2011 IEEE/ASME International Conference on
Advanced Intelligent Mechatronics (AIM2011)
Budapest, Hungary, July 3-7, 2011

Semi-autonomous Color Line-Following Educational Robots: Design and Implementation

Michail Makrodimitris, *Student Member, IEEE,* Alexandros Nikolakakis, and
Evangelos Papadopoulos, *Senior Member, IEEE*

***Abstract*—We describe an interactive game for children, employing semi-autonomous color line-following mobile robots. The design and implementation of the mobile robots is described in detail. Although the robots have intelligence that allows them to follow color lines, users interact with them and provide high-level commands such as what color line to follow after reaching a node, and the speed at which they should move. The paper describes the architecture of the game, focusing at the mechatronics aspects and the hardware and software implementation design choices made. It was observed that the game is more appropriate for children over seven years. To the best of the authors' knowledge, this is the first interactive game of this kind at this scale in the world.**

***Index terms*– Multi-color line-follower, human-robot interaction, educational mechatronic game, mechatronic design.**

I. Introduction

Today, there is a growing thrust towards the familiarization of children with technology. Popular interest in robotics has increased astonishingly recently and robotics is seen by many as offering major new benefits in education at all levels. Human-robot interaction is a novel and growing field of research, which has given the opportunity to teachers to develop new efficient and entertaining methods of teaching [1]. In addition, the children, by playing with robots and interacting with them, begin to familiarize themselves early with technology, which will be proven to them useful in the future.

A number of people build autonomous mobile robots and participate in line-following competitions. In order to make a line-following robot, one should combine knowledge from a variety of areas such as: programming, digital and analog electronics, drives and robotics. Line-followers also have many practical applications, for example logistic operations by fully autonomous industrial vehicles guided by paths. The paths can be colored lines, they can be black lines on a white surface (or vice-versa), or they can be invisible as in the case of magnetic fields or electrical wires embedded in the floor.

Almost all line-following robotic applications use black or white lines with a contrasting background. This is because it is easy to program a robot to recognize two colors with very different in terms of intensities. Robots following colored lines are rare, due to the difficulty for a robot to recognize a number of colors in a light-changing environment.

Some of the earliest Automated Guided Vehicles (AGVs) were line-following mobile robots. Some of the first were the Elmer and Elsie, a pair of autonomous robots that looked like turtles, built by W. Grey Walter in 1948. Elmer and Elsie were equipped with a light sensor, allowing them to move towards a light source, avoiding or moving obstacles on their way [2]. Twenty years later, the Stanford Cart was built. It was a line-follower mobile robot that was able to follow a white line, using a camera to see. Also, it was radio linked to a large mainframe that made the calculations [3].

The IEEE supports line-follower competitions and has a special homepage devoted to them [4]. Many designs for line-followers exist. There are line-followers which are based on analogue electronics and others based on digital electronics and microcontrollers [5], [6], [7], [8]. The latter lead to "smarter" robots, which have the ability to follow more complex rules during the game. Another criterion to classify line-followers, is the platform on which they are built. The most common platforms are the Lego line-followers [9], the sumo-bots line-followers and the handmade printed-circuit-made line-followers, which are based on microcontrollers.

In this paper, we describe the design and implementation of a game for kids employing line-following robots, based on a concept proposed by Prof. Ch. Kynigos of the Kapodistrian University of Athens. In this game, the robots are semi-autonomous: Although the robots have intelligence (machine low level) that allows them to follow colored lines, users interact with them and provide high level commands such as what color line to follow after reaching a node, and the speed at which they should move. The paper describes the architecture of the game, focusing at the mechatronics aspects and the hardware and software implementation design choices made. To the best of the authors' knowledge, this is the first interactive game of this kind at this scale.

II. Description of the Game

Current technology allows the development of new and entertaining methods of teaching. By using colorful, bright, sound-delivering and highly mobile robots, a teacher can easily attract the attention of children, familiarize them to new technologies, and teach them skills such as cooperation, patience and team spirit in a fun way. A robot-based

Michail Makrodimitris and Alexandros Nikolakakis are with the Department of Mechanical Engineering, National Technical University of Athens, Greece (e-mail: {mmakrod, alexisnik}@mail.ntua.gr).

