A. Kuturov, A. Yudin, I. Pashinskiy, M. Chistyakov "Team Development of an Autonomous Mobile Robot: Approaches and Results" in Research and Education in Robotics - EUROBOT 2011, ser. Communications in Computer and Information Science, vol. 161. International Conference, Prague, Czech Republic, June 15-17, 2011, Proceedings: Springer Berlin Heidelberg, 2011. pp. 187-201.

Team Development of an Autonomous Mobile Robot: Approaches and Results

Andrey Kuturov, Anton Yudin, Igor Pashinskiy, and Mikhail Chistyakov

Bauman Moscow State Technical University, IU4 Department,
2-nd Baumanskaya st., 5, 105005, Moscow, Russia
Lebedev Physical Institute of Russian Academy of Science,
Division of Solid State Physics
Leninskiy prospekt, 53, 119991, Moscow, Russia
kuturov@yandex.ru,skycluster@gmail.com,pashinsky007@mail.ru,mlazex@gmail.com
http://www.bearobot.org

Abstract. This article presents different approaches, used in mobile robot development by the beArobot team. The rules analysis practice is presented based on several years of the Eurobot competition participation experience. The choice of the best robot competition strategy is discussed along with a program solution for the same purpose. The article also presents several results of the team members achieved for the Eurobot 2011 project. It also gives a general idea of the mobile robot being developed. The team makes an effort to systemize its previous and current experience, thus making the article interesting for the educational purposes.

Keywords: Eurobot, mobile robot, education, analysis, modeling, motor control.

1 Introduction

Eurobot competitions [2] give an opportunity to try one's hands on a technical project, which is as close to real industrial development as possible. Such projects are even more interesting because they contain a fair share of creativity, and, moreover, because participants of the competition have to work in a team.

The beArobot team represented by the authors participated in the competitions for 3 consecutive years. Our goal is to develop a method of teaching beginners who never engaged in robots. And we make no distinction between university students and schoolchildren; on the contrary we try to find a universal approach to articulate the foundations for various levels of complexity.

This article is the consequence of positive results of the previous article [1], in which we for the first time tried to unite efforts of team members and to word our experience that includes: team work, robot development approaches, and design principles of mechanics, electronics and a mobile autonomous robot's control software. Experience in writing articles has proven to be very useful

both in terms of direct training of team members (each described the work, he was engaged in during a project) and in terms of development of common approaches to organization of work in a team. Thus, writing an article this year was an obligatory stage in the development of a robot for the Eurobot 2011 competition.

Next we will try to clarify the approaches to team development, which we find appropriate and also present some details of the current work, which is carried out by the team members this year.

2 Analysis of the Eurobot 2011 rules

The first action, which is mandatory for a team always includes analysis of regular rules of the competition. To save space, we do not give a description of tasks that can be found in the official competition rules[2]. The interested reader can always find this information by looking in the document.

2.1 General rules' task simplifications

In order to be able to evaluate and develop a behavior of a robot on the field with the use of software, at first we introduce some rules' task simplifications.

Firstly, taking into account that the main task is two-dimensional, we assume the movement of robots in the main field and in the starting areas discrete from one cells center to the other. For greater uniformity (and simplicity) lets render the starting zone cells and supporting zone cells equal to the cells in the main gaming area. Thus, the field takes the form as in Fig. 1.

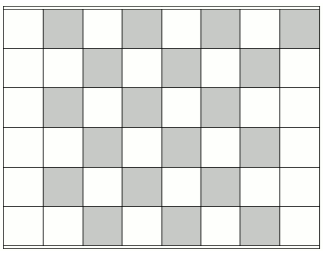

Fig. 1. The simplified gaming field for robot navigation.

Secondly, we agree to develop a static model of one robot-player, i.e. for a case, when dispositions of all objects depend solely on actions of one acting subject.

Thirdly, let's select a unit for evaluation of an object's displacement then various interaction actions of a robot will be estimated in it. Initially an expert's opinion can be used for an estimation and later it can be refined by measuring the actual design solution of a robot or its part.

Fourthly, we agree to put no more than one object in each cell of the field (towers in this case are considered to be single objects). Thus, when moving objects not in order to build a tower, the cells with already placed objects can't be used for a route.

