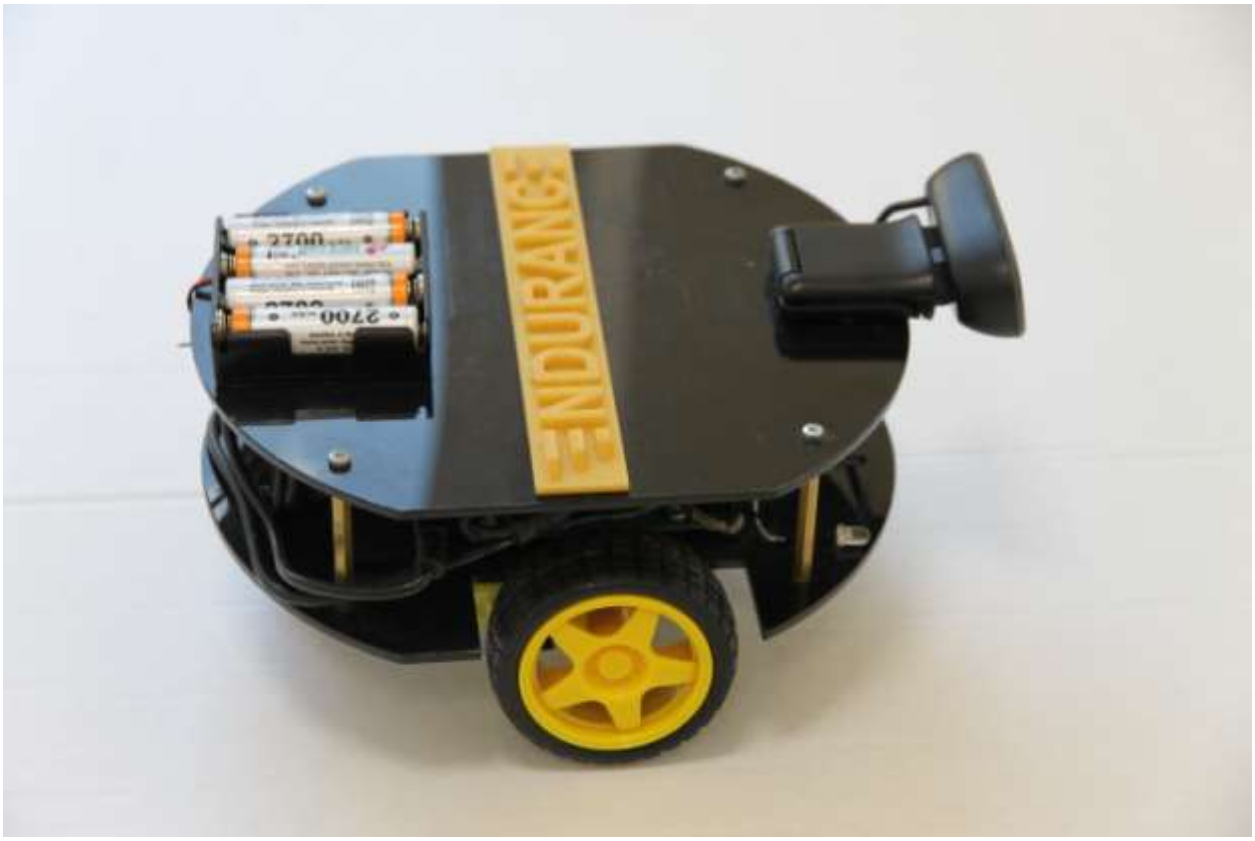
EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

A Full Description of the Small Wheeled Robot Rash 1 for Educational Needs



EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

Contents

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

What is a Rash 1 Robot?

The Rash 1 robot is a robot-constructor, a kit for amateur robotics enthusiasts. It is ideal for STEM centers and K-12 schools.

The educative Rash 1 robot was created by the Endurance team to help school and college students and enthusiasts build their own small wheeled robot.

Skills and knowledge (programming and schematics) acquired while working with the Rash 1 robot:

- ✓ understanding of the electric scheme of the device
- ✓ familiarizing with the components
- ✓ partial or full building of the main board of the device, the device main board assembly of the of the ready components
- ✓ familiarizing with the Arduino Nano platform
- ✓ familiarizing with the Arduino IDE software
- ✓ understanding of the software operation for the microcontroller
- ✓ familiarizing with the UART protocol
- ✓ the Arduino Nano microcontroller programming skill
- ✓ the skill of the device building and setting up

Knowledge and Skills Mastering while Working with the **Rash 1**

- ✓ organization of two-way communication with the device
- ✓ organization of the device operation indication, using, for example, an RGB-LED with a common anode
- ✓ organization of the direct hardware control, for example, with a button connection
- ✓ getting understanding of the work principle and getting experience of work with an ultrasonic distance sensor, for example, to prevent collisions
- ✓ getting understanding of the work principle and experience of the light sensor use, for example, for automatic turning on/off of the LED headlights.

Configuration

The robot consists of a double deck base platform, equipped with motor reducers, an electronics board, a video cam, a router, a battery unit, and two bright LEDs in the front. The electronics board contains an ATmega 328 microcontroller (seated on the Arduino Nano board), a motor driver, an ultrasonic depth sensor, and indicator-control means in the form of LEDs and buttons. A router is needed to organize connection with the robot. We use a TPLink MR3020 controlled by the CyberWRT system containing all the necessary modules for connection with the microcontroller.

Functioning

This model has two operation modes: autonomous and controlled.

In the autonomous mode, the robot moves straight forward until it hits an obstacle. The obstacle is detected at the distance of 20-40 cm in the direction of travel. In case of obstacle detection the change direction algorithm starts operating - the robot rotates, choosing randomly one of the four options. The robot determines the direction where no obstacles are detected and resumes the linear motion in the selected direction. If the straight forward movement is not possible due to the obstruction, the robot wheels stop turning after 20 seconds, and the robot pulls back.

In the remote control mode, the robot performs the operator's commands. The operator connects to the router (via Wi-Fi in the local network or via the Internet), and receiving a video stream from the robot's camera controls the robot by pressing the buttons: forward/backward, left/right, headlights on/off (bright LEDs installed in front of the robot). No special software is required for the connection. It is possible to connect to the robot from any browser.

Who Needs It

This model can be used by:

- ✓ amateur robotics enthusiasts,
- ✓ companies (for rapid development of a robot prototype for special tasks).

How to Use it

Though this model was developed by the company for it internal use of, third parties can also use it:

- ✓ as a test model for building and testing a prototype robot designed for specific tasks;
- ✓ for remote monitoring (remote area surveillance, data collection if the appropriate sensors are installed).

Features

Power supply: 3xAAA, 4,5V

Sizes (H X W X L): 10 X 17 X 23 cm

Weight: - 700 g

Driving distance: ~ 60 min.

Connection: via Wi-Fi or Internet

Video quality depends on the video cam: (2 mp) or HD720

Headlights: yes, operator controlled

Speed: operator controlled ~1-3.5 kmph

Autonomous mode: yes

Distance detection range: 20-40 cm (measurement accuracy +/- 2 cm)

Advantages

- ✓ easy implementation
- ✓ possibility of new devices assembling due to availability of the ready components
- ✓ ready software documentation, description of the interaction protocol between the microcontroller and the router.

Further Improvement

The project result analysis and further development of the solution is described in a separate document.

Summary of the tasks for further solution improvement:

- ✓ sound generation - development of a generator of audio signals with a certain frequency;
- ✓ wheel speed balancing at straightforward movement, using optical sensors and an optical encoder for wheels;
- ✓ automatic headlights turning on/off, using a light sensor;
- ✓ improvement of the reliability of the range finding by means of adding a third ultrasonic sensor;
- ✓ power supply organization on the basis of two 18650Li-Ion batteries and an integrated voltage stabilizer;
- ✓ elimination of creaking by means of gear motors lubrication with a silicone grease;
- ✓ assessment of the supply voltage increase up to 7,0-7,5V; recalculation and soldering of the LED resistors, to avoid exceeding of the maximum microcontroller output relevant;
- ✓ smooth speed change at the beginning or at the end of movement (the change function may be linear or exponential).

Integration Works

The presence of the described and known protocol of interaction between the microprocessor and the router allows representation of the robot as two independent parts - controlling and executive - each of which can be used in other projects that require the functionality of a mobile platform or a remote control unit.

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

Academic Design Implementation

According to the academic design concept, the developed system is divided into the end quantity of parts – that is, development modules. A teacher or curator of this curriculum is responsible for the number of modules quantity, their structure and interrelation.

Recommended Development Stages:

1) A teacher gives a student a primary technical task (specifications) for development of a certain module or a group of modules.
2) A student studies the task and asks the teacher questions if any.
3) A student sets to work (it's recommended to do it in stages), chooses the work procedure independently or coordinates it with the teacher or curator; when submitting each part of the work, a student presents intermediate technical details of the fulfilled stage.
4) Upon completion of the work a student presents a finished product accompanied with primary technical details.
5) The teacher or curator examines the technical details, checks them and improves, if needed, together with the developer;
6) Design finalization.

It is recommended to a student to compile accompanying information at each stage:

1) A teacher or curator states primary technical specifications;
2) The technical task is specified during the coordination process;
3) If the performed work is not divided into parts, no accompanying documentation is needed at this stage. If this is not the case, the result of every stage must be accompanied with an intermediate technical description;
4) A student presents a primary technical description formed at his sole discretion;
5) The final technical description is formed consulting with the teacher;
6) The design development is considered finished when there is a ready product with the corresponding technical description.

Development Stages Disclosure

1) Primary technical specifications formed by a teacher or a student include mainly general remarks and may not contain any technical details.
2) During the coordination process a student clarifies every unclear detail of the technical task, suggests corrections of the technical task or some of its parts in order to improve the technical specifications or to simplify it and cut the cost of the design.
3) On the basis of the volume of works and on the teacher's request a student can further divide the stages of his work into modules. For example, there's a need to

write a program for a mobile platform control. A student can break down the work on the module principle: wheel control – distance sensor control – control by external signals. In this case the program should be also written on the modular basis. The first module is the servos control. The second module is detection of the range to the object. The third module is the interface module of the movement command reception. While working at these two modules, a technological mode is set. That is, when the button on the platform is on, the platform moves forward until it hits an obstacle, then it turns around and moves in the opposite direction. This mode is required for debugging and evaluation of the program work.

For example, a teacher or curator wants to obtain results of their students' work at this stage. In this case the intermediate technical description may not contain detailed specifications. It is allowed to give only key parameters, the process of the technological mode (the button) turning on/off, the algorithmic platform movement in this mode. On the other hand, a high-quality intermediate technical description allows the developer to save time, because on completion of the work, he will be able to add this part to the initial technical description.

4) A student presents his initial technical description of the performed work. The initial technical description is an addition to the technical specifications. For example, the technical task says that the sensor is connected to the control board via the SPI interface. In this case the technical description should explain what sensor outputs are connected to the control board and to what particular parts of the board. The technical description should describe every interface interaction of the module: what, to what and how is connected, what protocols are used, what are the electrical features of the connection. A student can add to the description any details of the solution he considers important.
5) After the study of the initial technical description ambiguities are clarified. The initial technical description may be accepted as it is, without any corrections or improvements if the developer was meticulous enough about the details in the description of the design.

A Suggested Structure of the Academic Documentation

The documentation on the software component of the mobile platform Rash 1 may serve as an example.

Bearing in mind the concept of the robots development, it should be noted that the mentioned documentation corresponds actually only to the "platform software" module and does not fully reflect either the mechanical component or the server and client components of the robot, but it describes well enough some moments of the platform schematics in the technical descriptions of separate sensors and units created in the process of work.

The documentation is presented as graphic files. Below you will find files and folders with explanations of the contained material.

1-Functional scheme.jpg

The robot functional scheme from the control viewpoint.

2- Functional scheme of the platform control.jpg

The functional scheme of the program modules.

3-Structural scheme of the platform control.jpg

The structural scheme of the program modules.

The technical task for the program modules/interface module

The detailed technical task for the interface module.

The technical task for the program modules/range finder module

The detailed technical task for the range finder module.

The technical task for the program modules/control module

The detailed technical task for the platform control module.

The technical task for the program modules/motors control module

The detailed technical task for the motors control module.

The periphery description/motors driver

The detailed technical description of the servos driver.

The periphery description/ultrasonic depth sensor

The detailed technical description of the ultrasonic depth sensor.

The periphery description/Carduino Nano v7.jpg

The technical description of the mobile platform control.

Program

The program for the microcontroller of the mobile platform control board.

Archives to Download

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

Development

Development of the Robotized Mobile Platform Rash 1

Tasks of the Academic Design

Knowledge, skills and experience, the highlights of which are listed below will be acquired in the process of designing and manufacturing of the mobile platform.

1) Introduction to the hardware - the mechanical part.

2) Introduction to the hardware - the electronic part: an Arduino microcontroller, an ultrasonic range finder board, an engine driver board.

3) Acquisition of the skills of development (program writing and debugging) of the software in C language and the Arduino IDE environment, development of the style and structure of the software writing.

4) Getting to know the cyberWRT firmware and obtaining experience of the control signals transmission from the server Web interface, running on the cyberWRT, to the Arduino microcontroller.

5) Creation of a workable end result - a mobile platform - to assess their capabilities and resources to support and maintain the created solution.

The Development Result

1) A mobile platform is to be constructed:
    - The platform is able to move autonomously without collision with obstacles; an ultrasonic sensor prevents collisions: when an obstacle is detected an algorithm of stochastic detour is activated;
    - The platform is able to move on commands sent via the server Web interface operating on the on-board router (connection unit): forward/backward/left/right; the platform is able to generate an acoustic signal, turn on/off the headlights, select the speed in the range of 10% - 100% of the maximum value; plus perform five more commands.
    - A picture from an on-board camera installed on the robot is transmitted to the Web browser;
    - Selection of the control mode – on command or remotely – is set by means of a button on the robot.

- The robot has detection devices: a microcontroller (IC) power LED (red), an LED of movement commands performance (blue), an LED of distance detection (yellow); the nearer the object the brighter the LED;
- The visible range in the firmware of version 1.0 is 20-110 cm, the critical distance to the object is 40 cm; at this distance the algorithm of turning and obstacle detour is activated;
- In the autonomous mode a self-diagnostic system is employed (although not set up in a separate module): if the robot is stuck and does not "see" an obstacle to detour it, the time relay turns on, and after 20 seconds the robot moves backward and performs the algorithm of turning.

2) A kit of the engineering design documentation for the software part of the development will be prepared.
3) Video and photo materials showing capabilities of the mobile platform will be prepared.
4) Statistical data will be collected for analysis.

Design Results Analysis

In the process of the created platform testing the following shortcomings and nuances were brought to light.

