2009

Design of a Library of Motion Functions for a Humanoid Robot for a Soccer Game

King Fahd University of Petroleum & Minerals
Computer Engineering Department

COE 485: Senior Design Project

Ahmad Adnan Abu-Arafah
ID# 243432

Supervisor: Dr. Mayez Al-Mouhamed

KFUPM
Department of Computer Engineering
5/16/2009

ABSTRACT

This technical report describes a library of motion functions for a humanoid robot to play a soccer game. The report begins with describing the project in general, the tasks that need to be completed and the constraints involved. A literature survey on the topic of biped robot walking motion is presented along with a review of the hardware manual of the Kondo KHR-1HV robot and the hardware specs of various participating teams in Robocup 2008. This is followed by detailing the design of the library, the options available and the approach that was taken along with a discussion of problems met and the mathematical model developed. Finally, a conclusion will be made about the project and recommendations for future improvements.

ACKNOWLEDGMENT

To my supervisors Dr. Mayez Al-Mouhamed and Dr. Abdulhafid Bouhraoua for their time and effort and for making sure that my senior design project proceeds as planned. I would also like to thank Mr. Zuhair Khayat and Mr. Khaled Al-Hawaj, members of the Group of Interest on Robocup (GIR) for their cooperation and directing me in improving my skills and knowledge.

And to everyone who assisted me in senior design project and in preparing this report, thank you.

TABLE OF CONTENTS

I. INTRODUCTION

The aim of this project is to help establish the KFUPM Robocup team further by developing a library of motion functions that will allow humanoid robot(s) to play a game of soccer (football). Robocup™ (previously the Robot World Cup Initiative) is an international education and research competition that aims to advance robotics and AI research such that by the year 2050, a team of fully autonomous humanoid robots can play and win against the human world champion soccer team. Choosing a soccer game as the standard problem allows for various technologies to be integrated and evaluated, including: robotics, sensor fusion, strategy acquisition, multi-agent collaboration and autonomous agents. Robocup offers a software platform for research as well as the main task of developing a team of numerous fast moving robots playing under a dynamic environment.

The project is on the design of a library of motion functions allowing the Kondo to walk using its left and right legs, moving forward and backward, turning its body while walking and kicking a ball towards the goal. The Kondo KHR-1 is a humanoid robot, having 19 degrees of freedom (DOF) in its legs (6), arms (3) and head (1). The Kondo has a micro-controller allowing the servo control of all of its 19 DOFs. The Kondo Controller has a Serial Interface (RS232C) with a PC from where a user interface is provided, allowing the user to program the configuration of all of its 19 DOFs by setting the value of the motor angles in a specific table, then sending the values to the Kondo using the above mentioned serial communication interface.

The project tasks are:

1. **Task-1:** Review the Kondo mechanics, assembly, user manuals and get familiar with the provided servo interface (described above).

2. **Task-2:** Finding or developing a serial interface between a PC and the Kondo controller so that we can program a specific motion by sequencing (time) a series of angle configurations that are required to generate the desired motion.

3. **Task-3:** Program a library of motion functions allowing the Kondo to (1) walk using its left and right legs, (2) moving forward and backward, (3) turning its body while walking, and (4) kicking the ball towards the goal. It is clear that some variant of the above might be needed.

4. **Task-4:** Testing the above library of motion functions and tuning them to improve performance. Some evaluation is also needed regarding the duration of each motion and its reliability.

Certain constraints such as the performance, stability, acceptable speed of motion as well as reliability of software code and overall design must be met.

II. LITERATURE SURVEY

A. Biped Walking Motion Papers Review

Development of a Humanoid Robot Simulator and Walking Motion Analysis

Noboru Sugiura1, Masaki Takahashi

The researchers in this paper describe a customized humanoid robot simulator based on the Open Dynamics Engine (ODE) which enables them to adjust the ODE's various physics parameters. The ODE like most physics APIs offer various physics functions, i.e. rigid body dynamics, joints between rigid bodies, collision, friction, etc. The simulator was compiled on Visual C++ 2005 and has many functions that control the biped walking motion which implements the Linear Inverted Pendulum algorithm. To analyze the walking motion, the researchers have developed a humanoid robot model, robot motion editor, gyro sensor and inclination sensor model and simulated the walking motion. The following software libraries and environments are free and could be used to develop a simulator:

1) ODE physics API: Developed by Russell Smith
2) 3D-computer graphics: Rendered by OpenGL
3) Compiler: Visual Studio C++ 2005 Express

The model humanoid robot has 20 degrees of freedom, and all bodies are constructed by simple box joints (hinge type) and there exists a polygon view model as well. The total mass is 1.2 kg and the height is 0.38 m. The robot model has many physical parameters which can be edited (Body position, Body rotation, Body mass, Joint position, Joint axis). The simulator supports STL (polygon data format). The robot was designed by using Autodesk Inventor and the polygon data was exported to STL files. It is possible to edit STL data position, rotation and zoom.

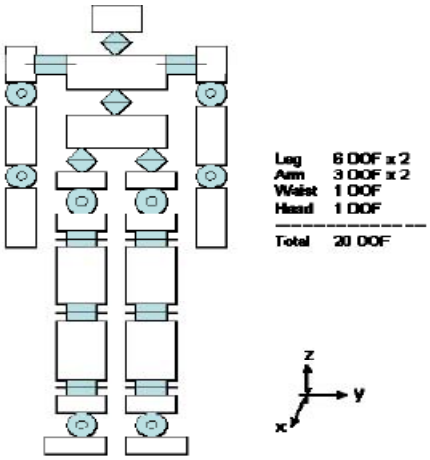


Fig. 1. Robot Model

(a) Box model (b) Polygon Model

Fig. 2. Modeling of Humanoid Robot

A motion editor with functions found in most humanoid robot kits was made with the following specifications:

1) To make a pose, set each joint angle.
2) To make a motion, set the poses and transition time.
3) If the motion is executed, the joint angle values are linearly interpolated between poses.

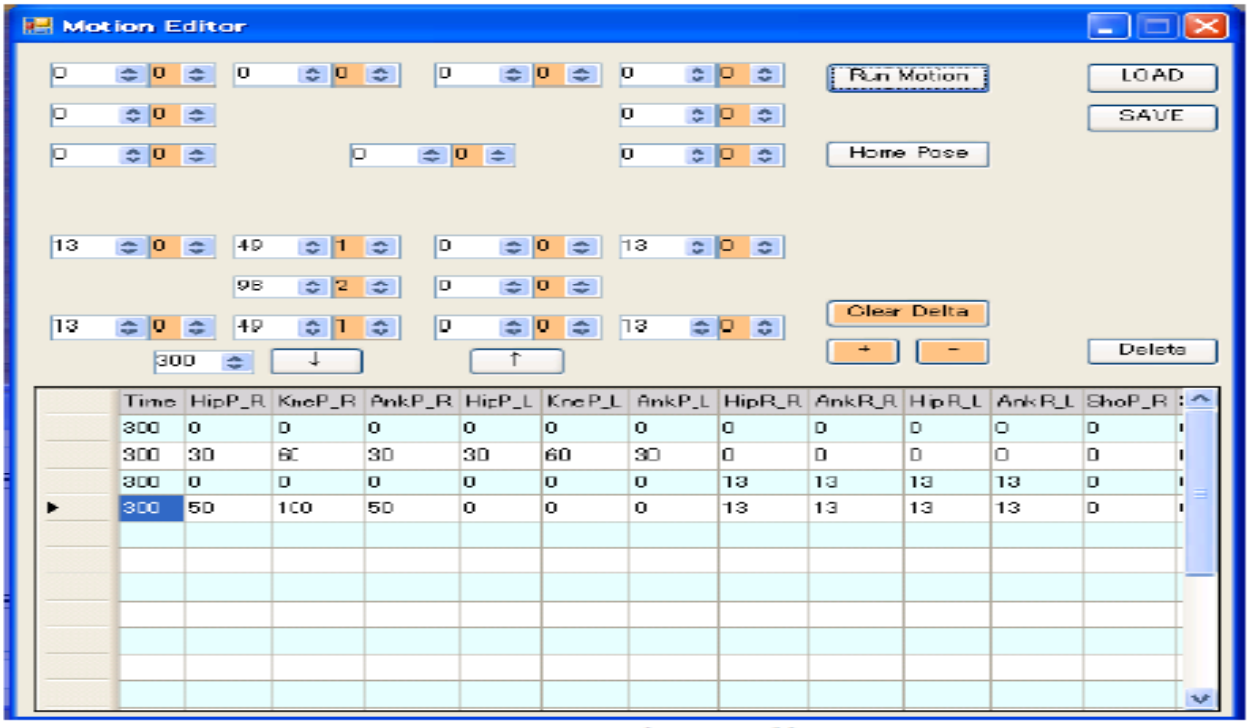


Fig. 3. Motion Editor

In this simulator, it is possible to calculate CG (Center of Gravity) and ZMP (Zero Momentum Point). ZMP is calculated from torque and force at ankle joint. A gyro and an inclination sensor model were implemented for this simulator, and to reduce some noise, both sensor models have a 2nd order low pass filter. Both of these sensors are important to control a robot. A gyro sensor is used for a gyro feedback controller. The controller improves attitude stability. An inclination sensor is used for detecting attitude when a robot is fallen. After detecting falling, the behavior controller changes the control state to stand-up motion.

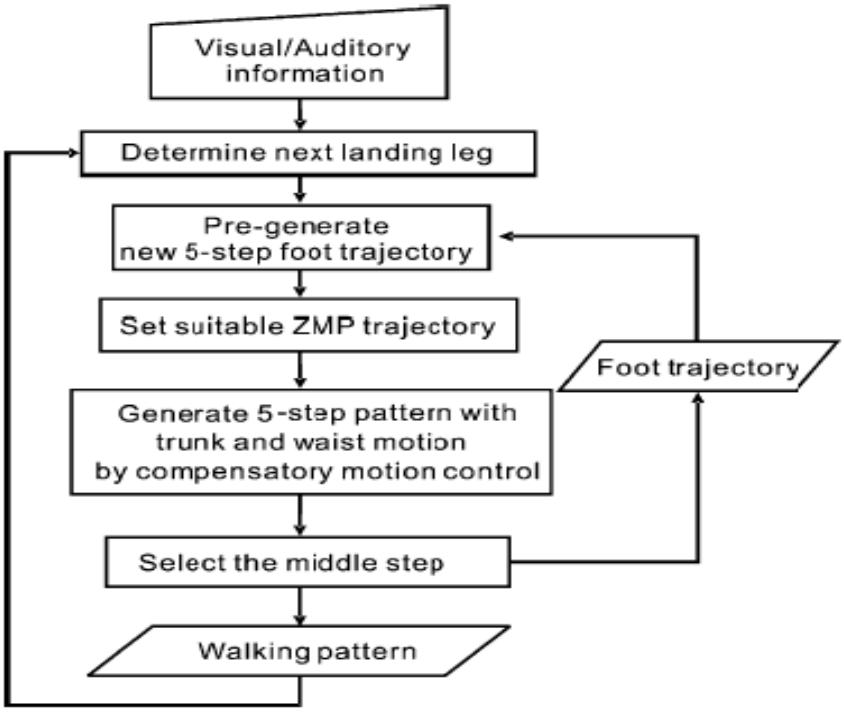
There are various humanoid robot walking algorithms, and the well known Linear Inverted Pendulum was chosen to generate the motion pattern. Most robot kits are programmed based on motion patterns which are series of several pose data. This is a good interface to create various motions. From Linear Inverted Pendulum algorithm the walking motion profiler is derived, and then the profiler was divided into eight points and approximated between points by linear interpolation. In general, a sine wave should be divided into about more than ten points.

Sensory-based walking motion instruction for biped humanoid robot

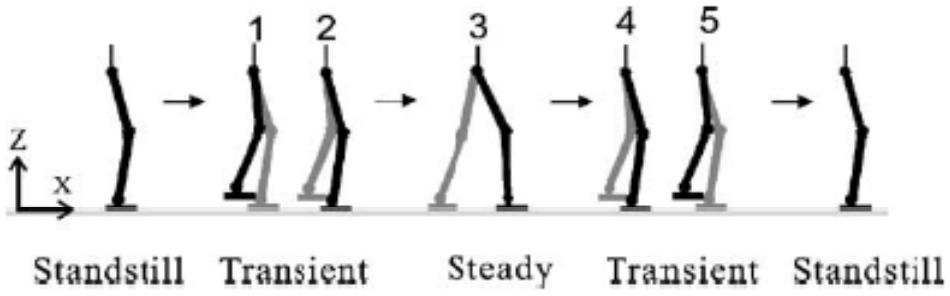
Yu Ogura, Shumpei Ando, Hun-ok Lim, Atsuo Takanishi

The paper describes a sensory-based biped walking motion instruction system where visual and auditory sensors are employed to generate walking patterns according to human orders. The sensors also memorize various walking patterns effectively and systematically. The motion of lower-limbs for locomotion is created by an online pattern generator based on the sensory information. At the same time, the motion of the trunk and the waist for stability is generated online by a balance control method. Combining these locomotive and balance motions, a complete walking pattern is hierarchically constructed and memorized on a database. The walking instruction is conducted through computer simulations.

Realistically, humanoid robots are expected to share the same working environments as humans and in order to achieve this kind of task, one of the main requirements for such biped robots is to have the ability to perform interactive locomotion. Biped robots should be able to generate a walking pattern according to the environments and human commands. The behavior learning method based on Bayesian Networks and the experience of interaction between a human and a robot have been presented; this behavior learning method does not need any prior knowledge of the robot. Therefore, it can be conveniently applied to a human-robot interaction model. However, few researchers have studied biped dynamic locomotion based on interactive technology. The first human-robot interactive walking was reported by the Waseda Biped Humanoid Group. In their work, walking motion patterns were generated offline before walking and saved in the database of the biped robot. A complete walking cycle consists of five phases as shown in the following figure: stationary, transient, steady, transient and stationary phases.



