# Designing a Low-cost, Expressive Educational Robot

**Thomas Hsiu, Steve Richards, Ajinkya Bhave[§], Andres Perez-Bergquist[§], Illah Nourbakhsh[§]**
[§]The Robotics Institute, Carnegie Mellon University
Pittsburgh, PA 15213, illah@ri.cmu.edu

**Abstract**

*The Trikebot is the result of a ground-up design effort chartered to develop an effective and low-cost educational robot for secondary level education and home use. This paper describes all aspects of the Trikebot, including chassis and mechanism; control electronics; communication architecture; robot control server and student programming environment. Notable innovations include a fast-build construction kit, indoor/outdoor terrainability, CMOS vision-centered sensing, back-EMF motor speed control and a Java programming interface.*

## 1 Introduction

In Summer 2002, the authors and others developed, taught and evaluated *Robotic Autonomy*, a seven-week introductory hands-on robotics course as part of Carnegie Mellon West's NASA-Ames campus in Mountain View, California [1]. The research surrounding this effort included robot design, curriculum design and ongoing, long-term educational evaluation. Although these and other authors have recognized and study the role of robotics in education, this work is notable in that all aspects of the robot mechanism, electronics, software and educational curriculum were subject to ground-up, coordinated design [2,3,4,5,6,7]. A total of 30 Trikebot robots were built and used during this program. They continue to be used by graduates of the course at home. This paper focuses on the robot mechanism, electronics and programming interface design processes.

## 2 Goals

The overall design goals are informed by the intended target audience for the educational course: high school students between their junior and senior year. Prerequisites are only basic mechanical dexterity (e.g. simple assembly and fabrication) and knowledge of a programming language (e.g. *Introduction to Programming*). Each student would be slated to take a Trikebot home to program and use at will for a year. Thus, the Trikebot would need to be designed not only for the beginning robotics student but for the continuing, sophisticated user. Following from the above overarching goal, particular requirements and concordant innovations at each level of robot design are in subsequent sections.

## 3 Mechanism

### 3.1 Physical Overview

The Trikebot chassis is a three-wheeled mobile robot base in a tricycle-like configuration, with a single driven steerable wheel and two fixed passive wheels (Fig. 1). Its major physical features are a tall camera mast with a pan and tilt mechanism and two large payload areas. Altogether the Trikebot has 4 control degrees of freedom—drive motor, steering, camera pan and camera tilt.

The *tread width*, or distance between wheel centerlines as viewed from the front or back, is 15.8 inches and the *wheelbase*, or distance between wheel axes as viewed from the side, is 10.9 inches. The wheels of the Trikebot are 6 inches in diameter, supporting a ground clearance of 2.2 inches. The nominal camera height is 18.3 inches and it can pan approximately ±90° and tilt +90°/-45°. The mechanical chassis alone, including servos and drive motor, weighs approximately 10.5 lbs.

### 3.2 Mechanical Design Objectives

The Trikebot chassis has three primary functions. As a CMUcam platform [8], our goal was to place the camera at least 18 inches above the ground plane. This was part of a decision to make the Trikebot a floor-based robot which students could interact with more dynamically than a smaller table-top size robot. The pan and tilt mechanism is critical for *diagnostic transparency* and affection; it enables the robot to indicate direction of gaze and widens the field of view [9].

We expect the Trikebot to operate indoors and on flat outdoor areas such as parking lots, sidewalks and lawns; it must be able to overcome obstacles such as electrical cables, door thresholds and gravel. To facilitate mobility in closed quarters, we required Trikebot to turn in-place within a 24 inch circle. To encourage student-robot interaction, the speed of the Trikebot was specified as comparable to a person's medium speed walk, 30 in/sec.

As a worst-case payload requirement, the Trikebot is designed to carry a laptop computer, six 7.2V Remote Control (RC) car battery packs and various onboard electronics.

Being assembled and maintained by students in a general classroom environment required that the majority of the components of the Trikebot be assembled

using simple hand tools and that they be robust enough to handle rough treatment. Of course, cost is always an issue, so appropriate manufacturing techniques were chosen for the quantities of parts used.