Evangelos Papadopoulos is with the Department of Mechanical Engineering, National Technical University of Athens, Greece (phone: +30-210-772-1440; fax: +30-210-772-1450; e-mail: egpapado@central.ntua.gr).

competition can also teach children how to set goals, plan strategies, and master decision-making under pressure. With these in mind, we proceeded to design the following game.

The game resembles the well-known 'pacman' software game. Physically, it consists of an arena, see Fig. 1, and eight robots, see Fig. 2, controlled separately by eight touch panels. The users are children. The goal is for each user to command its robot so as to gain as many points as possible within a specific time frame.

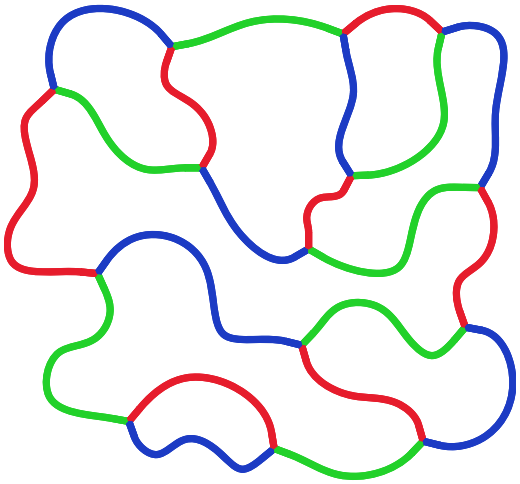

Figure 1: The game arena. Colored path lines intersect at nodes, three at a time. The area of the arena is 25 square meters.

There are two robot modes: 'predator mode' and 'prey' mode. Each player controls one robot, which is either in 'pray mode' or 'predator mode'. At the beginning of the game, four players have robots in 'predator mode', recognized by their red lights and a facial expression of angriness, and four players have robots at the 'pray mode', recognized by their green lights and a facial expression of happiness, see Fig. 2. During the game there are always 4 'predators' and 4 'prey' robots, which continuously interchange modes through their collisions.

When a robot operates in 'prey mode', points are gained while the robot remains 'not-hit' by a 'predator mode' robot. Furthermore, the points are acquired at the end of a set interval, according to the speed of the robot during this interval. The faster the robot moves, the fewer points it gains, as it is easier for it to evade the 'predator mode' robots. When the robot in 'pray mode' moves more slowly, it is easier for the 'predator mode' robots to hit it, and therefore more points are awarded for moving at a slower speed. The 'predator mode' robots gain points only when they hit a 'prey mode' robot. When such a collision occurs, the two robots exchange roles.

Each robot must be autonomous concerning the following requirements: (a) it must be able to follow the colored lines and identify and at each node, to follow the commanded color, set by the user, (b) it must change its speed according to the user's command and (c) it must be able to perform a 180^{o} turn on the color line. At the game end the player who has gathered the most points, wins.

(a) (b)

Figure 2: (a) A robot in 'prey mode' robot and (b) in 'predator mode'. Preys smile, have green eyes and side eyelashes, while predators show their teeth, have red eyes and eyelashes on top of the eyes.

The robots in 'predator mode' must try to hit a 'pray mode' robot so that they can gain points by 'catching' their pray (100 points) and then become 'pray mode' robots, which constantly gain points by not getting 'caught'. 'Pray mode' robots only need to avoid the 'predator mode' robots, gaining points as time passes (every 1 s 'prey mode' robots gain 1-5 points for moving with speed 5-1 respectively). Therefore, the best strategy for the prey-robot users would be to move as slowly as possible while trying to escape, by selecting the best colored path to follow at each node. For the predator-robot users, the best strategy would be to cooperate to catch some prey-robots. When two robots of the same role touch, nothing happens; they stand still for a few seconds and continue after a while. The players can see their name and score on a big LCD TV screen. Each game round lasts for a set period of time, usually set at 15 minutes.

