Putting Low-Cost Commercial Robotics Components to the Test

Development of an Educational Mechatronics/Robotics Platform Using LEGO Components

BY VASILIOS PAPADIMITRIOU
AND EVANGELOS PAPADOPOULOS

In this article, we investigate the possibility of using low-cost commercial material as a means of learning, research, and experimentation in fields such as mechatronics, robotics, and automatic control. The capabilities and limitations of the selected platform, i.e., of the LEGO elements, are studied via two projects that were designed and carried out, including a number of enhancements that address hardware and software limitations. The first project involves a robotic vehicle that can follow predefined paths, while the second concerns two robotic vehicles cooperating in a specific task. Algorithms and additional hardware were developed and the overall results are presented. It was found that the platform is suitable for teaching many diverse issues of central importance in the areas of interest.

In the last decades, revolutionary technologies have allowed the development of innovations deeply affecting our lives. Mechatronics, a synergistic procedure of mechanical, electronics, software, and control engineering, is having a significant contribution in the design and conception of innovative and versatile products. Robotics is becoming more important both in industry and everyday life. Teaching mechatronics and robotics, with an emphasis on design, optimization, experimentation, and development of real working systems, can be facilitated by the availability of a flexible and yet inexpensive rapid prototyping system.

The well-known LEGOLab at Aarhus University [1] as well as similar efforts at Brown University [2] and Tufts University [3] employ LEGO elements for prototyping and implementation of various ideas in control, robotics, and computer science disciplines. In addition, student groups at the Department of Automation at the Technical University of Denmark create and program experimental rigs made of LEGO elements for control systems courses [4]. Martin outlines the LEGO Mindstorms capabilities [5], while Butler et al. [6] and Lau et al. [7] emphasize the versatility and originality of the LEGO Mindstorms. Lund and Pagliarini created Robocup Jr., a contest for children intended at building soccer robots using LEGO Mindstorms [8]. Recently, Fiorini reported the use of the LEGO components at the University of Verona [9].

Digital Object Identifier 10.1109/M-RA.2007.901322

1070-9932/07/$25.00©2007 IEEE

Teaching mechatronics and robotics, with an emphasis on development of real working systems, can be facilitated by the availability of a flexible and yet inexpensive rapid prototyping system.

In this article, we investigate the possibility of using low-cost commercial material as the basis for an educational platform for mechatronics, robotics, or control applications. Having chosen the LEGO elements, two major projects were designed and completed as illustrative examples of its educational strengths. The first involves a robotic vehicle following predefined paths, and the second concerns two robotic vehicles cooperating in a specific task. Algorithms, original and additional hardware developed, and simulation and experimental results are presented. The overall resulting capabilities and limitations are studied, and important system parameters are identified or reverse-engineered for the purpose of design and learning. To expand the capabilities of the platform, or overcome specific limitations, engineering improvements and replacements were explored. It is suggested that the platform employed is capable of fulfilling our training mission at a low cost and in a learning-friendly environment.

Mechatronics, Robotics, and Engineering Education

To improve the learning experience as well as to promote innovative thinking of newborn engineers, controls, robotics, or mechatronics courses can include projects based on real hardware and related software. Such projects may be either predefined in their scope, or free and focusing on creative applications of theoretical class material. In the predefined project case, teams must find reliable and effective solutions employing their creativity and engineering knowledge, without failing to overcome practical problems and limitations. In the free project case, additional creativity, ingenuity, and innovation are required for defining the problem and solving it. In both cases, teams must come up with effective solutions employing both theory and experimentation, using ingenuity bounded by feasibility considerations. In addition, such projects inevitably expand student knowledge in areas beyond their immediate interests.

In such courses, a basic question is what hardware set to use for the projects. Two basic options include (a) use of nonstandard, off-the-self, structural, actuator, and sensor elements, accompanied by standard microcontrollers, and (b) use of a standardized and to the extent possible complete set, such as the LEGO Mindstorms/Robolab.

In a mechatronics/robotics course at the National Technical University of Athens (NTUA), we have tried both options. We have found that option (a) offers a richer engineering experience since it is closer to reality. However, this option is more suited to courses that extend beyond a single term, as it takes more time to use effectively elements that are not necessarily meant to work together. In addition, used elements are usually not reusable and must be replaced every time the course is run.

On the other hand, using option (b) has in principle the advantage of a relatively complete set of elements, sensors, actuators, and microcontrollers; is reusable; and is of low cost. In this option, the basic building elements and equipment are given to students, allowing teams working in parallel to use or construct additional equipment like sensors or structural components. To implement a variety of sophisticated designs, appropriate low-cost, safe, and simple equipment meeting certain specifications and a variety of electrical, electronic, mechanical, and structural elements are needed.

At first glance, the LEGO components offer interesting possibilities. The Mindstorms kits, aimed at advanced users, or the Robolab kits, introduced by the LEGO educational division, include relatively low-cost hardware and software suitable for implementing complex designs in the areas of robotics, mechatronics, and control. Students can become familiar with the set quickly and later even design and build their own add-on components, such as microcontrollers, sensors, structural elements, etc.

However, an important question to answer is whether the use of such a set, not initially meant for teaching engineering, is capable of teaching issues related to mechanical design, actuators and transmissions, sensor interfacing and/or development, software development, communications, high-level planning, servo control, and artificial intelligence. To this end, we approached the problem from the engineering point of view. First, we identified how key elements work, and next we developed two relatively complex projects that demonstrate the possibilities of the developed platform and the issues one can expect to tackle and teach.

Understanding the RCX Microcontroller

RCX Internals. Hardware Overview

At the heart of the Mindstorms/Robolab system lays the Robotic Command Explorer (RCX), an integrated microcontroller that can be programmed using a PC. The RCX (see Figure 1) draws power either from six 1.5-V batteries or from a power adaptor and sports three input and three output ports, an infrared (IR) transmitter/receiver, four control buttons, and an LCD screen.

Its core microprocessor, the Hitachi H8/3292, is ideally suited for real-time control applications. The H8/3292 has a clock rate of 16 MHz at 5 V and incorporates 16 kB ROM and 512 B RAM, eight 10-b A/D converters, and a serial communication unit for synchronous or asynchronous data exchange. Also the RCX includes input port circuits, a chip for driving outputs, and an external 32 kB RAM.

The input ports sample digital or analog sensors, and the output ports drive electrical actuators, such as dc motors. The input port circuit (Figure 2) is designed to sample active sensors and provide them with power or sample passive sensors; i.e., sensors that need no external power. When the input port is in active mode, the electronic switch S remains closed and opens instantly for almost 0.1 ms, allowing the A/D converter to sample the sensor. The sampling rate depends on the RCX operating system (OS). Under the standard firmware, the sampling rate is 333 Hz, whereas under the alternative brickOS operating system, ports are sampled at a 5.5-KHz frequency.

The output ports control is carried out by the Melexis ELEX 10402 motor driver IC that implements an H-bridge topology. The chip includes two pairs of MOSFET transistors and protecting freewheeling diodes, [Figure 3(a)]. Four driving modes are implemented, depending on the status of its two digital inputs. Assuming that the actuator is a permanent magnet dc motor, the four modes include forward, backward, stall, and floating mode. In the forward mode, two of the four transistors conduct, while in the backward mode the other two transistors conduct. When stalled, the motor pins are shorted, while in the floating mode the pins are free. The main drawback of this IC is the 1.2-V drop voltage across the output transistors.

The RCX designers created software controllers for more sophisticated control of the output ports, based on the pulse width modulation (PWM) technique. The output voltage is a pulse train, with a period T, in which the voltage is equal to the voltage source V_s for t_{on}, and zero for t_{off} seconds. If the PWM period is significantly lower than the time constant of the motor, then the equivalent voltage V_o sensed by the motor is given by

$$V_o = \frac{t_{\text{on}}}{t_{\text{on}} + t_{\text{off}}} V_s = \frac{t_{\text{on}}}{T} V_s = DV_s. \qquad (1)$$

The duty cycle D ranges from 0 to 100%. Software functions controlling the output ports support eight duty cycle levels between 0 to 100% under the standard OS, with a discrete step of 12.5%. The brickOS supports 256 duty cycles with a discrete step of 0.39%.