An online pattern generator creates a continuous walking pattern as shown in the figure. First, walking parameters such as step length, height and direction are determined by sensory information are inputted into the pattern generator. Second, the pattern generator creates a five-step pattern of the lower-limb and sets a target Zero Moment Point (ZMP) in the stable polygon. Third, the compensatory motion of the trunk and the waist is calculated by the walking motion control method based on the trajectories of the lower-limbs and the ZMP. Finally, the middle step of the five-step pattern with the trunk and waist pattern is selected as follows:



How a continuous pattern is generated online is detailed in the following steps:

(1) Five steps are made online as soon as the biped humanoid robot begins walking, which are composed of four transient steps and a steady walking step. The pattern generator selects from the first step to the third step.

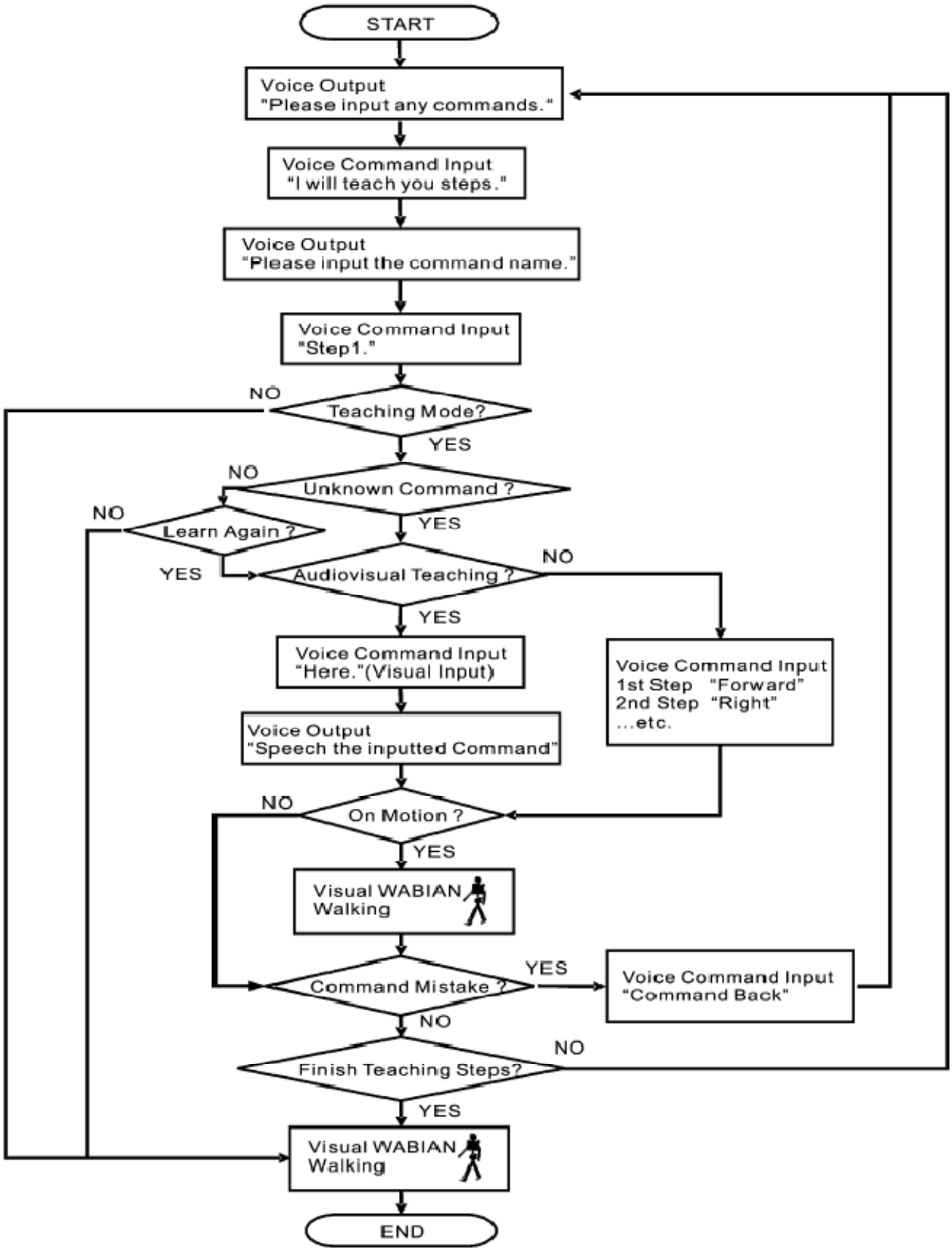
(2) On the second step, three steps (from the fourth step to the fifth step) are generated to satisfy the dynamic conditions of the second and the third lower-limb step. The fourth step is chosen.

(3) On the third step, three steps (from the fifth step to the seventh step) are generated again to satisfy the dynamic conditions of the third and the fourth lower-limb steps. The fifth, sixth and seventh steps are selected.

(4) A continuous walking pattern is generated online, repeating the above procedure.

In order to perform different ways of walking, a robot needs a huge database of walking patterns. However, this is not desirable since the user has to remember all the different ways of walking motion. Moreover, the robot needs to store a large amount of data which consists of redundant information. Therefore, we propose a macroscopic walking pattern, which is able to combine uniform-walking patterns hierarchically. This hierarchical pattern construction allows the biped robot to easily understand different waking patterns. In addition, it is useful if the biped robot is expected to have sufficient intelligence to walk according to human commands and sensory information regarding environments.

For effective walking performance, a biped humanoid robot must be capable of responding to a human command. So, an auditory system was developed to repeat a human voice command as soon as the biped robot hears it.

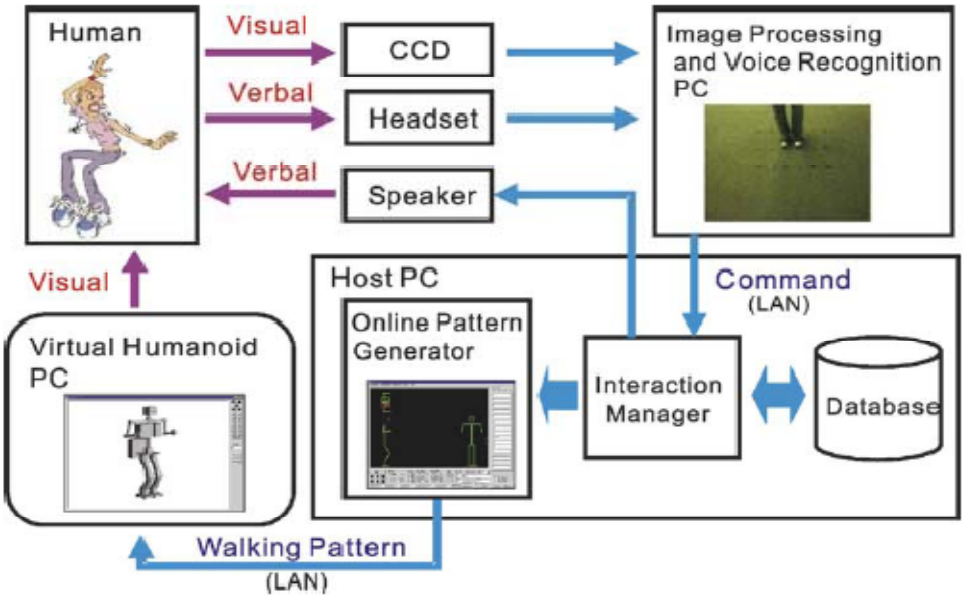

START
Voice Output
"Please input any commands."
Voice Command Input
"I will teach you steps."
Voice Output
"Please input the command name."
Voice Command Input
"Step1."
NO
Teaching Mode?
YES
NO
Unknown Command ?
YES
NO
Learn Again ?
YES
Audiovisual Teaching ?
NO
YES
Voice Command Input
"Here."(Visual Input)
Voice Output
"Speech the inputted Command"
Voice Command Input
1st Step "Forward"
2nd Step "Right"
...etc.
NO
On Motion ?
YES
Visual WABIAN
Walking
YES
Command Mistake ?
Voice Command Input
"Command Back"
NO
NO
Finish Teaching Steps?
YES
Visual WABIAN
Walking
END

This section presents a walking instruction strategy that consists of three different teaching modes: two on-motion modes and one off-motion mode. On-motion mode: the biped robot memorizes the walking pattern while it is walking. There are two sub-modes; a continuous mode and a step-by-step mode. In the continuous mode, a voice command is recognized while the robot walks continuously. For example, when command right 3 is given to the robot, it starts moving according to the Right 3, which is defined as three steps to the right. Then if a new command is ordered such as Forward 4, the robot executes Forward 4 after the previous command is performed.

In the step-by-step mode, only one voice command is registered at a time and a new command is given only after the previous command has been executed. This mode is useful when the robot is in a small space or performs complex walking. For example, when it is asked to walk to the right after one step forward, it would be better to move the right leg directly to the right than to move to the steady position. The off-motion mode is useful for higher layered commands. Voice commands are registered while the robot lifts up and down in the same place. If a higher-level command (e.g. Dance 4) is ordered, the robot performs the pattern of Dance 4 after "Start" is commanded.

The experimental system is divided into three parts: the software simulation system, the auditory system and the vision system. The simulation system (consisting of three PCs) is used to demonstrate the proposed interactive walking instruction methods. The first PC is the Information Processing PC, which processes visual images and recognizes voice information. The second PC is the Host PC, which generates walking patterns online according to the processed information and saves commands on a database. The last PC is a virtual humanoid PC, which shows OpenGL animation according to the generated patterns. The Host PC and others are connected by LANs. The commands along with the walking parameters are sent from the Information Processing PC to the Host, and the walking patterns, which consist of joint angles, are sent from the Host PC to the Virtual humanoid called Virtual WABIAN.

The ViaVoice system was used as the voice recognition engine. The voice input system recognizes only commands that are previously defined by a text file. When vocabularies are recognized, the system is able to generate the related pattern. In this study, walking instructions using visual information are carried out through recognition of human toes. This method utilizes the physical feature of biped humanoid robots effectively. A capture system mounted on the Information Processing PC recognizes visual images through a CCD camera and tracks the color of the index specified by the software.



Experiment results: interactive biped walking instructions through visual and voice recognition has been carried out using the Virtual WABIAN simulation system. A CCD camera is set at the height of about 1.5m from the floor, and a person who is wearing shoes with colored indexes instructs the walking motions in front of the camera. A step, which consists of the position and direction of a swing leg, is taught by visual image recognition and steps are memorized as a macroscopic walking pattern by an instructor's voice command. An instructor teaches the steps using step-by-step mode, seeing the monitor of Virtual WABIAN.

Step 1 teaching time

	Audio (s)	AudioVisual (s)
1	60	47
2	95	58
3	81	61
4	69	74
5	74	59
Average	76	60

The results in the above table show that visual and voice instructions take about 10–20 seconds less than voice only instructions. While voice only instructions require at least two voice commands to teach the simplest step such as the step length and direction, the visual and voice instruction requires the one word "here" to check the tracking point of the visual system. The ViaVoice system needs more than 2–3 seconds to recognize a vocabulary. The greater the complexity of our living environments, the more visual and voice instructions for the biped robots are necessary.

Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point

Shuuji KAJITA, Fumio KANEHIRO, Kenji KANEKO, Kiyoshi FUJIWARA, Kensuke HARADA, Kazuhito YOKOI and Hirohisa HIRUKAWA

The research on biped robot control and walking pattern generation can be classified into two categories. The first group requires the precise knowledge of robot dynamics including mass, location of center of mass (CoM) and inertia of each link to prepare walking patterns. Therefore, it mainly relies on the accuracy of the models. This group is labeled as the ZMP based approach since they often use the zero-moment point (ZMP) for pattern generation and walking control. On the other hand, the second group uses limited knowledge of dynamics, for example the location of total center of mass, total angular momentum, etc. Since the controller knows little about the system structure, this approach relies much on a feedback control. We can label this as the inverted pendulum approach, since they frequently use the inverted pendulum model.

Based on the second group's approach, the authors of this paper proposed a method of walking control and pattern generation by which a dynamic biped walking motion was successfully realized by simulations and experiments. However, since their method generated a stable gait by changing foot placements from their original assignment, it was not applicable to a situation like walking on stepping-stones where the foot must be placed on specified locations. Most of the inverted pendulum based methods suffer with this problem while the ZMP based methods can handle such complex situations. The paper introduces a novel walking pattern generation method that allows arbitrary foot placements as a mixture of the ZMP based and the inverted pendulum based approaches. It is shown that by using the preview controller, we can take into account the precise multi body dynamics even though the method is based on a simple inverted pendulum model.

Walking pattern generation for given a ZMP can be solved as the different three problems:

1) Pattern generation as an inverse problem

2) ZMP control as a servo problem

3) Pattern generation by preview control

A robot can be represented as a cart in the cart-table model and plot the cart motion as the trajectory of the center of mass (CoM) of the robot, which allows for an easy calculation of the ZMP by using the ZMP equations. On the other hand, the walking pattern generation is the inverse problem of this. That is, the cart motion should be calculated from the given ZMP trajectory, which is determined by the desired footholds and step period. Takanishi et al. proposed to solve this problem by using Fourier Transformation. By applying the Fast Fourier Transformation (FFT) to the ZMP reference, the ZMP equations can be solved in the frequency domain. Then the inverse FFT returns the resulted CoM trajectory into the time domain.

Kagami, Nishiwaki et al. proposed a method to solve this problem in the discrete time domain. They showed that the ZMP equation can be discretized as a trinomial expression and can be efficiently solved by an algorithm of complexity *O(N)* for the given reference data of size N. Both methods are proposed as batch processes that use a ZMP reference of certain period to generate the corresponding CoM trajectory. To generate continuous walking patterns for a long period, they must calculate the entire trajectory off-line or must connect the pieces of trajectories calculated from the ZMP reference divided into short segments.