2.2 General definitions

Before we go directly to the analysis, we must define some concepts that will be used later on. As one of the ultimate goals of the analysis can be formulated as finding alternative strategies that can win in a match, we should define a strategy in this case.

Strategy - a set of tactics in a strict sequence repetition to achieve the global goal (winning) of a robot in the competition.

Tactic - a means to select a set of alternative actions to achieve some goal.

2.3 Complexity levels

Based on experience of participation in previous Eurobot competitions we conclude that there are several levels of robotic solutions' implementation. These levels vary in complexity of a robot's actions and its physical components. In fact, a level involves such areas of development as: mechanics, electronics, programming and project management. Each next level should include the previous, so that the levels reflect evolution of any robot's creation within a team's considered goal. In our case the goal is gradual studying, moving from less complex technical problems to more complex.

We give an expert evaluation of possible solution levels:

- Level 1 - a robot can push a pawn.
- Level 2 - a robot can move a pawn in any direction.
- Level 3 - a robot can perform the "funny configuration" action.
- Level 4 - a robot can build towers of objects.

This scheme of complexity levels can be used both for human controlled "robots", and for real autonomous robots.

2.4 Model parameters

In competition tasks we can find the following possible parameters for a model: number of objects, number of points, complexity, and time.

On closer inspection, it appears that finally only two independent basic parameters of the model can be identified and which will be able to enter the selection criteria for alternative solutions. Those parameters are: points awarded for productive activities and time spent on all activities of a robot (productive and supportive).

"Number of elements" is a direct alternative to the "number of points" parameter, because the first can be expressed in terms of the second. We come from

the assumption that the number of points in this case is a more convenient form of successful actions' evaluation due to a possibility of combining objects, leading to difficulties in determining the "number of objects" parameter. In case of the "number of points" parameter it is done a lot easier by introducing a translation table of all possible combinations of objects into points and indicating possible values for each level of solution's complexity.

A fact which confirms that there is no direct correlation between the proposed options is that actions of a robot include both productive actions which are evaluated in points, and auxiliary actions, which are not evaluated in points.

Duration of productive actions can be predicted with fair accuracy, but due to the fact that it is possible to combine objects resulting in a significant score change, this time component in this case has no direct connection with the point system.

Duration of auxiliary actions cannot be accurately evaluated due to the presence of objects that are positioned randomly on the field. It can only be predicted with a certain degree of probability for time's maximum and minimum limits.

Let's show the maximum possible limits of points per game for the previously considered levels of solution:

- Level 1 - from 0 to 180 points.
- Level 2 - from 0 to 310 points.
- Level 3 - from 0 to 350 points.
- Level 4 - from 0 to 500 points.

Maximum limits on the time parameter are from 0 to 90 seconds.

Thus, for further consideration, we have a two-parameter mathematical model with certain restrictions on the parameters.

2.5 Tactics of a robot

Let's consider some of a robot's tasks, which it must solve on the field in order to reach its goal. These tasks may include:

1. Moving from the current point to a point of a next task selection.
2. Searching for another "free" object on the field.
3. Capturing an object for manipulation.
4. Moving an object to a desired point on the field.
5. Performing the "funny configuration" action.
6. Building a tower.

To accomplish each of the above tasks one or more tactics can be developed. The number of tactics for solving a task shows how adequate a robot can be in a situation in a match and in fact characterizes the robot's intelligence.

The minimum requirement for the number of tactics, depending on a solution complexity (note that complexity of a tactic itself depends on the level):

- Level 1 - one tactic for tasks from 1 to 4. Resulting in 4 tactics.
- Level 2 - one tactic for tasks from 1 to 4. Resulting in 4 tactics.
- Level 3 - one tactic for tasks from 1 to 5. Resulting in 5 tactics.
- Level 4 - one tactic for tasks from 1 to 6. Resulting in 6 tactics.

2.6 Tactics of a robot

For this tactic, we assume that the scope for an algorithm that implements the tactic is a two-dimensional array of 5 by 5 items. In this case, a robot is always in the central cell of the array. Thus, while the robot is moving on the field, the array is also "moving" after it, including all new objects appearing in the scope of the array. Fig. 2 shows one of the possible situations with a disposition of objects on the field.