To identify the parameters of the PWM under both OSs, two experiments were conducted. First, the output port pins were directly connected to an oscilloscope, and the response was captured for various duty cycles (power levels) 0–8. As an example, the response for level 3 is shown in Figure 4(a) (blue curve). The experiment was repeated with a resistive load R_L connected to the output [Figure 4(a) (green curve)].

As shown in these figures, the response is periodical with a period $T = 8$ ms. In both responses, the voltage remains stable (phase on) for 4 ms and then it drops (phase off).

The equivalent total internal resistor value r of the output port circuit was estimated as follows. A resistive load was connected to the output port and the current flow was measured. Various resistor values and duty cycles were applied, and using Ohm's law, the resistor value was found to be approximately $r = 6.5\ \Omega$.

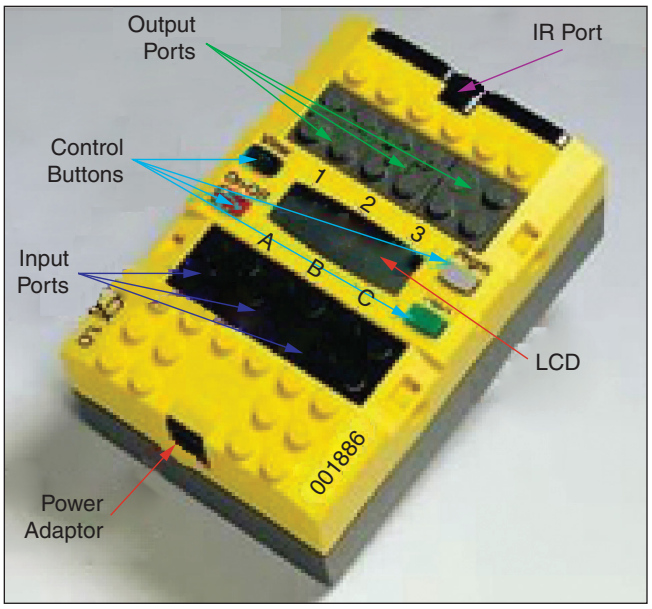

Figure 1. *The input and output ports of the RCX.*

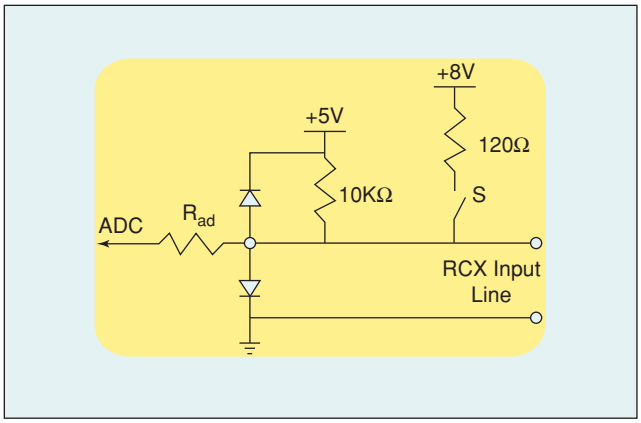

Figure 2. *The input port circuit in passive or active mode.*

In addition to the input and output ports, the RCX has communication capabilities. The brickOS (ex legOS) network protocol is a set of elementary routines supported by brickOS, enabling communication among two or more RCXs and remote PCs. The RCX can send or receive data through the IR transmitter/receiver port at a maximum distance of about 6 m and at a varying viewing angle. The communication link works at a maximum rate of 2,400 bps. In addition, data can be fowarded to specific agents (other RCXs or PCs), enabling sophisticated control of a team of agents.

RCX Software Architecture-Alternative Languages and OSs

The RCX is supplied with a standard firmware that lies in the external 32 kB RAM, occupying almost half of it, and providing high-level control and access to the resources of the integrated system. User programs are stored in a 6-kB memory space and the remaining 10 kB are used by the system for various purposes, such as temporary memory. The

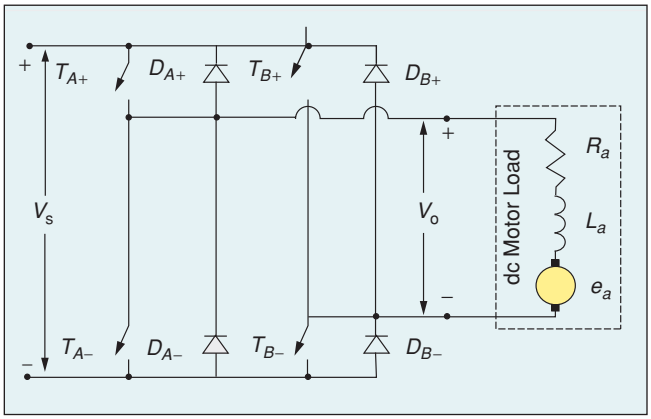

Figure 3. *Topology of a typical H-bridge.*

16 kB ROM incorporated in the H8 μC includes low-level routines for hardware control. User code is created and debugged on a PC running the Robolab environment. It is then downloaded to the RCX RAM through an IR tower and interpreted step by step by the bytecode interpreter lying in the firmware and executed. This procedure causes an execution delay that reduces its suitability for real-time applications.

The programming language is an essential matter as it affects a user's development capabilities. Robolab sets are accompanied with the graphical programming language Robolab, developed by the Center for Engineering Educational Outreach at Tufts University in association with National Instruments [10]. Here, symbolic icons representing functions and variables are dragged on a canvas and connected with one another to form a logical program. This language is easy to learn, but is difficult to use for complex programs and is subject to various limitations. Thus, the necessity for a script language with extended features emerges. The most successful script language working under the standard firmware is NQC [11]. NQC has a simple syntax and resembles C. However, it is subject to some limitations, such as the small number of exclusively integer variables and the lack of mathematical functions, matrices, or arrays.

Limitations originate not only from the language but also from the original firmware. To address these issues, LEGO enthusiasts created various alternative languages and operating systems [12]. Kekoa Proudfoot, a student at Stanford University [13], fully decoded the system ROM, allowing others to produce new software. Today, the most powerful alternative OSs include the brickOS [14], the lejOS [15] (which enables programming in Java), and the pbForth allowing programming in Forth [16]. The brickOS was designed to allow programming in C or C++ as well as to maximize the hardware's exploitation. It occupies less RAM memory space (maximizing user program space), uses the H8 processor at full clock speed, and maximizes the input sampling rate at 5.5 KHz. In addition, user-downloaded programs are precompiled to machine code and execute faster than in other OSs. The language supports five variable types including floating and double point; an unlimited number of variables, arrays and matrices; and functions for RCX-PC IR communications.

Basic Hardware Reverse Engineering

As already mentioned, the RCX has three output ports used to power and control electrical devices like LEGO motors. The most efficient, compact, and powerful LEGO motor is the high-torque 9-V permanent magnet dc gear motor.

Motor Identification Experiments

To identify key motor and rotation sensor parameters, specific experiments were conducted. The simplified dc permanent magnet motor model includes an internal resistance R_a and a back-electromotive force (EMF) that occurs in electric motors where there is relative motion between the armature of the motor and the external magnetic field connected in series and described by the motor torque constant k_T. These parameters are estimated next.

The motor electromechanical equations are

$$V_s = k_T \cdot \omega + i \cdot R_a, \tag{2a}$$

$$\tau = k_T \frac{V_s}{R_a} - \frac{k_T^2}{R_a}\omega, \tag{2b}$$

$$\tau_{\max} = k_T \cdot V_s/R_a, \quad \omega_{\max} = V_s/k_T, \tag{2c}$$

where τ is the developed electromagnetic torque, ω is the shaft angular speed, V_s is the driving voltage and the subscript "max" indicates the maximum value of a variable.