III. Game Architecture

In the game, eight robots and an equal number of touch panels exist. With the help of the touch panels, the users control the robot speed, the direction of motion (forward or reverse) and the colored line that the robot will follow after reaching a node. Each robot is controlled by a specific touch panel only. Apart from user commands, a touch panel constantly sends information to the robot, concerning its current status. When a collision occurs, the robot in addition to knowing its own mode, it should be aware of the mode of the other robot it collided onto, so that points are allocated correctly. In the case of multiple collisions, the system must be aware which robot collided with which. This information could not be sent from the touch panels to the robots because of bandwidth constraints.

To fulfill the above specifications, it was decided to use a centralized architecture: A central server was set up so that the points could be calculated and awarded to all robots, by determining which robot collides with which based on each robot's state. This is done using data collected from the control touch panels, which in turn communicate with the robots. The server connects to the touch panels with a star

architecture. The touch panels inform the server regarding the commanded robot speed, so that points are calculated and awarded to each robot after the lapse of 1 s. When a collision occurs, the robot sends a signal to its touch panel, which passes it through to the server. It is then determined which robot collided with each other by checking the collision time from each reported collision. If the two colliding robots are in different mode, they change mode and the corresponding points are awarded, see Figure 3. The server displays the total points for each robot on a large screen.

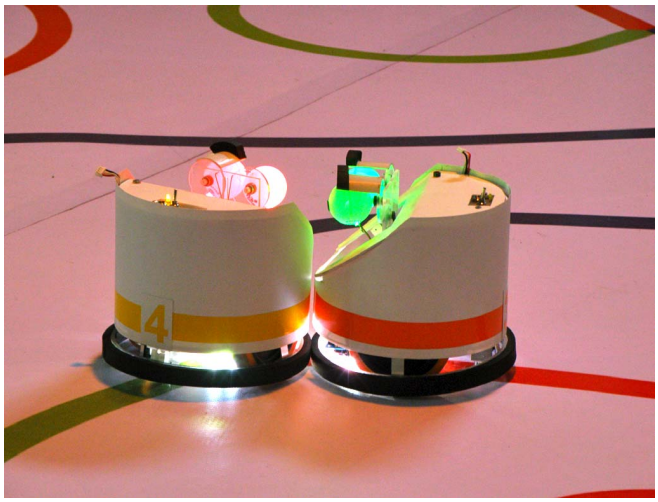

Figure 3: Collision between two robots, one in 'pray mode' (left) and one in 'predator mode' (right). Before the collision the two robots were both in the other mode.

IV. Hardware Design

Due to room limitations, the arena size had to be constrained by a 6 m x 6 m square and by the requirement that users should be able to recognize and watch their robots from their position easily. On the other hand, the arena should not be very small to allow for a number of players stand around it. Finally, a square arena with 5 m sides was constructed. The arena contains colored path lines (red, blue and green), see Fig. 1, that meet in groups of three at node points. The path line width was selected to be 4 cm for visibility.

A. Mechanical Design

From a mechanical point of view, each robot consists of four mechanical modules: (a) the chassis, (b), the motors (c) the wheels and (d) the collision bumpers.

During the game design phase, it was decided that the children should be able to lift the robots. Furthermore, the robots had to be big enough to be seen easily, but at the same time they should be small enough to be able to easily perform maneuvers on the selected arena. We selected plastic for our structural material of the chassis, to keep the robot as light as possible and the cost reasonable. The use of plastic reduces the weight of the robots to approximately 2 kg and at the same time increases their speed. The fact that the robots aren't heavy enables the children to easily put a robot aside or set up the game without the fear of possible accidents. Due to the above reasons, the robot chassis base diameter was selected to be 15 cm.

In order to select the motors we took the following into consideration: (a) the robots should have the ability to move and accelerate fast, (b) the robots should have the ability to make a 180° turn in the smallest possibly area, (c) the motors should be reliable and not too expensive.

In order for the robot to be able to make sharp 180° turns, the differential type of drive was selected, as robots with this type of drive can turn on the spot. Although tracks share this characteristic, they tend to skid and not be so controllable as wheels. Therefore, they were rejected. In the front and back of the robots, ball caster were placed (so that the robots will not tip over during acceleration or deceleration phases) while in the middle, two wheels to be driven by DC motors.