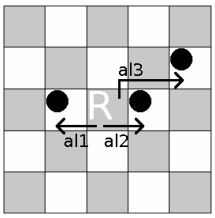

Fig. 2. Alternative actions for capturing objects in a robot's scope.

The figure shows that the robot (the "owner" of white cells) has 3 alternatives for capturing objects: al1, al2, al3 (the direction of an alternative is shown with an arrow). The number of alternatives is formed by the number of objects in the array's scope.

Let's explain what lies at the heart of the search tactic for the best possible alternative. We believe that we need to minimize the total match number of a robot's cell "steps" in the field. Then it is logical to choose such an action, which minimizes further movement of a robot, such as when it finishes necessary actions and is ready to search for another object, it has to be on an adequate distance from the object.

To implement the tactic, we introduce two numerical parameters:

- an award - for reduction of future movements;
- a penalty - for distance travelled to an object.

Then the best alternative's search function has the form:

$$f_{opt} = award - penalty\ . \tag{1}$$

and the parameters are calculated as follows:

$$award = \frac{1}{dist_1} + \frac{1}{dist_2} + ... + \frac{1}{dist_{n-1}}\ . \tag{2}$$

$$penalty = \frac{dist}{n}\ . \tag{3}$$

Where: n - the number of found alternatives; dist1, dist2, distn - the distances to the relevant alternatives, but from the target cell of the calculated

alternative; dist - the distance from a robot's current position to the target cell of the calculated alternative.

The alternative that has the maximum value of f_{opt} will be considered optimal.

2.7 General ideas of a software based strategy search

Having an opportunity to consult with specialists in such areas as math and robotics, as well as having some experience in development of robotic systems, the authors were able to identify possible ways to solve the task of finding an optimal strategy for a mobile robot. Those include: methods of artificial intelligence, the method of local variations, the method of partial repetition, brute force, and the trace method.

The brute force method from all of the above methods proved to be the best to solve the problem. Since the problem is solved on a computer, the use of this method allows a solution that other methods cannot provide at our current level of engineering. The main drawback of the method is that it is necessary to perform a large number of calculations, which leads to considerable time spent on calculations.

It is worth noting that the work also raises an issue of developing a more formal mathematical solution that would expedite the further action decision making process on a robot.

2.8 A strategy search algorithm

To solve the strategy search problem we further divide the field (Fig. 3, on the left) into cells, with the size of an object (a pawn). Each cell will be assumed as a point with specific coordinates (Fig. 3, on the right).

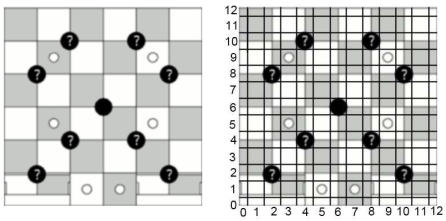

Fig. 3. The field with objects and bonus cells (on the left), a coordinate grid (on the right).

A trajectory of any movement is specified by coordinates of all individual segments of a polyline routed for a robot. Movement vectors are arranged using these coordinates. With lengths of the movement vectors and angles between them, we get the path traversed by a robot and its rotation angle.

The algorithm in Fig. 4 describes how to select an object, and then install it on a profitable cell.

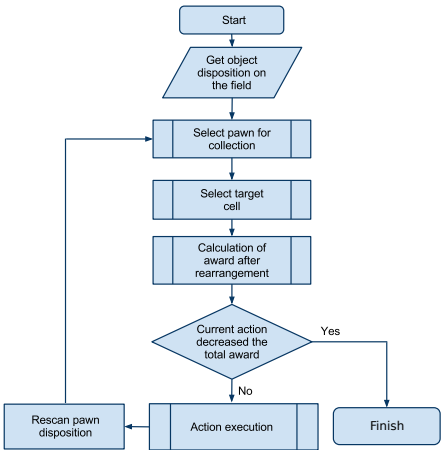

Fig. 4. The general software search algorithm.

To select an object we need to perform an object trace, i.e. to calculate how many times each object encountered in a robot's movement path for each destination cell of the field. And, accordingly, the most profitable object will be the one that appears more times in the way of a robot. As objects are not equal, each of them is assigned a coefficient that determines the score obtained for it on a usual no-bonus cell (Fig. 5).