Considering (2a), both parameters k_T and R_a can be estimated using the least squares method, with a number of mechanical loads applied to the motor under constant terminal voltage V_s and the angular velocity ω and the current i of the motor measured. The experimental setup is comprised of a capstan-equipped motor lifting various weights. The angular velocity ω is measured with a tachogenerator and scope, and the current i is measured with a multimeter. It was found that $k_T = 0.22$ Nm/A, and $R_a \simeq 24\,\Omega$. Supplying the RCX by a 9-V power supply, the zero load current was $i_o = 10$ mA, the stall torque in (2c) was $\tau_{\max} \cong 80$ mNm, and $\omega_{\max} \cong 390$ r/min.

Finally, the torque-speed characteristic was determined with the motor driven at the various RCX power levels. Simplifying Figure 3 to the one in Figure 5, during phase on, transistor T conducts and the motor is connected through an intermediate circuit to the batteries. In phase off, transistor T opens and the motor is in floating mode, causing current i to become zero. The scope connected to the motor terminals showed that during phase off voltage response is nonzero, matching the response of the motor back-EMF [Figure 4(b)]. This implies that during phase off, all transistors in Figure 3 remain open. Assuming that level X, out of the eight available, is used, phase on is applied for $X/8$ ms and phase off for $(8–X)/8$ ms. Using Ohm's law in both cases, the voltage across the motor terminals is

$$V_m = \frac{X\,(V_s R_a - k_T \omega R_a)}{8(R_a + r)} + k_T \omega, \;\; X \in \mathbb{N}. \qquad (3)$$

Substituting (3) in (2b), the torque becomes

$$\tau_{\text{mot}} = (V_s - k_T \omega)\frac{k_T}{8(R_a + r)} X, \quad X \in \mathrm{N}, \qquad (4)$$

where V_s is the battery voltage and r the internal resistance of the power supply circuit. The identified torque-speed characteristic for the eight power levels is shown in Figure 6. What is striking here is that, due to the method used in driving the motors, the characteristic curves for a constant voltage have the same maximum speed but different stall torques.

A final experiment was conducted to confirm the results predicted by (4). A motor was connected to the RCX, and various power levels were imposed for medium loads. The motor angular velocity and current flow were measured for each power level. The resulting experimental characteristic curves confirmed (4) with deviations less than 5%.

Sensor Identification Experiments

The RCX input ports are in position to sample various sensors, either LEGO or homemade types. The compact LEGO light sensor is used to measure the incident light intensity. An LED diffuses red and infrared light to the environment, and a phototransistor converts the incident light to an electric signal. It can also distinguish colors, such as white, black, or red.

The most sophisticated LEGO sensor is the rotation sensor, which is a shaft encoder with a resolution of 16 pulses/rotation and a speed operation limit around 6,000 r/min.

The upper speed limit for reliable and accurate operation was estimated using a synchronous motor revolving at 250 r/min and various gear-train assemblies. First, the sensor was tested when connected to the RCX running the standard OS, at speeds up to 1,250 r/min (the sampling rate under this OS is 333 Hz). It was found that the sensor's accuracy decreased dramatically above 1,000 r/min, confirming the fact that the sampling rate is quite poor. The same experiment was conducted by replacing the RCX OS with brickOS. The sensor was tested at a maximum speed of 3,125 r/min, and it was found that the sensor's relative error was steadily around 1–2%. Conclusively, based on these experiments, the sensor's accuracy is considered satisfactory at speeds up to 3,000 r/min.

The aim here is to construct a robotic vehicle to scan and read data from a Kandinsky painting and then interact with it, drawing lines, according to an intelligent algorithm.

Hardware Platform Design and Development

Model-Based Control of a Differentially Driven Platform

The first project developed involves the design and implementation of a high-level closed-loop control law to a dynamic system constructed of LEGO elements. The system's response was measured and evaluated.

A differentially-driven nonholonomic platform, called the Explorer, was designed and built with LEGO elements. Its task is to follow a predefined path, such as a sinus curve, with a desired velocity profile. The platform employs two sensor-equipped dc motors driving the wheels as well as a caster [Figure 7(a)]. The transmission system uses plastic chains and sprockets and is designed to minimize backlash. In addition, a transmission system, see Figure 7(b), connecting the wheel to the rotation sensors was designed to increase the sensor speed with respect to the wheel by a factor of 15 and the sensor resolution from 16 pulses/rev to 240 pulses/rev.

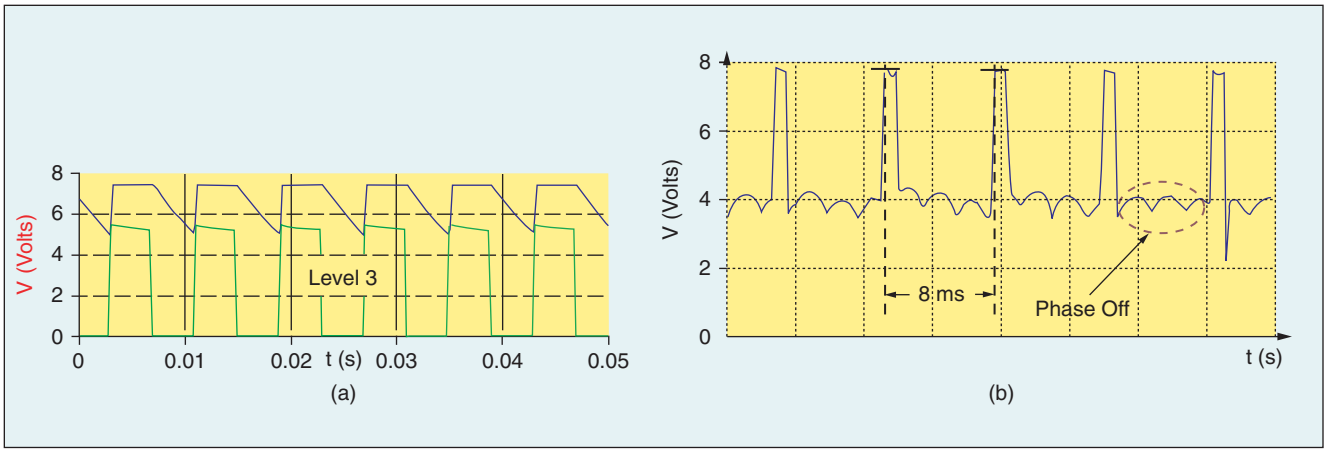

Figure 4. *(a) Voltage response of output port at level 3 under zero load (blue curve) and under a resistive load R_L (green curve). (b) Output voltage across motor pins subjected to a medium load.*

Considering Figure 8, the Cartesian velocity $(\dot{x}_F, \dot{y}_F)$ of vehicle point F can be written as a function of the two wheel angular velocities $\dot{\vartheta}_\ell, \dot{\vartheta}_r$ via a Jacobian matrix $\mathbf{J}$, which is a function of model geometric characteristics and orientation

$$\dot{\mathbf{x}}_\mathbf{F} = \begin{bmatrix} \dot{x}_F \\ \dot{y}_F \end{bmatrix} = \mathbf{J} \cdot \begin{bmatrix} \dot{\vartheta}_\ell \\ \dot{\vartheta}_r \end{bmatrix}. \tag{5}$$

The dynamic model used is in a useful form since it links the input motor torques τ with the accelerations $\ddot{\mathbf{x}}_\mathbf{F}$ [17]:

$$\mathbf{M}\ddot{\mathbf{x}}_\mathbf{F} + \mathbf{V} = \mathbf{J}^{-1} \cdot \tau, \tag{6}$$

where $\mathbf{M}$ is the inertia matrix and $\mathbf{V}$ is the vector of velocity-dependent forces. Equation (6) can be used to build a trajectory following model-based control law [18]. The torques are calculated as

$$\tau = \mathbf{J}(\mathbf{M}\ddot{\mathbf{x}}_\mathbf{F}^* + \mathbf{V}), \tag{7}$$

where the auxiliary accelerations $\ddot{\mathbf{x}}_\mathbf{F}^*$ are given by

$$\ddot{\mathbf{x}}_\mathbf{F}^* = \ddot{\mathbf{x}}_\mathbf{Fd} + \mathbf{K_v}(\dot{\mathbf{x}}_\mathbf{Fd} - \dot{\mathbf{x}}_\mathbf{F}) + \mathbf{K_p}(\mathbf{x}_\mathbf{Fd} - \mathbf{x}_\mathbf{F}) \tag{8}$$

and $\mathbf{K_v} = diag\{k_v\}$, $\mathbf{K}_p = diag\{k_p\}$ are diagonal gain matrices. Assuming exact parameter knowledge, the tracking error $\mathbf{e} = \mathbf{x_{Fd}} - \mathbf{x_F}$ converges to zero asymptotically.