Solving the ZMP control as a servo problem, a new variable u_x is defined as the time derivative of the horizontal acceleration of CoM:

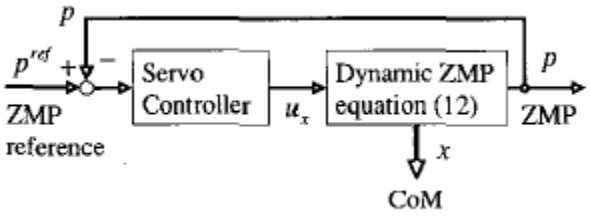
$$\frac{d}{dt}\ddot{x} = u_x$$

Regarding *U*. we can translate the ZMP equation into a strictly proper dynamical system as:

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_x$$

$$p_x = \begin{bmatrix} 1 & 0 & -z_c/g \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}.$$

u_y can be defined and the system can be obtained in the same form. By using the above dynamics, we can construct a walking pattern generator as a ZMP tracking system. This system generates the CoM trajectory such that the resulting ZMP follows the given reference:



However, we must consider a CoM of a robot that walks one step forward dynamically. The robot supports its body by hind-leg from 0s to 1.5s, and has support exchange at 1.5s followed by the foreleg support until 3.0s. Thus the reference ZMP should have a step change at 1.5s and obviously the CoM must start moving before this. Assuming the controller in above figure, the output must be calculated from the future input! Such situation is similar to driving on a winding road, where we steer a car by watching ahead, that is, watching the future reference.

A control that utilizes future information was first proposed by Sheridan in 1966 and was named as the "Preview control". In 1969, Hayase and Ichikawa worked on the same concept and solved a linear quadratic (LQ) optimal servo controller with preview action. A digital version of LQ optimal preview controller was developed by Tomizuka and Rosenthal in 1979 and was completed as the controller for MIMO system by Katayama et.al in 1985.

An optimal preview servo controller following the method proposed by Katayama et al. can be designed. First, we discretize the **system parameter** of the previous equation with sampling time of T as:

$$\begin{aligned} \boldsymbol{x}(k+1) &= A\boldsymbol{x}(k) + B\boldsymbol{u}(k), \\ p(k) &= C\boldsymbol{x}(k), \end{aligned}$$

Where:

$$\begin{aligned} \boldsymbol{x}(k) &\equiv [\ x(kT) \quad \dot{x}(kT) \quad \ddot{x}(kT)\ ]^T, \\ u(k) &\equiv u_x(kT), \\ p(k) &\equiv p_x(kT), \\ A &\equiv \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}, \\ B &\equiv \begin{bmatrix} T^3/6 \\ T^2/2 \\ T \end{bmatrix}, \\ C &\equiv [\ 1 \quad 0 \quad -z_c/g\ ]. \end{aligned}$$

With the given reference of ZMP $p^{ref}(k)$, **the performance index** is specified as:

$$J = \sum_{i=k}^{\infty} \{ Q_e e(i)^2 + \Delta \boldsymbol{x}^T(i) Q_x \Delta \boldsymbol{x}(i) + R \Delta u^2(i) \}$$

Where $e(i) = p(i) - p^{ref}(i)$ is the servo error, Q_e, $R>0$ and Q_x are 3x3 symmetric non-negative definite matrices. $\Delta x(k) = x(k) - x(k-1)$ is the incremental state vector and $\Delta u(k) = u(k) - u(k-1)$ is the incremental input. When the ZMP reference can be previewed for NL step future at every sampling time, the optimal controller which minimizes the performance index is given by:

$$u(k) = -G_i \sum_{i=0}^{k} e(k) - G_x \boldsymbol{x}(k) - \sum_{j=1}^{N_L} G_p(j) p^{ref}(k+j),$$

Where G_i, G_x and $G_p(j)$ are the gains calculated from the weights Q_e, Q_x, R and the system parameter. The preview control is made of three terms, the integral action on the tracking error, the state feedback and the preview action using the future reference. The following figure shows the gain for the preview action. The controller does not need the information of the far future because the magnitude of the preview gain G_p, becomes very small in the future farther than 2 seconds:

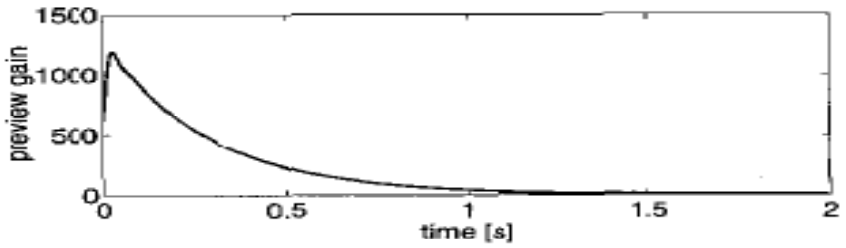


Figure 6: Preview controller gain G_p ($T = 5$[ms], $z_c = 0.814$[m], $Q_e = 1.0$, $Q_x = 0$, $R = 1.0 \times 10^{-6}$)

The next figure is an example of walking pattern generation with a previewing period of 1.6 seconds. The upper graph is the sagittal motion along z-axis and the lower graph is the lateral motion along y-axis. We can see that a smooth trajectory of CoM (dashed line) is generated and that the resulting ZMP (bold line) follows the reference (thin line) with good accuracy. The generated walking pattern corresponds to the walking of three steps forward. The ZMP reference is designed to stay in the center of support foot during single support phase, and to move from an old support foot to a new support foot during double support phase. To obtain a smooth ZMP trajectory in double support, we used cubic spline.

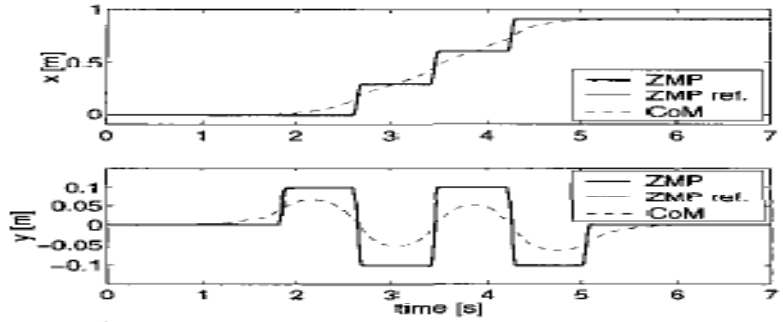
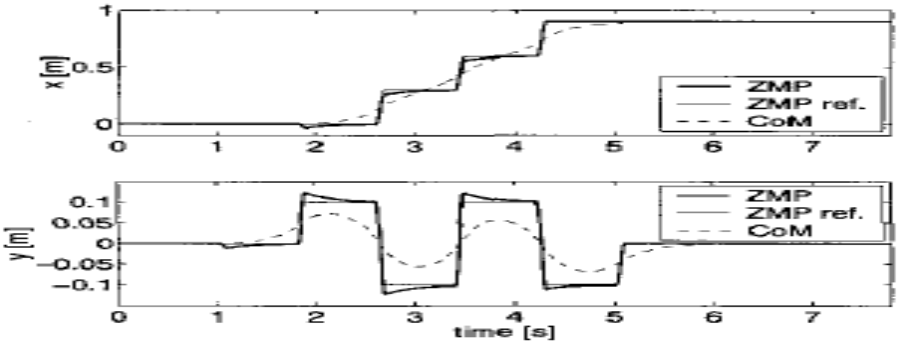


Figure 7: Body trajectory obtained by preview control, previewing period $T * N_L = 1.6(s)$

The last figure is the result generated with a previewing period of 0.8s, which is not sufficient for the ZMP tracking. In this case, the resulted ZMP (bold line) does not follow the reference (thin line) well. We observe undershooting in the sagittal motion and overshooting in the lateral motion. It should be noted that even though ZMP tracking performance is poor, the system still remains stable thanks to the term of the state feedback:



B. Kondo KHR-2HV Hardware Manual

Note: Kondo KHR-1HV's Hardware Manual has not been translated to English at the time this report was written, so the KHR-2HV was chosen being the most similar to the newest model.

Humanoid Robot Kit suitable for ages 10 and up.

Features

- 17 Degree of Freedom
- Metal and plastic construction is both strong and lightweight
- New graphic user interface software (Heart to Heart 3J)
- More than 30 standard pre-programmed actions (Inclusive of Serivce Pack 2)
- Comes With KRC-1 (Kondo Product Number 01110)

Includes

- 17pcs of KRS-788 HV servo motors (10kg.cm @ 0.14 Sec/60 deg)
- Metal and Plastic Brackets , Hardware and Plastic parts
- Battery (9N-300mah) and wall type charger (110V) (240V convertor available upon request)
- CD-ROM containing Software and Manuals (English versions available upon request)
- USB to serial downloading cable and driver

Specifications (General Tolerance of 10mm and 100g applies)

- Height - 353mm
- Width - 189mm
- Fat - 107mm
- Weight - 1270 grams

Tech notes

- Assembly time is approx 4.5 hours
- Requires small and medium size screwdriver

Comments

- Use of 3-cell (11.1V) Lithium Polymer is possible but user is advised to use one with high discharge capability (more than 10A continuous)

Option Parts (Available Separately)

- 01104 ROBO powercell HV D Type 9N-800mah (Ni-mh)
- 01110 KRC-1 Remote Controller (27mhz)
- 01068 KRG-2 Gyro

- 01116 KRG-3 Gryo
- 01076 RAS-1 Acclerometer

Price: $1950.00

Servo Motor: KRS-788HV

KRS-788HV, which is used to drive the joint in this kit, is a digital FET servo that improved KRS- 786ICS. In addition to the know-how which was accumulated with the radio control, by making it a servo applicable to HV, it became a more powerful and energy conservation servo only for a robot than former 6V drives. It can turn for approximately 180 degrees at the maximum. Functions and specifications are as follows:

* The functions of RedVersion are installed. The use of characteristics change and position capturing is possible.

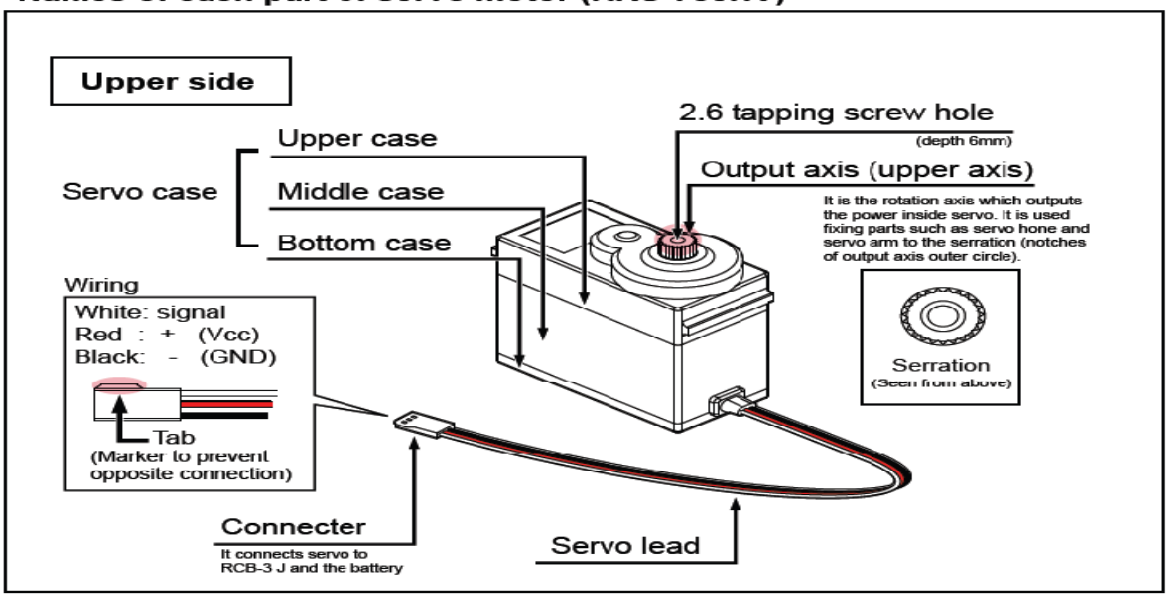
* By the use of ICS, setting from outside is possible.

* Fixing with the support of both axes is possible as a servo only for robots.

* Applicable to HV. It conserves more energy though the specifications are upgraded compared to former products.

Major specifications
External size ---------- 41×35×21(mm) *Without projections
Weight ---------- 47.5g
Torque ---------- 10.0kg/cm (when nicad 9 cell is used)
Speed ---------- 0.14sec/60°(when nicad 9 cell is used)
Proper voltage ---------- 9-12V

Names of each part of servo motor (KRS-788HV)



About RCB-3J:

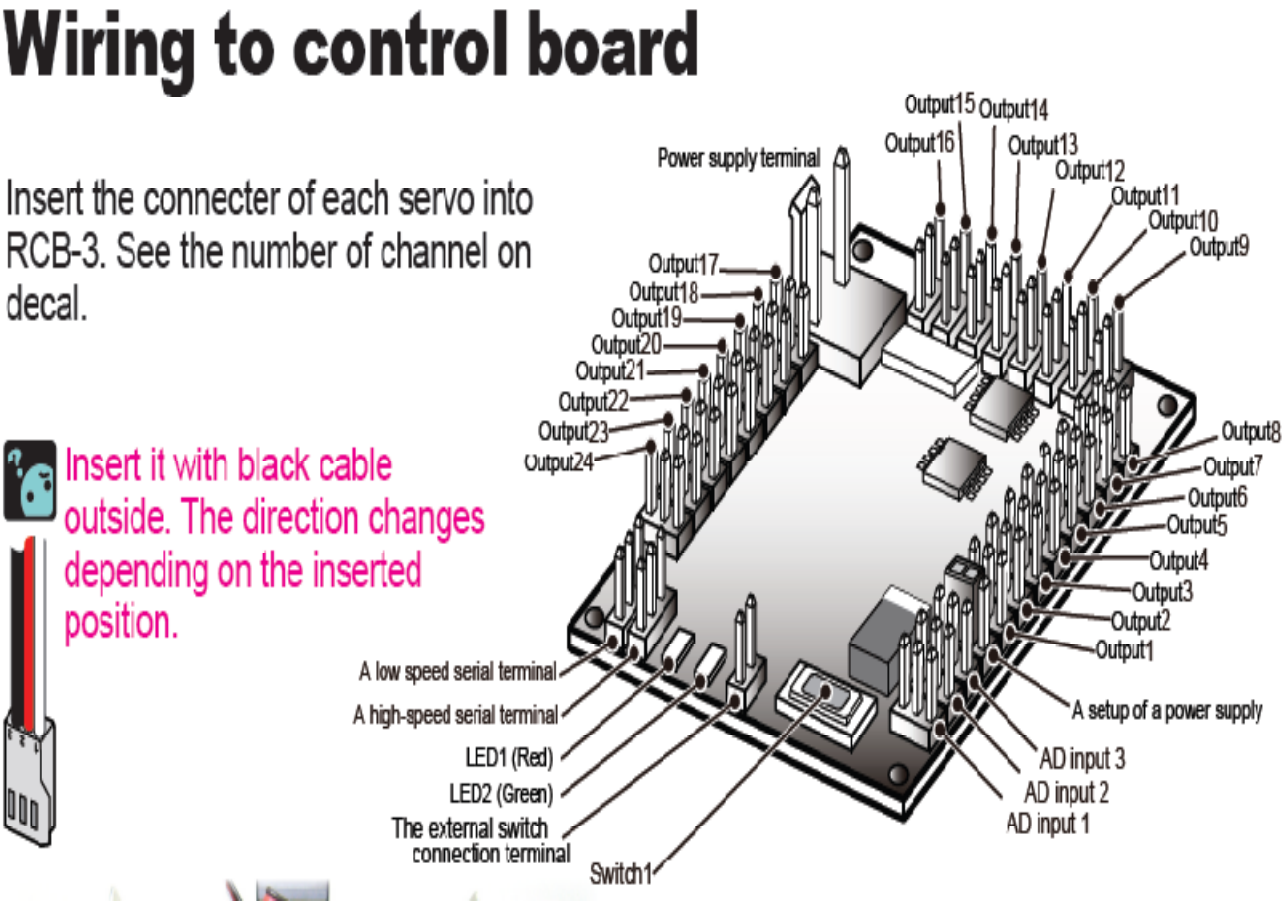
Major functions and specifications:

Size ---------- 45×35(mm)

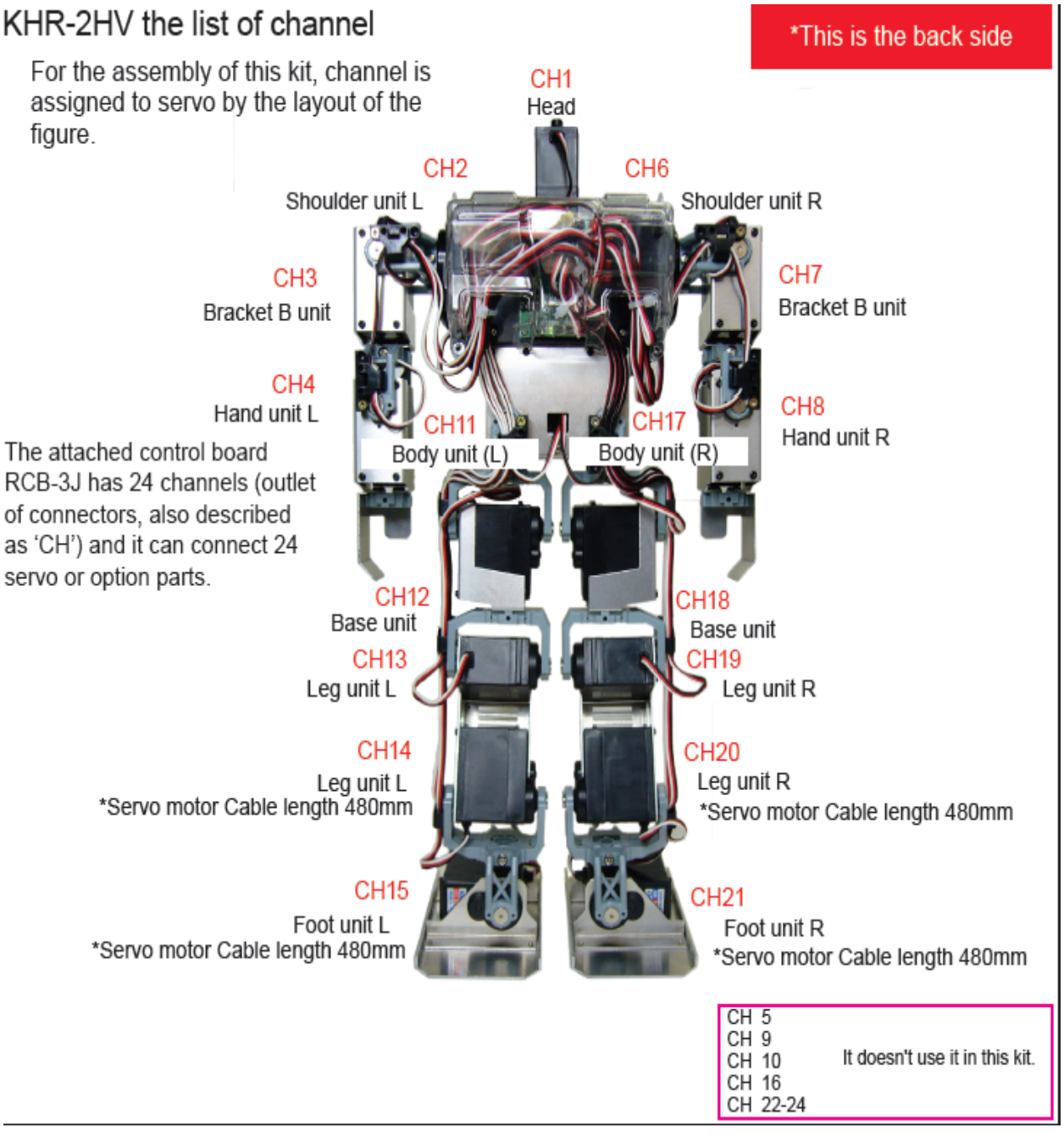
Weight ---------- 12g

The number of servo that can be controlled ---------- 24

Proper voltage ---------- Direct current 9-12V



Backside view of the robot along with the placing of active and passive channels:



C. Robocup 2008 Humanoid League Teams' Hardware Specs

Team	Computing Unit	Vision Unit
B-Human Germany	Fujitsu Siemens Pocket Loox 720 PDA 520MHz Intel XScale PXA272 128 MB RAM 1.3 Megapixel build in Camera with an used resolution of 320 x 240 Windows Mobile 2003 SE	Fujitsu Siemens Pocket Loox 720 Camera, 320x240
Bogobots Mexico	DSPic 30F4013	CMUcam3
DarstadtDribblers Germany	Microcontoller board: Hajime Researsh Institue, 32-bit microcomputer SH2/7145 50 MHz (160 MHz model exists) Onboard PC: DigitalLogic PC/104 AMD Geode LX800 500 MHz Windows CE 5.0/ Linux Wireless LAN, LAN	Philips SPC 1300 NC, 1.3 MP max, 90 fps max, angle 80 degrees
Fumanoid Germany	Central: CM5 containing ATMEGA8 Clocked @ 16 MHz Each Servomotor: ATMEGA8 Clocked @ 16 MHz Vision Module: ATMEGA8 Clocked @ 16 MHz	HaViMo, self developed vision module
CITBrains Japan	Main: Main: Brains Corp. mmEye-PPC (Freescale MPC5200) Sub: Hajime Robot HC3 (Renesas SH-2/7145F)	CCD with super-wide-angle lens
NimbRo Germany (1st Place)	**Computing unit: Sony VAIO UX1XN ultra mobile PC, Intel 1.3GHz Core Solo**	**Three WVGA USB2.0 cameras IDS uEye UI-1226LE, (with wideangle lenses, combined field of view restricted to 180◦)**
RoboPatriots USA	**Main Processor: 600 MHz**	**CMUCam3, 352x288**

(Kondo KHR-1HV)	**Verdex from Gumstix, Inc.** **Microcontroller 1: 16 MHz Robostix from Gumstix, Inc. Microcontroller 2: RCB 3 from Kondo**	**Resolution, 26 FPS**
REJunior Singapore	AMD LX800 500 MHz, 1GB Mem, Storage 4 GB, Interface RS 232, WiFi, USB	640x480 Resolution 30 fps Unspecified USB Camera
Pumas-UNAM Mexico	Micom MR-C3000 with ATmega 128 MPU 8bit RISC at 7.3728Mhz, with 32K flash memory and a 1Gb SD Memory Card.	CMU Cam 3 camera. This camera is equipped with a Philips LPC2106 processor, 64 Kb RAM, 128 Kb ROM, serial port and SD card slot. It can grab RGB and YCrCb images up to 352 by 288 pixels at rates of 26 FPS
SCUT100steps China	PC/104	Logitech pro 5000 Camera
SItiK Japan	"MOYURU" VAIO typeU (SONY: Zero spindle model) Celeron 1.06 GHz IEEE 802.11a “KENSEI-chan the 3rd “ VAIO typeU (SONY: HDD model) Celeron 1.06 GHz IEEE 802.11a "MoYURU the 2nd" VAIO typeU (SONY: Zero spindle model) Core Solo 1.20 GHz IEEE802.11a	"MOYURU" QVX-13N (Logicool) Resolution: 640x480 Color Space: YUV Frame Rate: 4 fps “KENSEI-chan the 3rd “ QVX-13N (Logicool) Resolution: 640x480 Color Space: YUV Frame Rate: 4 fps "MoYURU the 2nd" FFMV-03MTC-CS (Point Grey Research Inc.) 2 Cameras Resolution :640 x 480 Color Space: YUV, HSV FPS: 60, 30, 15, 7.5 Angle: Horizon 76.1, Vertical 60.8
TH-MOS China	Pocket PC, Windows Mobile	Model MOS 2007:

	5.0 and ARM processor at 400 MHz, WLAN	Logitech QuickCam 5000 Resolution 320x240 Color SpaceRGB 15 FPS Model MOS_III: CMU CMOS cam Resolution: 160×255 Color Space: RGB 50 FPS
TKU Taiwan	Sunplus-DSP 32.768 MHz Altera-NIOS 80 Mhz	CMOS Sensor, 160x120 Resolution
Team Jeap Japan	Main Controller: CPU PNM-SG3 ROM 4GB (Flash HDD) RAM 512 MB OS Linux Sub Controller: CPU VS-RC003 ARM (LPC 2148) ROM 512 KB RAM 40 KB OS HMDP RealTime MotionMachine	Quickcam
Team KMUTT	"Pawdee" Vision Board and Main PC * Manufacturer: Advantech * Processor: Geode LX-800 for PCM3353 (PC-104 format) * Speed: 500 MHz Motor controller * Manufacturer: Philips LPC-2138 * Processor: ARM7 TDMI-S * Speed: 60MHz "KM-1" Processing boards used:	"Pawdee" Manufacturer: Logitech Quickcam pro 4000 Specification: 12 frames per sec., 320 X 640 resolutions "KM-1" Manufacturer: Logitech Quickcam Pro9000 Specification: 15 frames per sec., 320 X 640 resolutions

	Vision Board and Main PC * Manufacturer: Advantech * Processor: Geode LX-800 for PCM3353 (PC-104 format) * Speed: 500 MHz * OS: LINUX Motor controller * Manufacturer: Philips LPC-2138 * Processor: ARM7 TDMI-S * Speed: 60MHz	
Team ROPE	"RO-PE-V" "RO-PE-VI" Kontron MOPSlcd7 Mobile Intel Pentium III CPU Speed: 700MHz	"RO-PE-V" "RO-PE-VI" Camera KPC-S700CP1 1/3" CCD / 480 TV Lines
Team Spirit Korea	Celeron-M 600 MHz ×1 TMS320F2810 × 1	CMOS Camera(Logictech Messenger) × 2
Team Uchile RoadRunners Chile	Fujitsu Siemens Pocket Loox N560 Intel® PXA270 624MHz 64MB RAM 128MB ROM	Philips ToUCam III – SPC900NC
TsinGhua Hephaestus China	PC 104	Logitech QuickCam Proo 4000 CCD Camera
Twobots Brainstormers Germany	Gumstix System on a Chip (SoC) with 600MHz, USB host, WLan and 128MB RAM	Stripped Quickcam 3000 Pro USB camera with modified spyhole lens providing a 180° fish-eye view
U of M Humanoids Canada	200 MHz Arm (Nokia 5500 phone) communication with micro-controller via custom designed IRDA infrared	CMOS Sensor 320x240
Victor TangoUSA	"DARwIn IIa" PC104+ Intel Pentium M Dothan 1.4Ghz CPU Board "DARwIN III" PC104+ Intel Pentium M Dothan 1.4Ghz CPU Board MicroController Board	"DARwIn IIa" "DARwIN III" 2 x Unibrain Fire-I IEEE 1394
ZJUDancer China	Mega128 & TI	CCD Camera

	TMS320DM642TM DSP	
Team Osaka Japan (2nd place)	**Main Controller:** **CPU: GeodeLX 800** **Rom: 4 GB** **Ram: 512 MB** **OS: Windows XP** **Interface: Storage device slot (CF card) x1, USB2.0 x 4, UART (RS232C) x 4, Analog video capture device x 2, Wireless network device IEEE802.11a x1, RGB display output x1, Speaker (1W) x2, Mic x1** **Sub Controller:** **CPU: LPC2148FBD64** **ROM 512 KB** **RAM: 64 KB** **Interface: UART x2, 10 bit AD converter x8, GPIO, Input capture, IXBUS x1** **Actuator Controller:** **CPU: C8051F411** **ROM: 32KB** **RAM: 2KB** **Interface: UART x1, 12 bit AD converter**	**Image sensor x2, 33 MP, 60FPS**
Persia Iran	Gumstix or CMU3 & microntroller (ICB: AVR from Atmel)	Webcam or CMU3
Parsargad Iran	Gumstix	CMOS webcam
Tsinghua Hrimthurs China	PXA 270 with Intel XScale 520 Mhz CPU	2x BOYI web cameras, resolution 320x240 pixel, up to 15 fps, white balance is automatic

III. DESIGN & TESTING:

A. Design Options

1. GIR RS232

We started in the beginning of the semester 081 to try and figure out the protocol that is used to communicate with the RCB-3 (Kondo Robot Controller Board) from the Heart to Heart program, which is a propriety software provided by Kondo. GIR members Zuhair Khayat and Khalid Al-Hawaj succeeded using an original electrical circuit to tap the communication between the microcontroller and the serial USB adapter and established a good, working knowledge of the protocol. We were able to move the head of the robot using a simple program made in Visual Basic that communicated with the microcontroller through RS232 (made by Khalid). Unfortunately, the interface could not be completed in the timeframe allowed for this project.