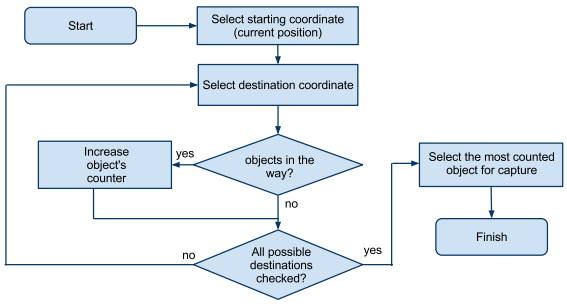

Fig. 5. The object trace algorithm.

A destination cell to put the object on is selected in a similar way (Fig. 6).

In the described algorithm (Fig. 4), a robot can perform the action immediately or calculate two or more actions forward, and then choose the most prof-

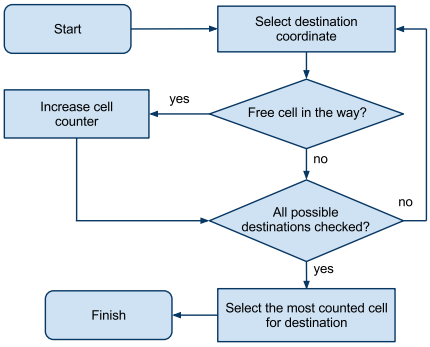

Fig. 6. The cell search algorithm.

itable branch. The number of actions calculated forward we'll call the "nesting" level.

An algorithm with the infinite nesting level continues to choose a point for a subsequent calculation until the most advantageous arrangement is reached, and the subsequent displacement of objects only make the outcome worse. The field is limited, so we have a countable number of options for the location of destinations for calculation, which suggests that such an optimal solution can be found.

2.9 The search algorithm implementation

The algorithm considered above was developed for needs of a mobile robot's navigation on the competition field. The base algorithm is implemented in C and is used to identify the most profitable tactics of a robot on a stationary computer. It is also used to study possible methods of solving the problem of an optimal strategy, because it gives a global solution rather than a local solution. Thus, we are able to reject various strategies that can give only a local, not an optimal solution.

In addition to a pure research usage the presented algorithm is used in a mobile robot's microcontroller based on the AVR architecture and is directly involved in the organization of the robot's action planning.

The introduced algorithm allows the robot to find the best object dispositions. Depending on the speed of a computer being used one can use different versions of the algorithm. Each increase in the nesting level directly affects the decision-making time. A margin of error of the algorithm is inversely proportional to its level of nesting, i.e. the greater the level of nesting, the more optimal the solution.

Although the proposed algorithm is developed and used for the needs of the mobile robot competitions, its use is not limited to that. For example, it can be adapted for warehouse work, where there is a need to place goods to certain

places. You can also adapt it for the needs of automatic placement of electronic elements on a PCB.

3 The Automatic Control System of a Robot Motor

Motors used for a robot's movement often have no speed control of a shaft, so when load increases on the shaft the speed decreases and accordingly the speed of movement decreases also.

It is worth noting that the problem of movement on top of a flat surface has its specifics, which identifies opportunities for organization of a robot's control. This time we are introducing an intelligent system for controlling an electric motor for the needs of robot navigation on a flat surface. Having an opportunity to receive data from various independent sensors, an attempt is made to develop universal software that would ensure minimal errors in regulating drive wheels' rotation speed of a robot.

The developed control system stabilizes frequency of rotation and helps to minimize loss of speed and time, which improves efficiency of a robot. On the basis of this system, one can develop the most effective control algorithms for an electric drive.

The peculiarity of the system is an ability to optimally tune control parameters of the servo drive using the developed software. In this approach the combination of a properly configured PID algorithm incorporated in a microprocessor system, and the program debugging and diagnosis achieve optimal performance.

3.1 The system composition

The general block diagram of the control system is shown in Fig. 7.

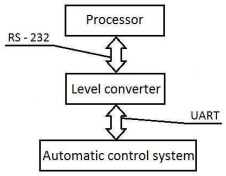

Fig. 7. The general block diagram of the system.