The implementation of this controller is done as follows. For a desired point F velocity, wheel desired angular velocities are derived by inverting $\mathbf{J}$. Using (7), control torques are calculated, and using (4), the corresponding power levels $X(t)$ are set. However, standard firmware supports only eight discrete power levels $X(0–7)$, while brickOS supports 256. As will be discussed later, these severe limitations were taken into account and proper solutions were given.

The Kandinsky Project

The core concept of this project, called the Kandinsky Project, was suggested to the authors by Prof. M. Santorineos of the School of Fine Arts. Kandinsky's paintings are based on the use of geometric primitives, such as triangles, rectangles, lines, circles, or ellipses filled with various colors. The aim here is to construct a robotic vehicle to scan and read data from a Kandinsky painting and then interact with it, drawing lines, according to an intelligent algorithm.

The developed scenario includes two autonomous robotic vehicles, a remote PC, and a simple painting acting as the workspace. The painting workspace contains a few basic geometric shapes, called targets, filled with a single color each, and unknown to the agents. Vehicle 1, the Explorer, is responsible for exploring the workspace and collecting all necessary data to fully identify the geometric characteristics and topology of the targets. Vehicle 2, the Painter, acts on the painting after having received and processed information about the characteristics of the workspace. The host PC is a remote computational node performing all necessary heavy calculations, overcoming the limited computational capabilities of the two mobile agents.

The Explorer [Figure 7(a)], is a differentially driven platform that includes a front color sensor and two rotation sensors connected to the wheels through gear trains. These yield a $22.5 \times 16 = 360$ pulses/wheel rev. The maximum wheel speed is restricted to 110 r/min, yielding a sensor speed of 2,475 r/min (less than the 3,000 r/min limit). The color sensor identifies target colored areas, while the vehicle's Cartesian position is provided by rotation sensor odometry.

The robot-painter [Figure 9] is a six-wheel double car-like vehicle that provides stability during the drawing process. The car includes two motors, one for steering and one for motion. Two rotation sensors are employed for odometry and navigation.

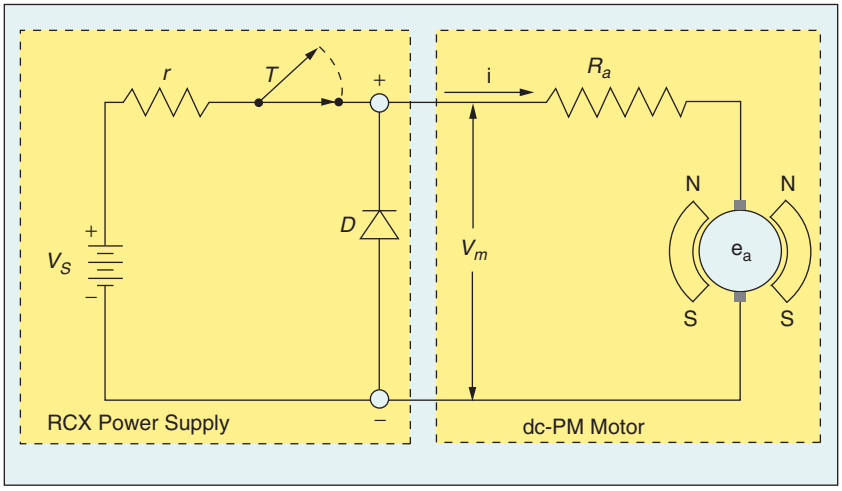

Figure 5. *The two phases (ON-OFF) of the motor driven by the RCX.*

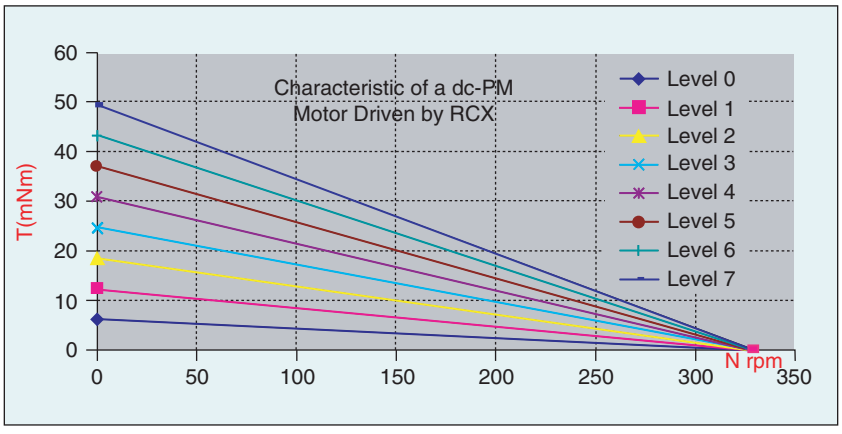

Figure 6. *Characteristic of a dc-PM motor driven by the RCX.*

This vehicle has two coupled pairs of steering systems that allow sharp vehicle turns. Each directional system is comprised of a worm-crown unit that rotates two links, which displace an Ackerman-style steering system forcing the wheels to turn. Another gear unit powers an opposite-placed directional system, resulting in front and back wheel turns of equal but opposite angles. As a result, the radius of rotation of the car is small. The pair of wheels in the middle of the vehicle is powered. During turning, a differential splits power to these wheels. To avoid the use of obstacle detection sensors and algorithms, the Explorer and the Painter work one at a time.

A color sensor was needed for this project. The sensor had to meet certain specifications, most important of which was the ability to distinguish six colors reliably: white, black, green, blue, yellow, and red. In addition, the sensor had to be lightweight, compact, and draw little current. The sensor is supposed to evaluate target-reflected light, when the target is illuminated by a full visible spectrum source of light (400–660 nm). A monochromatic target ideally absorbs all the visible spectrum zone of the incident light, except from the zone that corresponds to its color. Ideally, an illuminated target reflects back to the photodetector light that corresponds to its color while the photodetector should respond to each color univocally. However, various factors affect the system's efficiency, such as the reflectivity of the target's material, the angular and planar misalignments of the sensor-source of light, the environmental light, and mechanical vibrations.

The sensor comprised four parts; i.e., the sensing device, the source of light, the electronic circuit, and the power unit. Photodetectors, such as photodiodes and phototransistors, consist of a semiconductor photosensitive layer that transducts incident light to current, depending on light wavelength and intensity. The photoresistor's resistance changes according to the intensity and wavelength of the incident light. A more sophisticated sensor uses three photodiodes, one for each of the three basic colors [red, green, blue (RGB)]. The photosensitive layers of these photodiodes are covered by a filter, allowing only a specific color zone light to pass. Figure 10 shows the current response of the RGB photodiodes as a function of light wavelength.

Candidate light sources include the incandescent bulbs, the fluorescent lamps, and the white LEDs. The incandescent bulb intensity–wavelength response is a monotonous increasing function of wavelength and approximates that of daylight. Fluorescent lamps have a discontinuous response with discrete narrow zones of radiation. White LEDs radiate light in the visible area across the infrared area. The measured response depends on both the light emitted and the sensor characteristics.