To conclude which motors were suitable for our application we modeled the robot drive dynamics. Neglecting Coulomb friction, the equations of motion for a motor with the load reflected at its side are given by:

$$J_{eff}\dot{\omega} + b_{eff}\omega = \mathrm{T} \qquad (1)$$

where T is the motor shaft torque, $\dot{\omega}$ is its angular acceleration, ω its angular velocity, J_{eff} is the effective inertia of the robot as seen by the motor, (includes the motor and wheel inertias, and half of the reflected mass of the robot, which depends on the radius of the wheels and the gearbox ratio). Similarly b_{eff} is the effective viscous friction of the mobile robot, which is a function of the motor bearings viscous friction, and the wheel viscous friction. By having a rough estimate of the effective viscous friction and by calculating the effective inertia, we obtained the motor angular velocity response corresponding to Eq. (1) as a function of a number of parameters. The linear speed of the robot v was then given by:

$$v = d\omega / 2n \qquad (2)$$

where d is the wheel diameter and n the gear ratio. It was found that if the motor torque equals 100 mNm, the wheel diameter is 10 cm, and the weight of the robot is approximately 2 kg, then the maximum velocity of the robot is about 32 cm/s. Using the torque-speed requirements for typical motor tasks, the Faulhaber surplus motors type 1524E006S123, with 15/5S141:1K832 gearheads and HES164A encoders were found, which met our specifications. The maximum robot velocity was constrained further in order for the robot to follow the color lines without overshooting at steep turns. The motor encoders produce 4 pulses per revolution but because of the gearbox, we take 564 pulses per revolution, which is more than adequate for our application (less than 0.64^{o} for 1 pulse). In order for the robot to move correctly, the two wheels must share the same rotation axis. Otherwise the robots have a difficulty in moving along a straight line. The small chassis diameter does not allow motors to be installed on the inner wheel side without modification. Therefore, CNC drilling was employed to accurately place the two motors on the chassis using L brackets, placing the motor vertically. All the

electronic parts were connected on the plastic chassis.

As mentioned earlier, the wheels had to have a diameter around 10 cm. To this end, 3 7/8 inch Banebot wheels were selected, as the wheel material is selectable. To avoid wheel slipping or fainting out the colors of the arena due to friction between the wheels and the arena's surface, we selected a medium hardness wheel (Medium 40 Shore).

A vital part of the implementation of the game was the ability of the robots to 'feel' the collisions. We implemented a rubber bumper around the robot, combined with four mechanical switches (normally-closed) at its base, to detect collision from all directions. When a robot collides onto another, the bumper sensor moves accordingly and opens the corresponding mechanical switch. So the robot is not only capable to know that it collided to another robot but also the approximate location at which the collision took place. The collision bumper was reliable, but another implementation of the collision subsystem is more appropriate: the usage of accelerometers. The reason is that each of the eight robots should be tuned with a bumper sensor of equal stiffness, which is practically difficult. Furthermore if the collision sensor spring is soft, it will inform wrongly that a collision happened during accelerations of the robot and if it is stiff, it will not inform about collisions even if they did occur.

B. Electronics Design

The robots should have the ability to communicate with the touch panels and recognize the color currently detected by each sensor. They should also have the ability to operate for hours without the need for recharging and have an appropriate appearance in order to impress children. For these reasons each robot consists of the following different electronic components:

(a) The communication module, (b) the navigation module, (c) the power module and (d) the appearance changing module. The four modules are connected to a microcontroller, which realizes the desired robot motion.

For the communication module, we wanted to use a low power, easily implementable protocol, which will be supported by inexpensive chips. To this end, the Zigbee protocol was chosen. This protocol operates in the 2.4GHz band, is very low power-consuming, supports mesh networking, packet retransmission and each module is described using a unique MAC address. The price for each transceiver is low (under $20) and there is a large support community for it. The chip on which we based our communications is the XBee 1 mW Wire Antenna. The Zigbee transceivers were reliable and controlling one or two robots was smooth. However, when we implemented the full game with all eight robots and touch panels, interference problems occurred, as there was a lot of concurrent packet traffic. This problem was tackled by placing the touch panels higher from the ground transceivers, which resulted in less signal deflection.

