18th IEEE International Conference on Control Applications
Part of 2009 IEEE Multi-conference on Systems and Control
Saint Petersburg, Russia, July 8-10, 2009

Realistic simulation of a Lego Mindstorms NXT based robot

José Gonçalves, José Lima, Paulo Malheiros and Paulo Costa

Abstract— **This paper describes the realistic simulation of a Lego Mindstorms NXT based robot. The presented approach does not replace the training with hardware but is an important complement, since it allows to develop robot software without accessing to the real hardware.**

I. INTRODUCTION

Lego Mindstorms is a powerful educational tool [1] [2], being used by the authors in many activities. It has been used for teaching mobile robots introductory concepts to undergraduate and graduate students and in several demonstrations. The undergraduate students while attending summer courses prototype their own robots and develop robot software based on the Lego Mindstorms educational software. The graduate students while attending robotics classes also prototype their own robots, with the difference that the developed robot software is done resorting to high level programming languages. It has also been used in demonstrations with the goal of captivate and inform undergraduate students, concerning the areas that involve technology. As an example it is shown in Figure 1 a demonstration at the Live Science Center of Bragança where are shown two mobile robot prototypes based on the Lego Mindstorms NXT Kit.

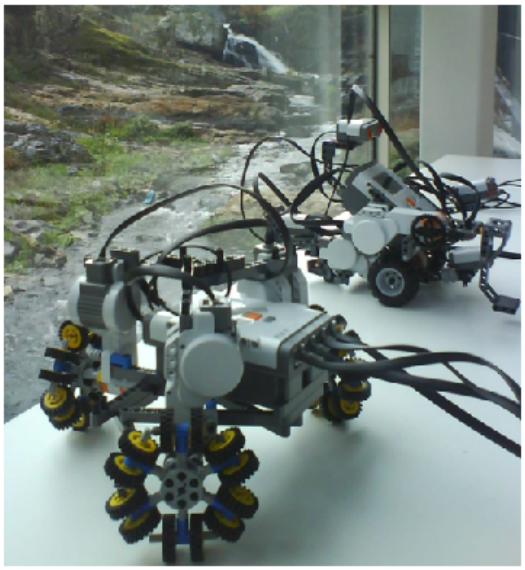

Fig. 1. Demonstration at the Live science center of Bragança

José Gonçalves and José Lima are with the Polytechnic Institute of Bragança, Department of Electrical Engineering, Bragança, Portugal {goncalves,jllima}@ipb.pt

Paulo Malheiros and Paulo Costa are with the Faculty of Engineering of the University of Porto, DEEC, Porto, Portugal {paulo.malheiros,paco}@fe.up.pt

Code migration from realistic simulators to real world systems is the key for reducing the development time of robot control, localization and navigation software. For this purpose it was developed the SimTwo realistic simulator (available for download at [3]). Due to the inherent complexity of building realistic models for the robots, its sensors and actuators and their interaction with the world, it is not an easy task to develop such simulators. In this paper it is described the modeling of an NXT based robot and its realistic simulation. In order to validate the presented approach a real robot and its simulation are compared through a challenge.

The presented approach does not replace the training with hardware but is an important complement. Students can develop and test robot controllers even at their homes, without the need of having access to hardware. This situation is important because usually students have access to hardware only when they are attending classes or when they have tutorial classes at the robotics laboratory. The simulation is also important to monitor some interesting variables, that are much more difficult to access in the real world. As a direct benefit the robot behavior can be monitorized in a more accurately way.

The Lego modularity makes the rapid prototyping of different robot configurations easier [4] [5]. This easiness presents itself as an extra motivation for the persons who are taking their first steps in the world of mobile robotics. The Lego parts allow connectivity, suppressing the need of using glue or screws. It's also an ecological tool because although it's not easy to recycle plastic, the Lego parts can be reused. It's a modular tool, it is possible for the same part to be a different thing depending on the application, motivating those who are using it and it is presented at a relatively low cost. Added to all the presented advantages of the Kit Lego Mindstorms, simulation can also be a very important teaching aid.

II. DEVELOPED PROTOTYPE

The developed prototype, presented in Figure 2, is a differential robot, composed by two wheels and a light sensor. Its movement is controlled by varying the velocity of each wheel independently.

The differential robot, presented in Figure 3, is probably the most used mobile robot. It is composed by two wheels with their shaft passing by the same axle, their movement is controlled by varying each wheel velocity independently.