To choose appropriate sensors, experiments were conducted that included a Cadmium Sulfur (CdS) photoresistor, an SIlicon photodiode with an IR filter, and a mazeT RGB color photodiode, while the light source included an incandescent void lamp, an incandescent xenon lamp and a white LED. In the experiments, six colored targets were used and sensor geometry parameters were varied to obtain maximum

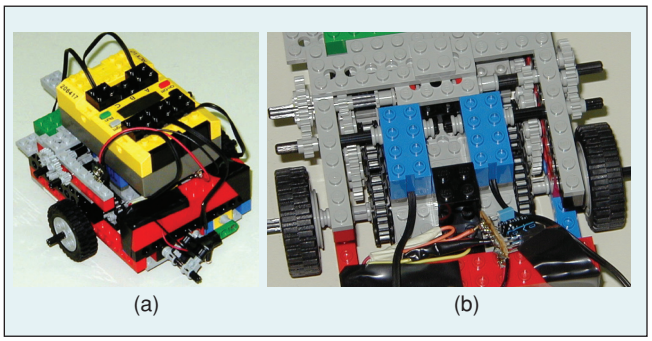

Figure 7. *(a) Overview of the Explorer. (b) The vehicle transmission and the rotation sensor gear system (blue elements).*

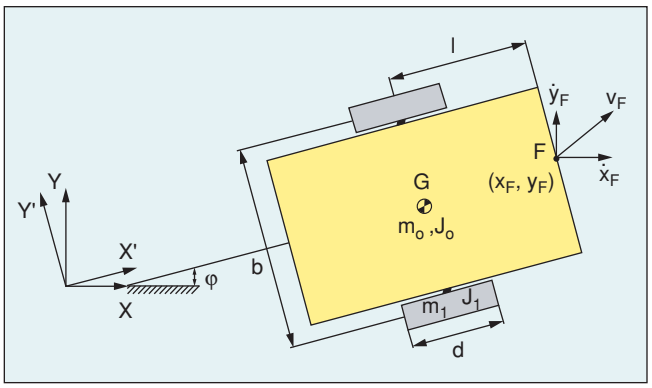

Figure 8. *Schematic of a differentially driven nonholonomic platform.*

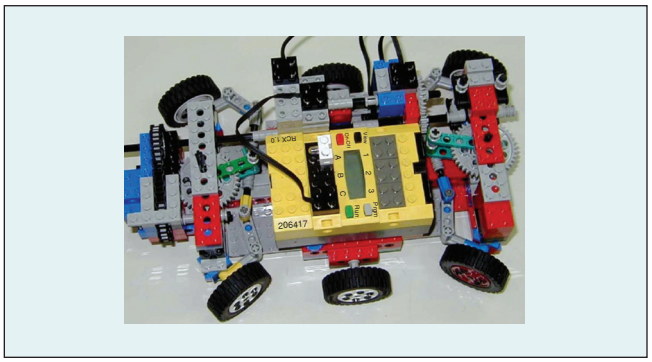

Figure 9. *Overview of the Painter.*

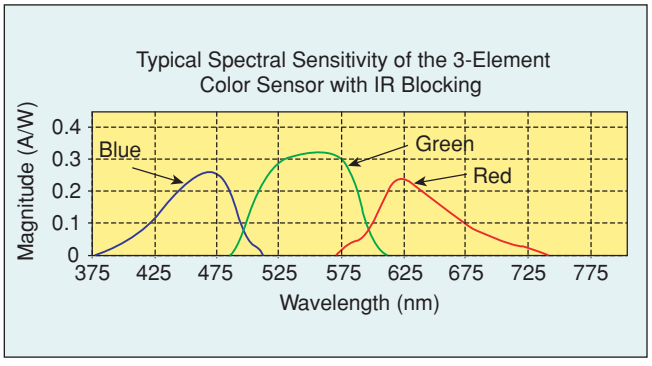

Figure 10. *Spectral response of an RGB photodiode.*

Is the use of a set not initially meant for teaching engineering capable of teaching issues related to mechanical design, actuators, sensor interfacing, software development, high-level planning, and artificial intelligence?

voltage differences identifying colors. Experiments with Labview showed that the three-diode sensor, sampled in Labview, had by far the best performance. More specifically, the white LED improved measurement accuracy and with it, all six color targets could be recognized.

An electronic circuit was developed to sample the RGB diode sensor and to interface it with the third input port of the RCX on the Explorer. The circuit is comprised of three op-amps, a PIC microcontroller (μC), a voltage regulator, a 9-V battery, and indicator LEDs. The μC samples the diodes and compares the values to the range for each color stored in its EEPROM. Using certain criteria, a decision is taken, and using a pseudo-analog output (PWM), the μC sends a voltage to the RCX univocally representing one of the six colors.

Algorithms and Simulations

Model-Based Control of a Differentially Driven Platform

Simulations were performed in Matlab/Simulink to design an open-loop controller for path following. The standard firmware supporting eight discrete power levels was used and the desired RCX power levels for a given path were calculated. In Figure 11(a) the blue continuous curve $X(t)$ corresponds to the power levels as calculated using (4).

However, these are in a continuous form and thus are not usable; a discretization method is required. The simplest method is to round the power levels [Figure 11(a) (black curve)]. The obtained discretized power levels along with (4) and (6) generate the resulting trajectory. However, the deviation from the desired trajectory is quite severe and therefore a new approach is needed.

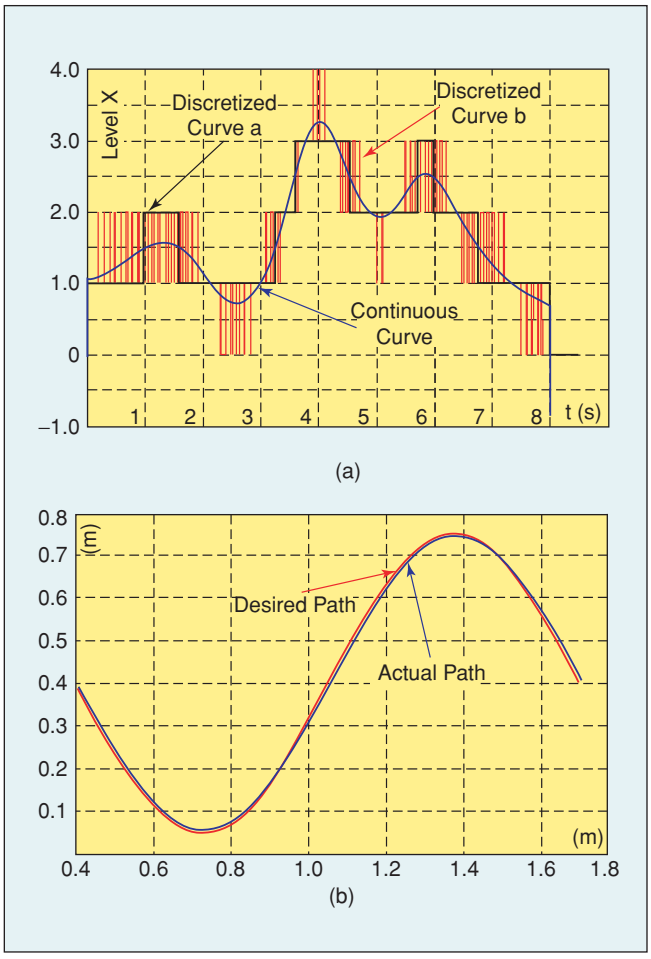

Figure 11. *(a) Continuous and discrete power levels X. (b) Simulated desired and actual path under the open-loop law.*