1) When moving straight the robot always deflects to the left. As the power consumption increases, the deviation angle becomes greater. Obviously, this is due to the motors parameter spread, perhaps, due to the different operation of the motors driver channels. To eliminate this behavior of the robot it is necessary to connect optical encoders and on the basis of the data from the encoders adjust the algorithm of the motor control module when moving forward or backward. Furthermore, for the power supply it is necessary to have an integrated voltage regulator, for example, to transfer the power supply to the pair of Li-ION elements using the stepdown converter.
2) The robot very often does not “see” an object in front of it. This is due to the narrow viewing angle of the ultrasonic depth sensor. Obviously, the accuracy and reliability of the distance detection can be increased by installing two or three ultrasonic sensors. To determine the distance the sensor readings are polled serially with short breaks to avoid the ultrasonic signals interference.
3) With version 1.0 we twice observed the turning algorithm stalling: the robot was rotating and did not move forward even if there was no obstacle in front of it. To break the situation it was necessary to stop the robot either by pressing it down with a hand or by lifting it above the surface so that it could “see” a clear path. To eliminate this kind of situations it is necessary to broaden the criteria of the robot behavior self-diagnosis with its enforced re-initiation in case of issues. The WatchDog should be used to eliminate IC stalling.
4) Sometimes the robot does not stop when you release the movement control button on the keyboard controlling the robot remotely. In this case we recommend quickly pressing and releasing the movement button. Perhaps, the stop command is lost in the input buffer, or the router does not generate the command. To avoid these failures it is necessary to repeat the stop command by pressing and releasing the movement button a few times.

5) Serial execution of the program operations is the specificity one faces when developing IC firmware. To perform the FORWARD command it is necessary to serially poll the interface modules: the distance detection module, the control module and then the movement module. In this mode the distance detection module causes a delay up to fifty milliseconds. In the event of an excessive growth of the functionality or addition of more ultrasonic sensors the delay may become unacceptable.

   In the FPGA this problem does not occur. Every unit there works simultaneously. It is possible to receive commands, process the distance, display the data and perform a dozen of other operations. In the IC a parallel performance of even unrelated functions is not possible. However, FPGA firmware designing is a much more painstaking work, which requires a certain qualification. Practically any programmer can write a code in C, while for a code in VHDL/Verilog it is necessary to understand the logics of the microcircuit operation.
6) During the remote control a picture from the on-board camera changes too much making it too difficult to get one's bearings. To reduce this effect it is necessary to apply smoothing and/or reduction of the display window.
7) During testing a poor design of the mechanical part was revealed. When fully loaded (the platform + the "second floor"), the platform dynamics drops, screeches can be heard when the platform turns. Perhaps the situation can be fixed by raising the voltage supply. In addition, a rear wheel is frequently jammed on turns and ball bearings greasing is of no help because of the poor design.
8) During the manual control if you stop the robot when it moves forward and quickly reverse it, the robot tends to sharply bend forward loosing rear footing.

Improvement of the technical solution in the given hardware form

1) Development of a generator of audio signals with a certain frequency.
2) Wheel speed balancing at rectilinear movement, using optical sensors and an optical encoder for the wheels.
3) Headlights automatic turning on/off by means of a light sensor.
4) Improvement of the distance detection accuracy by increasing the number of ultrasonic sensors to three.
5) Power supply organization on the basis of two 18650 Li-Ion batteries and an integrated voltage stabilizer.
6) Elimination of screeches by means of greasing the gear motors with silicone.
7) Supply voltage increase to 7,0-7,5 v with the subsequent recalculation and re-soldering of resistors for LEDs, to avoid exceeding the maximum relevant output of the IC.
8) Smooth speed change (smooth start and stop), the change function can be linear or exponential.

Resource consumption

This is an educational open – source project.

To purchase robot components and supplies one will spend 150 - 200 dollars depending on the choice of the supplier.

Components
Platform (wheels, motors, fasteners)
IC
Sonar
Motor driver
Mother board
Battery compartment
Router
Video camera
Cables, slots, ports, LEDs, solder alloy, spirit, etc.
4 AA batteries
Equipment to be used
Soldering station
Squeezing upsetting
Hot glue gun
Clamp meter
Gas heated soldering iron
Additional equipment
Laboratory power supply source
Digital storage oscilloscope

Time consumption

How much time a high school or university student will spend working at the development depends on his knowledge of and experience with the Arduino and C language.

The educational course is designed for 72 academic hours.

Next solutions to be developed in line

1) Development of a 4 wheel mobile platform. Usage of steppers for precise control of wheels rotation, movement and wide speed range settings. Substitution of the controlling IC for an FPGA.
2) Replacement of a communication unit or integration of a communication unit and a control unit in a single one under the control of a unified operational system, ROS, for instance.
3) A new direction is **remote monitoring**. This platform, after upgrading and improving can become a prototype for designing a remote monitoring robot. This robot could remotely monitor houses, apartments, and other premises, if equipped with a servo to rotate its camera, if cleared off the revealed shortcomings, if its charging system upgraded, so that the robot could go and

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

connect to the charging dock on its own. Such solutions are not present on the market today.

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

Assembled Robot Operation Instruction

- ✓ Turn on the robot.
- ✓ The CyberBot access point will pop up on the computer/tablet. Connect to it with a phone/tablet/laptop/PC; the password: endurance.
- ✓ Open a browser: Chrome/Opera and write in the address bar: 192.168.100.100.
- ✓ Click “Robot-spy 2” in the upper menu of the opened window.
- ✓ You will see a picture transmitted from the robot camera. Using the W/A/S/D key buttons or similar buttons on the sensor screen, start controlling (driving) the robot. Using the V button, you can turn on/off the light.
- ✓ The autonomous mode is activated with a button.

Revealed limitations

You cannot control the robot using Firefox because of java. In some versions of Internet Explorer the robot operation is unstable.

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

A developer's article

What does it mean to design a robot?

Репост с geektimes

Perhaps when reading about robots and programing, you think, "Would be cool to do anything of this kind by myself!" But the really possessed by this idea read more articles, watch more videos, study who and how made their robots. On pictures everything seems clear. Videos usually demonstrate ready products and show the making technology in short. So it looks easy and simple: saw off here, screw there, solder, assemble, program and here you are.

The more enthusiastic and advanced choose an easier, at a first glance, way: spring into action at once and make their first robot by way of copying it. It's very important to start doing something by oneself. In the process of making you will face many various hitches, up to the fact that you are unable to order/purchase this or that hokey-pokey because you don't know exactly how it is called. And soldering tiny connectors is very irritating. Why does it go so smoothly on the video? The creation process often gets longer and longer, but a persistent robotics beginner somehow achieves a certain result: even though a first meaningful start in-line pattern.

With the first robot assembly you come to understand why do this or that, and in a given succession. The creation process is now comprehended and can be described. From this moment you start thinking about development of the next model, of the second generation.

I also read a lot of articles about robots with interest and keep on reading! Especially, I like articles about copters: from the ground to the sky! However, to be honest, they only provoked my thoughts but didn't push me to action. Moreover, it was necessary to understand programming. I had worked with C when studied at the university, so I

plumb forgot it. I know what the Arduino is but have never seen it “in the flesh”. I have technical education and deal with DSP on the FPGA, so there are no technical obstacles for me to sort it out but I have more than enough of it at work. In short, I needed a good impulse to encourage me to creativity. And another person’s demand became my impulse. With good reason they say that real creation starts when you give its fruit to other people.

One person I know, the founder of Endurance, approached me with a number of technical questions concerning a telepresence robot. The guys from Endurance have already presented a prototype of an inexpensive telepresence robot (a phone + a wheeled plastic platform on an Arduino = a very budget solution), but they wanted to create a better, upgraded thing. So that was the background of my participation in the project.

I began with reading articles, studied what and where people bought and what made. But the picture remained incomplete. In general, it looked quite clear, but it only looked. It was not clear how to control the robot from a computer, how to transmit video to the computer. I’m not a software guy, and I didn’t feel like spending time on the software subject from a zero level. On the other hand, it could be interesting for I had successfully solved all my tasks before, using the AutoIt script language. My Internet search resulted in finding two key resources: the carduino Internet-shop and the cyber-place forum. The shop had in stock all the necessary components.

The prices in comparison with Ebay or Ali are top dollar, but what is more important one can buy there a platform, all the accessories to it, and every petty ware. It’s also easy to order and get the purchase. On the forum in the sections: "Do It Yourself" \ "Robotics" and "DIY» \ «CyberWrt» you can get general information and learn in details how to organize the communication of the computer with your robot via Wi-Fi. Ten years of experience in the specialty began telling. Instead of copying somebody’s ideas, I took a pencil and a sheet of paper…

What does it mean “to design”? What means “to make a robot” is clear – just start making. And what is to design, to develop?

Below I tried to describe the process of designing, developing a robot. Robot creation as manufacturing is understandable. You take separate components and assemble them together fastening, screwing, etc. Suddenly in the process of assembling you see that you did it in the wrong way and you have to unscrew several details and re-assemble them in a different way. So you assemble your robot by trial and error. But let’s try to do it like grown-up persons: first think through what to do, and then do.

So what is this article about? I will not write how I made my robot. There are enough articles about it. No need to write about it once more. So I decided to expose the technology of my thought, the method according to which you could make a robot from zero. So that a beginner robotics enthusiast would understand the whole process and

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

see the picture as a whole at a zero point.

Let's imagine a novice who has not whatsoever experience in robotics. But he has a friend of a solder, a multimeter, a set of screwdrivers, nippers and what not. Imagine? Good! Off we go…

Stage one — understanding

What do we want to do? What for?

I didn't have to answer those questions. They had been already answered: I was just a responsible party. I had to make a wheeled device with a video cam, which could connect to a PC via the Internet or Wi-Fi and I could see pictures from the device cam on my PC screen, and control the device, making it move around. Everything looked clear but rather boring. Something seemed missing. Remote control. A kind of a robot? What is a robot? A robot is an autonomous mechanism that operates on its own. How does it apply to my task? Let the device drive by itself! Let it go as it likes. However, if it goes the way it likes, it will collide with objects. That is absolutely wrong. So, the device should be able to see what is in front of it and by-pass obstacles, preventing collisions. How can I do that? It is possible to process the data from the on-board camcorder. Uh-uh ... somehow it is not clear how to get at it. Probably it is not easy. What do we still have? There are some sorts of infrared and laser distance sensors. You can buy one and read its data somehow. Sounds wild but why not...

After some mental work a concept of the device was born. We need a device which:

- is able to move around by operator's commands transmitting him video of what its on-board camera "sees" in front of the device;

— has an autonomous mode: can move in any direction;
— monitors the situation in front of it;
— avoids detected obstacles en route, ignoring the operator's command, or changes the movement direction in the autonomous mode;

Now as the goal is set, it's time to take a solder and accommodate it on the working table. Let it get used to the environment.

Stage two — reality exploring

Now as the basic functionality is clear and we know what to do, new questions arise: How to do it? What will it look like?

Sweeping eyes over my room I could firmly declare to my folks that I would need a table top, recliner wheels, a pair of Chinese chopsticks, an elastic band, 20 cm of curtains, and what not. Give it all to me for the sake of progress! But there's a more effective way to help progress. The tool named Internet. I enter "Robot with one's own hands" in the search bar, click and start reading, viewing, listening, and investigating.

This stage lasts longer than the first one. It's more painstaking to figure out how to implement the idea than to come up with an idea. Besides, while implementing it you expand horizons, learn more and produce new ideas and solutions.
The result of stage two is a detailed list of the components we need to enflesh our idea. So, our device will comprise of:

— a three-wheeled platform with two motors, two driven wheels and one as a support;
— AA batteries (1,2-1,3v x 4 = about 5 v – as a power supply for motors);
— an Arduino Nano microcontroller (IC) to control the device (to control motors via a motors driver, 5 v can power the IC);
— an ultrasound depth sensor (5v power supply);
— a cross-flashed router (5v) to work as a hotspot to connect to a webserver, give commands to the microcontroller and get video images from the webcam connected to the router;
— an on-off power switch and on-off autonomous mode switch;
— an LED, which will light up at obstacle detection;
— a pair of bright headlights – why not? Let the robot see in the dark;
— a webcam connected to the router.

This is a short list of robot components based on the ideas we got after studying the market. Now we can imagine what we can get as a result.

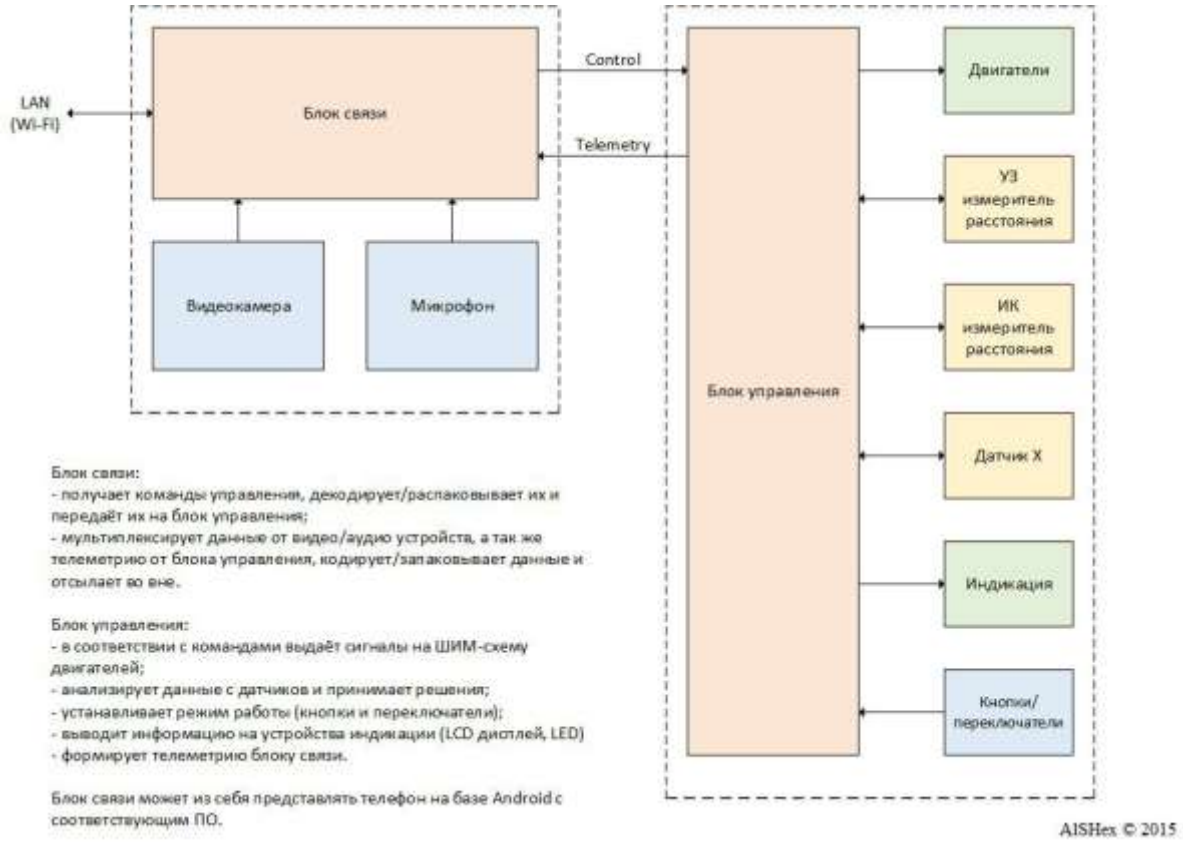
It is possible to buy everything in carduino and setup the router, as said on the cyber-place forum, in order to organize connection with the robot. At least we have an idea of the structure and looks of the robot.