Structurally, the system can be represented by three large blocks: the processor, level converter (LC), and the automatic control system (ACS) itself. ACS controls, regulates and stabilizes the necessary parameters. An important feature of the unit is a connection of the ACS microcontroller with the external environment via RS-232. Signals from the ACS are received by the LC unit, where they are converted, and transferred to the processor for carrying out the necessary calculations and forming commands. The processor sends commands

through the LC to the ACS to setup the necessary parameters of regulation, to define the rotation frequency for stabilization or to switch the drive control algorithm.

The processor is actually a computer. The system is configured and debugged in different modes with the help of it and the developed software. After the debugging is finished, computer can be replaced with the microcontroller (MC). MC via UART receives and transmits all the necessary data, performs adjustment of the regulation parameters in accordance with the algorithms, already tested on computer in initial configuration of the system presented on Fig. 7.

For the convenience of tuning the system parameters, data visualization program was established in Object Pascal namely Delphi 7.0, which transmits, receives the data, carries out its processing and outputs in an accessible form on the computer screen. The layout of the software program is shown in Fig. 8.

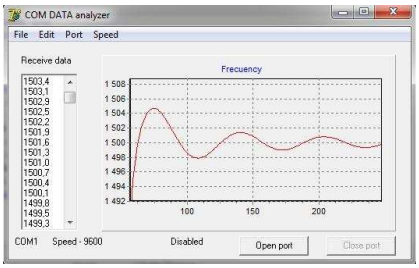

Fig. 8. The layout of the main computer program.

UART/RS-232 level converter converts the signal levels of the microcontroller UART to RS-232 standard. Fig. 9 shows waveforms before conversion (blue line), and pulses after the conversion (red line). System design, software and algorithm of the program were developed in the Proteus VSM. Modeling and debugging were also carried out with the help of Proteus VSM.

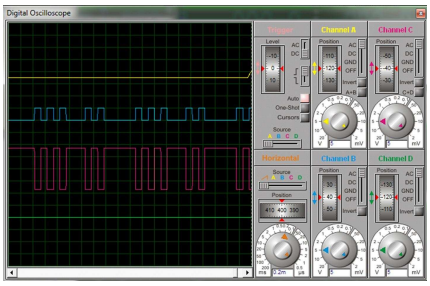

Fig. 9. Waveforms in Proteus VSM (Digital oscilloscope Tool).

The signals from the LC are received at the COM-port. Options of the data transmission used: 9600 baud, 6 data bits, 1 stop bit, no parity.

Let's see how the frequency is set on the drive shaft. We assume 1507.9 rpm is the needed frequency. The transfer is carried out as follows: start bit (St), 6-bit data (transmitting "1") stop bit (Sp), start bit (St), 6 bits of data (transmitting "5"), stop bit (Sp), etc.

Using RS-232 link is justified because currently RS-232-USB adapters are widespread. This allows the communication with a computer, even if it has no serial port. Computer interface greatly expands the capabilities of the motor control system, it allows you to monitor system settings, and most importantly: there is a means of diagnosing and adjusting system parameters.

3.2 The automatic control system

Fig. 10 shows the structure of the ACS unit.

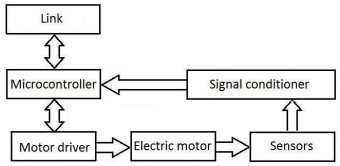

Fig. 10. The structure of the automatic control system.

Microcontroller (MC), using the built-in CCP (Compare-Capture-PWM), generates a signal pulse-width modulation (PWM) and submits it to the motor driver. Motor driver increases the amplitude and power of the received signal and passes it to the motor drive. The rotor of the electric motor comes into rotation, and the sensor unit reads the current value of speed.

An array of sensors of different physical nature is called a database. In addition to the base control loop for the frequency of the motor shaft rotation (the scheme uses high-precision three-channel encoder), the database implements the control loops for current and speed of the robot relative to the field.

Additional control loops allow the regulation tailored to suit not only the mode of operation, but also features of a motor. Depending on the type of sensor it converts a certain physical value, which characterizes the speed of the motor shaft, to a corresponding electrical signal. Resulting electric impulse is transmitted to signal conditioner. It adjusts the signal: changes pulse amplitude, performs filtering, smoothing. If the signal does not require processing signal conditioner can be excluded from the system.