The mechanical structure of the differential robot, presented in Figure 3, does not allow movements along the axle that cross wheel shafts [6]. Considering that there is no lateral slip (wheel velocity in the ground contact point

Fig. 2. Driving Base of the Lego Mindstorms NXT Educational KIT

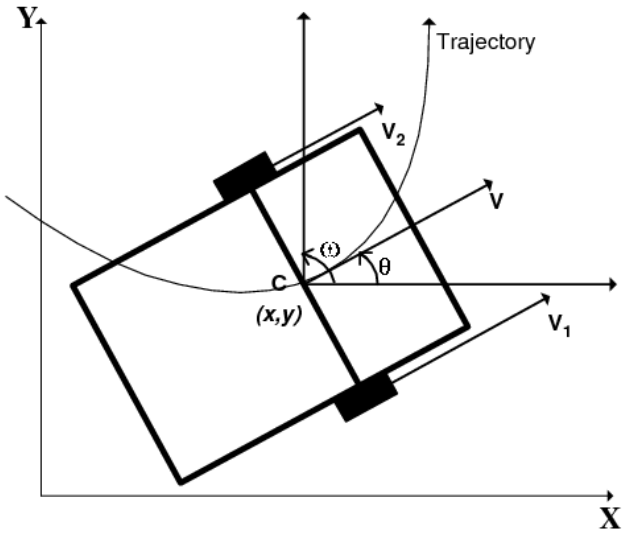

Fig. 3. Differential robot kinematics

perpendicular to the axle along the wheel shaft), the state vector given by equation (1) can be obtained.

$$X(t)^T = \begin{pmatrix} x(t) & y(t) & \theta(t) & v(t) & w(t) \end{pmatrix} \quad (1)$$

In equation (1), $x(t)$, $y(t)$ and $\theta(t)$ represent the point C in the plane and $w(t)$ represents the angular velocity. Another possibility to represent the chosen state variables is the use of the next equation:

$$X(t)^T = \begin{pmatrix} x(t) & y(t) & \theta(t) & V_1(t) & V_2(t) \end{pmatrix} \quad (2)$$

In this case $V_1(t)$ and $V_2(t)$ are the velocities measured in the contact point between the wheel and the ground. There are two alternative representations, being possible to change between them using equation (3) and equation (4). In equation (4), b represents the distance between the ground contact points.

$$v(t) = \frac{V_1(t) + V_2(t)}{2} \quad (3)$$

$$w(t) = \frac{V_1(t) - V_2(t)}{b} \quad (4)$$

Considering the non slip situation the differential robot kinematics is described by the following equation:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v(t)cos(\theta(t) \\ v(t)sin(\theta(t) \\ w(t) \end{pmatrix} \quad (5)$$

Equation (5) allows, using equations (3) and (4), to write the linear $\dot{x}$, $\dot{y}$ and the angular $\dot{\theta}$ velocities using each wheel measured velocity [7].

III. ROBOT MODELING AND SIMULATION

Design behavior without real hardware is possible due to a physics-based simulator implementation. The dynamic behavior of the simulated robot is computed by ODE Open Dynamics Engine, a free library for simulating rigid body dynamics [8] [9]. The simulator architecture is based on the real Lego NXT robot. The body masses and dimensions are followed in order to build a simulated robot like the real one. The robot weighs 0.540 Kg, each part weight is specified in Table I.

Robot parts	Weight (Kg)
Motor	0.079
Brick	0.255
Light sensor	0.013
Wheel	0.017
Other parts	0.080

TABLE I
WEIGHTS OF THE ROBOT PARTS

A. Actuator modeling

The Lego Mindstorms NXT kit provides three servomotors with built-in rotation sensor and a gear ratio of 1:48. A Lego Mindstorms NXT servomotor is shown in Figure 4.

Fig. 4. Lego Mindstorms NXT servomotor.

The servomotor can be resumed to a DC motor model, presented in Figure 5, where U_a is the converter output, R_a is the equivalent resistor, L_a is the equivalent inductance and

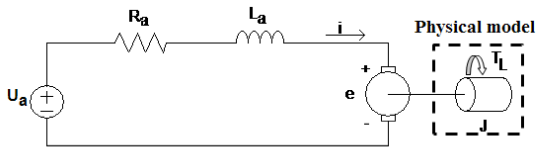

Fig. 5. DC motor electric model.

e is the back emf (electromotive force) voltage as expressed by equation (6) [10].