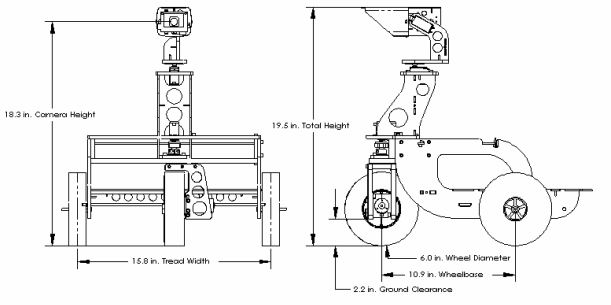


Figure 1: Trikebot Chassis Dimensions

### 3.3 Design Features

The Trikebot's final design derives from the above design objectives. The following describes how the various elements of the Trikebot chassis meet these objectives.

***Wheel Configuration***. A tricycle configuration with a single driven steering wheel gives the Trikebot very good agility using a single gearmotor as its drivemotor and a single high power servo for steering. The servo can steer the driven wheel through 180° allowing the robot to turn nearly in place.

We chose the tricycle design in lieu of the other common three-wheeled differential drive configuration to avoid several problems. One problem is that a trailing caster wheel can restrict the freedom of movement of the robot in certain situations. Furthermore, two driven wheels must match their speed profiles exactly in order for the robot to travel in a straight line. The tricycle design eliminates both issues.

The single wheel forward arrangement was chosen for agility over obstacles. The driven wheel can more easily grip and climb over an obstacle at slow speeds, subsequently dragging the rear wheels over the obstacle.

One final advantage of a three-wheeled design is reduced torsional stress on the chassis. In a four wheeled chassis, a single wheel can be raised above the others when traversing uneven terrain, twisting the chassis (and its payload). A three wheeled chassis undergoes less twisting, meaning the chassis can be simpler and lighter.

***Wheels*** With wheel diameters of 6 inches and a ground clearance of 2.2 inches, the Trikebot can drive over obstacles such as power cords, uneven sidewalks, and even gravel paths. The traction element of the wheels consists of closed-cell foam rubber *tires*. These tires provide adequate stiffness and traction, yet are still light and help absorb shocks. The rear passive wheels and front wheel hub are stock RC model airplane parts and car parts, utilized to minimize costs.

***Drivetrain*** The drivetrain consists of a 19.5:1 gearmotor directly coupled to the drive wheel. The gearmotor's output bearings are adequate for the loads expected to be delivered by the Trikebot and direct drive provided the simplest design (Fig. 2).

The drive wheel assembly turns about a kingpin which is centered above the contact patch of the drive wheel so that no steering torque is generated when the drive motor is engaged. A high-torque RC servo directly drives the kingpin, providing steering control.

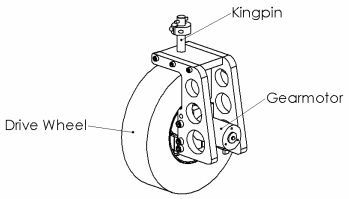


Figure 2: *The Drive Wheel Assembly*

***Camera Mast and Pan and Tilt*** The camera mast incorporates a pan and tilt mechanism and elevates the camera to above its desired 18 inch minimum height. The positioning of the mast to the front of the chassis allows the camera to scan slightly in front of the front wheel while looking down. This facilitates activities such as line following or object-in-path detection. The camera is centered above the camera's pan axis and the camera's centerline passes through the tilt axis. This simplifies the analysis of the camera's view relative to the robot. Both the pan and tilt are directly controlled by stock RC servos.

Figure 3: *The unassembled components of one Trikebot; 30 assembled Trikebots (right)*

***Payload Area*** The Payload areas of the Trikebot are positioned low and to the rear of the camera mast in order to place the fully loaded robot's center of gravity as low as possible and roughly 1/3 of the wheelbase behind the front wheel. A low center of gravity maximizes the stability of the Trikebot and placing the center of gravity 1/3 of the way behind the front wheel helps provide traction to the front driven wheel. The battery racks are located below the lower payload tray, again to lower the center of gravity and to provide access.

***General Construction*** Most of the Trikebot chassis is constructed of lasercut acetal (Delrin) sheets (Fig. 3). Aluminum machined parts were used for a few items, such as the drive hubs and motor clamps, but machining was minimized as it is ten times the price of lasercutting. However lasercutting has its drawbacks, allowing only cuts perpendicular to flat sheets of material like paper or plastic. To accommodate this, the Trikebot's parts fit together with tabs and slots, not unlike paper or cardboard models. Self-tapping screws wedged into slots hold the plastic parts together. This system enabled most of the Trikebot to be assembled and repaired by the students using hand tools. Using rapid manufacturing technologies such as lasercutting, combined with using stock parts such as RC servos and wheels, enables the Trikebot chassis to be produced economically (approximately $500 per chassis) and quickly in the required quantities.