Processed signal is fed to a microcontroller (MC). MC, via the CCP block captures the impulse (on the rising edge for example), calculates the pulse period and finally calculates the frequency.

In the navigation tasks speed of the motor shaft rotation is not always directly connected to the current speed of the robot (slipping, skidding). In such cases, the MC uses the PID algorithm and generates a control signal, taking the external noise into account.

ACS unit using RS-232 interface as a communication channel transmits the current values of monitored parameters and accepts command signals.

3.3 The algorithm of the control system

Fig. 11 shows the algorithm of the microprocessor system program. The program is written in C for the AVR microcontroller.

The algorithm initializes the variables, sets up the settings of the MC, configures the interrupts, reads the frequency of the motor shaft constantly and calculates the control action applied to the motor driver.

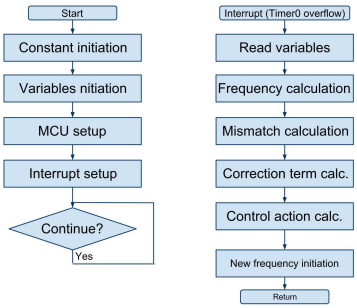

Fig. 11. Algorithm of the basic (on the left) and background (on the right) programs.

Initially the necessary number of pulses per time unit is set, and then the time constant is set - a sampling control constant (assumed to be 10 ms). During this period of time, the MC counts pulses from the sensor and then calculates the error, defined as the difference between the number of pulses set and received from the sensor. Next, the correction term and the control action are calculated according to the PID regulator model. The calculated value of the control action is the number of rectangular pulses generated by the MC in a time period equal to the sampling control constant.

During the sampling time the calculation of all necessary parameters takes place, and on interrupt from the timer, which specifies the sampling control constant, changes are applied to the PWM duty cycle, and as a consequence - frequency of the shaft rotation changes and thus the robot changes its speed.

3.4 Diagnostics and setup of the system

The essence of the diagnostic system is as follows: the shaft speed and the PWM duty cycle are transmitted to a computer via RS-232 link; computer software

records those parameters in a file and processes the data received. According to the file data the software program builds graphics on them, the operator can assess the applicability of the system settings for the current mode of operation. If necessary, he can run the configuration feature over the system. The essence of the setup algorithm is the selection of the optimal coefficients of regulation on the basis of the characteristics of the system in a certain mode. The program evaluates the control speed and accuracy. Fig. 12 shows the results of the system configuration program.

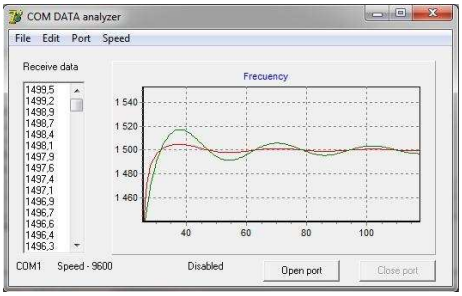

Fig. 12. Transient characteristics of the system at different settings.

Before tuning the system response is represented by the green curve. In the transient characteristic the damped response is observed. The speed and accuracy of control are reduced. Configuration program passed the whole design range of coefficients, and obtained the brown chart characteristic. In this transient characteristic a small release and quickly damped oscillations (1-2 periods) are observed. This type of transient response provides good performance and fast access to the specified speed.

4 The principles for the mechanics development

Rules of a next Eurobot competition are always aimed at changing the mechanical structure of a robot. It is therefore important to learn and understand a universal approach to development of mechanics rather than a specific mechanical solution.

When designing a robot's mechanics the first thing to determine is the functional requirements. They should be chosen based on a strategy of a robot. With clearly defined functional requirements, it becomes possible to determine the range of tasks for designers of mechanical assemblies.

The basic principle for mechanical design is a combination of the following requirements:

– Compactness (minimum volume occupied by mechanical assemblies);

– The minimum number of degrees of freedom while retaining full functionality;
– Reliability and durability of the construction;
– The minimum weight, the lowest center of mass possible for the robot;
– Use simple industrial drives.

The main task of designing a robot's mechanics is in satisfying all of the above requirements.