$$U_a = e + R_a i_a + L_a \frac{\partial i_a}{\partial t} \tag{6}$$

The motor can supply a torque T_L and the load has a moment of inertia J that will be computed by the physical model ODE. Current i_a can be correlated with the developed torque T_d through equation (7) and the back emf voltage can be correlated with angular speed through equation (8), where K_s is a motor parameter that can be found by an experimental setup as presented in subsection III-A.1 [11].

$$T_d(t) = K_s i(t) \tag{7}$$

$$e(t) = K_s \omega(t) \tag{8}$$

In fact, the real developed torque (useful) that will be applied to the load (T_L) is the developed torque subtracted by the friction torque (T_c).

$$T_L = T_d - T_c \tag{9}$$

1) DC motor model measurements: It was used the NXT Lego Mindstorms motor as the base of the simulator. The R_a and L_a values can be directly measured (R_a=7.6 Ω and L_a= 4.88 mH). The K_s motor parameter can be found by an indirect measure. For several angular speeds, it can be measured the emf voltage while the motor is in open circuit. The Table II presents the data for 7 measures.

rot (rto/min)	rot (rad/s)	e (V)	k=e/rot
0	0	0	
66	6.91	3.4	0.491
115	12.04	6.0	0.498
155	16.23	7.9	0.487
195	20.42	9.8	0.479
233	24.39	11.9	0.487
265	27.75	13.85	0.499

TABLE II
SPEED AND EMF VOLTAGE.

Figure 6 shows graphical data of the K_s line and its trend line. The average value for K_s (line slope) is about 0.4919 V.rad/s. It is possible to observe that the error in the origin for the presented approximation it is negligible, being only 0.0218 Volt. Despite being a small error it is possible to increase the precision in the K_s parameter estimation,

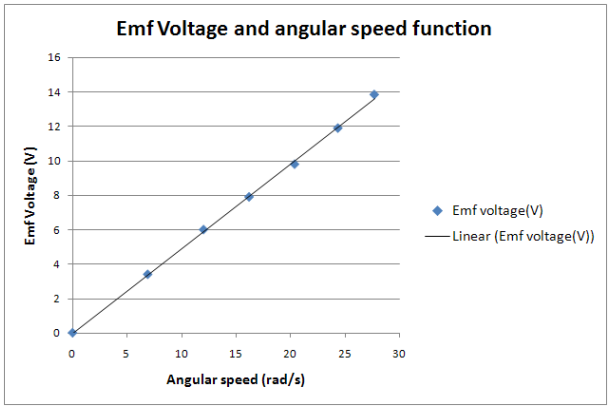

Fig. 6. K_s value for DC motor model.

forcing the trend line to pass in the origin. The obtained value for K_s is nearly 0.4908 V.rad/s.

In order to calculate the friction it was obtained the motor angular velocity for different voltages at its terminals. After a linear regression it was obtained equation (10).

$$rot = 1.9248v - 0.2519 \tag{10}$$

Where rot is the angular velocity in rad/s and v is the voltage at the motor terminals.

Assuming that the developed torque (T_d) is equal to the torque generated by the static friction (T_c) it is possible, resorting to the linear regression presented in equation (10), to obtain the necessary voltage to equal the static friction (v_c), by replacing rot by zero. For this situation it was not necessary to include in the model the viscous friction, because it only exists for angular velocities different from zero. As there is no movement there is no voltage generated by the emf and as the current has no variation it is possible to obtain the produced torque by the static friction resorting to equations (6) and (7), resulting in equation (11). The torque due to the static friction has a numerical value of 3.558E-4 Nm.

$$T_c = K_s \frac{v_c}{R_a} \tag{11}$$

After obtaining the static friction there is only left, to complete the motor model, the parameter estimation that relates the viscous friction with the motor angular velocity, as shown in equation (12).

$$T_\omega = B\omega \tag{12}$$

Placing the motor spinning without load the developed torque will given by equation (13).

$$T_d = T_c + B\omega \tag{13}$$

As the developed torque is proportional to the current as shown in equation (7) it is possible to conclude, from equation (13), that the current can be given by equation (14).

$$i_a = \frac{T_c + B\omega}{K_s} \tag{14}$$

The voltage applied to the motor terminals can be given by equation (6), but as in steady state the current variations are small, it can be obtained equation (15) for the motor terminal voltages.

$$U_a = K_s\omega + R_a i_a \tag{15}$$

where e was replaced by $K_s\omega$ as exemplified in equation (8).

From equations (14) and (15) it is obtained equation (16).

$$\omega = \frac{U_a - \frac{R_a T_c}{K_s}}{\frac{R_a B}{K_s} + K_s} \tag{16}$$

Minimizing the sum of the absolute error between the real angular velocity and the obtained by the model (equation (16)), it is obtained a numerical value for B (1.92E−3).