The solder on the table is looking with interest at the pictures above. Its interest is understandable. It's going to solder it all!

Stage three — creativity: general development

So much time spent and no visible results! Just images and thoughts in our heads. Is not it better to materialize the thoughts? We know what the robot will look like. We know its main components. But how to connect and assemble them all? How will the components communicate with each other? A robot is not a lamp, which lights as soon as you flip the switch. What happens when you turn on the toggle and supply electric power? Here, dear readers, our engineering design is divided into three components:

engineering development, electrical engineering, and programming.
There will be no engineering development as we use ready components. Our design work is reduced to successful gluing and bolting loose parts together. If truth be told we should have an assembly drawing. But we have a ready-made platform to install all the necessary components and parts, and pictures with images of the future robot. That's why we are not going to discuss that part of work in this article.
Below we present a suggested block diagram of our robot:



I apologize for the quality of our pictures. Choosing between quality and quantity I found the golden mean and laid out on the Internet as many pictures as I could, provided they were readable and understandable.

What do we see on the pictures?

Our robot will consist of two independent parts: a communication unit and a control unit. The communication unit communicates with the external operator, receives his commands and transmits them to the control unit. The control unit gets and executes commands. That is, we have divided control into two levels: low-level (platform control, interaction with the sensors) and high-level (interaction with the operator and command control of the low-level part). The control task is also divided into two parts, which can be solved separately: first comes a mobile platform as an Arduino microcontroller, and then a communication unit as a reconfigured router.

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

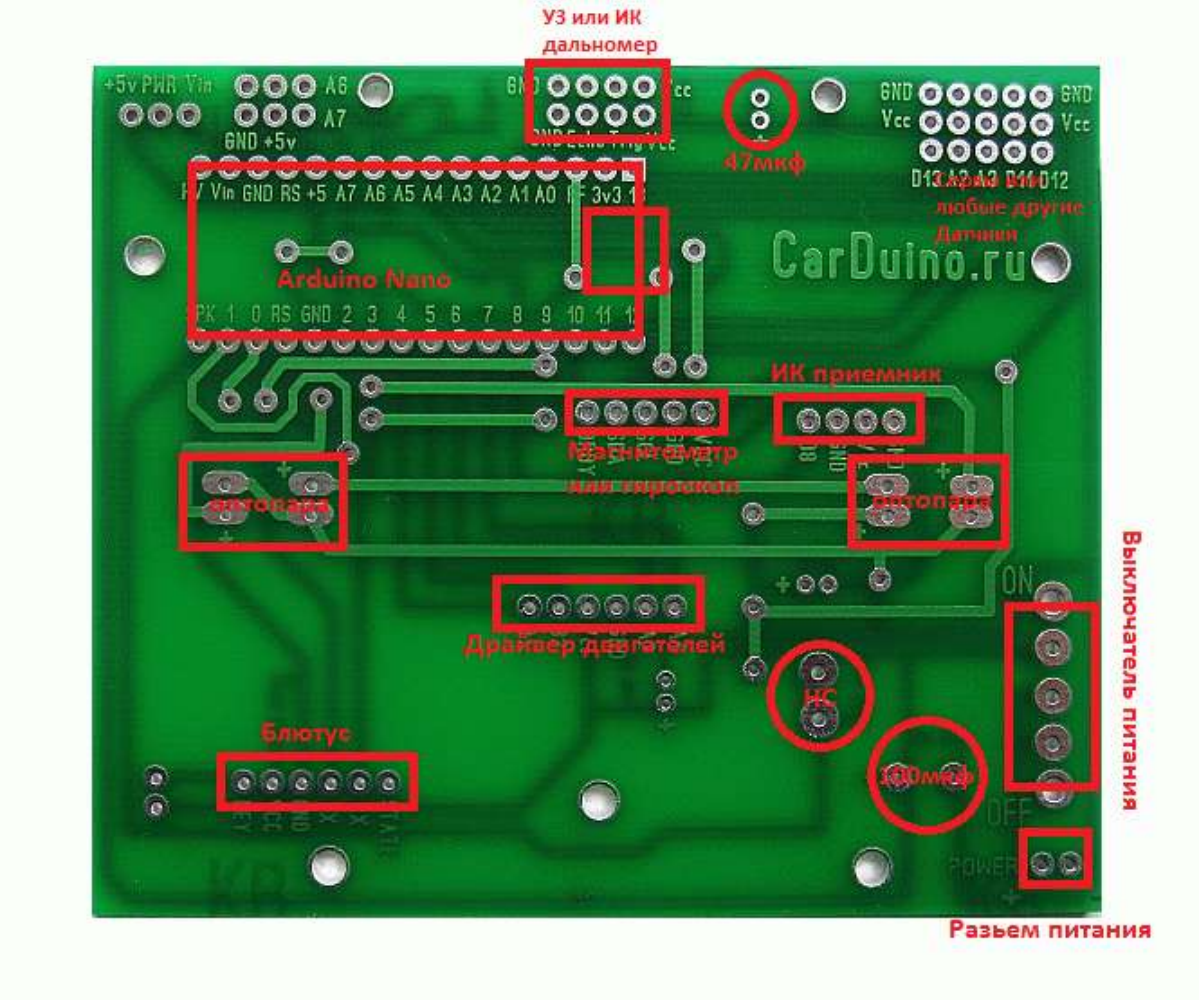
A communication unit

1) Electrical engineering. Here the picture is clear: the cam provides video and sound and connects to the router via hub. The Arduino also connects to it via hub. It's not a big deal to plug the cable into a USB port connector. The probability to do it in the wrong way is quite small. Though, some do manage to do it in the wrong way. But it's not about us!!! We check what we are doing: what and where plugging. If it does not fit we'll force it fit. And it will go to fit!

2) Programming. Here we use a ready firmware for the router and complete modules for the robot. The interface protocol with the control block has been already defined for us. It is necessary only to sort them out. How to upgrade the firmware and what to setup, we'll find it out later.

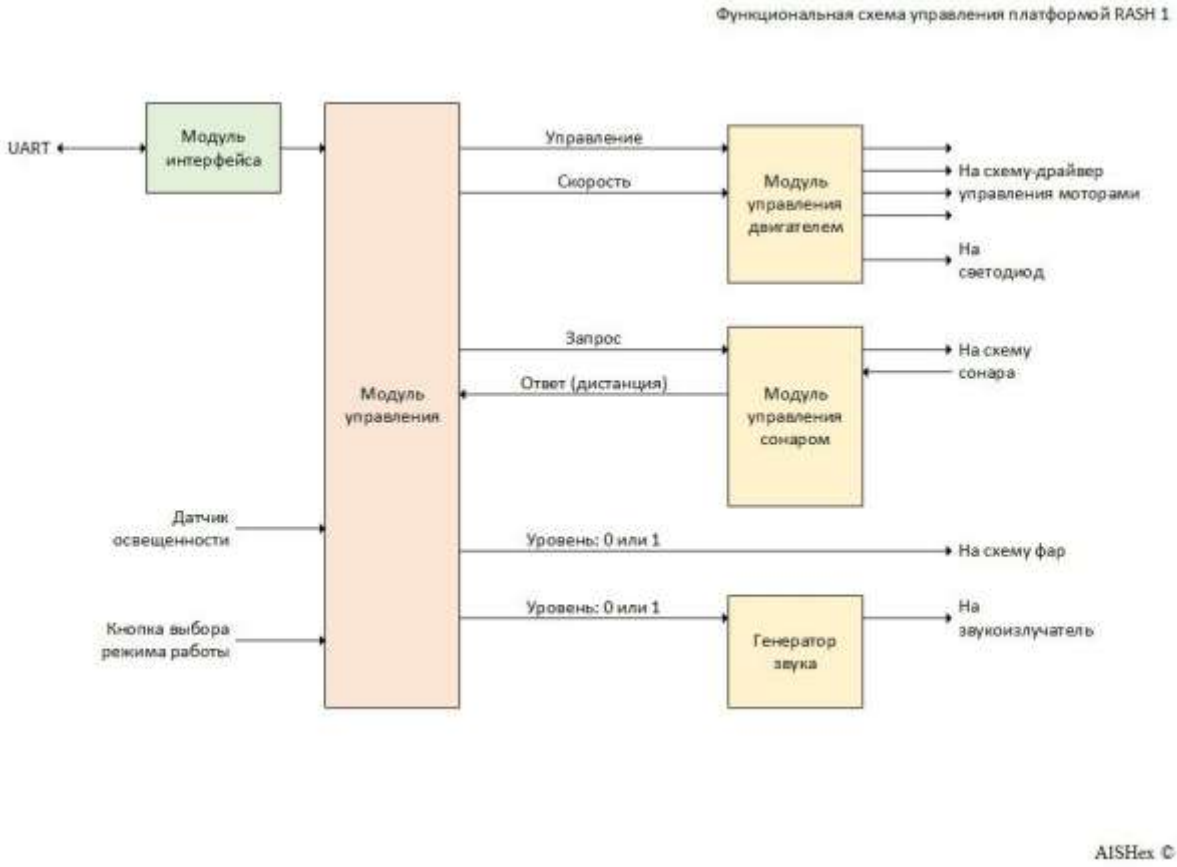
A control unit

1) Electrical engineering. Here everything looks clear too. We have a motherboard, where we do all the soldering work. It's only necessary to find out where to connect the sensors to the Arduino and not to mistake "ground" for "power". No particular problems with the connection. However, it takes some time to sort out what is to be connected and to what particular place on the board. But the work of masters and the picture below make everything fall into place. What and how it all is placed, we'll see into it later. In any case we can always modify it if a problem occurs.



2) Programming. Here we have a problem. What and how is to be going there? Here comes a command and what? Once again, we take a pencil and start drawing. Here is a

pattern of a functional program schematics (of the platform only).



More small squares. This time they depict program modules. This is an image of something that you cannot finger or touch, but it exists, for the robot moves based on the logics of this abstraction level.
The interface module should get a command and identify the incoming data as something that makes sense. When identifying a command, the module should inform the control module about the type of the command. The control module should perform this command: start moving, stop, "see" if there are obstacles on the way, turn on or off the headlights, etc.

When the autonomous mode button is activated, the control module should detect if there are no obstacles on the way and command the motors module to move ahead or perform some other action in case of obstacles detected.

Stage four — creativity: detailed development

Now we understand what to do and how. And out of what. We also understand what functions our robot will possess. So it’s time. Time to set to work! We’ve read so much, studied, explored, assessed, considered, invented, and discarded. And the solder on

the table lies naked looking at us askance, as if wondering, “What are you doing there? Take me and solder!”

With a sigh we cover it with a sheet of paper. “Go asleep. Not everything’s ready so far.” We still need to know not just the functions but also the structure. In other words, we need a detailed understanding of everything.

A communication unit

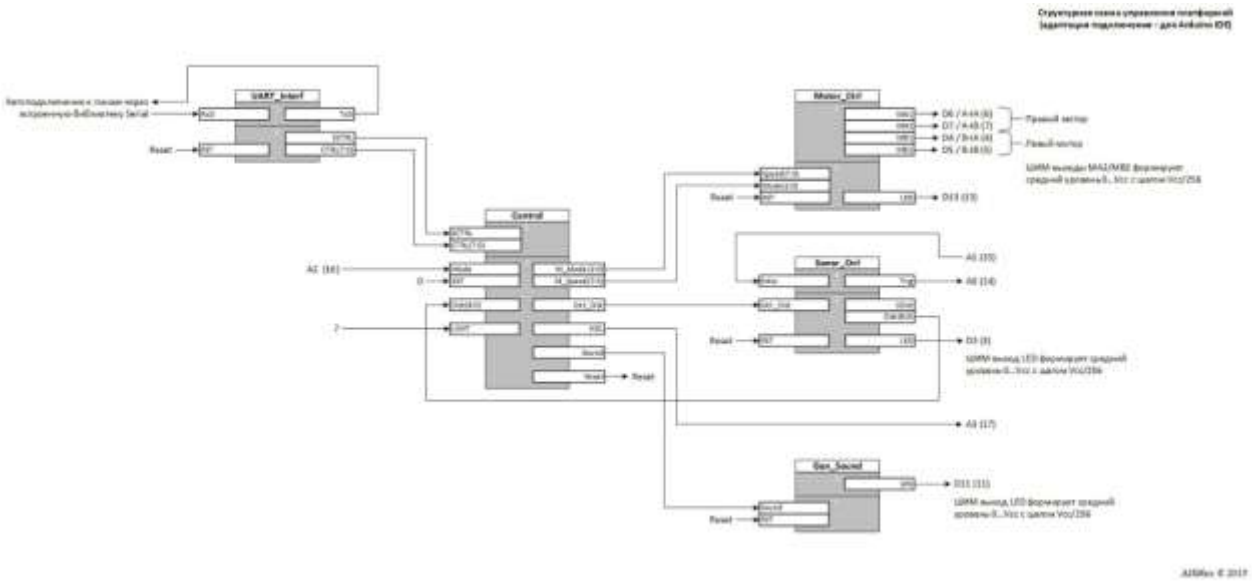
We have solved many issues at the previous stage. On the cyber-place forum we get acquainted with this topic and know now how to upgrade the firmware of the TP-Link MR3020 router and how to setup it. We download the router firmware, create files with the description of what IP to enter and where, what and where to click and push. The only problem is what particular commands will be sent to the microcontroller. And again the forum comes to help. Thank you to everybody who tried to assist me. We write down all the commands to the same text file, making notes what command is responsible for what. Everything appeared rather simple. When you push the button “Forward”, a certain line of the ASCII code is sent, when you release the button, the “Stop” command is given (what particular bytes are transmitted is shown below).

A control unit

We already know what to connect to the control unit. The heart of the control unit is an Arduino nano. What is that? What are these lamps and those connectors are responsible for? To what outputs the sonar pins are connected? And how to deal with the sonar?

We start to study the components info. No need to go to the library. Everything can be found on the Internet. We study manufacturer’s datasheets and make our own datasheets with our components and their detailed technical description.

After some mental work we draw schematics of the software modules connections, which shows what, how and where is connected (to what microcontroller pins).