To recognize the colored lines on our arena, our robots are equipped with 3 RGB sensors, which are placed in a triangular array underneath the robots. The number of color sensors was the minimum required so that the robots can follow a line and select the correct color when reaching a node. Increasing the number of sensors would increase the cost and make their mounting difficult, due to space limitations. Sensor placing was done in such a way that the central sensor is above the color line and the right and left sensors are above the white area of the arena, just at the edge of the color line. Choosing and finding a reliable color sensor was very important for the design of the whole design. In the beginning, we experimented with various inexpensive color sensors that did not have features such as focus etc. We tried various RGB color sensors with a wide cost range (from $2 up to $200 each one). As we wanted to minimize the total cost and the color sensors were an expensive part, we tried to make our own color sensor. This was based on a plain RGB sensor, light-emitting diodes (LEDs) for the required luminance, all soldered on a printed circuit board. This greatly decreased the price of each robot. One problem we faced was that the sensors did not focus on a specific area, and therefore detected the color of the entire field of view. Another difficulty we faced upon was the continuous change in light intensity of the surrounding environment. At first we tried to overcome this problem by software auto-tuning the sensor reading according to the environmental light. This procedure was difficult due to the constantly changing parameters.

As these sensors were not reliable enough, we decided to use a commercial solution, the TAOS TCS230 modules at $40 each. The TCS320 modules have a built-in manual focus lens, which solved the focusing problem. They also have two integrated LEDs, which solved the ambient light problem, by providing a stable ambient light environment for each sensor, (since surrounding light intensity is much lower compared to that of the focused LEDs). Their output is a square pulse with variable frequency according to the detected color, which can be measured using electronics. With the help of these color sensors, the robots were able to successfully detect the colors underneath them.

Besides the RGB sensors and the wheels, the navigation module consists of two other parts: The H-Bridge ICs (SN754410), which amplify the PWM commands of the microcontroller and the motors that are responsible for the movement of the wheels.

Concerning the robot autonomy, we wanted the batteries to be capable of moving the robots for 4 hours without being recharged. Lithium-ion polymer batteries were the only viable solution. We selected Li-Po batteries with 4000 mAh, at 11.1 V (3 cells x 3.7 V = 11.1 V). Recharging the robots is easy: flipping a toggle switch and connecting the batteries to a charger through a connector in the upper part of the robot, recharges the robot. Once this is accomplished, the toggle is flipped back to its original position.

When children are involved, it is very important to develop robots with an exciting appearance, which would attract their attention. A mechanism for changing the robot appearance was designed, see Fig. 2. The robots were

equipped with two-color ultra-bright LEDs that can be seen from a long distance, in order to allow the player to distinguish what kind of mode the robot is in (prey or predator), at each moment. Preys have bright green LED eyes, while predators, bright red. A simple mechanism based on an RC servomotor was installed, which makes the robot smile when it is a prey and showing its teeth when it is a predator. In addition, robot emotions are intensified by employing controllable eyelashes, see Fig. 2.

For controlling the robots, we used an 8-bit PIC16F877A Microchip microcontroller. All modules are connected to the microcontroller, see Figure 4, which is fast enough to ensure smooth robot motion as it succeeds to receive the commands of the players, adjust the parameters of the movement of the robots (velocity and color of the line of its robot to follow after the next node) and process data from the RGB sensors.

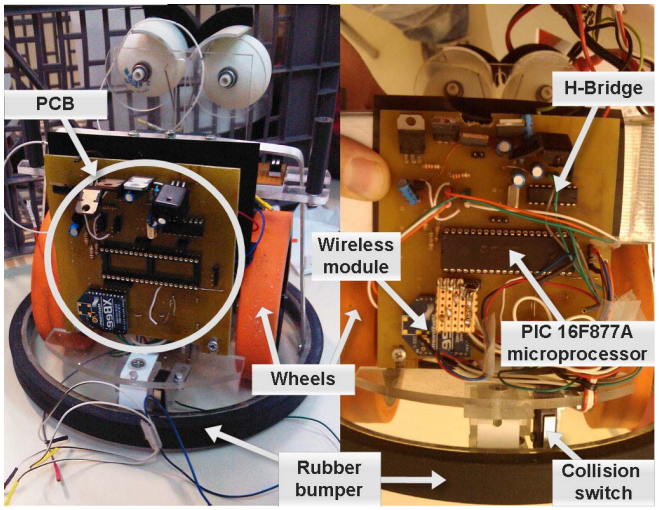

Figure 4: A robot with its cosmetic shell removed.