The estimated motor parameters are shown in Table III.

Parameters	Value SI units
K_s	0.4908
T_c	3.558E-4
B	1.92E−3
R_a	7.6
L_a	4.88E−3

TABLE III
MOTOR PARAMETERS

The different obtained motor models can be observed in the graphics shown in Figure 7.

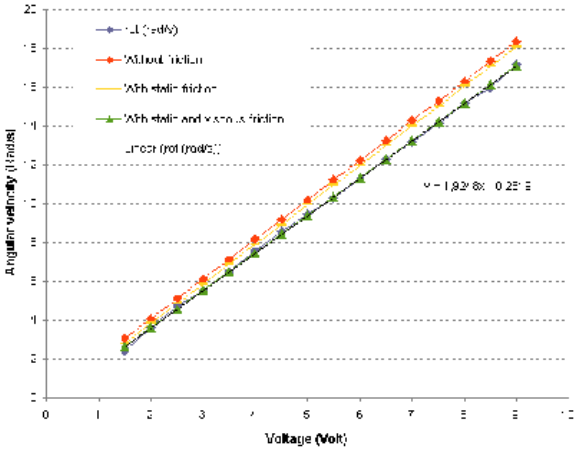

Fig. 7. Different obtained models.

2) DC motor nonlinearities: In a way to map the reality closer, where variables cannot assume all values, the model must have some limitations. The first one, the voltage applied to the supply terminals U_a. This voltage should be limited to the batteries voltage. Further, current i should be limited once it is related to the torque through equation (7). Figure 8 shows the limitations presented in the servomotor model.

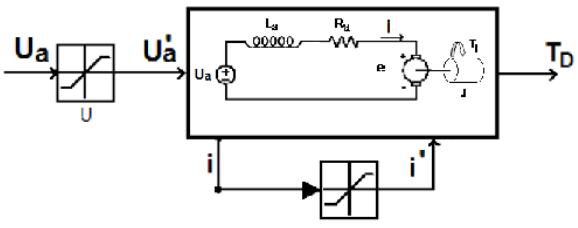

Fig. 8. Servomotor model nonlinearities.

B. Light sensor Modeling

The light sensor enables the robot to distinguish between light and dark. It can read the light intensity in a room and measure the light intensity of colored surfaces. The light sensor reacts as the characteristic shown in Figure 9, for measures across a gradient printed in an A4 page. It was observed that for different room lightning the rate that the sensor data evolves remains the same, but an offset value must be added, depending on the room lighting. The gradient that is going to be used in the proposed robot challenge can be observed in the next section, in Figure 12.

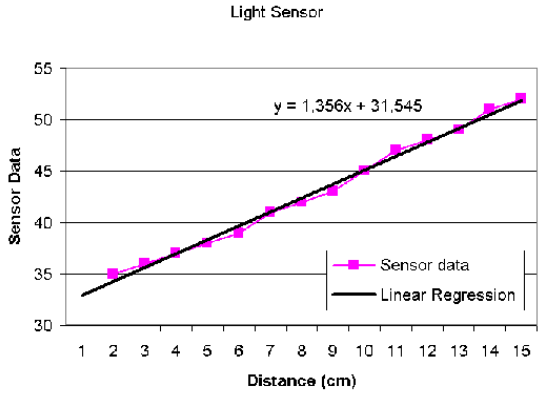

Fig. 9. Light sensor characteristic.

C. Robot simulation

For the simulation of the NXT Robot it was used the SimTwo, being a versatile robot simulation environment that allows rapid test and design of differential, omnidirectional, industrial robots, humanoids, etc., developed in Object Pascal. SimTwo has a set of predefined components such as motors where the model specified in page 2 is inputed. The Open Dynamics Engine [8] [9] is used for the simulation of the rigid body dynamics. The robots look and behavior are defined in XML format files. The virtual world is represented using GLScene components [12], these provide a simple implementation of OpenGL.

The NXT robot was defined using boxes and cilynders as shown in Figure 10. These are the objects used by the ODE to determine the collisions and friction.

SimTwo allows the replacement of the solids by any 3DS model which gives a realistic look of the robot as shown in Figure 11.