This schematics is actually the ground for the code. It is designed for the FPGA, but it is also good enough for the C-programming as well. So any platform will suit us.

Now let's read the schematics from the left to the right. The interface module gets data via the UART. If the data is a command it sends the control byte (CTRL) under the strobe pulse SCTRL to the control module. On getting the strobe, the control module understands that a command has arrived, reads the CTRL input, and executes the command. If it is the command "Forward", the pulse signal Get_Dist is sent to the output and the range data is expected at the input Dist. We are not prohibited to constantly issue Get_Dist pulses, in this case, we can read the range data from the Dist port input when we need it. If everything is okay with the range data, there are no obstacles in view, we can ride with the breeze. The motor control module via the Mode bus gets a signal, telling that we need to move forward at a given speed.

When measuring the distance and moving, a high level (a logical unit) is also transmitted to the corresponding LEDs for the work indication and understanding what is happening with the system (it will be useful for debugging).

The module schematics is ready! We have it all at our finger tips. Now we need to describe what every module is to do and when.

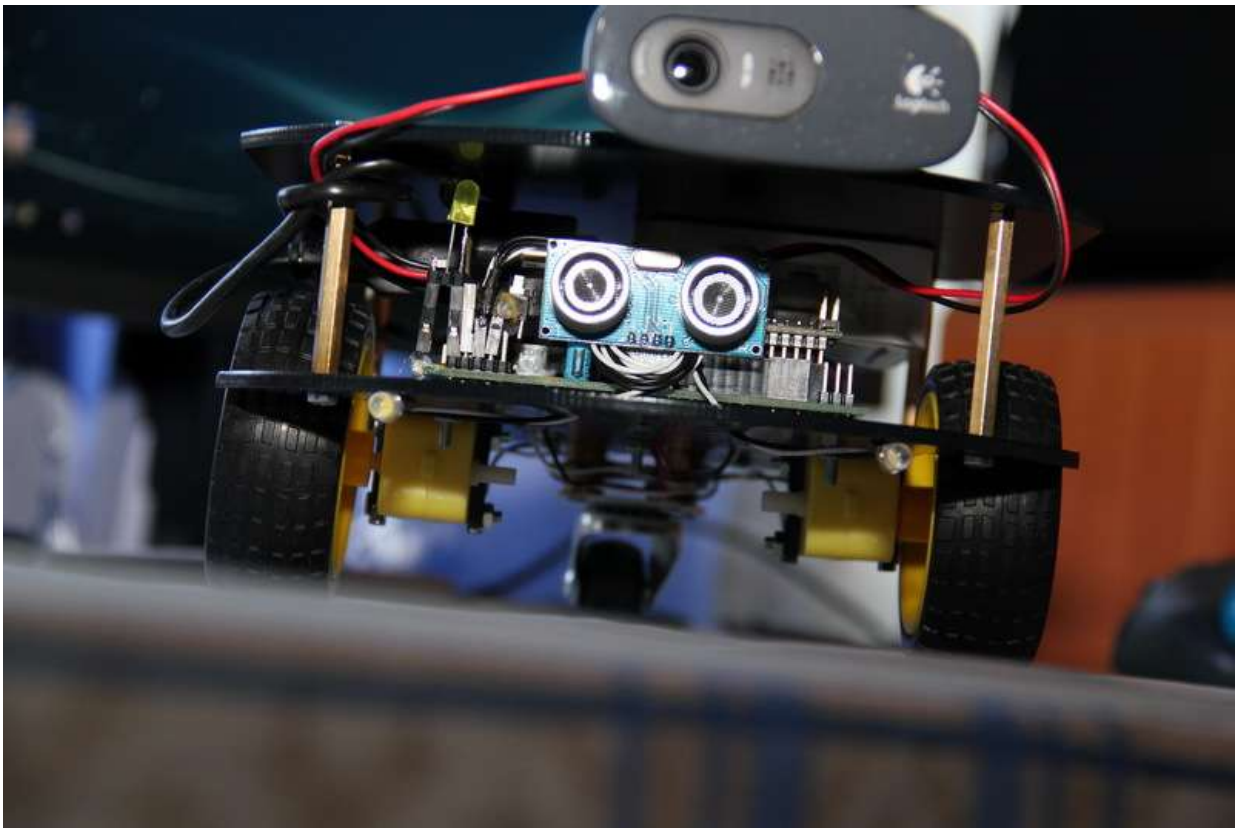
A solder's tail with a plug fell down from the table top and hung swinging. The sheet of paper covering the solder shifted a bit opening the sad face of the solder. The solder looks at us with the expression clear without words. *Nuts. Bad luck with the master*.

Stage five — creation

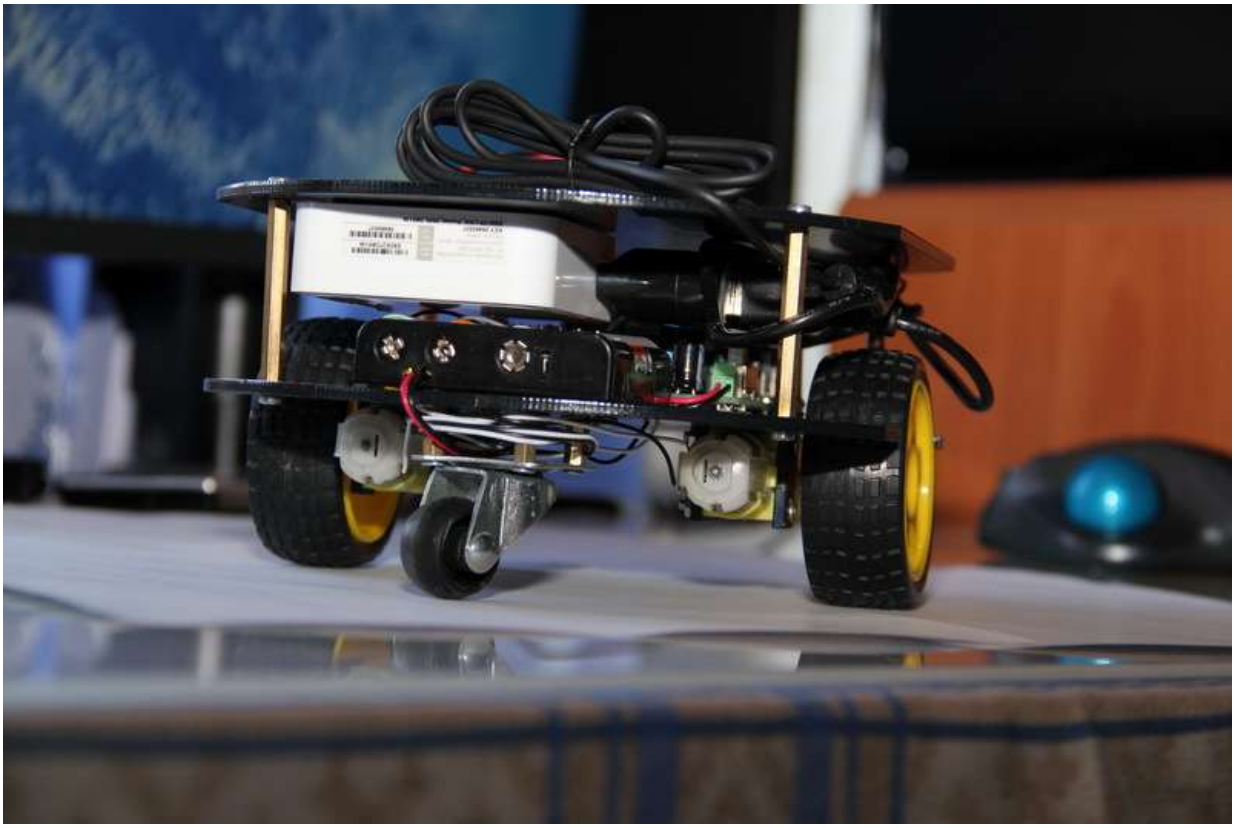
EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

The engineering design has been almost finished. Everything is clear. One thing is left – to pull off a magic act and assemble something out of this heap of spare parts and a pile of paper sheets with pictures, schematics and diagrams, something for the sake of which everything has been undertaken.

We grab our sad solder and connect it to a power supply. The solder flux and alloy start happily dancing on the table. The sun shines brightly lightening the location of the SMD resistor pinpoint soldering on the board. Life is beautiful when everything runs cool!

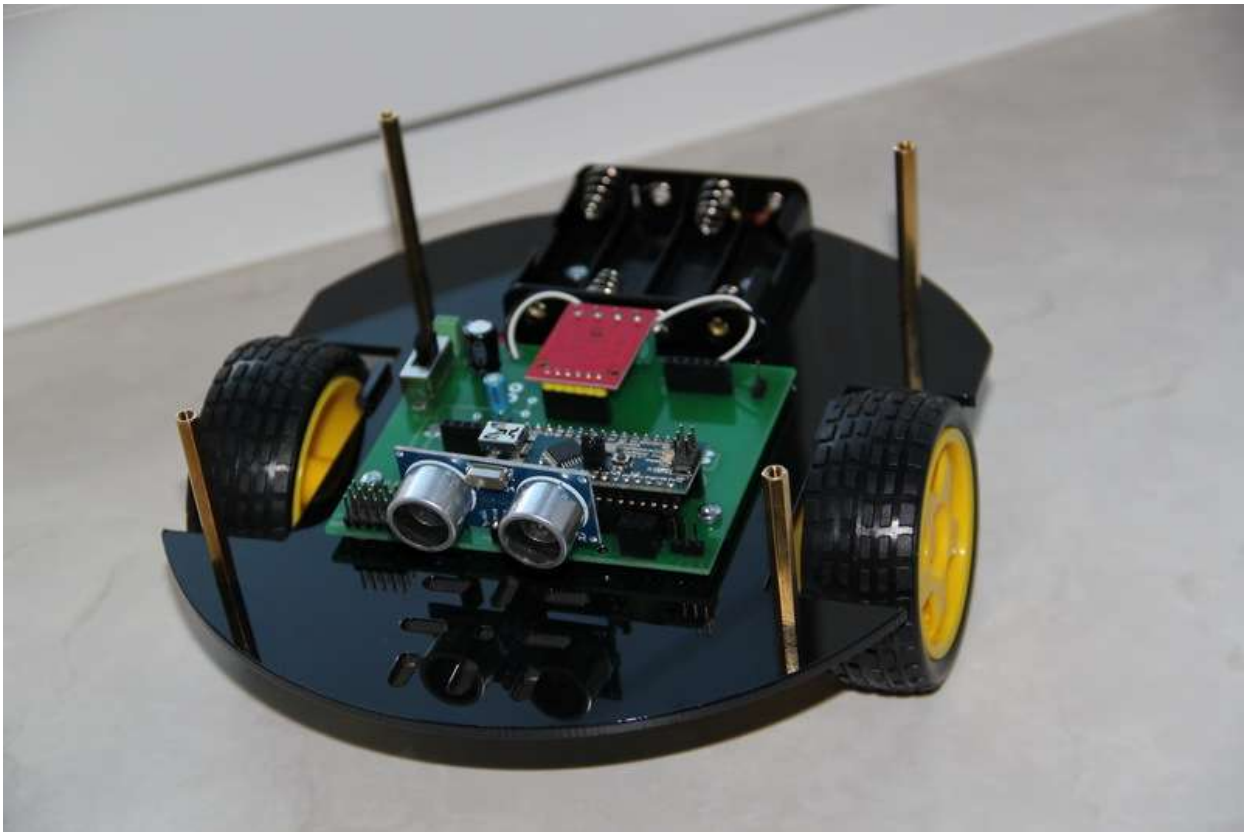


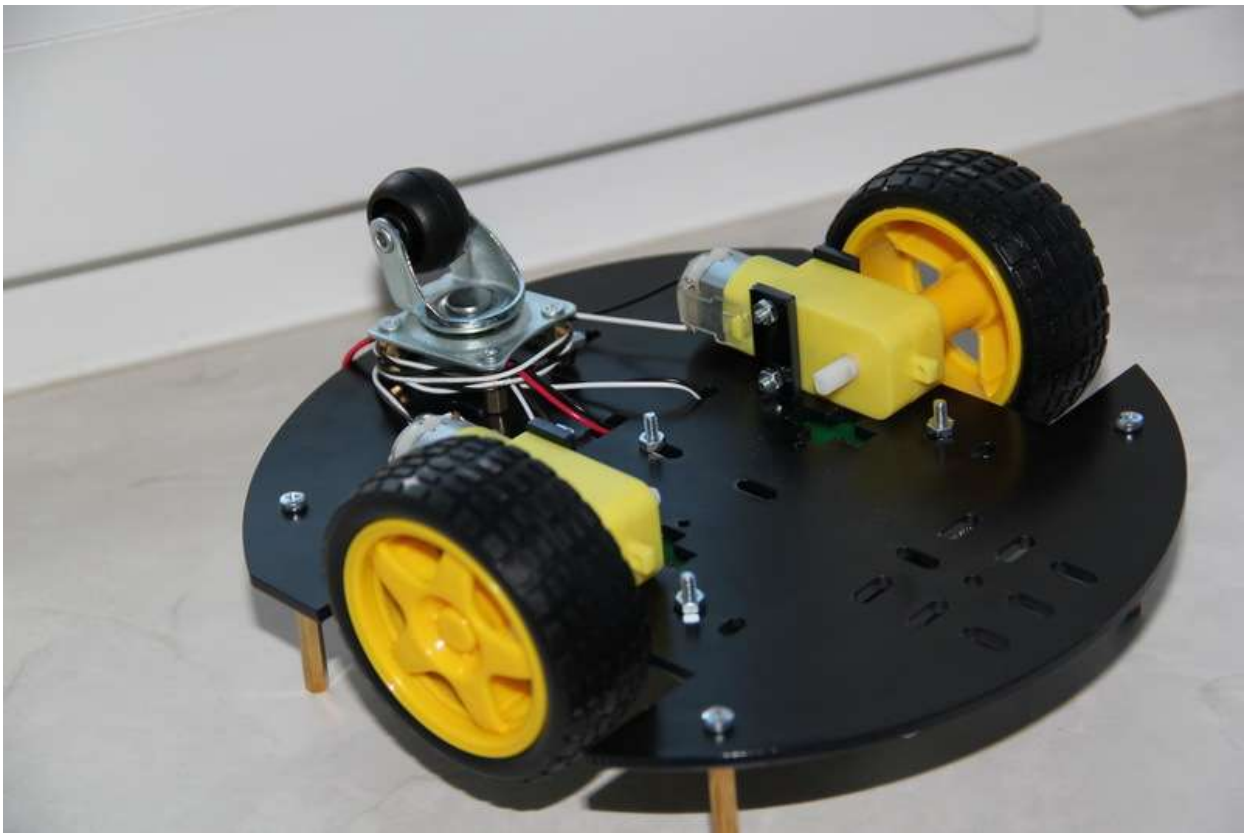
EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302



This is the most spectacular stage of the engineering development when you see a real performance of a real thing. The performance that will leave no one indifferent. This is not drawing some strange diagrams and tables, looking at which one can't but wondering how it can result in something tangible. And here… How great it is when your first robot covers its first inches on the table and stops respectfully at 20 cm from the edge because you blocked the edge of the table with your hand. Smart thing turned out to be! Knows that it cannot move on!