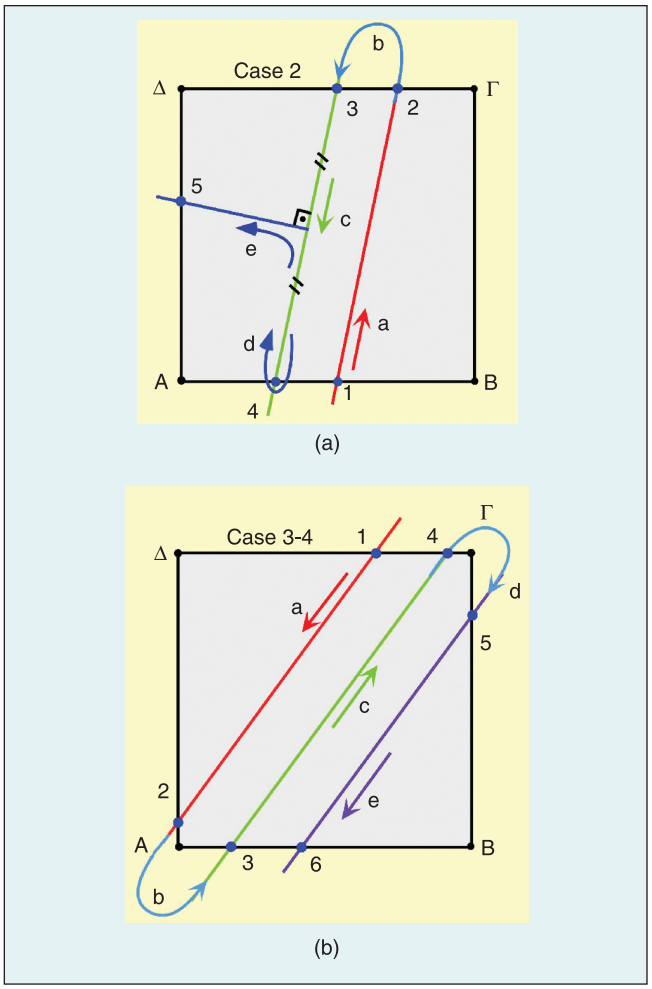

Figure 12. *Examples of vehicle data-gathering paths.*

To this end, the continuous (blue curve) is divided in κ equal time intervals of T seconds duration each, and for each interval $(\kappa T, \kappa T + T)$, the average value of $X(t)$ is calculated. The minimum power level $X_{\min}$ represents the integer value that is always lower than $X(\kappa T, \kappa T + T)$. The $X_{\max}$ is defined similarly. To determine the times $t_{\min}$, $t_{\max}$ during which $X_{\min}$, $X_{\max}$ are applied, the following system is solved:

$$\left.\begin{aligned} X_{aver} &= \tfrac{1}{T}\int_{\kappa T}^{\kappa T+T} X(t)\,dt = X_{\min}\,t_{\min} + X_{\max}\,t_{\max} \\ t_{\min} &+ t_{\max} = T \end{aligned}\right\}. \quad (9)$$

The discretized curve is shown in Figure 12(a) (red curve). To further optimize the results, an algorithm is created that tests in each interval all possible results of time durations $T_{\min}$, $T_{\max}$ of $X_{\min}$, $X_{\max}$ and selects the combination that yields the minimum trajectory error. The results are shown in Figure 11(b). The desired trajectory is sinusoidal of length 1.3 m and amplitude of 0.35 m. The resulting errors do not exceed 5–7 mm.

The Kandinsky Project

Having completed the spot color sensor design, a target recognition methodology was developed. Four types of targets were selected: blue circles, red squares, yellow equilateral triangles, random green quadrilaterals, all pasted on a white workspace. It was assumed that the color sensor recognizes the color of a single point at (x_i, y_i). During vehicle motion, the sensor locates points at the periphery of each colored target. This ability is used to identify the kind and location of the target in the workspace. Indeed, the Explorer is programmed to collect the minimum necessary number of points at the periphery of each target to estimate the vertex coordinates of polygonal shapes, or the center coordinates and radii of circles. The vehicle identifies the type of the shape by its color and performs appropriate predefined maneuvers, according to analysis studies, collecting the necessary number of data points for further processing. For example, in the case of a circle, geometry suggests that only three circumference points must be collected by the explorer to solve for its radius and location.

The identification of a square target is more challenging since there is no unique algebraic equation representing its shape. Instead, a system of equations may be used taking into account two types of restrictions; i.e., all sides must be equal in length and vertical to successive sides. When the vehicle identifies the first point P_1 at the square's periphery [see Figures 12(a),(b)], it continues moving straight until it senses a color alteration. Then, it collects data for the second point P_2. It is completely unknown on which side P_2 lies, meaning that P_2 may be on a parallel or a vertical side with respect to the side of P_1. The vehicle is then programmed to make a U-turn and move straight in parallel to its first path collecting two more points. After the vehicle has collected four points, the relation between vectors $\overrightarrow{14}$, $\overrightarrow{23}$ is determined. These topological uncertainties lead to several possible cases.

Having completed successfully two complex projects, we can evaluate the feasibility of using the LEGO platform for teaching robotics/mechatronics and control issues.

Three main cases with several subcases exist. In Case 1, vectors $\overrightarrow{14}$ and $\overrightarrow{32}$ are vertical, in Case 2 they are parallel, and in Case 3 they intersect, yielding thus 10 distinct subcases. For instance, in Case 1, the path followed by the Explorer is the $a \to b \to c \to d \to e$, so that the system of equations yield a unique solution.

In the case of an equilateral triangle the same methodology is followed and several cases result. Finally, a random convex polygonal shape has no special geometric constraints and thus a more common method is pursued. When such a shape is detected, it is scanned with parallel linear paths and many data points on its periphery are collected. The system is then easily solved.

Another critical issue concerning the Explorer is the wandering technique in a confined workspace. The goal is to build a simple algorithm that can be implemented in brickOS minimizing computational and memory needs. In addition, manipulated randomness is chosen to determine the progress of the procedure. The most obvious technique is to program the vehicle to scan the workspace uniformly, for instance in parallel or diagonally, but this technique yields long times.

Given the LEGO hardware constraints, it was decided to use sequences of on-the-spot turns of φ_o degrees and straight motions for ℓ_o meters. At first, an algorithm determines in random the values (φ_o, ℓ_o) allowed to fluctuate between specific limits, using pseudo-random functions. This technique is very simple and quite effective but is based on fixed pseudo-random functions (Matlab in simulation, brickOS in experiments) that may yield low performance (identical repeated number sequences). Another serious drawback is that already explored minor areas are not excluded for the rest of the searching procedure, decreasing its effectiveness.

For these reasons, the workspace is divided in 4ℓ-side squares, with ℓ defined in Figure 7, forming virtual identical small hives, and each of them having a condition value. If the vehicle crosses an area-hive, then its condition value changes, becoming an undesirable destination. The selection of the final destination for every movement cycle (turn–straight movement) is made randomly between groups of neighboring hives, excluding the undesirable ones. Having selected a hive, a final destination point in it is chosen. This way, the wandering becomes more effective and fast but requires more computations and memory space for saving hive conditions.

Simulations conducted in Matlab were made to verify the target recognition methodology and optimize the wandering algorithms. The inverse kinematic model of the differential platform, a trapezoidal angular velocity profile for each

wheel, real maximum velocity data, and realistic parameter values, were used. At workspace limits, the vehicle was programmed to rotate on the spot, admitting an orientation normal to the boundary line, and then move away for a random distance. Figure 13(a) shows the performance of the wandering algorithm.

After having found a square, the vehicle performs successfully the necessary motions for a square (Case 3), and the equations yielding the square coordinates are solved successfully, resulting in target identification [Figure 13(b)].

Experiments and Results

Model-Based Control of a Differentially Driven Platform

In an experiment, the open-loop controller is used for a sinus trajectory length of 1.3 m and amplitude of 0.35 m. The response is satisfactory for the first quarter, but then, due to parameter errors such as the vehicle's moment of inertia around the z-axis, friction, and RCX limitations, it deviates significantly.