## 4 Control Electronics

### 4.1 Overview

The role of the control electronics was to create a clean interface between the physical robot layer and the high-level Java programming interface the students would use to program the robot. The electronics abstract away most of the communication overhead, interface control and motion control aspects of the Trikebot. Our solution accomplished this abstraction while allowing flexibility for expansion, lower level control and design modularity.

Fig. 4 depicts the connectivity of the Trikebot's control electronics. An iPAQ 3650 serves as the 802.11b wireless link between the robot electronics and the students' laptops. Laptop to iPAQ communication is achieved over TCP/IP, with the resulting serial stream multiplexed between the CMUcam, which provides visual perception services, and the Brainstem network, which provides motion and sensing control. Note that the iPAQ is a very expensive 802.11-serial router in this application. In time, we hope to replace this costly choice with a PIC-based 802.11b controller.

### 4.2 Brainstem™ Architecture

In the Trikebot, the BrainStem network is primarily a slave controller. The student's laptop performs high-level decision making and sequencing, in turn requesting control outputs and inputs from the Brainstem network using a Java API. The BrainStem architecture offers rich I/O capabilities in slave mode but can also function independently via TEA (tiny embedded application) programs which use ANSI C syntax to run on small virtual machines located within the BrainStem module's controller [10].

The Trikebot's steering and camera pan/tilt servos are driven by the BrainStem GP 1.0 module. This board also supports the GP2D02 IR distance ranger. Both of these tasks are managed by the GP 1.0 module which encapsulates the serial clocking of data from the digital IR sensor, dampens the motion input to the servos, and manages the servo ranges and offsets.

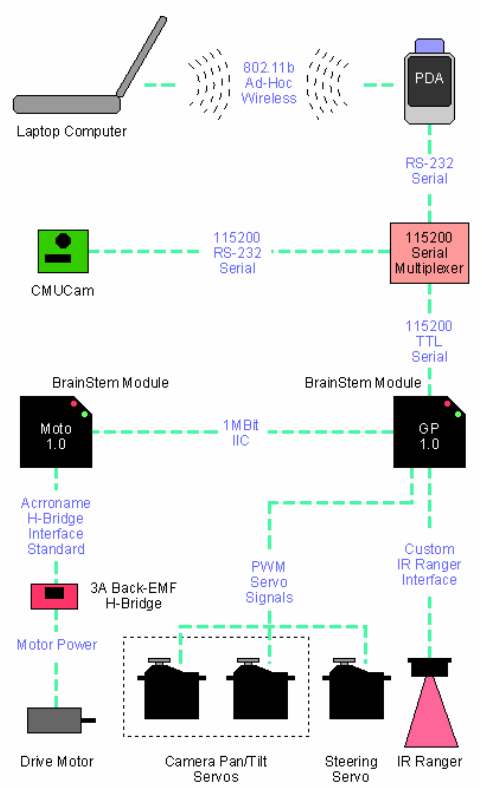


Figure 4: *The Trikebot control electronics' connectivity*

The GP 1.0 module also acts as a serial to I2C router to communicate with the other BrainStem Module, the Moto 1.0 board. This approach allows all commands to be sent to the BrainStem I2C network via a single serial connection. The Moto 1.0 module handles the closed-loop motion control of the Trikebot's motor using an H-Bridge daughterboard.

### 4.3 Back-EMF based speed control

One unique ability of the H-Bridge and Moto 1.0 module used in the Trikebot is Back-EMF speed measurement. This approach uses the natural characteristics of a spinning motor to derive a feedback voltage that is linearly proportional to the speed of the motor.

Most precision robotics applications use motors with encoders offering quadrature position sensing. This approach is effective but the combination of the precision encoders and quadrature decoding chips on the motion controller makes this approach costly. Using

Back-EMF control allows feedback-based PID speed control while using a simple gearmotor with no encoder.

The basic idea behind Back-EMF speed control is that while a motor is being driven, the H-Bridge windings that actually offer the connection to the drive current for the motor can be "floated" or left disconnected. When this occurs, the induction developed in the windings of the motors quickly collapses and the motor transitions to a generator of current due to the residual inertia in the mechanical drive system. Once this transition has occurred, the output voltage from the motor is directly related to the speed of the motor.