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

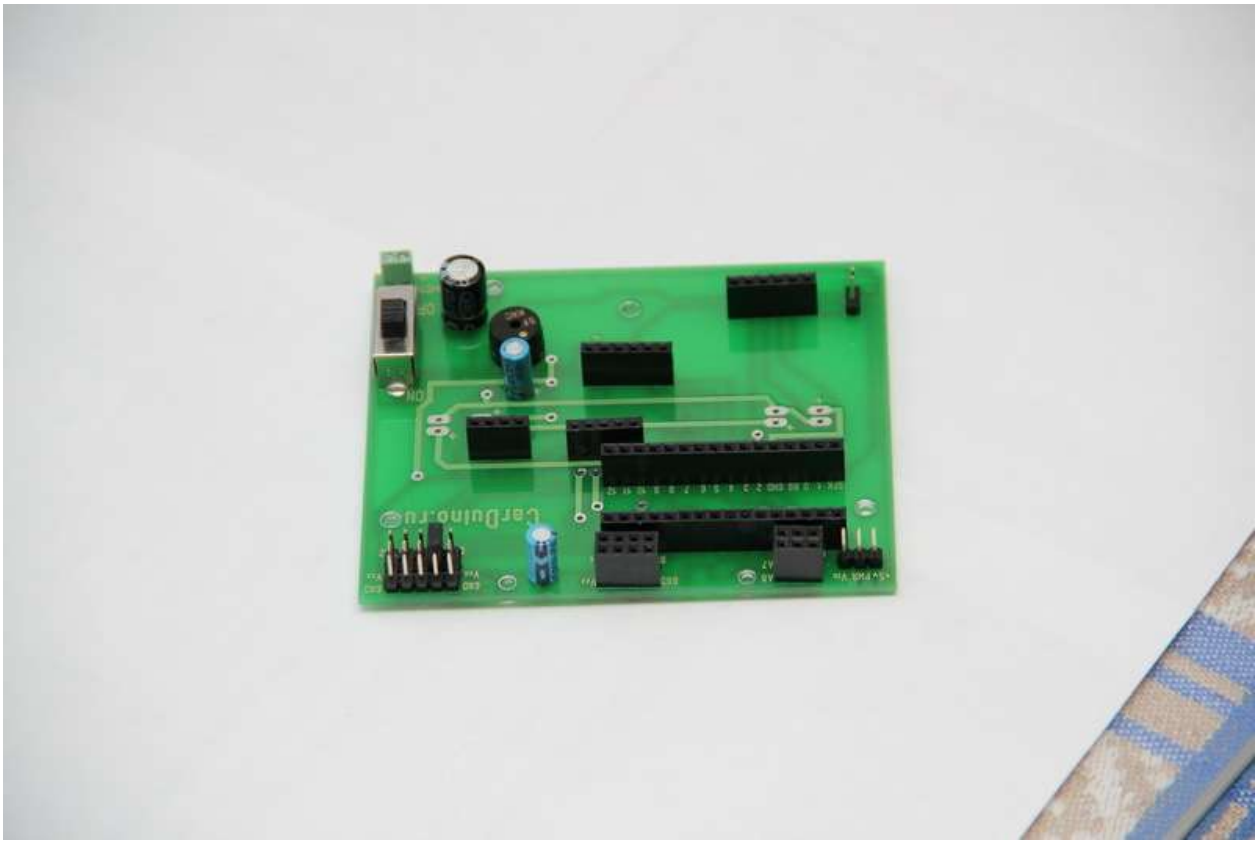


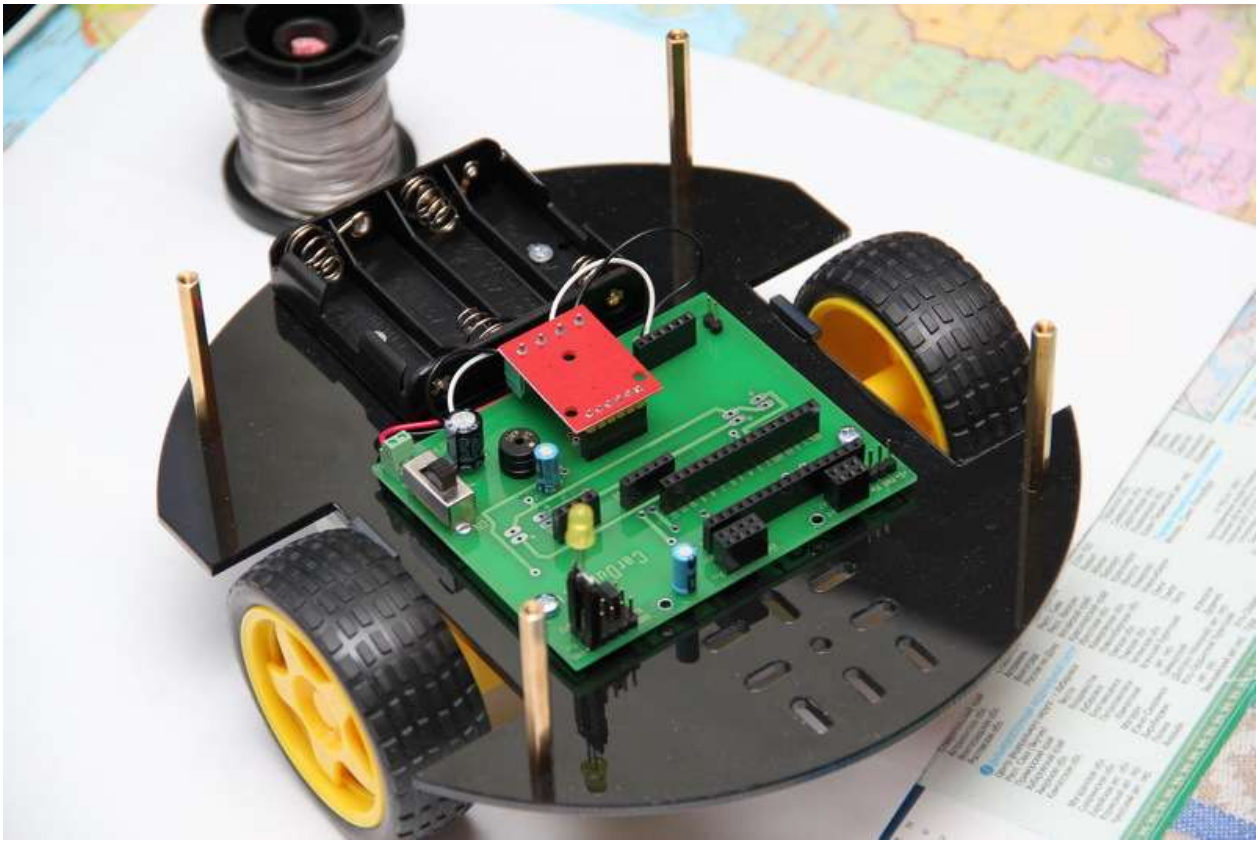


EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

At this stage we assemble the mechanical construction: solder the wires, program the microcontroller. Test the result. Get puzzled. Think it over again. Correct the technical specifications. Re-assemble. Re-test. Get puzzled anew. Keep our chins up. Check the batteries. Discharged. Change them and feel happy.

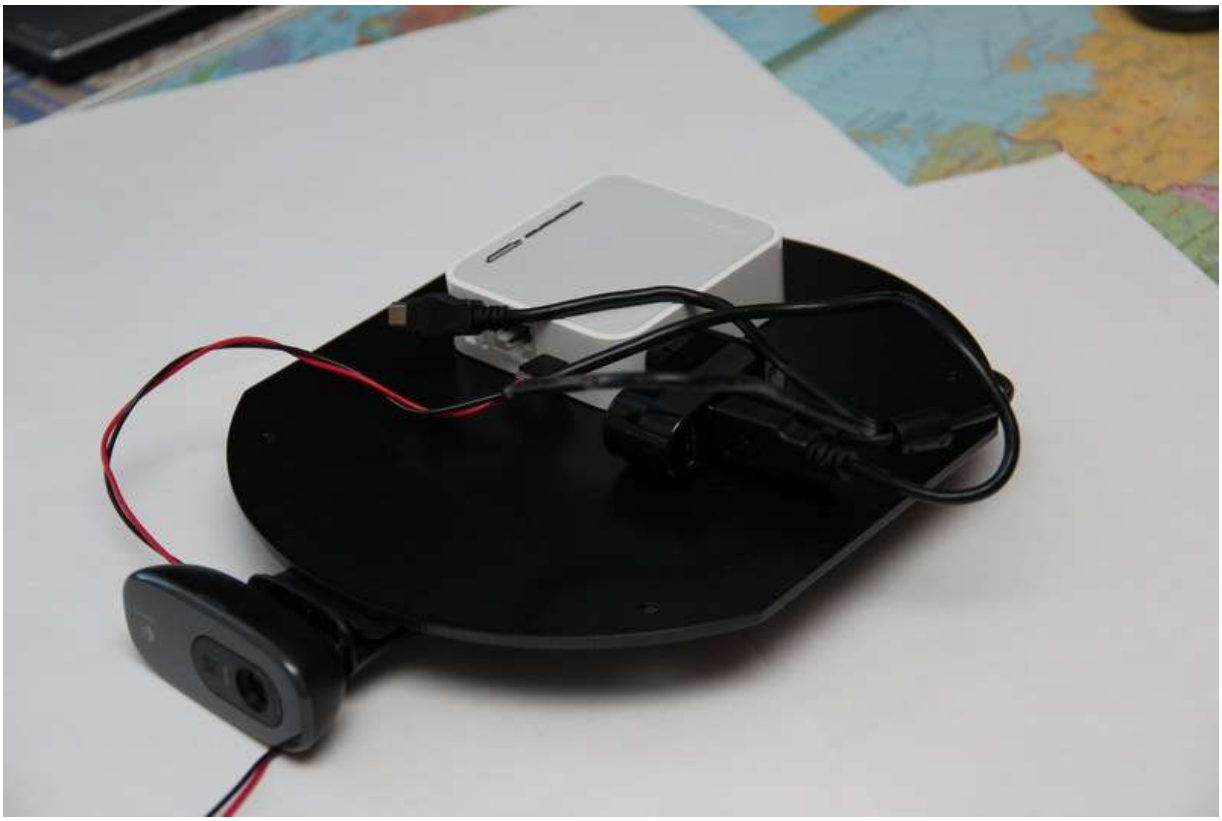
EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