The cost of each robot is under $130, excluding the three RGB sensors and the batteries, while the total cost is $340. The expensive RGB sensors could not be avoided, due to the game requirements for abrupt curves in the arena and large speeds for the robots. The cost could be reduced if one would accept lower robot speeds or even if the path colors were restricted to just two. Another path in reducing the cost could have been selecting batteries that would require recharging much more often.

As mentioned earlier, each robot is controlled exclusively by a single touch panel. The touch panels consist of two parts: a printed circuit board (PCB), which houses all the appropriate electronics and a plastic panel. The PCB of the touch panel consists of the PIC16F877A microcontroller and of an XBee transceiver. The microcontroller detects the touch of the player's hands on capacitance buttons, and sends the corresponding command to the robot.

The panel is made of colorful plastic underneath of which, twelve capacitive sensors, LEDs and a liquid crystal display (LCD) are placed. By touching the plastic above the capacitive sensors, a player can send commands to his/ her robot and receive feedback on the LCD in English. Apart from communicating wirelessly with the robot, the touch panel communicates with the server through an RS232 port. Excluding its stand, a touch panel costs about $100.

Finally, the point awarding server, is a printed circuit board, which consists of eight serial port through which the eight touch panels connect, and of a PIC microcontroller that keeps the score of points knowing the status of the robots and which robot collided onto which. The server sends the scores to an LCD TV screen through a VGA connector.

V. Software Design

The software implementation of the color line-following robot game involves three different programs. (a) The program of a robot, (b) the program of a touch panel and (c) the program of the server. As each of the above three modules is equipped with a PIC microcontroller, the programming was made using C with the help of a MicroC compiler. Programming a touch panel is easy as its only task is sending ASCII letter commands and implementing easy serial protocols or communicating with an LCD, which our compiler provides us with easy library commands.

The software challenge was to write the motion algorithm, during which three major points had to be overcome. The first challenge was to correctly measure the output frequency of all three RGB sensors at the same time. The program subroutine starts with setting up the microcontroller timer. Then it waits for the digital status of the sensor to go high and then low and then it restarts the timer. The time of the pulse is clocked so we can find the frequency of a color. For example, the white color is on the range of 0-50, the red 65-100, the green 110-250 and blue 270-400. The pseudocode for measuring the color sensor data is the following:

Initialize timer,
Wait for pin to go high,
Wait for pin to go low,
Set timer to 0,
Wait for pin to go high again,
Wait for pin to go low,
Timer value is the pulse period
Depending on the timer value categorize the color reading (0 – 50 is white, 65 – 100 is red, 110-250 is green and 270 – 400 is blue).

A second difficulty we faced was due to a problem that occurs only in color line-followers. If a sensor is overlooking a single color, then it returns the number ranges mentioned above. However, if it is overlooking the boundaries between two different colors, for example white and green, then the sensor sees the "average" color, which in this case is interpreted as red. Then the robot assumes that is over a node and may start executing a node subroutine (rotation on the spot). In order to deal with this problem, the color sensor data is checked for several milliseconds before taking any decisions. Here is the code we used (node turning subroutine):

Check if the all sensors have different colors,

If yes, then if the status is the same for a small delay,
If yes, then turn according to the selected color

The third point to overcome had to do with optimizing the control code running on the robot PIC microcontroller. A common practice would have been to write the code so as to take care of all the possible situations that the three RGB sensors could face. For example, the initial code resembled the following pseudocode:

If the left sensor is white and the center sensor is white and the right sensor is a color, turn right,
Else if the left sensor is white and the center sensor is red and the right sensor is white, go straight,
Else if … etc.