The choice of materials is also of importance to the mechanics. One of the most common and convenient material is aluminum, it can be found in a form of rods, tubes, angles and strips of different shapes, sizes and dimensions. Its convenience is in the simplicity of processing (drilling, milling, etc.), durability and reliability.

5 The robot mechanics for the Eurobot 2011 competitions

This year our robot has a cylindrical shape (Fig. 13). The chassis consists of two drive wheels driven by stepper motors and 3 slippery stanchions to prevent the forward and back inclination of the robot. The drive wheels are located on the robot diameter and are as distant from each other as possible. Slippery stanchions form a triangle and are situated as distant from each other as possible for a good stability of the robot.

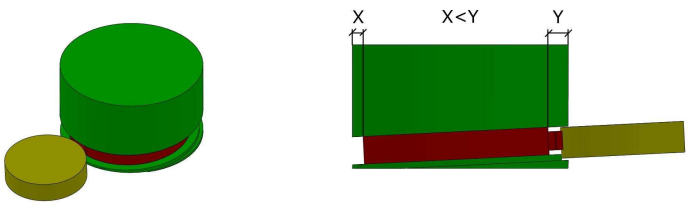

Fig. 13. The general robot idea.

The seizure of objects is realized by a side cylindrical surface with a suction cup attached to a toothed belt covering the robot's perimeter. An object capture is possible because of the onboard pump working in a "vacuum" state. The pump is also able to generate pressure as well. After the capture was successful the toothed belt rotates to a position from which it is planned to put the object on the field. If building of a tower is needed, the belt is rotated from the initial position (X) for 180 degrees in a parking lot (Y), located behind the robot. The "parking" is a niche inside the robot, equipped with a mechanism for reliable clamping of an object for its retention while rising to build a tower.

The rise of an object is possible by means of lifting the whole cylindrical shell of the robot with the attached belt, sucker, and "parking". The hull rises relative

to the robot's inside massive frame. Lifting is organized by means of multiple pneumatic cylinders, working from the same pump as the sucker.

Implementation of the "funny configuration" is as follows. The robot is located in an immediate vicinity of a pawn, the robot's hull is lifted to a height slightly greater than the height of the pawn, and the robot runs into a pawn so that it is in the cavity underneath. With the help of pneumatic cylinders the robot, based on the pawn, lifts itself in the air.

6 Conclusions

While preparing for the Eurobot competitions, the beArobot team has once again presented its vision of a mobile autonomous robot's development. We continue to develop approaches that in future we hope will be united in a methodology of teaching robotics to everyone.

This article presents the results of the analysis of the Eurobot 2011 competition rules, which allow us to represent the competition's tasks in a convenient form, ranged by difficulty. We believe that developing such an approach could lead to an independent method of the Eurobot rules' analysis in future, which will supply teams with an objective approach of formulation of development tasks, as well as evaluation of a robot's actions on the field.

In particular, one of the results of the analysis was the optimal moving strategy search algorithm with a robot and objects in its scope.

In the course of this work a system of automatic motor control was developed. This system has high reliability and performance, due to the use of a special diagnostics and configuration system. It is planned to improve the system further.

Separately, we want to note the experience of using Proteus VSM program in our project. It was used to simulate individual robotic systems, debugging and software configuration. In the initial stages of development, when you need to quickly simulate any electronic system of the robot, as well as for the needs of learning this package fits perfectly.

In conclusion, the basic mechanics design principles were given as the most volatile region in the competitive practice. The robot, developed for the Eurobot 2011 competitions was described.

References

1. Demidov, A., Kuturov, A., Yudin, A. and others: Autonomous Mobile Robot Development in a Team, Summarizing Our Approaches. Technical report, 3rd International Conference on Research and Education in Robotics (2010)
2. Eurobot, international robotics contest, `http://www.eurobot.org`
3. Gerhard, S.: Beginners Introduction to the Assembly Language of ATMEL-AVR-Microprocessors (2011)
4. Thomas, B.: Embedded Robotics Mobile Robot Design and Applications with Embedded Systems, Springer, New York (2003)
5. Ted, V.S.: Programming Microcontrollers in C. Second Edition, EMBEDDED TECHNOLOGY Series, LLH Technology Publishing (2001)