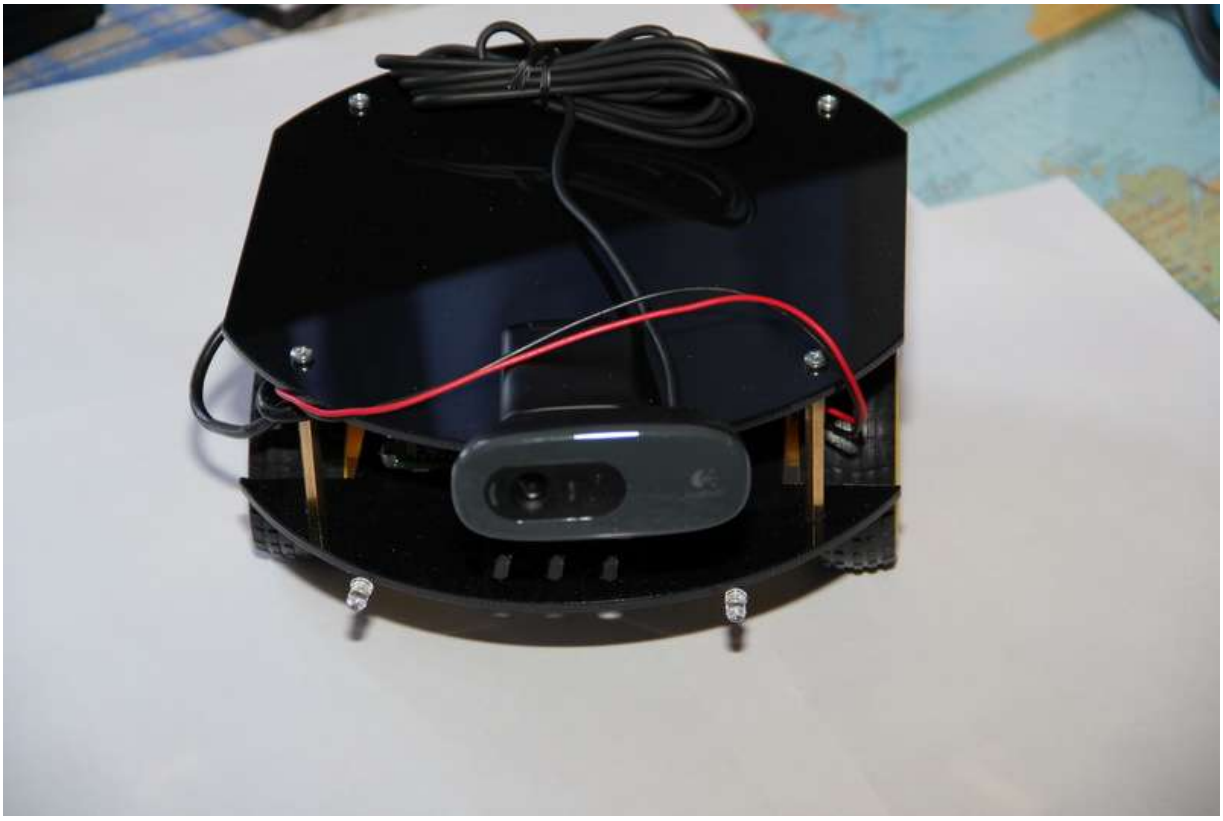




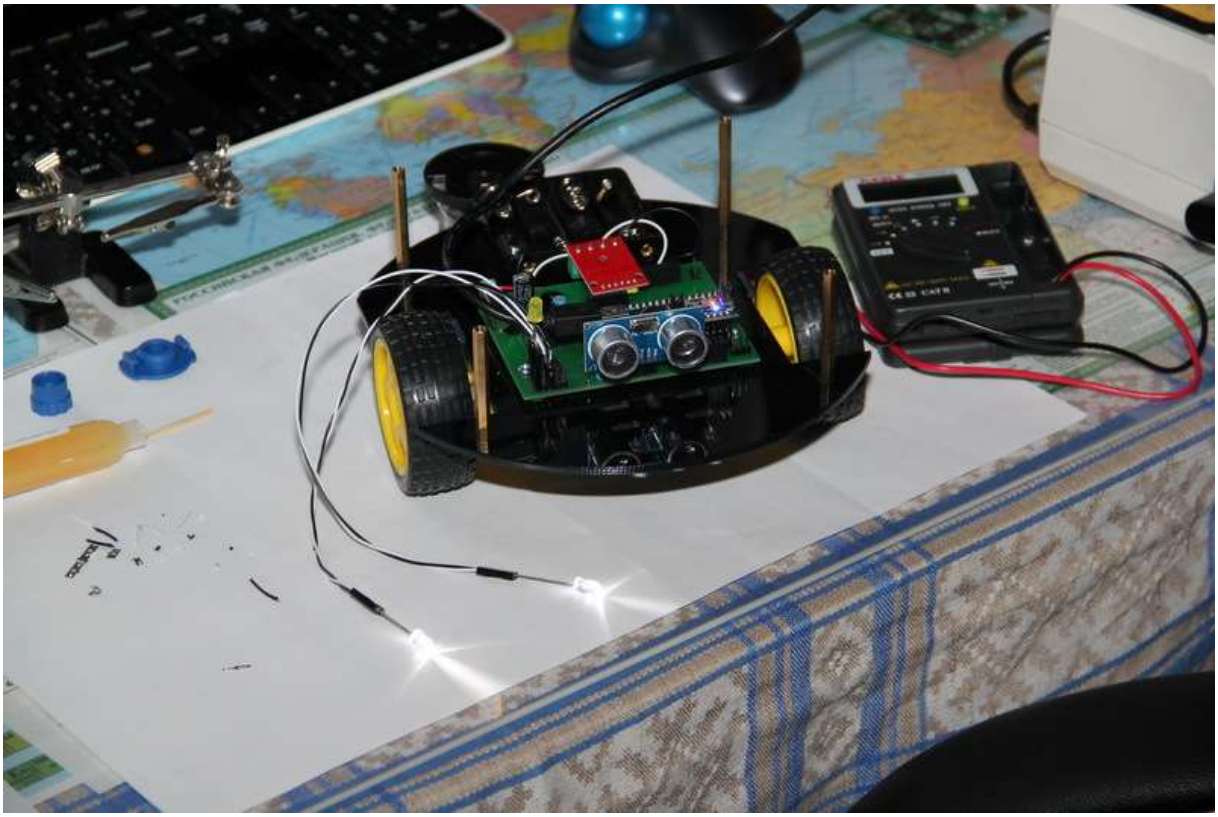
EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

At the end of this stage we need to make a list, which together with the rest of the paperwork will ensure repeatability of the result. The list of the components given somewhere above includes only the key parts, but when placing an order I also added chicken feed, such as: a power switch, an adapter for the battery unit connectors, splits, etc.





Besides, we used various tools and small materials: wires, soldering flux and alloy.



I wrote the program twice.
First time I read actively the arduino.ru website, refreshed the C language in my memory, sorted out the syntax and swore at the code execution sequence (I got used to the parallel execution of the code, when all the computing units work simultaneously, instead of one at a time.

When I started programming for the second time, I organized the program structurally, though not always justified. Still, you have to keep to the given style. The principle is... in the repetition of the program modules drawn above in the structure code. The variables come first in the code then the setup constants are prescribed. Then the initiation block «void setup» follows, and the main «void loop», which task is continuous modules reading. Each module has its own input and output variables affecting its operation. These variables are controlled in the main loop.

The interrogated button sets the modules operation mode. A command received by the interface is transmitted to the other control modules if the autonomous mode is not activated. At first I had only two modules: motors control and light control. Then a sound generation module came to being, but I didn't have time to finish it. Here the structural organization of the program appeared to be useful. It was easy to add or delete modules.

Stage six — understanding of the creation

What did we want to make?

Let's see:

Not bad as a whole. It moves around. Stops. Turns. But… not everything is as good. The robot moves cheerfully load free, but gets slow loaded. Besides, it constantly plays away, deviating to the left. As if hinting, it's no good to be all by itself. Is not it time to take care of the species variety?

Summarizing, I can't but admit that there is much to improve.

1) When moving straight the robot always deflects to the left. As the power consumption increases, the deviation angle becomes greater. Obviously, this is due to the motors parameter spread, perhaps, because of the different operation of the motor driver channels. To eliminate this behavior of the robot it is necessary to connect optical encoders and on the basis of the data from the encoders adjust the algorithm of the motor control module when moving forward or backward. Furthermore, for the power supply it is necessary to have an

integrated voltage regulator, for example, to transfer the power supply to the pair of Li-ION elements using a stepdown converter.

2) The robot very often does not “see” an object in front of it. This is due to the narrow viewing angle of the ultrasonic depth sensor. Obviously, the accuracy and reliability of the distance detection can be increased by installing two or three ultrasonic depth sensors. To determine the distance the sensor readings are polled serially with short breaks to avoid the ultrasonic signals interference.

3) Twice with version 1.0, we observed the turning algorithm stalling: the robot was rotating and did not move forward even if there was no obstacle in front of it. To break the situation it was necessary to stop the robot either by pressing it down with a hand or by lifting it above the surface so that it could “see” a clear path. To eliminate this kind of situations it is necessary to broaden the criteria of the robot behavior self-diagnosis with its enforced re-initiation in case of issues. The WatchDog should be used to eliminate IC stalling.

4) Sometimes the robot does not stop when you release the movement control button on the keyboard when controlling the robot remotely. In this case we recommend quickly pressing and releasing the movement button. Perhaps, the stop command is lost in the input buffer, or the router does not generate the command. To avoid these router failures it is necessary to repeat the stop command by pressing and releasing the motion button a few times.

5) Serial execution of the program operations is the specificity one faces when developing IC firmware. To perform the FORWARD command it is necessary to serially poll the interface modules: the distance detection module, the control module and then the movement module. In this mode the distance detection module causes a delay up to fifty milliseconds. In the event of an excessive growth of the functionality or addition of more ultrasonic sensors the delay may become unacceptable. In the FPGA this problem does not occur. Every unit there works simultaneously. It is possible to receive commands, process the distance, display the data and perform a dozen of other operations. In the IC a parallel performance of even unrelated functions is not possible. However, FPGA firmware designing is a much more painstaking work, which requires a certain qualification. Practically any programmer can write a code in C, while for a code in VHDL/Verilog it is necessary to understand the logics of the microcircuit operation.

4) During the remote control a picture from the on-board camera changes too much making it too difficult to get one’s bearings. To reduce this effect it is necessary to apply smoothing and/or reduction of the display window.

1) During testing a poor design of the mechanical part was revealed. When fully loaded (the platform + the "second floor"), the platform dynamics drops, screeches can be heard when the platform turns. Perhaps the situation can be fixed by raising the voltage supply. In addition, a rear wheel is frequently jammed on turns and ball bearings greasing is of no help because of the poor design.

2) During the manual control if you stop the robot when it moves forward and quickly reverse it, the robot tends to sharply bend forward loosing rear footing.

What is there on the to-do list?

1) The acoustical signal. Development of a generator of audio signals with a certain frequency.

2) Wheel speed aligning at rectilinear movement, using optical sensors and an optical encoder for the wheels.
3) Headlights automatic turning on/off with the help of a light sensor.
4) Improvement of the accuracy of distance determining by increasing the number of ultrasonic sensors to three.
5) Power supply organization on the basis of two 18650 Li-Ion batteries and an integrated voltage stabilizer.
6) Elimination of screeches by means of greasing the gear motors with silicone.
7) Increase of the supply voltage up to 7,0-7,5 v with the subsequent recalculation and re-soldering of resistors for LEDs, to avoid exceeding the maximum relevant output of the IC output.
8) Smooth speed change (smooth start and stop), the change function can be linear or exponential.

Stage seven — growth

go to «Stage two»

Here we are, almost at the end of the story.

I successfully demonstrated the robot and passed it with an accompanying documentation and description of the shortcomings and ways of fixing them to the company. In my opinion, these technical descriptions and technical specifications should help get the correct idea of the electronics "inside" the device, as well as remove the vast majority of subtle technical issues. It seems everybody was satisfied.

Unfortunately, I do not always have enough free time to seriously participate in the developments of Endurance. As far as I know the company looks for students who are interested in robotics. If you are interested and have some experience in the field, as well as ability and desire to participate in robotics, you will be certainly welcome there.

Of course, the use of the robot in this embodiment is seen in ironic light in terms of its industrial applications. However, the platform may be useful. I believe I could use such a robot in the countryside for remote monitoring of by country home. Equipped with an improved system of power supply and servos to rotate the onboard video camera, this robot could be used to monitor the situation in the house and outside.

It is possible to equip the robot with rechargeable Li-ION batteries. In this case the robot could be recharged on a charging dock being put there either by hand or being taught to go there independently.

I wish you success in robotics!
Don’t be afraid to start.

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

A developer's article

What does it mean to develop a robot?

A driver is designed to control two DC motors.

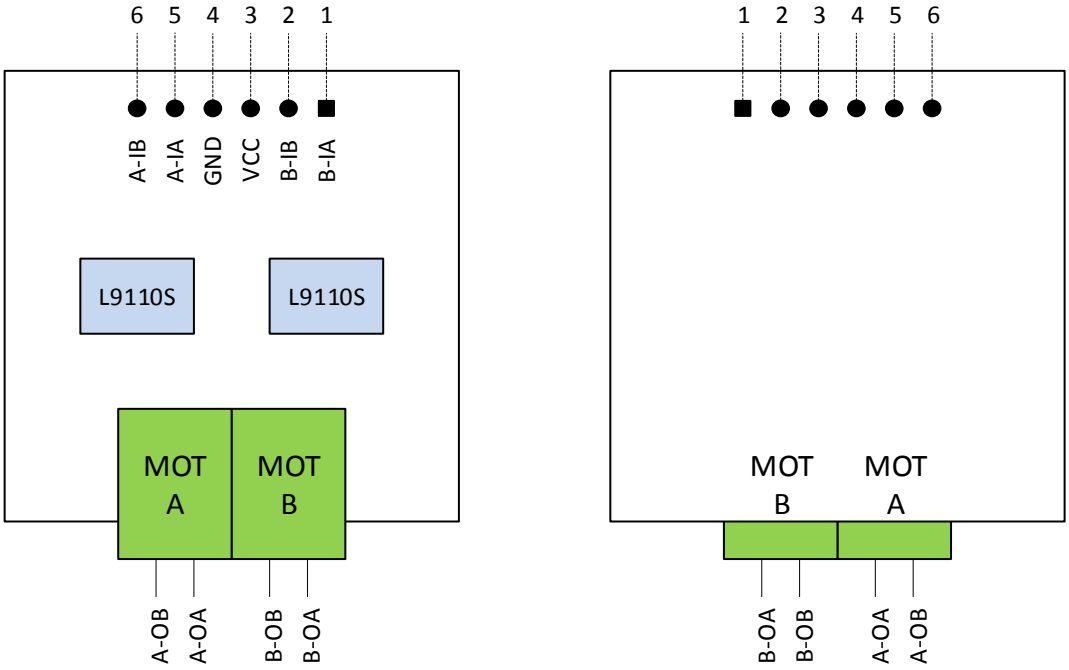


Fig. 1. General view of the double-bridge driver on L9110S base (view from both sides of the board)

General characteristics of the driver

Voltage supply: 2.5v... 12v

Rated relevant for one channel: 800mA.

Peak short-term relevant for each channel: 1500mA.

The module is controlled by the TTL/CMOS logical levels.

Operational temperature: 0 … +80C.

The module size 30x22x12.

Special features

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

The driver can be connected directly to the terminals of the controller.

Built-in output protection diodes.

An LED power indicator.

A convenient screw-shaped clamp for connecting electric motors.

Table 1. Driver's operation logics

IA	IB	OA	OB
0	0	0	0
0	1	0	+
1	0	+	0
1	1	0	0

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

An ultrasonic depth sensor

HC-SR04

A technical description

The circuit is designed for measuring and displaying the distance to the object.

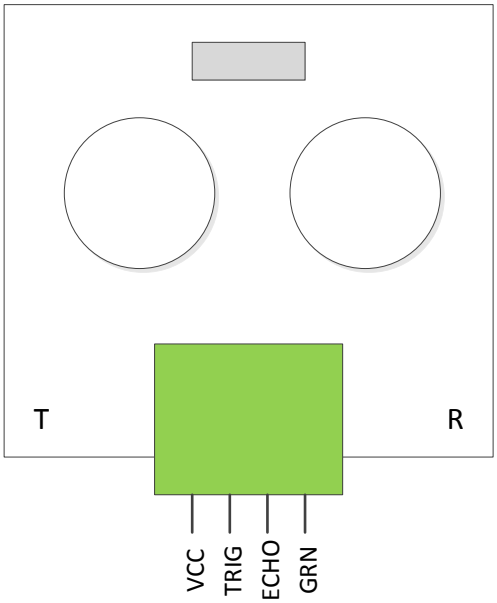


Fig. 1. General view of the ultrasonic depth sensor

General features of the driver.

Supply voltage: 4.8 … 5.5V DC.

Stop relevant: <2mA.

Effective angle: <15°.

Distance detection range: 2 … 400 cm.

Resolution: 0.1 … 0.3 cm.

Ultrasonic frequency range of operation at 40 kHz

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

Driver's operation logics

1) 10 microseconds pulse is supplied to the TRIG pin, and the sonar emits 8 pulses at 40kHz.

2) The ECHO pin issues a signal and its Techo duration is measured. When an obstacle is detected the signal duration is in the range 150 microseconds – 25 milliseconds. When there are no obstacles the signal duration is 38 milliseconds

Converting the pulse duration to the metric distance is done by the following formulas:

Distance in cm: Techo/58;

Distance in inches: Techo/148.