Next, the model-based controller in (7) is applied for a 0.6-m linear path and a half sinus of length 0.6 m and amplitude 0.3 m. The brickOS is employed since it offers 256 power levels for output port control. The main drawback of this OS is that its control signal refresh rate is set to 255 ms. Even with this limitation, both the linear and sinus paths were followed with a very small final position error. For example, when the linear 0.60-m path was executed in 5 seconds, the final position error was 0.035 m, while when it was executed in 2 seconds the error was 0.11 m. This firmly proves that the refresh rate is very low, resulting in increased errors at high velocities. Finally, in the case of the half sinus path (length 0.6 m, amplitude 0.3 m) the final position error was only 5–10 mm. Here, the vehicle during the run presented occasional initial path deviations of around 0.08 m that were gradually and drastically reduced after the vehicle progressed along its trajectory. Generally, we can conclude that kinematic systems having a relatively low time constant (fast systems) present difficulties with reference to their control, due to brickOS and computational power limitations in combination with the limited available power of the actuators. In addition, limited and occasionally unstable memory management decreases the RCX's reliability in solving inverse kinematic or dynamic models of such systems. However, a sophisticated compromise under engineering criteria may satisfactorily overcome such drawbacks.

The Kandinsky Project

In the final experiment, three agents (the two autonomous vehicles and a remote PC) participated. A serial cooperation protocol was implemented between the two robotic vehicles, as parallel cooperation requires sensors and computationally heavy software for collision avoidance. The

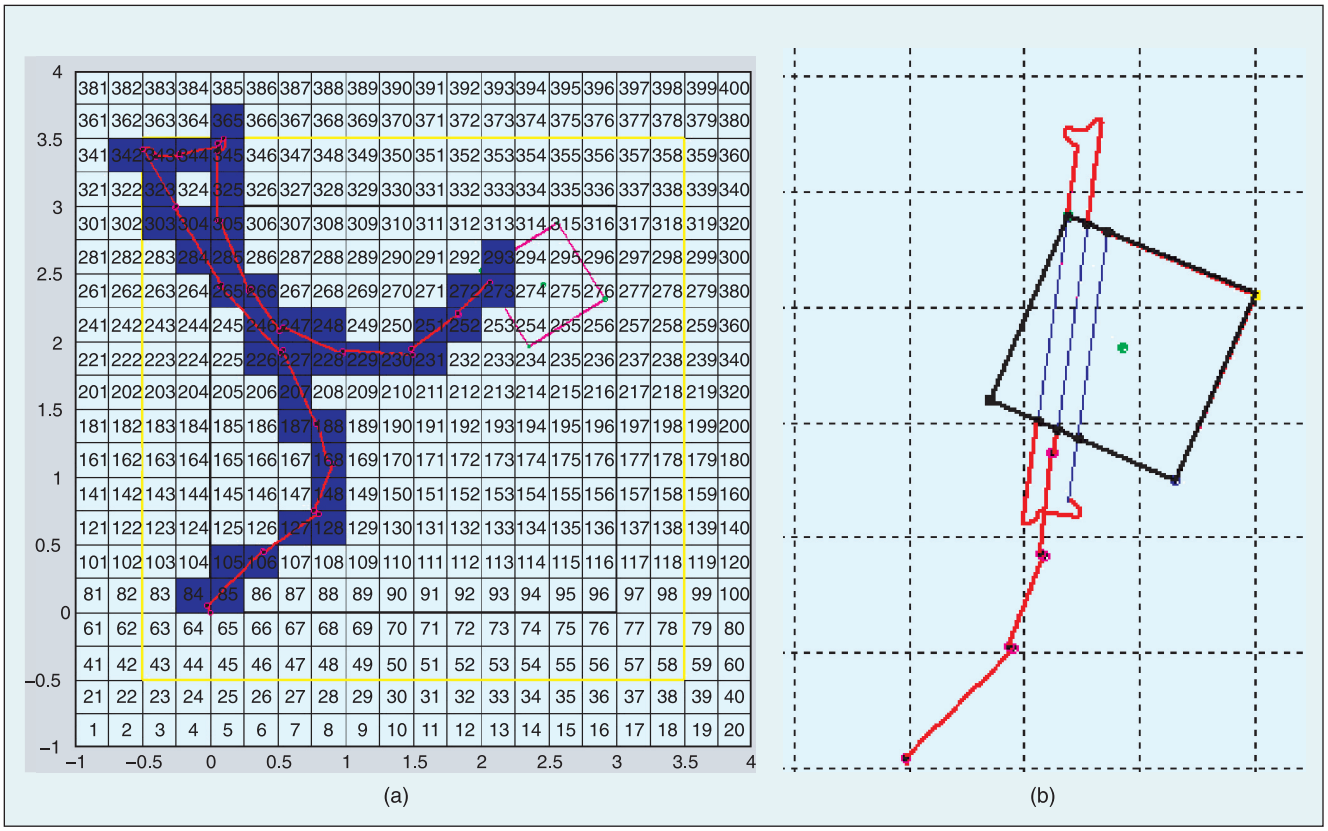

Figure 13. *(a) Wandering algorithm simulation results. (b) Simulation results of the wandering and target recognition methodology. The red line represents the Explorer path. The blue line represents explored colored hives considered as undesirable destinations.*

Explorer wandered in the workspace and when a target was detected, it performed the necessary maneuvers for data collection. When done, it sent data to the PC and moved away from the target. The PC identified the target (kind, location in Cartesian plane) and designed a simple path for the Painter, parameterized by the length of a straight motion ℓ_1 by the angle of a subsequent rotation φ, and finally by the length of another straight motion ℓ_2. The set ℓ_1, φ, ℓ_2 was transmitted to the Painter, previously in standby mode, which computed its trajectory and started reading its rotation sensors and moving. Two controller functions were needed, and since the brickOS supports multithreading, they worked in parallel and continuously at the background. The first sampled periodically the color sensor and updated a relevant global variable, while the other sampled the rotation sensors to update the vehicle position (x_F, y_F, φ) using the platform kinematic model.

For the Explorer, a behavior-based controller was chosen [18]. Such a controller consists of specific, hierarchically prioritized behaviors that are triggered, one at a time, when certain predefined conditions are satisfied. These include *Wander, Avoid Limits, Found Shape*, and *Send Data.*

Avoid Limits is triggered when the workspace limits, defined by a black stripe, are detected and drives the vehicle away from them. *Found Shape* performs the necessary maneuvers to collect data points. *Wander* employs the developed wandering algorithm for workspace exploration. *Send Data* sends test data to confirm the communication channel with the PC. If communication fails, the vehicle performs an on-the-spot minor turn and tries again until communication is established. Then the real data and checksum are sent. If this test fails, the PC demands a new delivery, ensuring uncorrupted data transmission.

The first activated behavior is *Wander*, and it runs until the color sensor identifies a new color. If the color detected is nonblack, *Found Shape* takes over until the necessary data is collected. Next, *Found Shape* is killed, and *Send Data* takes over to transmit data to the PC. After successful completion of the previous tasks, the Explorer moves away from the target and rests. Next, the Painter receives from the PC ℓ_1, φ, ℓ_2, and moves to the center of the target, drawing a line across it.

In the final experiment, a red circle with radius $R = 0.1$ m was used as a target, with center coordinates $(x_c, y_c) = (0.6\text{ m}, 0.4\text{ m})$. Results proved very successful. The Explorer detected the target and the proper maneuvers were performed. The data sent to the PC were processed and the center and radius of the circle were estimated. More specifically, in one case the Explorer found the target in 12 seconds with estimation errors $[e_{x_c}, e_{y_c}, e_R]^T = [3.25\%, 3.5\%, 0.1\%]^T$, while in another case the Explorer spotted the circle in 77 seconds with errors $[e_{x_c}, e_{y_c}, e_R]^T = [14\%, 15.5\%, 0.2\%]^T$. Finally, path parameters were calculated and successfully sent to the Painter that moved across the center of the circle with a final position error of 0.01–0.02 m in both cases. The main problem experienced was that in the cases in which the vehicle wandered for a very long period of time, the point locations collected were totally unreliable due to accumulated position errors, and therefore the circle location was calculated incorrectly. This led to nonvalid data sent to the second vehicle, rendering it immobile.

The overall impression of the LEGO mechatronics platform is positive, and if the mentioned drawbacks are tackled, its capabilities will be boosted significantly.