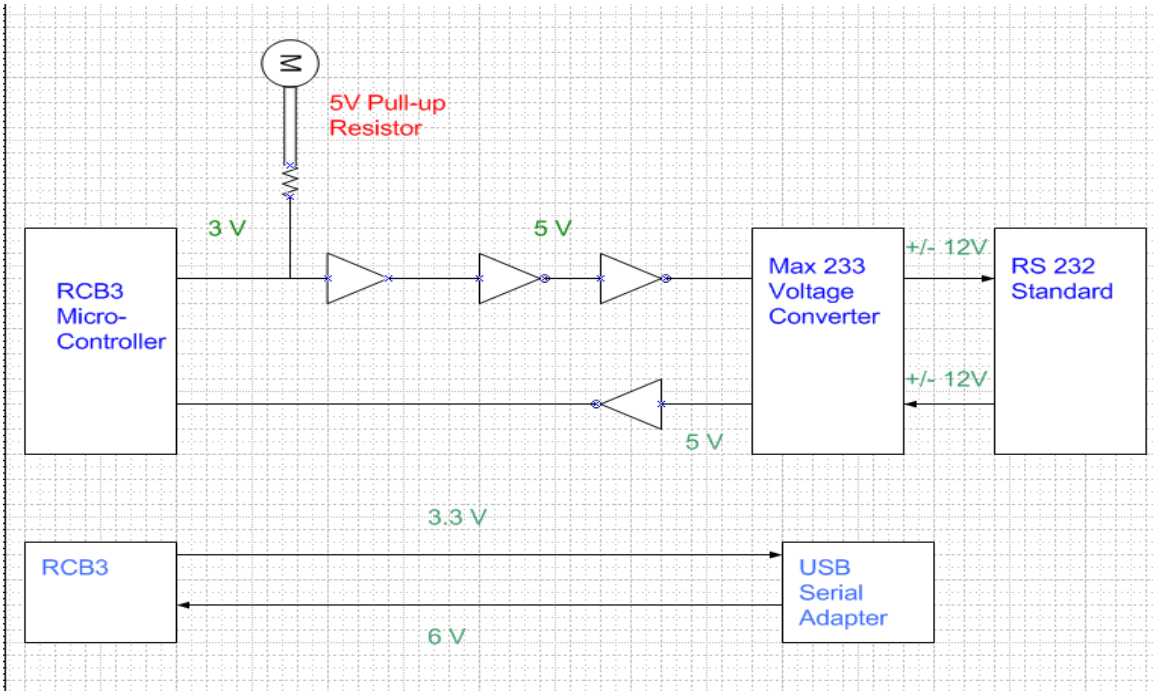


Figure: Overview of electrical circuitry used to tap into the RCB3-J

Overall, these are the main issues discussed and achievements accomplished in our meetings during 081:

Achievements:

1. Tapping the robot's serial communication and understanding protocols and commands used to program the robot.

2. Controlling the robot using VB program and RS232 without using H2H tool.

Problems:

1. Communication problem between the PC's serial port and the robot's serial port. It is possible that it is an electrical problem (solved afterwards).

2. Image capturing is a data extensive operation. It is going to take long time to capture an image and transfer it to PC if we used normal serial communication. To solve this problem, we could use i2c interface as Khalid suggested. The rabbit supports i2c interface but it is not confirmed that CMUcam can use i2c interface. The maximum data rate of i2c interface can reach up to 3.4 Mbps.

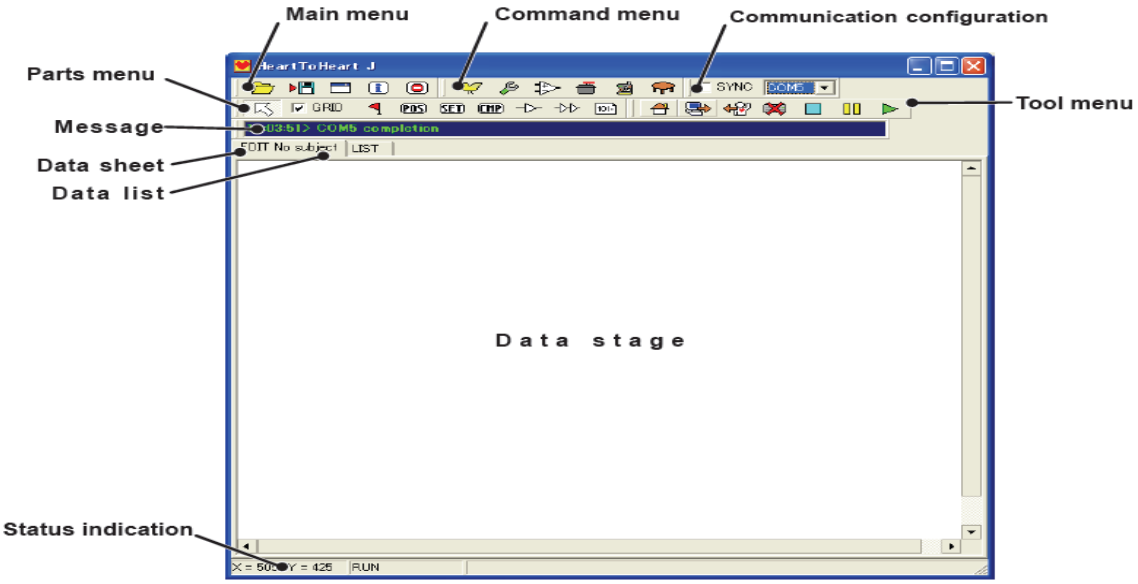
2. HeartToHeart3J

The HeartToHeart3J is the native serial interface and motion mixer that is provided with the Kondo robot kit. It communicates with the microcontroller board through the USB serial adapter. Among its many advantages, it has a user friendly GUI, learning how to use the software is easy and there are many customization options allowed in the program. Through the interface, you can make a total of 80 motions (30 positions maximum for each motion) and 5 scenarios (e.g. walking than turning left). You can edit the contents of the microcontroller memory, adjust the gyro sensors and analog input, enter the value of angles for each position manually or by capturing the values after adjusting the robot physically (teaching mode). The files saved and loaded have the extension .RCB. The collections of positions are connected through wires, and you can make conditional branches and loops as well. The main disadvantage however is that the robot must be connected through the USB adapter when using this program, which makes adjusting the motions very difficult. The following is a preview of the interface:

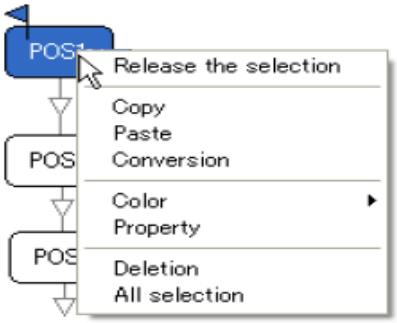
Data stage:

A data stage has a function like canvas. Editing actual data is done by double-clicking on the arranged parts. The tab above the sheet is used to switch the data sheet into a list. The list if the numerical value of arranged parts which are compiled. The name of the item being edited is indicated in the tab of the data sheet:

The start of software



Objects on the Data Sheet: a menu would be indicated when you right-click the each parts arranged on data sheet. You can set up the name or the color of item arranged by selecting property and so on:



Tools Menu: 

1) A setup of a trim adjustment option: It adjusts a home position of each servo corresponding with the 24 output terminals.

2) Set up of options: Lists the various setups that can be used RCB-3J.

3) Analog configuration: There are 3 systems of analog input in the RCB-3J. This program sets up a movement from the output of the sensor connected to this terminal. There are 2 kinds of movements by analog input:

A. Real Time Mixing: It is the way to add the specified magnification of the input signal to the output terminal. It is used, for example, when you use a gyro sensor.

B. Motion Interruption: It distinguishes the value of the signal and executes the motions set up. For example, it could be used in the situation where automatically picking the body up is composed of using the acceleration sensor.

4) ICS setting

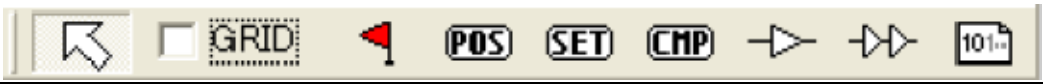
5) Receiver button: In this window you can monitor a signal from the receiver when a transmitter is used. Though, in case of using a usual button-type (command) transmitter, previously the 2 byte signal was functioned as a command, in RCB-3J however, it is expanded more, and all analog quantities can be allotted to 1 byte signal.

6) Data table indication: The list of the motions or the scenarios in the internal memory of RCB-3J can be displayed by executing reading (using an icon in screen) the data table at present. RCB-3J does not only have data but also the name of data, date of last update, etc.

Communication Configuration: 

1) Synchronization Switch: the position of the servo being adjusted on software can affect the RCB-3J in real time. In case the switch is turned off (when the checkbox is empty), the changes could be sent to the RCB-3J if you push the transmission button after the modifications.

2) Communication Port: Here you can choose the communication port. A serial USB adapter is recognized as a virtual serial port on the personal computer. It can't communicate with RCB-3J if you choose the wrong COM number.

Parts menu: 

1) Choice Tool: It is the icon used to choose parts or moving objects around the data canvas.

2) Grid: when a check is in, a grid becomes effective, and each part would be arranged onto the grid.

3) Indication of a starting point: You can specify the start point (what is carried out as the first position) of each motion. When more than one motion exists on the data sheet, only the motions having "a starting point" are actually executed.

4) Position

5) Setup value arrangement (SET object): you can set up the change of the mixing setup, counter (the number of repetitions) in the case of a loop.

6) Compare tool: the setup of branching is done using this part. All the setups are indicated as "if true, jump". A value is established with a SET object, and a loop counter is subtracted with CMP. A comparative register is the value set up with a SET object, too. It jumps if the setup condition is satisfied. The branching connection wiring part listed below would be used for this jump.

7) Connection wiring: it would be used to connect the positions arranged onto the data sheet. You can make continuous motions by connecting the positions that are arranged separately.

8) Branching connection wiring: the starting point would always be a branching setup (CMP), which is different from ordinary wiring. When a condition is satisfied, the point united by branching connection wiring would be executed.

9) Compilation tool: the motion data would be sent as binary values when it is sent to the RCB-3J. When you click on the compilation tool, the present data would be compiled, and would be indicated in the data list.

Command menu: 

1) Home position: when this button is pressed, the condition of RCB-3J is returned to the home position set up at present.

2) Writing: write the contents of data arranged on the present data sheet to the RCB-3J. Specify the number of motion or scenario when the dialog box of "data writing" opens. Writing would be done by clicking the OK button on the dialog box.

3) Reading: this reads and indicates the motion of RCB-3J or the contents of the scenario.
As the dialog box of "data reading" shows up, specify the number of motion or scenario to be read and then push OK button, and reading is done.

4) Delete: this deletes (or eliminates) the motion or a scenario written in RCB-3J.
When a data deletion dialog is indicated, select the motion or a scenario to be deleted and then push the OK button, and deletion is done.

5) Stop: this stops a motion or a scenario being regenerated.

6) Pause: causes a motion or a scenario being played back to pause.

7) Start: push the start button again to stop the pausing. When clicked, the dialog box of the regenerated number designation opens. Specify the number and push OK button, and the motion of the number specified is play backed.

How make a simple motion:

1) Connect the serial USB adapter to the computer and the RCB-3J.

2) Select and place a position on the data canvas.

3) Double click on the position object and change the values of each servo according to the first motion desired.

4) Repeat the above step for each position until the motion is complete.

5) Place the start flag on the position you desire to begin first.

6) Wire the different positions together.

7) Write the data into the selected memory slot of the microcontroller.

8) Play back the motion and adjust accordingly.

The following is a description of the channels that control the Kondo KHR-1HV servos:

Channel	Representation	Negative Angle	Positive Angle	Mirror Channel
Ch 1	Head	Left	Right	----------
Ch 2	Left Shoulder	Down Rotation	Up Rotation	Ch 6
Ch 3	Left Arm	Detract	Extend	Ch 7
Ch 4	Left Elbow	Bend Up	Bend Down	Ch 8
Ch 6	Right Shoulder	Up Rotation	Down Rotation	Ch 2
Ch 7	Right Arm	Extend	Detract	Ch 3
Ch 8	Right Elbow	Bend Up	Bend Down	Ch 4
Ch 10	Left Hip	Clockwise	Anti-clockwise	Ch 16
Ch 11	Left Tilt	Left	Right	Ch 17
Ch 12	Left Leg	Down	Up	Ch 18
Ch 13	Left Knee	Down	Up	Ch 19

Ch 14	Left Foot	Up	Down	Ch 20
Ch 15	Left Ankle	Rotate Right	Rotate Left	Ch 21
Ch 16	Right Hip	Clockwise	Anti-clockwise	Ch 10
Ch 17	Right Tilt	Left	Right	Ch 11
Ch 18	Right Leg	Up	Down	Ch 12
Ch 19	Right Knee	Up	Down	Ch 13
Ch 20	Right Foot	Down	Up	Ch 14
Ch 21	Right Ankle	Rotate Right	Rotate Left	Ch 15

3. Laurent C Library and Button Box

Laurent Lessieux (software developer) developed a library that allows any C/C++ developer to communicate with any RCB3 driven robot using the serial link from a PC running under the Windows OS. I downloaded the 1.05 version to study its main features and to test its various functions while attempting to control the Kondo KHR-1HV robot by using a simple program.

Reported Features:

1. Small DLL that will enable any C/C++ developer to communicate with an RCB3 driven robot.

2. Control multiple robots in one program in the event you have more than one connected (e.g. Bluetooth) since it is necessary to specify the ports.