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

The program

```
////////////////////////////////////////////////////////////////////////////////////////////////////

// Robot Platform RASH 1 © AISHex

// 1.0

//

// Create: 23/03/2015

// Modification: 11/04/2015

//

// Description: Program of the RASH 1 platform control

//

////////////////////////////////////////////////////////////////////////////////////////////////////


// pinout

const int LEDtech = 13; //technological signal output to LED

const int LEDdist = 3; //sonar signal output to LED

const int Sonar_t = 14; //data to the sonar

const int Sonar_r = 15; //sonar data

const int Mode = 16; //mode button connection – autonomous or manual: =0 - manual; 1= autonomous

const int MB1 = 4; //left motor - digital

const int MB2 = 5; //left motor - pwm

const int MA1 = 6; //right motor - pwm

const int MA2 = 7; //right motor - digital

const int HDL = 17; //signal output to headlights

const int SPK = 11; //signal output to the speaker

//const int LGHT = ; //light sensor data
```

```
//interface module setting

const long UART_Speed = 9600; //speed UART


//sonar module setting

const int D1 = 20; // PWM function coefficients for sonar data output to LED

const int D2 = 110;

const float A = 2.5;

const float B = 305;


//control module setting

const byte Byte_forward = byte('W'); //command: forward

const byte Byte_back = byte('S'); // command: backward

const byte Byte_left = byte('A'); // command: to the left

const byte Byte_right = byte('D'); // command: to the right

const byte Byte_stop = byte('x'); // command: stop

const byte Byte_sound = byte('C'); // command: acoustic signal/tune

const byte Byte_light = byte('V'); // command: headlights

const int Dist_min = 40; //minimal distance at which the robot starts the detour obstacles algorithm [cm]

const int Cycle_lightoff = 1000; // program repetition number after which the headlights turned on automatically in the autonomous mode, are turned off.

const int Speed_default = 255; // default value, which will be used after turning on the robot until the speed setting command comes.  The speed is reset to the default value when switching to the autonomous mode.


//general settings

const int Delay_prog = 10; //delay in the program repetition, [ms]

const int M_stop = 0; //constants for the motor module (for ease of use)
```

```
const int M_forward = 1;

const int M_back = 2;

const int M_left = 3;

const int M_right = 4;


//modules and functions descriptions


//========== Interface ==========

void _UART_Interf(unsigned int RST, unsigned int *Data); //interface data: Data[0]=1 – command presence; Data[1] – command data

//RST - reset: 0= normal functioning; 1= reset

//Data – interface data array: Data[0]=1 – command presence; Data[1] – command data

//

//Connection to UART is done with the help of the Serial library


//========== Motor ==========

void _Motor(unsigned int RST, unsigned int Mode, unsigned int Speed);

//RST - reset: 0= normal functioning; 1= reset

//Mode - mode: 0= stop; 1= forward movement; 2= backward movement; 3= left turn; 4= right turn

//Speed: motor speed – level formed by PWM: 0= zero level; 255= max.  level

//

//MA1, MA2, MB1, MB2 – motor pinout signals, MB1 и MA2 - digital, MB2 и MA1 - analog (0/255)

//LEDtech – a digital pinout signal for the technological LED is on when a command is performed and is off when the Stop command received.


//========== Sonar ==========
```

unsigned int _Sonar(unsigned int RST);

//RST - reset: 0= normal functioning; 1= reset

//

//Sonar_t – a digital pinout signal for a Trig sonar

//Sonar_r – a digital pinout signal for an Echo sonar

//LEDdist – an analog (0/255) pinout signal for LED, to which a PWM signal will go for distance visualization (the closer an object the brighter the LED)

//========== Control Motor ==========

void _ControlM(unsigned int RST, unsigned int SCTRL, unsigned int DCTRL, unsigned int Mode);

//RST - reset: 0= normal functioning; 1= reset

//SCTRL – data relevance signal DCTRL: 0= non-relevant; 1= relevant

//DCTRL – interface module data (command)

//Mode – control mode: 0= command control; 1= autonomous mode

//

//Module communicates with Sonar, Motor и Rotate modules

//LEDtech – a digital pinout signal for the tech LED blinks once at the speed setting command

//========== Control Light ==========

void _ControlL(unsigned int RST, unsigned int SCTRL, unsigned int DCTRL, unsigned int Mode);

//RST - reset: 0= normal functioning; 1= reset

//SCTRL - data relevance signal DCTRL: 0= not relevant; 1= relevant

//DCTRL – interface module data (command)

//Mode – control mode: 0= command control; 1= autonomous mode

//

```
//HDL – a digital pinout signal for headlights control: 0= headlights off; 1= headlights on


//========== Control Sound ==========

void _ControlS(unsigned int RST, unsigned int SCTRL, unsigned int DCTRL, unsigned int Mode);

//RST - reset: 0= normal functioning; 1= reset

//SCTRL - data relevance signal DCTRL: 0= non-relevant; 1= relevant

//DCTRL - interface module data (command)

//Mode – control mode: 0= command control; 1= autonomous mode

//

//SPK – an analog (0/255) pinout signal for PWM signal output


//========== Rotate ========== - a rotation algorithm, an independent module separated from the ControlM

void _Rotate(unsigned int RST, unsigned int Speed);

//RST - reset: 0= normal functioning; 1= reset

//Speed: motor speed – level formed by PWM: 0= zero level; 255= max. level


//variables declaration

unsigned int CMD[2] = {0,0}; //interface data: CMD[0]=1 – command presence; CMD[1] – command data

unsigned int CTRL[8] = {0,0,0,0,0,0,0,0}; //control data array

//CTRL[0]: 0= normal functioning; 1= modules reset

//CTRL[1]: 0= manual mode; 1= autonomous mode


void setup() {

  //pinout

  pinMode(MB1, OUTPUT); digitalWrite(MB1, LOW);
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
pinMode(MB2, OUTPUT); analogWrite(MB2, 0);
pinMode(MA1, OUTPUT); analogWrite(MA1, 0);
pinMode(MA2, OUTPUT); digitalWrite(MA2, LOW);
pinMode(Sonar_t, OUTPUT); digitalWrite(Sonar_t, LOW);
pinMode(Sonar_r, INPUT); digitalWrite(Sonar_r, LOW);
pinMode(LEDdist, OUTPUT); analogWrite(LEDdist, 0);
pinMode(Mode, INPUT); digitalWrite(Mode, LOW);
pinMode(LEDtech, OUTPUT); digitalWrite(LEDtech, LOW);
pinMode(HDL, OUTPUT); digitalWrite(HDL, LOW);
pinMode(SPK, OUTPUT); analogWrite(SPK, 0);

//modules initialization
Serial.begin(UART_Speed);
_UART_Interf(1, CMD);
_UART_Interf(0, CMD);
_Sonar(1);
_Sonar(0);
_Motor(1, 0, 0);
_Motor(0, 0, 0);
_Rotate(1, 0);
_Rotate(0, 0);
_ControlM(1, 0, 0, 0);
_ControlM(0, 0, 0, 0);
_ControlL(1, 0, 0, 0);
_ControlL(0, 0, 0, 0);
_ControlS(1, 0, 0, 0);
_ControlS(0, 0, 0, 0);
```

```
}



void loop() {
  //Mode button poll and operation mode installation
  if (digitalRead(Mode) == LOW) {
    if (CTRL[1] == 1) { CTRL[0] = 1; } else { CTRL[0] = 0; } //when switching from a different mode,  reset is transmitted to all the circuits. During the next pass reset is cleared.

    CTRL[1] = 0;
  } else {
    if (CTRL[1] == 0) { CTRL[0] = 1; } else { CTRL[0] = 0; }
    CTRL[1] = 1;
  }

  //interface poll, result available in D_Interf
  _UART_Interf(CTRL[0], CMD);

  //control circuit execution
  //motors control
  _ControlM(CTRL[0], CMD[0], CMD[1], CTRL[1]);
  //headlights control
  _ControlL(CTRL[0], CMD[0], CMD[1], CTRL[1]);
  //sound control
  _ControlS(CTRL[0], CMD[0], CMD[1], CTRL[1]);
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
  delay(Delay_prog);
}



//========== Interface module ==========
void _UART_Interf(unsigned int RST, unsigned int *Data) {

unsigned int DUART;
static unsigned int cnt_byte;

  if (RST == 0) {
     if (Serial.available() != 0) {
       DUART = Serial.read();

       switch (cnt_byte) { // check of the packet integrity, in case of at least one failure - reception reset and search for a new packet header
          case 0:
            if (DUART == byte('t')) { cnt_byte++; } else { cnt_byte = 0; }
            Data[0] = 0; Data[1] = 0;
            break;

          case 1:
            if (DUART == byte('x')) { cnt_byte++; } else { cnt_byte = 0; }
            Data[0] = 0; Data[1] = 0;
            break;
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
case 2:
  if (DUART == byte('_')) { cnt_byte++; } else { cnt_byte = 0; }
  Data[0] = 0; Data[1] = 0;
  break;

case 3:
  if (DUART == byte('c')) { cnt_byte++; } else { cnt_byte = 0; }
  Data[0] = 0; Data[1] = 0;
  break;

case 4:
  if (DUART == byte('o')) { cnt_byte++; } else { cnt_byte = 0; }
  Data[0] = 0; Data[1] = 0;
  break;

case 5:
  if (DUART == byte('m')) { cnt_byte++; } else { cnt_byte = 0; }
  Data[0] = 0; Data[1] = 0;
  break;

case 6:
  if (DUART == byte('=')) { cnt_byte++; } else { cnt_byte = 0; }
  Data[0] = 0; Data[1] = 0;
  break;

case 7: //if you reach the end, the packet is correct, and command issue is possible
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
          cnt_byte = 0;
          Data[0] = 1; Data[1] = DUART;
          break;
       }
    } else {
      Data[0] = 0; Data[1] = 0;
    }
  } else {
    cnt_byte = 0;
    Data[0] = 0; Data[1] = 0;
  }
}



//========== Sonar module ==========
unsigned int _Sonar(unsigned int RST) {

unsigned int Duration;

  if (RST == 0) {
    digitalWrite(Sonar_t, HIGH); //ultrasonic sensor initialization
    delayMicroseconds(10);
    digitalWrite(Sonar_t, LOW);
    Duration = pulseIn(Sonar_r, HIGH); //ultrasonic sensor data receipt (pulse width in
ms)
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
    //distance data output to the LED
    if (Duration/58 > D1 && Duration/58 < D2) {
      analogWrite(LEDdist,int((-A*float(Duration/58)+B)));
    } else {
      if (Duration/58 < D1) {
        analogWrite(LEDdist, HIGH);
      } else {
        analogWrite(LEDdist, LOW);
      }
    }

    return Duration/58; //distance data output in cm
  } else {
    digitalWrite(LEDdist, LOW);
    return 0;
  }
}



//========== Control Motor module ==========
void _ControlM(unsigned int RST, unsigned int SCTRL, unsigned int DCTRL, unsigned
int Mode) {


unsigned int Dist;
static unsigned int Speed;
static unsigned long Time_forward;
```

```
if (RST == 0) {


  if (Mode == 0) { //"manual" command mode

    if (SCTRL == 1) { //if a command received

      switch (byte(DCTRL)) { //command decoding

        case Byte_forward:

          Dist = _Sonar(0);

          if (Dist > D1) { _Motor(0, M_forward, Speed); } //if it is possible to move forward
– we move

          break;


        case Byte_back:

          _Motor(0, M_back, Speed);

          break;


        case Byte_left:

          _Motor(0, M_left, Speed);

          break;


        case Byte_right:

          _Motor(0, M_right, Speed);

          break;


        case Byte_stop:

          _Motor(0, M_stop, Speed);

          break;
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
      default:
        break;
    }

    if (DCTRL > 47 && DCTRL < 58) { //speed info
    Speed = (DCTRL-47)*25+5;
      digitalWrite(LEDtech, HIGH);
      delay(1000);
      digitalWrite(LEDtech, LOW);
    }
  }
}

 if (Mode == 1) { //autonomous mode
  Speed = Speed_default;
  Dist = _Sonar(0);

  if (Dist > Dist_min) {
    if (millis()-Time_forward < 21000) {
      _Motor(0, M_forward, Speed);
    } else {  //if it is moving only forward for 20 sec. it is necessary to move it backward. Most probably the robot got stuck in a small room.  :)
      _Motor(0, M_stop, Speed);
      delay(300);
      _Motor(0, M_back, Speed);
      delay(600);
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
        _Motor(0, M_stop, Speed);
        delay(300);
        _Rotate(0, Speed);
        _Motor(0, M_stop, Speed);
        delay(300);

        Time_forward = millis()-1; //-1 for sure millis()-Time_forward will be always a positive number
      }
    } else {
      _Motor(0, M_stop, Speed);
      delay(300);
      _Rotate(0, Speed);
      delay(300);

      Time_forward = millis()-1;
    }
  }
 } else {
  Dist = 0;
  Speed = Speed_default;
  Time_forward = 0;

  _Sonar(1);
  _Motor(0, 0, 0);
  _Rotate(1, 0);
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
    digitalWrite(LEDtech, LOW);
  }
}
```

```
//========== Control Light ==========
void _ControlL(unsigned int RST, unsigned int SCTRL, unsigned int DCTRL, unsigned int Mode) {

static unsigned int Light; // 0= headlights off; 1= headlights on

  if (RST == 0) {

    if (Mode == 0) { //"manual" command mode
      if (SCTRL == 1) { //if you got a command
        switch (byte(DCTRL)) { //command decoding
          case Byte_light:
            if (Light == 0) {
              Light = 1;
              digitalWrite(HDL, HIGH);
            } else {
              Light = 0;
              digitalWrite(HDL, LOW);
            }
            break;
```

```
        default:
          break;
       }
     }
   }

   if (Mode == 1) { //autonomous mode
     //block operations
   }
 } else {
   Light = 0;
   digitalWrite(HDL, LOW);
 }
}
```

```
//========== Control Sound ==========
void _ControlS(unsigned int RST, unsigned int SCTRL, unsigned int DCTRL, unsigned int Mode) {

 if (RST == 0) {

   if (Mode == 0) { //"manual" command mode
     if (SCTRL == 1) { //if you got a command
       switch (byte(DCTRL)) { //command decoding
         case Byte_sound:
           //block operations
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
          break;


        default:
          break;
      }
    }
  }

  if (Mode == 1) { //autonomous mode
    //block operations
  }
 } else {
  analogWrite(SPK, 0);
 }
}



//========== Motor module ==========
void _Motor(unsigned int RST, unsigned int Mode, unsigned int Speed) {
 if (RST == 0) {
  switch (Mode) { //command decoding and action execution
    case 0: //stop
      digitalWrite(LEDtech, LOW);


      digitalWrite(MB1, LOW);
      analogWrite(MB2, 0);
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
    analogWrite(MA1, 0);
    digitalWrite(MA2, LOW);

    break;

  case 1: //forward
    digitalWrite(LEDtech, HIGH);

    digitalWrite(MB1, HIGH);
    analogWrite(MB2, 255-Speed);

    analogWrite(MA1, Speed);
    digitalWrite(MA2, LOW);

    break;

  case 2: //back
    digitalWrite(LEDtech, HIGH);

    digitalWrite(MB1, LOW);
    analogWrite(MB2, Speed);

    analogWrite(MA1, 255-Speed);
    digitalWrite(MA2, HIGH);

    break;
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
    case 3: //left
      digitalWrite(LEDtech, HIGH);

      digitalWrite(MB1, LOW);
      analogWrite(MB2, Speed);

      analogWrite(MA1, Speed);
      digitalWrite(MA2, LOW);

      break;

    case 4: //right
      digitalWrite(LEDtech, HIGH);

      digitalWrite(MB1, HIGH);
      analogWrite(MB2, 255-Speed);

      analogWrite(MA1, 255-Speed);
      digitalWrite(MA2, HIGH);
      break;

    default:
      break;
  }
} else {
  digitalWrite(LEDtech, LOW);
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
    digitalWrite(MB1, LOW);
    analogWrite(MB2, 0);

    analogWrite(MA1, 0);
    digitalWrite(MA2, LOW);
  }
}



//========== Rotate ========== - rotation algorithm
void _Rotate(unsigned int RST, unsigned int Speed) {

unsigned int Dist;
static unsigned int Num;
unsigned int cnt;
unsigned long Now_time;

  if (RST == 0) {
    do {
      if (Num%2 == 0) { //parity computing
        if (Num >= 0 && Num < 128) {
          _Motor(0, M_right, Speed);
          delay(100);
          _Motor(0, M_stop, Speed);
          delay(100);
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
    }
    if (Num >= 128 && Num < 255) {
      _Motor(0, M_right, Speed);
      delay(200);
      _Motor(0, M_stop, Speed);
      delay(100);
    }
  } else {
     if (Num >= 0 && Num < 128) {
      _Motor(0, M_left, Speed);
      delay(250);
      _Motor(0, M_stop, Speed);
      delay(100);
    }
    if (Num >= 128 && Num < 255) {
      _Motor(0, M_left, Speed);
      delay(150);
      _Motor(0, M_stop, Segment);
      delay(100);
    }
  }
  cnt++;

  Dist = _Sonar(0);

} while (Dist < Dist_min && cnt <= 3); //we exit the cycle, if the distance is clear or we rotated 3 times.
```

