con il robot. È stato inoltre implementato un semplice esempio di come il Robovie-X possa essere usato a scopo didattico; infatti con la seconda interfaccia grafica è possibile tramite semplici calcoli trigonometrici calcolare il valore degli angoli in cui i giunti devono posizionarsi per assumere la posizione decisa dall'utente tramite l'inserimento dei valori di scostamento e rotazione rispetto alla posizione naturale della gamba. Sono quindi stati raggiunti gli obbiettivi di questo lavoro: creare usa solida base per eventuali future implementazioni di applicazioni sul Robovie-X e al tempo stesso fornire già un software in grado di fare semplici applicazioni su di esso. Per migliorare ulteriormente il software bisognerebbe prima di tutto approfondire maggiormente la conoscenza del protocollo USB di comunicazione tra il robot e il pc, e in particolar modo capire come viene mandata al robot l'istruzione di eseguire il movimento in un determinato periodo di tempo, dato che per adesso i movimenti del Robovie-X sono istantanei. Successivamente si potrebbe quindi implementare la parte di *motion area* presente nel software *RobovieMaker2* e assente nella sua controparte in ambiente Linux. Permettendo così la creazione di movimenti e soprattutto di camminate semplici che possono essere oggetto di future applicazioni a scopo didattico.

Bibliografia

[1] M. Deitel Harvey and J. Deitel Paul, *C++. Fondamenti di programmazione.*
Casa editrice Apogeo, 2005.

[2] R. Bonetto, *Applicativo open source per il controllo del robot RB2000.*
Università degli studi di Padova, 2008.

[3] A. Casasola, *Realizzazione di un traduttore da un linguaggio pseudo-naturale ad un file di comandi per il robot umanoide Robovie-X.*
Università degli studi di Padova, 2010.

[4] C. Tavian, *Algoritmi di stabilizzazione di camminata di robot umanoide.*
Università degli studi di Padova, 2011.

[5] F. Dalla Libera, T. Minato, H. Ishiguro, E. Pagello, and E. Menegatti, *A software toolset for quick humanoid motion prototyping.*
Proceedings of the 4th Workshop on Humanoid Soccer Robots, workshop of the 2009 IEEE-RAS Intl. Conf. on Humanoid Robots (Humanoids 2009), Paris, France, 2009.

[6] *Gtkmm Tutorial.*
http://developer.gnome.org/gtkmm-tutorial/, 2005-2011.

Ringraziamenti

Colgo l'occasione per ringraziare il Prof. Emanuele Menegatti che mi ha seguito nella stesura del presente elaborato, dimostrandosi sempre cordiale e disponibile a fronte delle mie richieste di consigli e di suggerimenti.
Ringrazio i ragazzi di IT+Robotics, in particolare Matteo Finotto, per la disponibilità e la pazienza che mi hanno sempre dimostrato. Un ringraziamento va anche a Stefano e Matteo dell'IASLab, per la simpatia e l'aiuto che mi hanno dato nel proseguire il mio lavoro.
Un grazie di cuore ai miei genitori che hanno creduto nelle mie capacità e che mi hanno sempre sostenuto, sia economicamente che moralmente.
Infine un ringraziamento va ai miei amici e anche ai miei animati, per avermi sempre aiutato ad affrontare i momenti di difficoltà della mia carriera universitaria con incoraggiamenti e sorrisi contagiosi.