That solution proved to be unsuitable as there are so many if/ else-if cases and the program becomes too large to handle. As a result of this, combined with the previous problem, it was impossible to make the robot follow the line correctly. In the black and white line-followers we do not face this difficulty, as the color is one and so if-else if cases are less. In order to solve this problem we did the following: First we matched the colors to numbers (white=0, red=1, green=2, blue=3). We then divided the program in two parts, based on whether the central color sensor was above a colored line or not, and by comparing numerically the right with the left color sensor, we greatly improved the structure of the program. The pseudocode of the robot is like this:

Interrupt routine in case a character is received:
In relation with the character received, adjust status and appearance (predator or prey), speed, next color followed.
Interrupt routine in case a collision is sensed:
Send to touch panel its identity.
All touch panels must send collision data to a server and then following the same route comes a character to change or not the status.
Main program loop:
Get data from RGB sensors,
If the central color is white then:
If the left color is white and the right is something darker, turn more to the right
If not then:
If the right color is white and the left is something darker, turn more to the right.
If not then all colors should be white, so the robot has lost the line. It should make a counterclockwise circle until at least one sensor is not white.
If both sensors (left and center or right and center) give a color, turn a little left or right respectively.
If only the left (or right) sensor gives any color (not white), turn a little left (or right).
If the center sensor gives any color, then:
If the other two are white, go straight,
If only the right is white, turn left,
If only the left is white, turn right
If none of the above, after a delay, turn according to selection of the player.

VI. Field Results

By observing children playing the game, one can draw a number of conclusions. The children are enthusiastic and impressed by the fact that they can control mobile robots. In no way they are afraid of the robots while there is a noticeable familiarization even at small ages, i.e. 5-7 years old. However, for these ages certain problems arise. First, small children cannot follow the game rules as they are too complex for them. Also, small children do not have a specific goal when playing the game. In certain cases, they are so thrilled by the robots that they forget to give new commands using the touch panels, or they just want to touch the robots in the arena. However, children above seven years old aim at winning, keep the score, and make efforts in understanding the rules and improving their gaming skills. Therefore, a game such as this should target children older than seven years old.

VII. Conclusions

The implementation of an innovative color line-following game with semi-autonomous mobile robots is described. To the authors' knowledge, no other game with color line-following robots at this scale exists. Implementing the game was challenging and required special hardware and software design. However, the game proved to be very reliable. The robot cost was kept to a minimum, despite the use of costly RGB sensors for providing crucial information regarding robot location. The overall reliability can be improved using accelerometers instead of mechanical bumper switches. It was observed that this type of game is more suitable for children older than seven years old.

VIII. Acknowledgments

The authors would like to thank Prof. C. Kynigos for proposing the game and Mr. L. Bahas for assistance in producing the robots. This work was partly supported by Polydiadrasi SA.

References

[1] Mirats Tur, J.M., Pfeiffer, C.F., " Mobile robot design in education," *IEEE Robotics & Automation Mag.*, v. 13, no 1, March 2006, pp. 69.
[2] http://www.ias.uwe.ac.uk/Robots/gwonline/gwonline.html
[3] http://www.stanford.edu/~learnest/cart.htm
[4] http://ieeerobotics.wikidot.com/line-following-competition
[5] Cook, D., *Intermediate Robot Building*, Apress, 2010
[6] Kettler, A., Szymanski, M., Liedke, J., Wörn, H., "Introducing Wada-A new robot for research, Education and Arts", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, 18-22 Oct. 2010.
[7] Konaka, E., Suzuki, T., Okuma, S, "Line-following control of two wheeled vehicle by a symbolic control," *Proc. of the 40th IEEE Conference on Decision and Control*, Orlando, USA, Dec. 4 - 7, 2001.
[8] Cardeira, C., Da Costa, J.S.," A low cost mobile robot for engineering education," *32nd Annual Conference of the IEEE Industrial Electronics Society,* IECON 2005, pp. 2162-2167.
[9] Papadimitriou, V. and Papadopoulos, E., "Development of an Educational Mechatronics/ Robotics Platform Using LEGO® Components," *IEEE Robotics and Automation Magazine*, Vol. 14, No. 3, September, 2007, pp. 99-110.