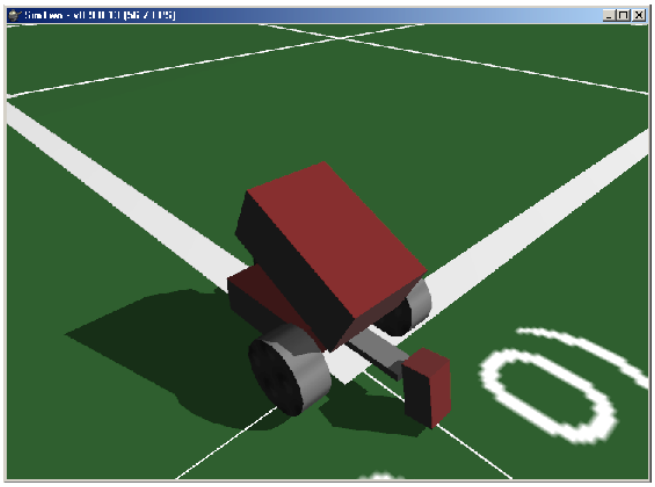

Fig. 10. NXT robot modeled in the SimTwo

Fig. 11. NXT robot modeled in the SimTwo with 3DS models

IV. PROPOSED CHALLENGE

In order to validate the presented approach, a real robot and its simulation are compared through a challenge. The proposed challenge is the robot to follow a path based in a gradient. The robot must follow the path by the center of each section. The path for the real robot is presented in Figure 12.

It was applied the same controller for the simulated and for the real robot. Given the sensor measurements it is possible to estimate, at each sample time, the distance error (de) from the path center. The applied controller maintains the robot linear velocity constant and the angular velocity proportional to the distance error. The chosen velocities for each wheel are described by equations (17) and (18).

$$V_1(t) = k_v + k_w de \tag{17}$$

$$V_2(t) = k_v - k_w de \tag{18}$$

Where k_v and k_w are constants that can be tunned.

Fig. 12. Challenge real robot.

Given equation (3) the robot linear velocity results in equation (19) and given equation (4) the angular velocity results in equation (20).

$$v(t) = k_v \tag{19}$$

$$w(t) = \frac{k_w 2de}{b} \tag{20}$$

A. Simulation

The controller was implemented in the simulator using the embedded script functionality. It is possible to observe that the trajectory is done correctly by the analysis of robot angle, presented in Figure 13, being provided by the simulator. The simulator provides several variables but the angle is the most representative to show the results of the proposed challenge.

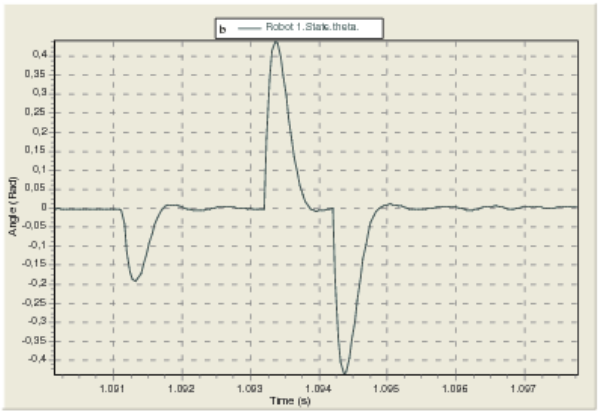

Fig. 13. Simulated robot angle.

B. Real robot

The chosen programming language for the control software was Lejos. The control program consists in one thread, the periodical events are generated placing the thread to sleep during a specific amount of the time. After this time the thread wakes up, makes all the operations, calculates the time left to the next sample and goes back to sleep. The used control architecture, presented in Figure 15, is generic and can be applied to several control challenges other than mobile robotics. The thread has a sampling time associated, chosen so that the controller can be updated as fast as possible. The choice was 40 ms being enough to fulfill the proposed real time requisites, being a sampling time commonly used in mobile robotics. After several tests it was observed that the Brick toke always less than 40 ms to execute the controller. The robot position estimation is based on the odometry calculation and registered resorting to the Datalogger class, that allows to store float values and then transmits the data from the Brick to a PC via Bluetooth or USB. The position estimate error is cumulative, since odometry calculation is a relative measurement. Despite the cumulative error it is possible to observe that the trajectory is done correctly by the analysis of the estimated robot angle, presented in Figure 14.