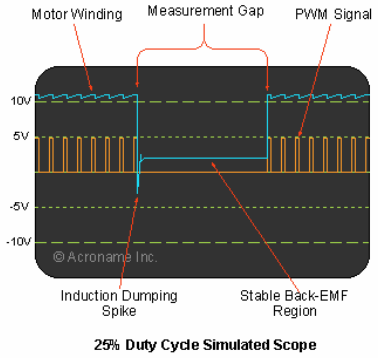


Figure 5: *A single back-EMF speed measurement*

The Back-EMF circuit built into the 3A H-Bridge used in the Trikebot's Moto 1.0 board measures the voltage from the motor and converts it to a logic voltage centered at 2.5 volts. When the motor is running full speed in one direction, the voltage drops to ~0.0 volts and, in the other direction, the voltage rises to ~5.0 volts. This is read by a 10-bit analog input on the Moto 1.0 module and used as the feedback for the PID loop. After the A/D measurement, the motor is again driven via PWM. As shown in Fig. 5 , the motor winding initially dives as the induction developed in the motor collapses. This inductive drop is rapid and thereafter the motor begins generating a current and a voltage proportional to the speed of the spinning motor. The Back-EMF circuit can then measure voltage to infer motor speed. This concludes the measurement gap in the PWM signal and then the Moto 1.0 resumes driving the motor via PWM.

Note that these diagrams show one side of the motor winding that we are measuring for a given direction. In a typical H-Bridge arrangement, one side of the motor is grounded while the other is rapidly switched from floating to powered. Changing direction reverses the roles of these two sides of the motor connections in this arrangement. Also note that the motor can maximally be driven at only 95% duty cycle as some dead time is always required for the Back-EMF measurement gap.

As with any PID system, proper tuning of the parameters is required and expanded to include the frequency of the Back-EMF reading, the delay in the measurement gap for the measurement, and the DC offset of the feedback signal. These parameters are exposed (Fig. 6) on the Moto board and are adjusted with a user-interface available for numerous host computers. Once configured, the settings are saved to an EEPROM on the Moto module. As a fail-safe precaution, the EEPROM can be actively locked (write protected) with a jumper.

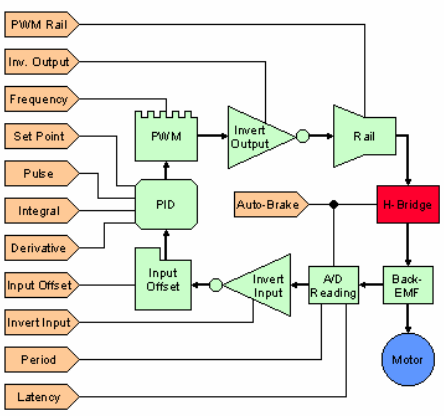


Figure 6: *Back-EMF PID tunable parameters*

### 4.4 iPAQ Robot Server

The iPAQ ARM-based processor serves as both an 802.11b to serial bridge and a real-time controller on-board the robot. There are a number of reasons to avoid placing the student laptop directly onboard the Trikebot. First, reducing the payload requirements enables a longer running time for the robot and reduces the chances of damage in the case of collisions. Second, the laptop provides direct diagnostic feedback to the student during program execution. Finally, an off-board laptop can serve as a teleoperation input device.

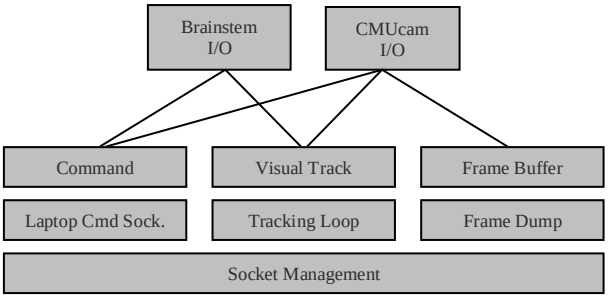


Figure 7: *Functional layers of the iPAQ firmware*

The fundamental problem of removing the laptop and thus the high-level control program from the Trikebot concerns communication latency. Even in the best of cases, roundtrip communication latency via 802.11b can

easily exceed 150 ms. This is too slow for fast-feedback control loops such as visual pan-tilt tracking of moving objects using the Trikebot's CMUcam.