Discussion

Having completed the reverse engineering of key LEGO components, and having completed successfully two complex projects, we can evaluate the feasibility of using the LEGO platform for teaching robotics/mechatronics and control issues. To our surprise, we found that all of our initial aims were covered beyond our expectations. With projects such as the ones described here, one can study mechanical design, (robustness of the system, balancing, compactness, complex kinematic designs, friction issues), actuators and transmissions (motor-load dynamics, transmission ratios, tension, backlash), sensor analysis and development (novel low-cost sensors better than the LEGO ones, or unavailable), sensor interfacing (matching sensors to RCX inputs, using more sensors than ports, PIC/RCX interfaces), software development, (real-time control, compact code development, use of powerful OS routines, sensor and actuator I/O), communications (physical and bandwidth considerations, driver development), high-level planning, (analysis of complex systems, code development for command generation), servo control (closed-loop control of motors, error minimization), and artificial intelligence (high-level intelligence, algorithm development, and testing in practice).

To develop demanding designs, our experience showed that one will need the following LEGO components: at least two RCX and matching IR towers, rotation sensors, gear motors, light sensors, touch sensors, structural and transmission elements, LEDs, and wires. Non-LEGO components include various sensors (photodetectors, color sensors, encoders, distance, temperature, or ultrasonic sensors, a camera, IR detectors, etc.) and PIC μCs for I/O expansion. One will also need to bypass the software environment provided in Mindstorms or Robolab and use instead brickOS, lejOS, or similar OSs. Also needed are cross compilers between Hitachi μC machine code and C/C++, programs for IR communications, Linux OS, compiler for C/C++ under Linux/Win32, software for electronic design, RCX internals manuals, PIC μCs manuals, etc. However, most of the software needed can be found easily on the Web and is free. In addition, this material can be made available to class students through a course Web site.

Summarizing, the platform developed can fulfill the hands-on requirements in teaching robotics/mechatronics/controls at low cost and is especially suited to single-semester courses. It is expected that the development of the new LEGO NXT controller, with three motor ports and four sensor ports, will offer new possibilities and excitement to the related engineering learning experience [19].

Conclusions

Results drawn from the developed projects prove that the capabilities of an inexpensive set of elements, such as the one by LEGO, are satisfactory and of low cost. The plethora of various mechatronics or robotics projects developed by students with LEGO elements in universities or by sophisticated LEGO enthusiasts further supports this conclusion. With reference to the LEGO internals, the fact that the RCX μC can run alternative OSs and be programmed in many high-level languages increases user options. On the other hand, more memory space for user programs and input ports are certainly needed for more complex tasks. Available sensors are reliable but cover only basic needs. The most important of them, the rotation sensor, has a very low resolution, and without special hardware/software enhancements, it limits closed-loop control applications. Motors provide a satisfactory combination of high torque, small size, and reliable operation, but they could be more compact in size providing higher output torque. In combination with the wide variety of structural elements, a quite extended range of robotic and mechatronic prototypes can be constructed rapidly in compact sizes. The overall impression of the LEGO mechatronics platform is positive, and if the mentioned drawbacks are tackled, its capabilities will be boosted significantly. Finally, there is no doubt that such a platform, despite its standardized elements, can inspire even a graduate engineering student or a sophisticated user to develop new sensors or actuators to expand the I/O RCX ports, to write compact and intelligent code, and to create systems with extended abilities beyond expectations.

Keywords

Mechatronics, robotics and control education, LEGO elements, low-cost experimental training.

References

[1] LEGOLAB, Univ. Aarhus, Dept. Computer Science [Online]. Available: http://www.daimi.au.dk/.

[2] Serious LEGO [Online]. Available: http://jpbrown.i8.com/.

[3] Tufts University, Dept. Mechanical Eng. [Online]. Available: http://ase.tufts.edu/mechanical/.

[4] Abildgaard, M., Andersen, N.H., Hansen, K.S. and Utzen, C., *LEGO Mindstorms in Control System Education*, Lego-Team Report 2. Dept. Automation, Technical Univ. Denmark, 2001.

[5] Martin, F., "The Art of LEGO design," *The Robotics Practitioner: The Journal for Robot Builders*, USA, Spring 1995, vol. 1, no. 2, pp 1-20.

[6] Butler, D., Martin, F. and Gleason, W., "Empowering Minds by Taking Control: Developing Teacher's Technological Fluency with LEGO Mindstorms," in Society for Information Technology & Teacher Education Conference, Charlottesville, VA, 2000, pp. 598-603.

[7] Lau, K.W., Tan, H.K., Erwin, B.T., and Petrovic, P., "Creative Learning in School with LEGO Programmable Robotics Products," in *29th ASEE/IEEE Frontiers in Education Conf.*, San Juan, Puerto Rico, Nov. 1999, pp. 26-31.

[8] Lund H.H. and Pagliarini L., "RoboCup Jr. with LEGO Mindstorms," in *Proc. of IEEE Int. Conference on Robotics and Automation*, San Francisco, CA, 2000, pp. 813-819.

[9] Fiorini, P., "LEGO kits in the lab," *IEEE Robotics and Automation Magazine*, December 2005, vol. 12, no. 4, p. 5.

[10] Center for Engineering Educational Outreach, Tufts University: http://www.ceeo.tufts.edu.

[11] Not Quite C, [Online]. Available: http://bricxcc.sourceforge.net/nqc/

[12] Lego Users Group Network (2000), *LUGNET News*. [Online]. Available: http://news.lugnet.com/robotics/rcx.

[13] Proudfoot, K., (1999), RCX Internals [Online]. Available: http://graphics.stanford.edu/~kekoa/rcx/.

[14] brickOS [Online]. Available: http://brickOS.sourceforge.net/.

[15] lejOS [Online]. Available: http://lejOS.sourceforge.net/.

[16] Hempel R., *Forth for LEGO Mindstorms*. [Online]. Available: http://www.hempeldesigngroup.com/lego/pbFORTH/, 2000.

[17] Papadopoulos E. and Poulakakis J., "Planning and Model-Based Control for Mobile Manipulators," in *Proc. Int. Conf. Intelligent Robots and Systems*, Takamatsu, Japan, Nov. 2000, pp. 245–250.

[18] Mataric, M., "Behavior-Based Systems: Main Properties and Implications," in *Proc. IEEE Int. Conf. Robotics and Automation*, Workshop on Architectures for Intelligent Control Systems, Nice, France, May 1992, pp. 46-54.

[19] LEGO Mindstorms [Online]. Available: http://mindstorms.lego.com/.

Vasilios Papadimitriou received his Diploma in mechanical engineering from the National Technical University of Athens (NTUA) in 2001 and his M.S. in automatic control systems and robotics from NTUA in 2003. Currently, he works as a mechanical engineer for the food industry in Athens, Greece. His interests include robotics, industrial automation systems, and mechatronics. He is a member of the Technical Chamber of Greece (TEE).

Evangelos Papadopoulos received his Diploma from the National Technical University of Athens (NTUA) in 1981, and his M.S. and Ph.D. degrees from MIT in 1983 and 1991 respectively, all in mechanical engineering. He was an analyst with the Hellenic Navy from 1985–1987. In 1991 he joined McGill University and the Centre for Intelligent Machines (CIM) as an assistant professor. Currently, he is an associate professor with the Mechanical Engineering Department at the NTUA. He teaches courses in the areas of systems, controls, mechatronics and robotics. His research interests are in the area of robotics, including space, field, and underwater robotics; modeling and control of dynamic systems; haptic devices; and mechatronics. He has published more than 110 articles in journals and conference proceedings. He is serving as an associate editor of the *IEEE Transactions on Robotics and Automation* and of the *Theory of Machines and Mechanisms*. He is a senior member of the IEEE and of the AIAA and a member of the ASME, the Technical Chamber of Greece (TEE), and Sigma Xi.

Address for Correspondence: Prof. E. Papadopoulos, Department of Mechanical Engineering, National Technical University of Athens, 9 Heroon Polytechniou Str., 15780 Athens, Greece. Tel.: +210-772-1440. E-mail: egpapado@central.ntua.gr.