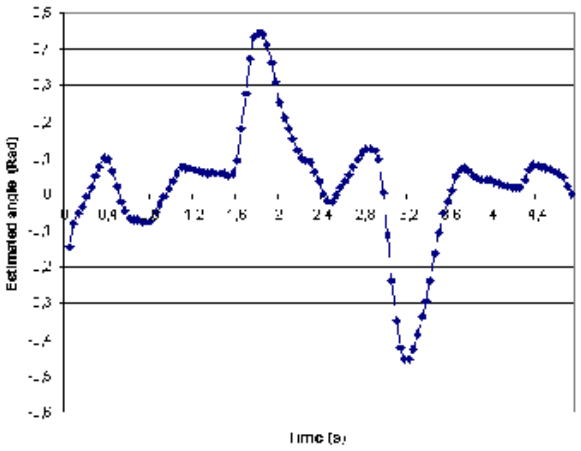

Fig. 14. Estimated angle of the real robot.

V. CONCLUSIONS

For the simulation of the NXT Robot it was used the SimTwo, being a versatile robot simulation environment that allows rapid test and design of differential, omnidirectional, industrial robots, humanoids, etc., developed in Object Pascal. The presented approach does not replace the training with hardware but is an important complement, since it allows to develop robot software without accessing to the real hardware. The used control architecture is generic and can be applied to several control challenges other than mobile robotics. The robot controller was implemented in the simulator using the embedded script functionality. It is also possible to control robots using a remote application that communicates with the simulator using the UDP protocol.

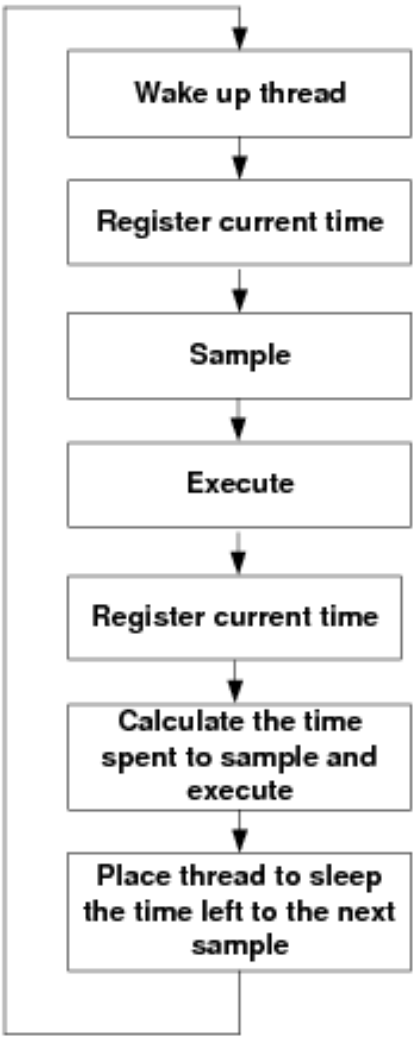

Fig. 15. Generic control architecture.

REFERENCES

[1] Á. Valera, M. Weiss, M. Vallés, J. Díez *Proposal of a low-cost mobile robot control laboratory experiment,* 7th IFAC Symposium on Advances in Control Education, Madrid, 2006.

[2] P. Gawthrop, E. McGookin *Using Lego In Control Education,* Plenary Lecture 7th IFAC Symposium on Advances in Control Education, Madrid, 2006.

[3] "SimTwo" http://www.fe.up.pt/∼paco/wiki, 2009.

[4] G. Reshko and M. Mason and I. Nourbakhsh *Rapid prototyping of small robots,* Technical Report, Carnegie Mellon University, 2000.

[5] J. Gonçalves, J. Lima, P. Costa *Rapid prototyping of mobile robots extending Lego mindstorms platform,* 7th IFAC Symposium on Advances in Control Education, Madrid, 2006.

[6] G. Dudek and M. Jenkin, *Computational Principles of Mobile Robotics,* Cambridge University Press, 2000.

[7] J. Borenstein, H. Everett, L. Feng, *'Where am I?' Sensors and Methods for Mobile Robot Positioning,* Technical Report, The University of Michigan, 1996.

[8] "ODE" http://www.ode.org/, 2009.

[9] B. Browning, E. Tryzelaar, *UberSim: A Realistic Simulation Engine for RobotSoccer,* Proceedings of Autonomous Agents and Multi-Agent Systems, Melbourne, 2003.

[10] A. Conceição, A. Moreira, P. Costa, *Dynamic Parameters Identification of an Omni-directional Mobile Robot,* Proceedings of the International Conference on Informatics in Control, Automation and Robotics, Setúbal 2006.

[11] R. Bishop, *The Mechatronics Handbook,* CRC Press, New York, 2002.

[12] "GLScene" http://glscene.sourceforge.net/wikka/HomePage, 2009.