In addition to providing communication services to each downstream electronic device (Fig. 7), the iPAQ serves three other functions. When the laptop requests an image dump, the iPAQ acts as an intermediate image buffer to collect and send that information via a dedicated image socket. Second, the iPAQ can serve as a pan-tilt feedback controller, utilizing CMUcam to measure the visual displacement of a tracked object, then commanding the pan and tilt servos to visually center the object. Feedback to the laptop regarding the tracked object and the Trikebot's neck position uses a separate TCP/IP socket. Third, the iPAQ uses a timer for each servo to power down the servo once it has reached the commanded position, saving power.

## 5 Robot Control and Programming

### 5.1 Goals

As the interface between student and robot, the laptop environment is critical for students to learn successfully and enjoyably. One objective is that the interface enable students to directly control the robot's motion as easily and quickly as possible. Second, assuming basic programming skills a student should be able to program the Trikebot for autonomous motion with as shallow a learning curve as possible. The goal is to rapidly surmount the obstacles of learning machine-specific programming and compilation details, instead devoting the intellectual effort to exploring the space of autonomous and interactive robot behaviors. Finally the third objective is that the interface should provide maximal diagnostic transparency during program execution so that the student is empowered to improve the performance of the Trikebot (Nourbakhsh 2000a).

### 5.2 Control and Diagnostic UI

The Trikebot UI, shown in Fig. 9, enables direct teleoperation of the Trikebot. This is critical to the ongoing diagnostic process for students. By dumping images from the Trikebot's CMUcam, for example, students can visually inspect the quality of the video signal on which they are attempting computer vision operations.

At the control level, the UI enables the student to drive the Trikebot directly, control the head's pan/tilt position and dump images from CMUcam. During each of these control operations, the interface displays and continuously updates the same sensor values that students use during programming: motor speed, current and rangefinder distance.

The Trikebot UI was implemented outside of any high-overhead IDE, ensuring that the finished product can be compiled and executed using simple command-level calls in Java 1.4 or beyond. This ease of compilation is key to the *User Controls* window that is also part of the UI (Fig. 9). This window provides the student with a series of buttons and input/output textfields so that, without spending time on GUI development, the students can launch their programs, observe Trikebot state during program execution and halt their programs from the UI. This coupling of the teleoperation and control UI to the buttons and fields used to interact with student code is a critical aspect of the success of the Trikebot as an educational, programmable robot. The complete JAVA source file-set is available at [12].

### 5.3 Programming Interface

Although the JAVA client for the Trikebot UI spans a large number of source files, the *User Controls* panel is implemented as a separate source file. In order to write the JAVA functions that are triggered when those buttons are pressed, the students modify a second contiguous block in one other file (Fig. 8). Students are thus able to program the Trikebot by making direct modifications to two files using a text editor such as JEXT [13], then compiling and executing from a command line using javac and java. This programming process removes the complexity of teaching students to use an all-purpose IDE such as FORTE.

```
private void Action1() //dumb wander
 {
  int refreshes = 0;
  String Debugging;
  theWindow.quit = false;
  while (theWindow.quit == false)
  {
   trikebot.RefreshState();
    //get the state variables
   if(trikebot.state.Range() <= 150) {
     trikebot.Drive(20,0);
    //wander forward at a slow speed
   } else {
     trikebot.KillMotor();
   }
  }
 }
```

Figure 8: *An example of a student code fragment from the summer 2002 course.*

In previous work the authors have taught robot programming using LISP, C, C++ and JAVA. The most effective language was LISP because of its functional nature, similar in spirit to recent robot languages such as GRL [14], and because of the Listener Window interactivity. This ability to execute a portion of robot code in order to diagnose surprising robot behavior is extremely important in robotics, and the Trikebot's UI serves this purpose.