3. Parameters are checked and appropriate errors are reported in case of a violation of the contract (mainly the RCB3 Command References in the Japanese Version).

4. Any exception caused by invalid pointers passed will also be caught and reported.

5. All RCB3 documented functions are implemented

How to use:

To access the robot using this library, you must follow this sequence:

1) Create an RCB3 Interface using this command:

```
bool CreateRCB3Interface(int in_comPort,int in_comSpeed,int in_model,UINT
&out_rcb3Interface);
```

The comSpeed should always be 115200. The model should be one of the two currently supported (RCB3_MODEL_NORMAL or RCB3_MODEL_J). The out_rcb3Interface is the handle that will describe your new interface. From now on, this number will be passed to all functions. This function will try to open the COM port and communicate with the RCB3 to see if it is connected. If the creation failed, take a look at the Error message to see what went wrong.

2) Send a command

3) Check the boolean value for failure/success

4) Send other commands as needed

5) Destroy the RCB3 Interface using this command:

```
bool DestroyRCB3Interface(UINT in_rcb3Interface);
```

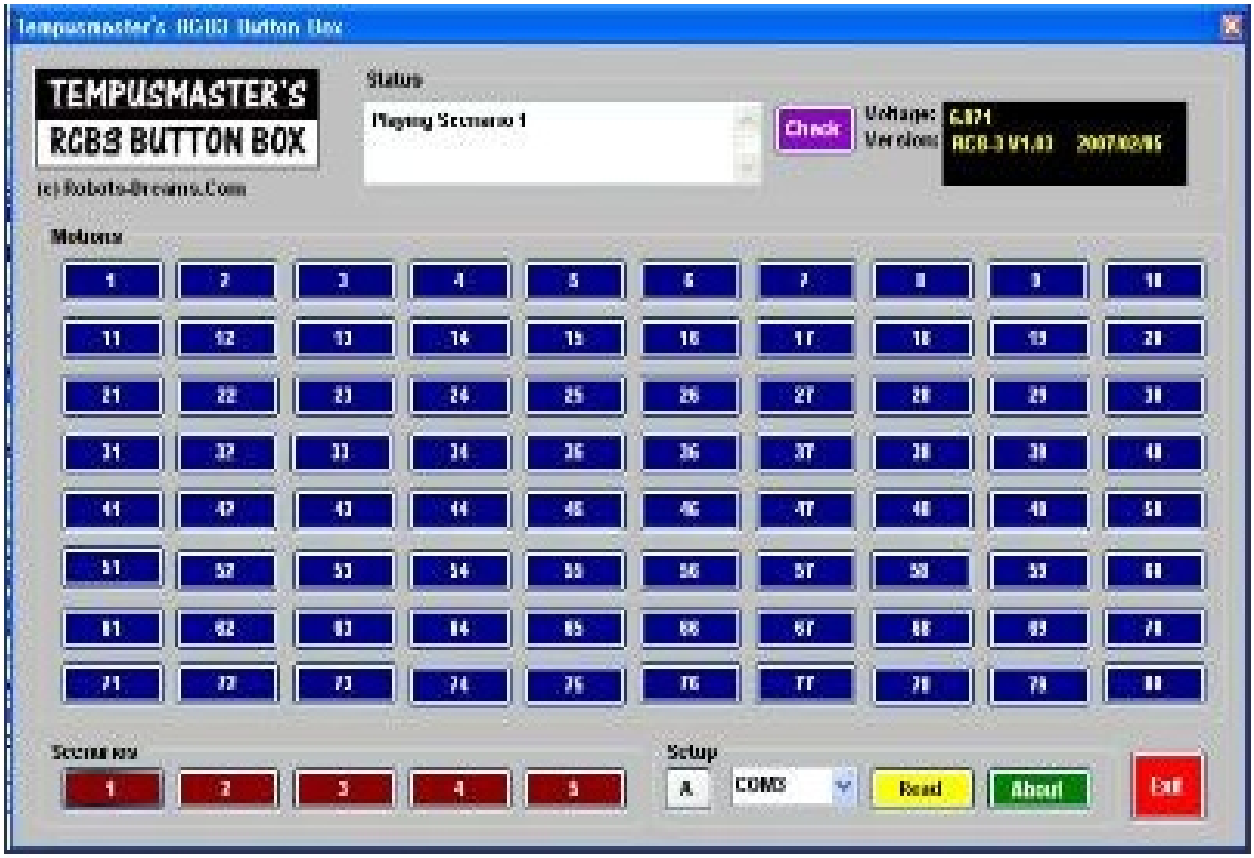
Testing:

I wrote a simple C language program that creates an interface with the same port the H2H is connected to (port 2), receive a full description of the last error in string format and terminates (destroys) the interface. Unfortunately, I kept receiving a tstcon32.exe warning and a blank, ActiveX Control window. However, when I used the C# library, the program was able to playback any motion that is stored in the microcontroller's memory. I was also able to add time delays between each motion play back. Functions that send individual angle positions are available; unfortunately due to the limits of the RCB3 EEPROM and the possibility of loops that iterate more than (write) 100 times per execution makes these functions (possibly) hazardous. Upon further search, I found this versatile application that was built upon Laurent's library:

Button Box C API:

The RCB-3 Button Box application developed by Tempusmaster is a simple control application that features the ability to trigger motions and scenarios stored in the Kondo RCB-3 controller, and reports errors and status information. It can read and display the controller's motion and

scenario data including the assigned locations, size, assigned remote control code, and label. The button label display can be switched between numeric and the actual names with a single click.



B. Solution Implementation

After carefully examining the available options, I have decided to design the library of motion functions by first compiling (mixing) the motions and scenarios using the HeartToHeart3J interface, write them into memory, and then playback the motions using the Laurent library or the Button Box API. This solution has the following advantages:

1) It allows us to use the excellent options and capabilities the HeartToHeart3J provides into making complex and accurate motions and scenarios through a variety of means, for example by setting the angle positions for each channel or positioning the robot physically and capturing the values through the teaching mode. The program also allows us to control and use gyro and accelerator sensors and to edit the memory content of the RCB-3J controller.

2) Depending on the hardware module that will be used to control the robots and the OS they run on, we can either use the Laurent Library (C Library) to playback back the motions depending on the behavior program in Linux or the Button Box API if we use Windows.

Afterwards, I started compiling and editing the following motions:

1) Walking Forward

2) Walking Backwards

3) Step right/left

4) Turn right/left

5) Swat right/left

6) Kick right/left

7) Bow

8) Push-ups

1) Walking Forward:

There are many walking motion algorithms, some of which are described in the Literature Survey and Mathematical Model sections. However, for the purpose of our project which is to make a library of motions and not a walking pattern generator, I followed this basic bipedal robot walking principles:

1) Home position is only a point of reference, used to compare poses and movements, not a good pose for natural walking

2) Start and end in a balanced, natural position with the knees slightly bent

3) Use the arms for balance while in motion

4) Shift weight from side to side

5) Lift the feet up off the surface

6) Upper body should remain level and perpendicular to the surface

7) Adjust lower servos to keep the upper body positioning

8) Keep the soles of the feet parallel to the floor surface

9) CoM should stay within the foot/sole balance region

After taking into consideration constraints such as speed, reliability, balance and symmetry, the following is the final configuration for the **walking forward motion**:

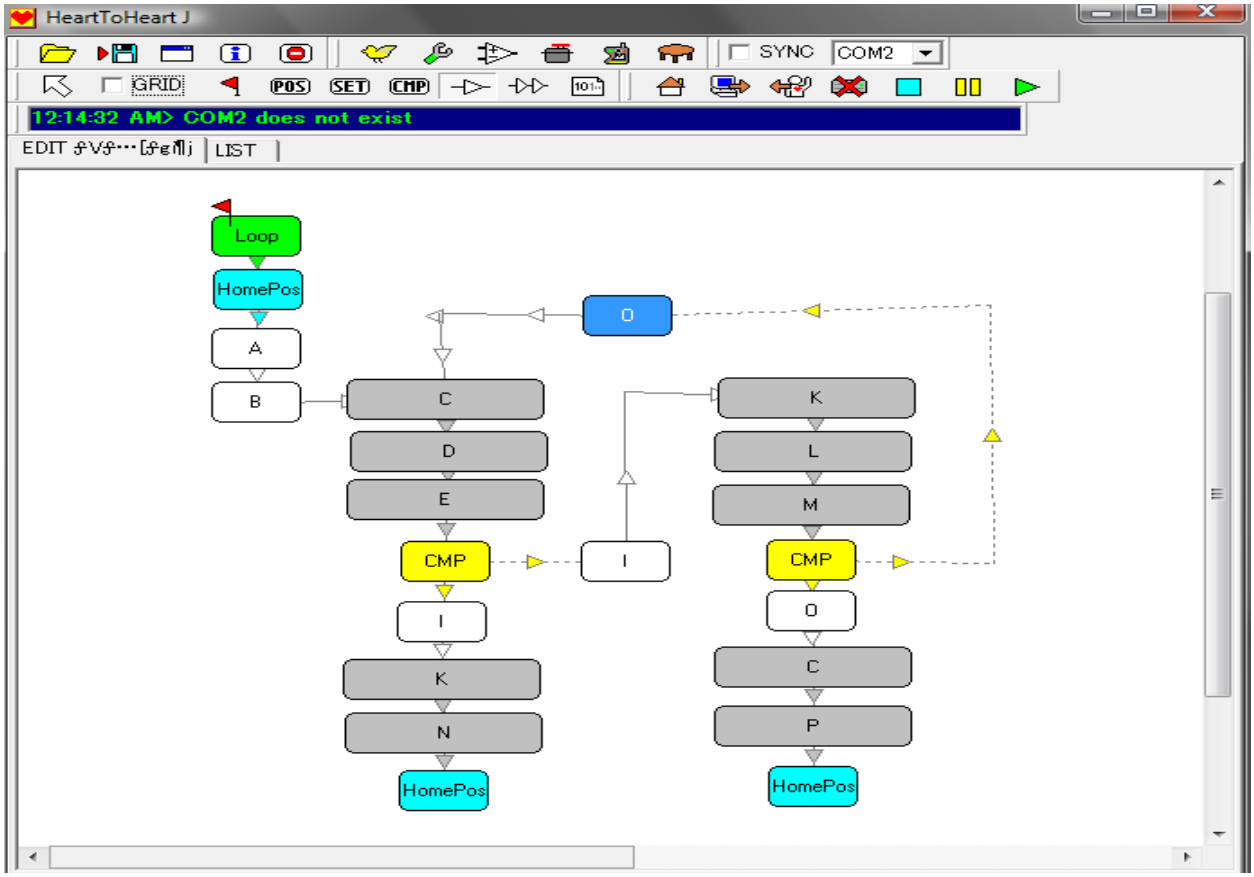
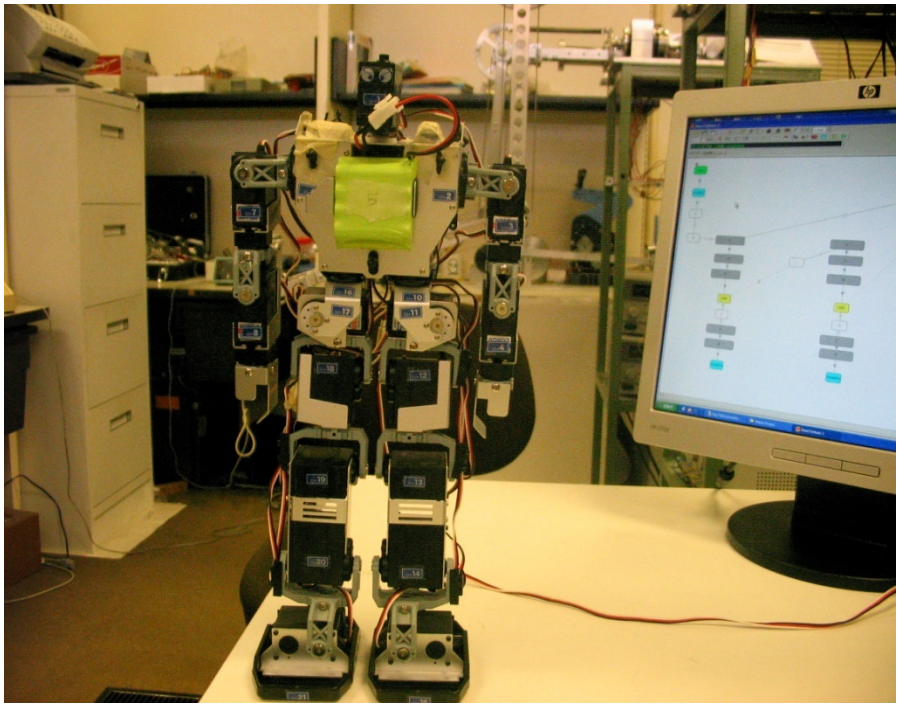
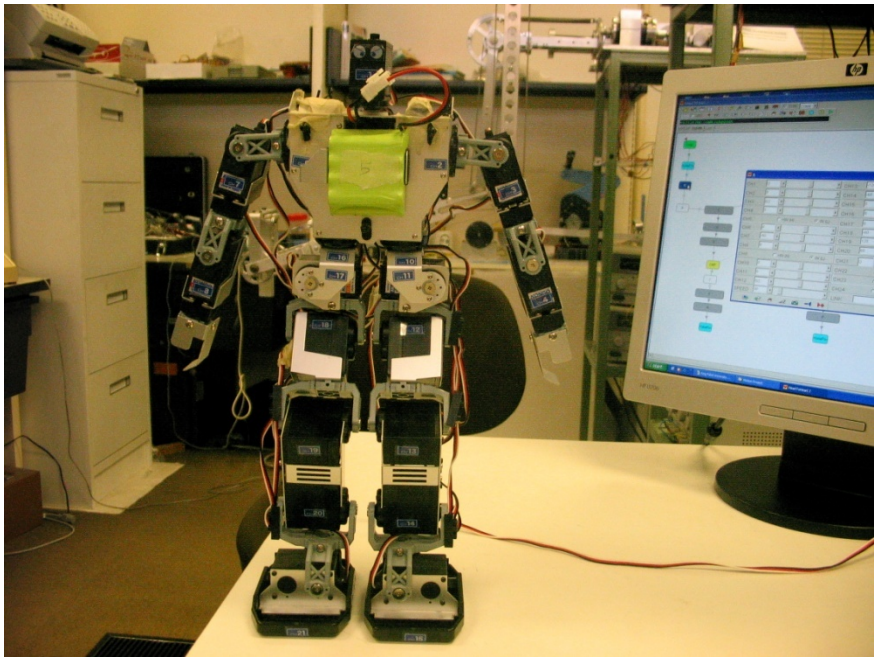