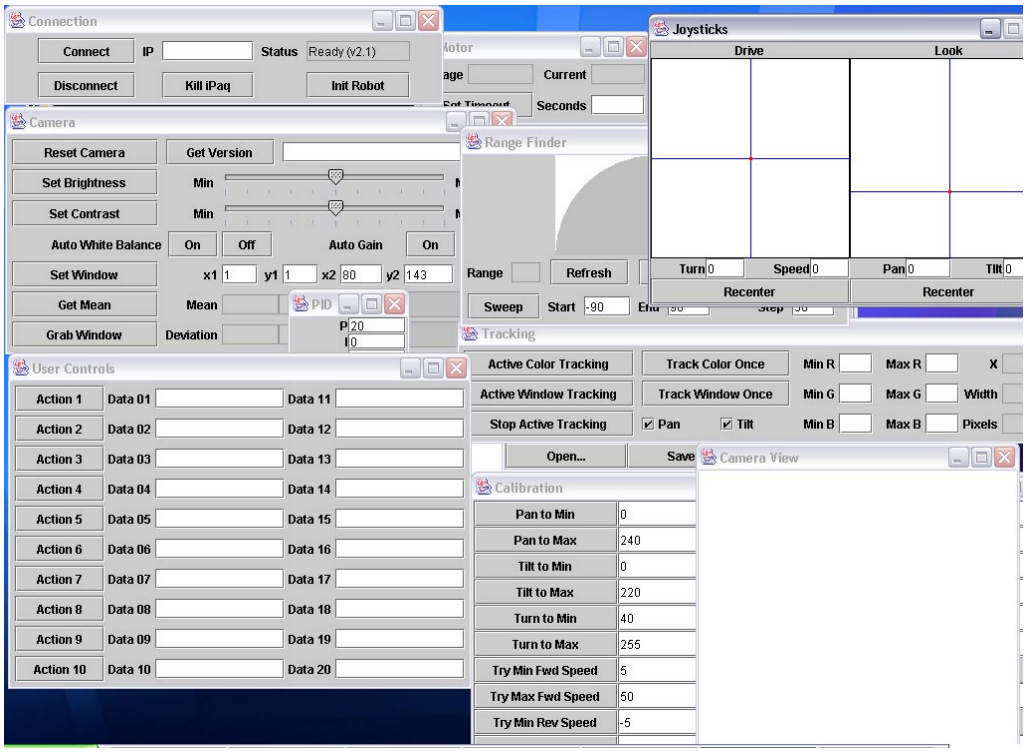


Figure 9: *The Trikebot laptop UI*

## 6 Conclusions

The Trikebot was designed, fabricated and tested with thirty students in an ongoing project to inspire youth with regards to science and engineering and provide an ongoing creative outlet for robotic exploration. Physical and electronic robot innovations enabled an affordable price point while achieving necessary robustness for daily use. The simple JAVA programming environment has demonstrated sufficient richness for challenge-based curriculum assignments. We hope that this effort will prove useful for future educational robotics endeavours.

## Acknowledgements

The back-EMF speed control solution was first proposed by Randy Sargent and demonstrated at EPFL AMR-DST. Funding was provided by NASA-Ames Research Center. Thanks to Adriana Cardenas, Maylene Duenas, John D'Ignazio, Neeti Malhotra, Raj Reddy and Peter Zhang.

## References

[1] RASC 2003. Web reference: *http://www.cs.cmu.edu/~rasc*

[2] Beer, R., Chiel, H., Drushel, R. Using autonomous robots to teach science and engineering. *Communications of the ACM*, June 1999.

[3] Kumar, D. & Meeden, L. A robot laboratory for teaching artificial intelligence. In *Proc. of 29th SIGCSE Symposium on Computer Science Education*, 1998.

[4] Murphy, R. *Introduction to AI Robotics.* MIT Press, 2000.

[5] Nourbakhsh, I. When students meet robots. Essay in *IEEE Intelligent Systems and Their Applications*, 15(6), p15. 2000.

[6] Nourbakhsh, I. Robotics and education in the classroom and in the museum: On the study of robots, and robots for study. In *Proceedings Workshop for Personal Robotics for Education.* IEEE ICRA 2000.

[7] Wolz, U. Teaching design and project management with Lego RCX robots. In *Proc. SIGCSE Conference 2000*.

[8] Rowe, A., Rosenberg, C. and Nourbakhsh, I. A low cost embedded color vision system. In *Proceedings of IROS 2002.* August 2002.

[9] Fong, T., Nourbakhsh, I., Dautenhahn, K. A survey of socially interactive robots. *Robotics and Autonomous Systems* 42 (3-4), pp. 143-166, 2003.

[10] Acroname. Web reference: *http://www.acroname.com*

[11] Nourbakhsh, I. Property Mapping: A simple technique for mobile robot programming. In *Proceedings of AAAI 2000.* July 2000.

[12] TRIKEBOT. Source code download site. Web reference: *http://www.cs.cmu.edu/~rasc/RA/TrikeBackg.htm*

[13] JEXT. Web reference: *http://www.jext.org*

[14] Horswill, I. Functional programming of behavior-based systems. In *Proceedings IEEE International Symposium on Computational Intelligence in Robotics and Automation.* 1999.