Figure: The 25 Positions that compose the Walking forward motion using the H2H software

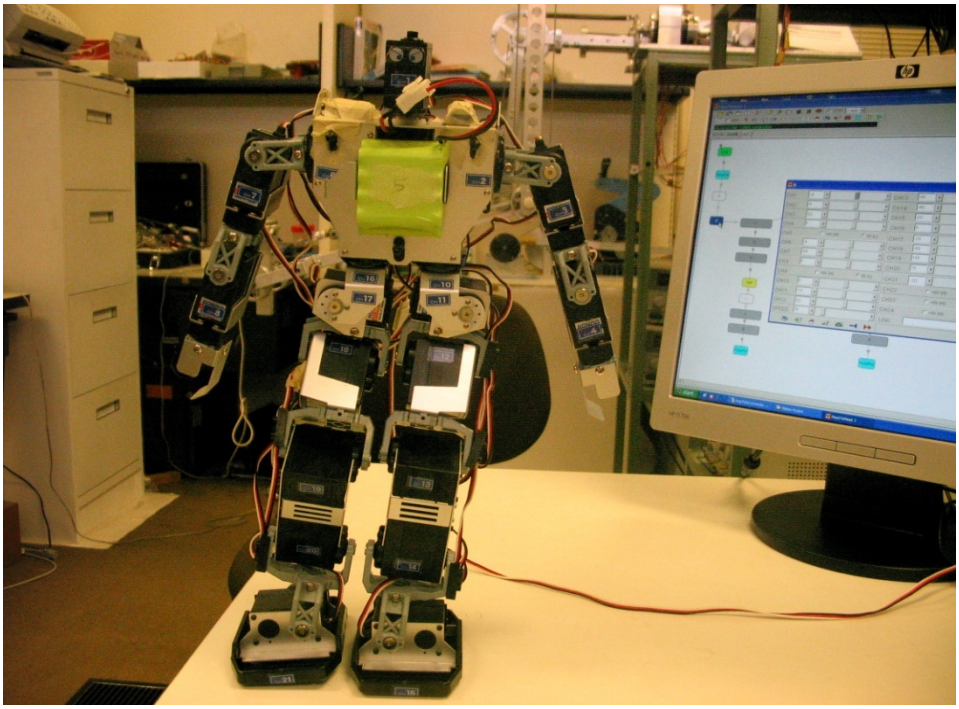
Home Position:



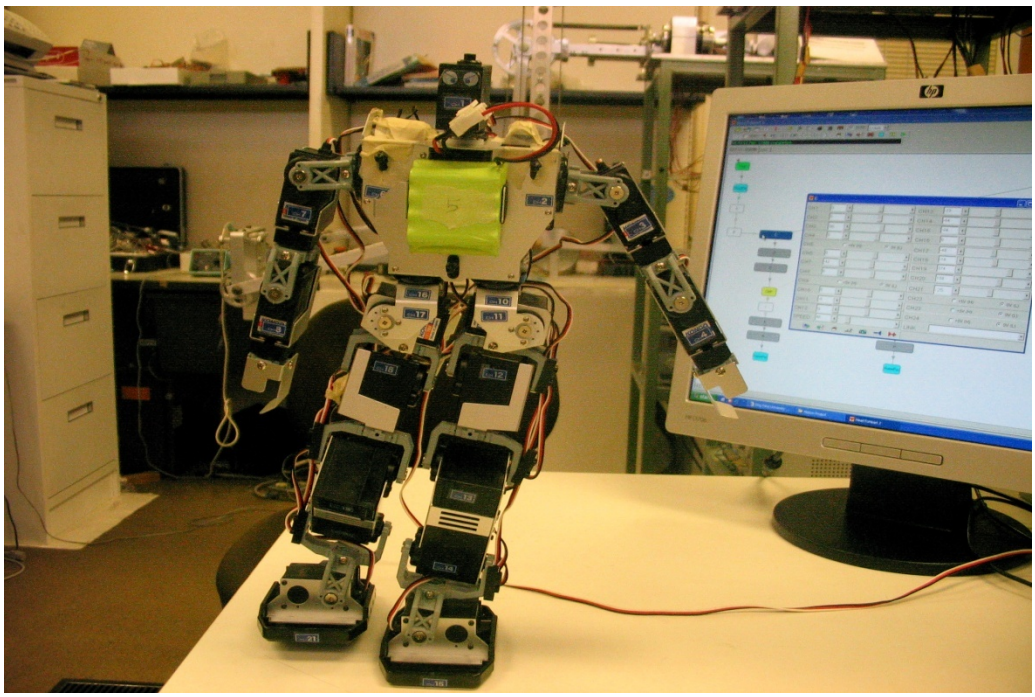
A: Bend both knees slightly:



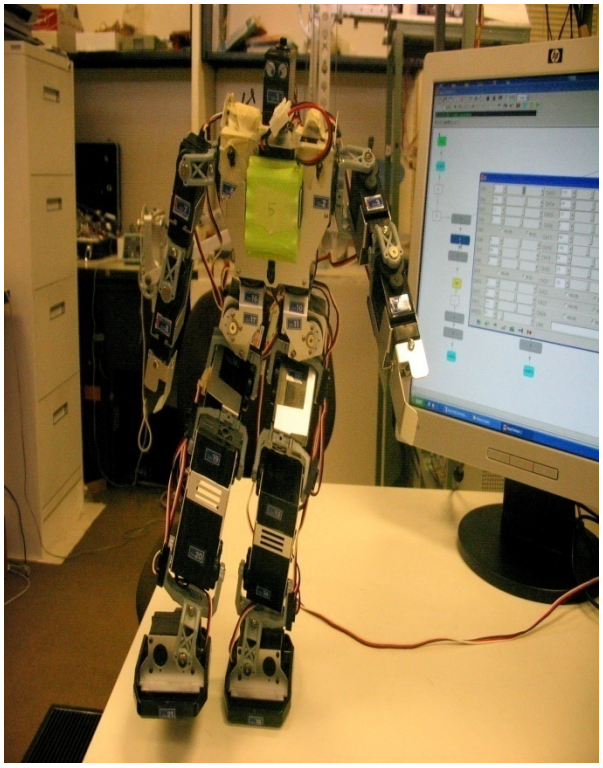
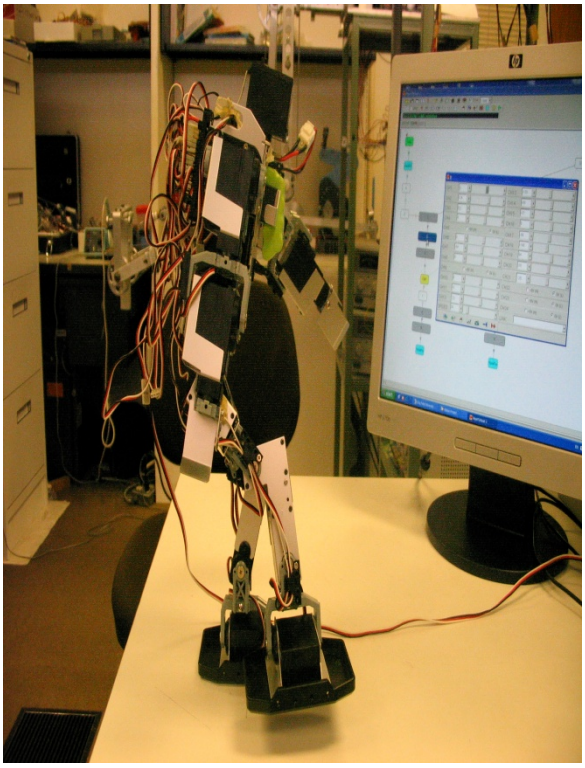
B: Prepare for right step: Tilt to left leg to make the CoM on the left sole, both arms spread away equally and right knee slightly bent with ankle rotating to the right:



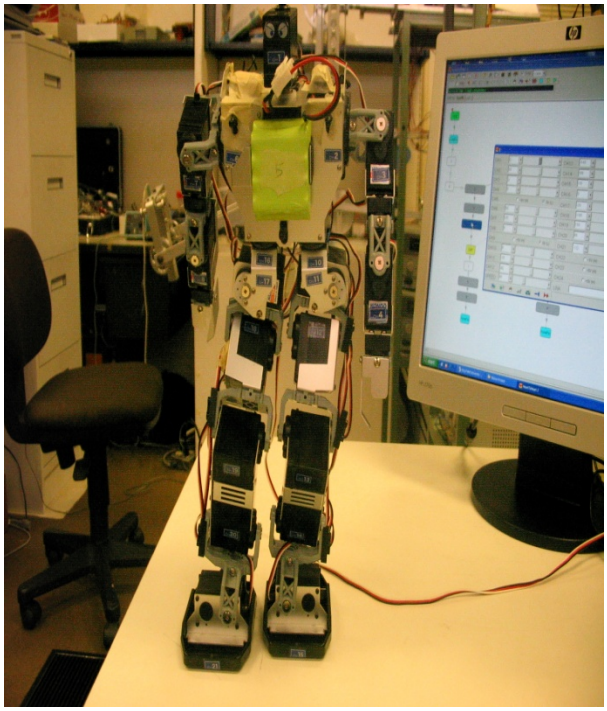
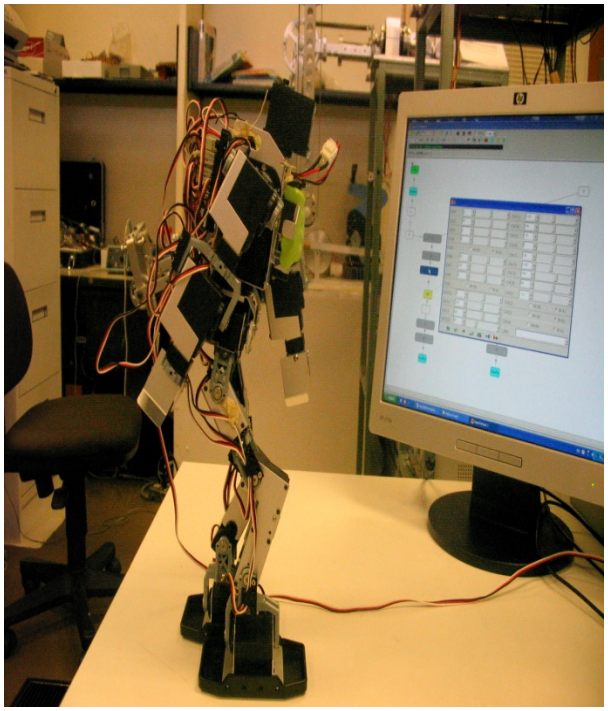
C: Right Arm closer to the body, left arm further away, right leg slightly forward:



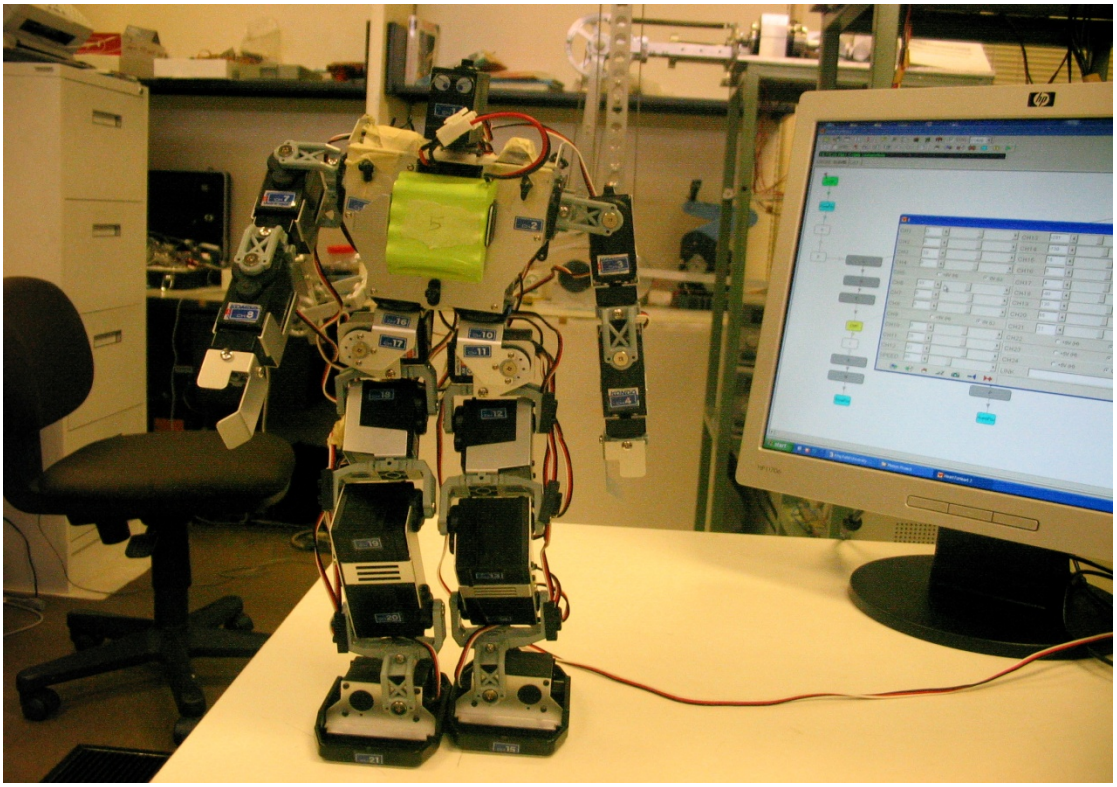
D: Left arm to the front for balance, right arm extended away slightly, right leg extended further to make the step:

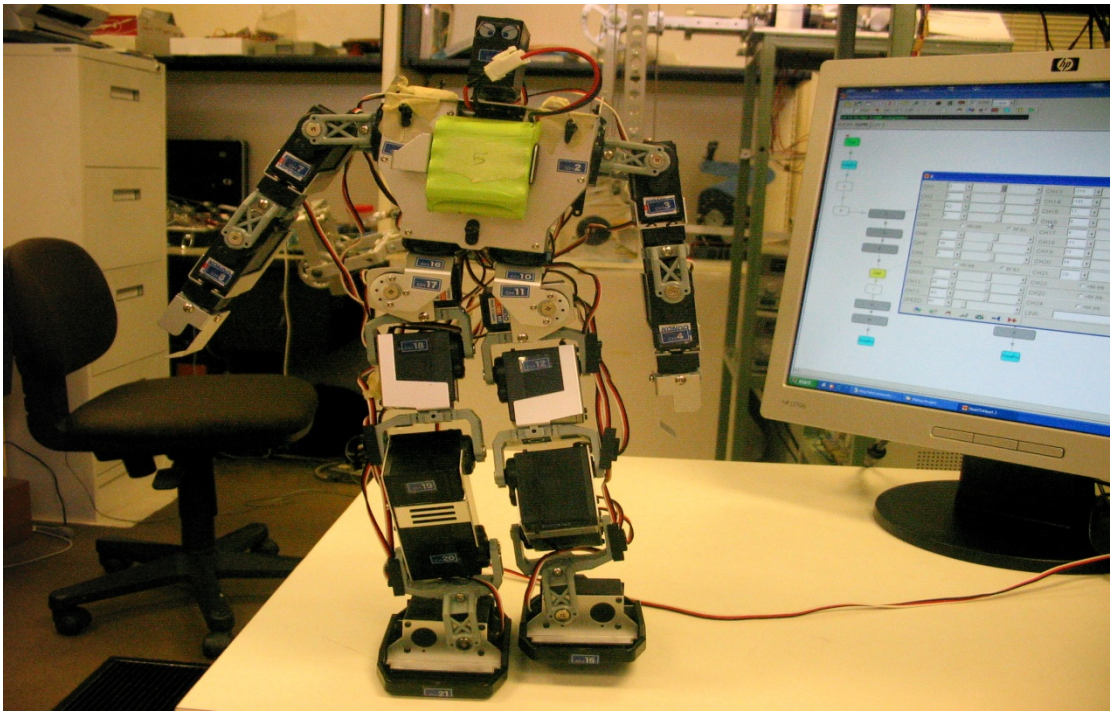
E: Right leg lands firmly, left knee bends forward, left arm goes back, and right arm goes back further:

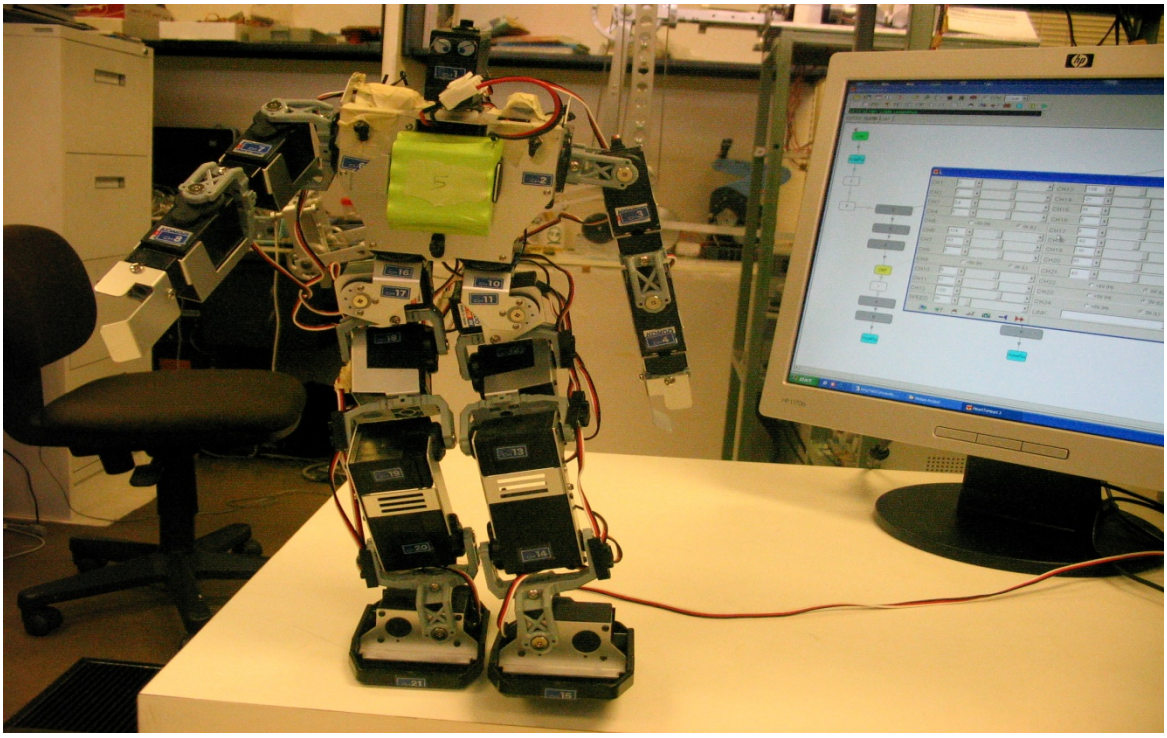
I: Center of gravity shifts to the right sole, left arm aligned closely with the body, right arm front and away from body, left knee slightly bent with left ankle rotating to left:



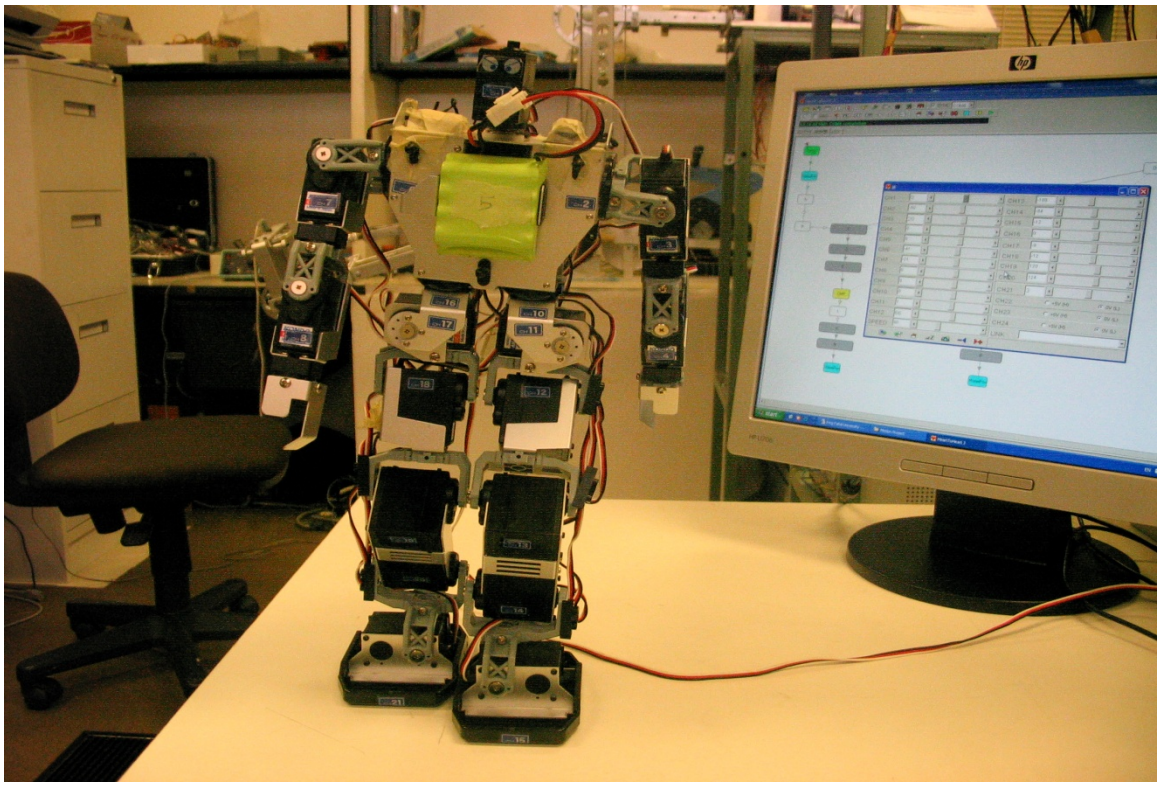
K: Left foot raised up, right arm further away:



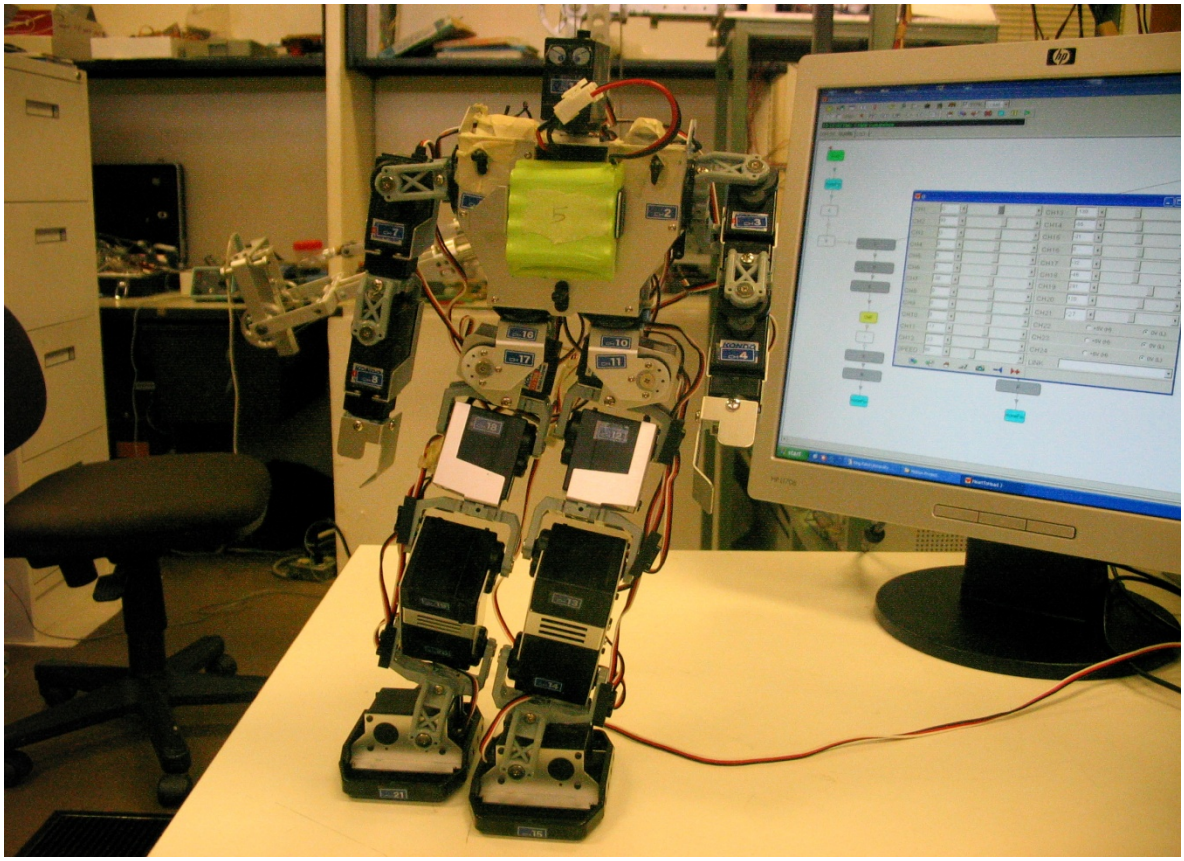
L: Left leg extended for left step, right arm to the front:



M: Left leg lands firmly, left arm backwards further, right arm slightly backwards and right knee bends forward:



O: Center of mass shifts to the left sole, right arm aligned close to the body, left arm front and away from body, right knee slightly bent with right ankle rotating to the right:



Note: The above positions constitute one step with right leg (C-D) and one step with the left leg (K-M). Depending on the value that is set in the position **loop** (e.g. x = 5), the position **cmploop** will decrement x by 1 and compare it with 0 when it is reached. If true, the motion will follow the following sequence:

1) If the last executed position is **E**: Make a partial left leg step by executing positions K and N in preparation for the **homeposition.**

2) If the last executed position is **M**: Make a partial right leg step by executing positions C and P in preparation for the **homeposition.**

So, for example if the value in loop is set to 5, then the following motion will be observed:

Right step, left step, right step, left step, right step, *half* left step and then return to home position.

2) Walking Backwards:

Number of Positions: 15

Description: The robot moves backwards by first shifting his center of mass to the left sole of his feet, takes a step backwards with its right leg, shift the center of mass to the right sole, and stay in that position until the robot is stable again, then repeat again but now with a left step. Depending on the value set in the counter, the robot will continue to execute this motion until the value is decremented to 0. Finally, the robot prepares itself to return to the home position by taking a half step backwards.

Refer to the directory of motion videos or the appendix to view the angle values of each position that is an element of this motion.

3) Step Right/left

Number of Positions: 7

Description: The robot moves backwards by first shifting his center of mass to the left sole of his feet, takes a step backwards with its right leg, shift the center of mass to the right sole, and stay in that position until the robot is stable again, then repeat again but now with a left step. Depending on the value set in the counter, the robot will continue to execute this motion until the value is decremented to 0. Finally, the robot prepares itself to return to the home position by taking a half step backwards.

Refer to the directory of motion videos for more detail or the appendix to view the angle values of the individual positions that compose this motion.

4) Turn right/left

5) Swat right/left

6) Kick right/left

7) Bow

8) Push-ups