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

```
    cnt = 0;


    Now_time = millis(); //time reading
    while (Now_time > 255) { //time limitation
                    Now_time -= 255;
    }
    Num += Now_time; //new random number generation: old number+time
    while (Num > 255) { //new number limitation if necessary
      Num -= 255;
    }


  } else {
    Dist = 0;
    Num = 0;
    cnt = 0;
    Now_time = 0;
    _Motor(1, 0, 0);
    _Sonar(1);
  }
}
```

The control module

The control

A technical description

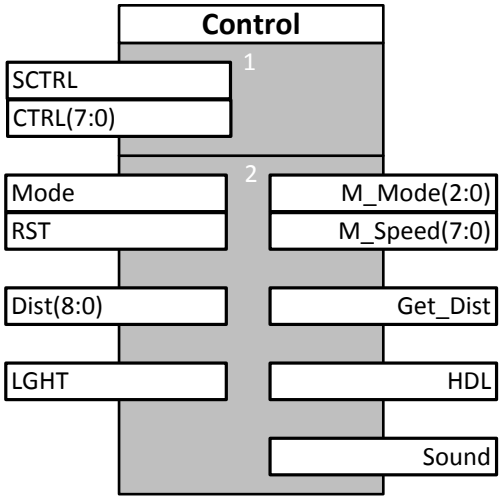


Fig. 1. Control module

Table 1. Module ports

Name	Description		I/O
External control (1)			
SCTRL	Data relevance signal CTRL [1 cycle]		I
	0	Not relevant	
	1	Relevant	
CTRL(7:0)	Control command byte		I
External control (1)			
Mode	Platform operation mode [level]		I
	0	External commands control	

Name	Description		I/O
	1	Autonomous mode	
RST	Resent signal[1 cycle]		I
	0	Normal work	
	1	Module reset	
Dist(8:0)	Distance to the obstacle		I
LGHT	Light sensor signal [level]		I
	0	Sufficient light	
	1	Insufficient light	
M_Mode(2:0)	Control command to the motor module		O
	0	Stop	
	1	Forward	
	2	Backward	
	3	To the left	
	4	To the right	
	5	Reserve	
	6	Reserve	
	7	Reserve	
M_Speed(7:0)	Motor speed		O
Get_Dist	Request for the distance to the obstacle [1 такт]		O
	0	No request	
	1	Request	
HDL	Headlights control signal [level]		O
	0	Headlights off	
	1	Headlights on	
Sound	Beep signal to the acoustic generator [1 cycle]		O

Name	Description		I/O
	0	No sound generation	
	1	Sound generation	

Table 2. Module parameters

Parameter	Limitation	Default data	Description
Given target parameters			
Dist_min	2 < X < 400	40	Minimal distance to the obstacle, at which the movement direction should be changed (cm).
Speed_default	0 < X < 255	255	Speed value, set after the module start and in the autonomous mode
Calculated parameters			
-	-		-

A general description of the module operation

The control module:

- Receives the control byte from the interface module and performs the following commands: platform movement, headlights and sound control;
- Switches (by the control button) to the autonomous mode, moves the platform, receives the light sensor data and controls the headlights turning on/off.
- Controls forward movement in any operation mode by continuous polling the depth sensor; changes forward movement in the autonomous mode.

A general description of the operation algorithm

After the start the operation mode is checked.

If the regular mode is set, the interface module is polled. A received command is executed. For safety reasons, before performing the command forward, the path ahead is checked for the presence of obstacles (poll of the distance sensor module). If there is no obstacle, the command is executed.

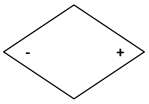
If the autonomous mode is set, the light sensor is polled. Depending on its data the headlights are turned on or off. Then the sonar module is polled. If there is no obstacle ahead, the platform moves forward, otherwise the rotation algorithm, which tries to turn the platform, is activated. In a closed space the platform may come to a dead standstill when moving forward for a long time. In this case the return motion with the following activation of the rotation algorithm is performed. The rotation algorithm allows making turns at any angle.

Then the sonar module is polled. If there is an obstacle on the way, then 2 more turns are performed in the same direction, and exit from the algorithm. If the obstacle is still on the way, another option may come into action during the next performance of the algorithm, which may help out.

Due to the fact that the code is implemented in C, and the program is based on the sequential operation, the module algorithm takes into account the sequence of the modules start-up and data acquisition.

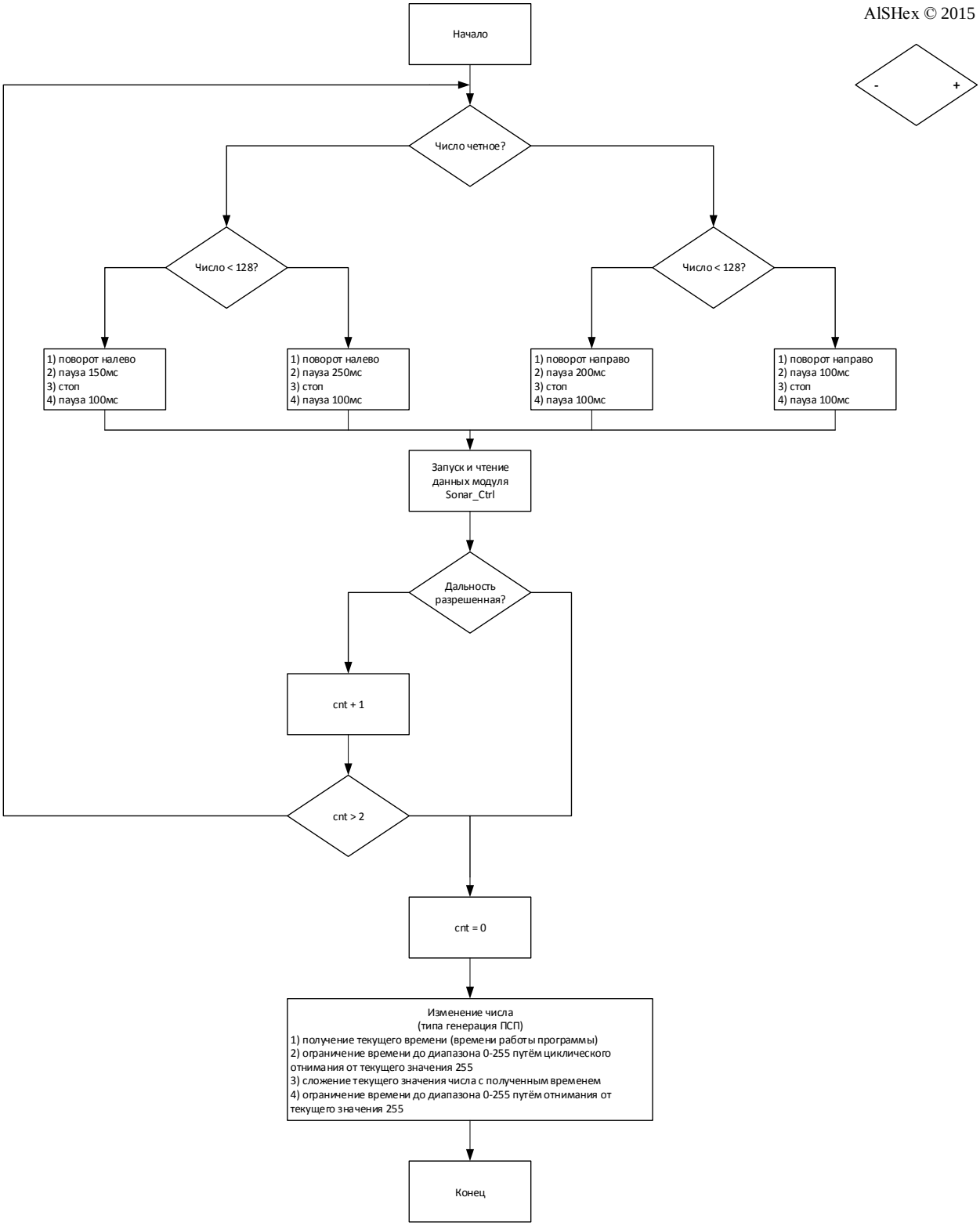
The operation algorithm of the control diagram:

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302



EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

The rotation algorithm:



The motor control module

Motor_Ctrl

A technical description

The operation logics of the module is similar to the control of the double-bridge motor driver Motor Shield on the basis of the L9110S microcircuit. The module is designated for the coordinated control of the pair of the motors for the wheeled platform movement.

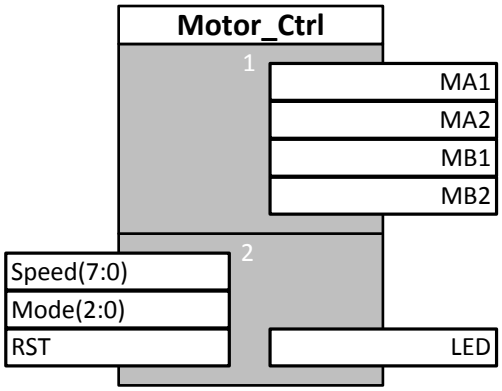


Fig. 1. Motor_Ctrl module

Table 1. Module ports

Name	Description		I/O
Module ports (1)			
MA1	Signal to A motor		O
	0	Low level	
	1	High level	
MA	PWM-signal to A motor		O
	0	Low level	
	1	High level	
MB1	Signal to B motor		O
	0	Low level	

Name	Description		I/O
	1	High level	
MB2	PWM-signal to motor B		O
	0	Low level	
	1	High level	
Control (2)			
Speed(7:0)	The speed value sets values for PWM		I
Mode(2:0)	Motors operation mode		I
	0	No movement	
	1	Forward movement with a preset speed	
	2	Backward movement with a preset speed	
	3	Turn to the left with a preset speed	
	4	Turn to the right with a preset speed	
	5	Reserve	
	6	Reserve	
	7	Reserve	
RST	Module reset		I
	0	Normal work	
	1	Module reset	
LED	Module operation indication signal		I
	0	Low level	
	1	High level	

A general description of the module operation

Motor_Ctrl module

- Receives speed values and movement direction for the motors control;
- Controls the motors taking into account the preset speed and uses the available ports of the PWM forming.

The module controls the **Speed** and **Mode** ports, and when the operation mode is set, generates levels to its output ports.

The PWM is formed taking into account the 8-bit DAC. If the speed is zero, the output signal is also zero. If the Speed is 255, the output signal has a high level during the whole PWM period. In case of the intermediate Speed values the corresponding PWM signals are formed. The PWM signal frequency is about 500Hz (when using an Arduino mini with a 16MHz clock generator).

When levels are transmitted to the module output, a high level is transmitted to the LED output. At the output reset the signal is also reset. Thus, commands performance is indicated. When the platform is moving, the LED is glowing, in the opposite case it is not.

At the reset signal the module must issue zero levels to the motors, that equals to the **Mode**=0 command (stop).

Table 2. Module operation logics

Mode	Motor A – right motor		Motor B – left motor	
	MA1	MA2	MB1	MB2
0 / stop	0	0	0	0
1 / forward	0	Speed	0	Speed
2 / backward	1	255-Speed	1	255-Speed
3 / to the left	0	Speed	1	255-Speed
4 / to the right	1	255-Speed	0	Speed

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

The control module of the ultrasonic depth sensor

Sonar_Ctrl

A technical description

The operation logics of the module is built on the basis of the ultrasonic ranging module Sonar H-SR04 circuit control. The objective of the module is generation of an activating Trig pulse to the sonar, receiving a response from the sonar and displaying the distance to the object in cm.

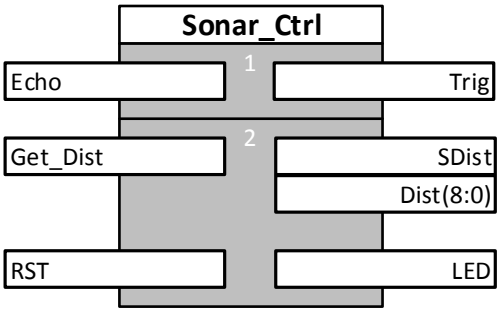


Fig. 1. Sonar_Ctrl module

Table 1. Module ports

Name	Description		I/O
Interaction with the sonar (1)			
Trig	Signal to the sonar		O
	0	Low level	
	1	High level	
Echo	Signal from the sonar		I
	0	Low level	
	1	High level	
Control (2)			

Name	Description		I/O
Get_Dist	Measurement request [1cycle]		I
	0	No request	
	1	Distance request	
RST	Module reset		I
	0	Normal work	
	1	Module reset	
SDist	Data relevance signal Dist [1cycle]		O
	0	Non relevant	
	1	Relevant	
Dist(8:0)	Distance to the object in cm: 2 … 400		O
LED	PWM signal proportional to the Dist value		O
	0	Low level	
	1	High level	

Table 2. Module parameters

Parameter	Limitation	Default value	Description
Given parameters			
D1	0 < X < 2^30	20	Minimal controlled distance in cm
D2	0 < X < 3000	110	Maximal controlled distance in cm
A	0 < X < 3000	2,5	PWM function coefficient
B	0 < X < 2^24	305	PWM function coefficient
Calculated parameters			
-	-		-

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

A general description of the module operation

The sonar_Ctrl module:

- Receives a distance measurement request and displays the value in cm;
- Interacts with the sonar circuit.

The module constantly monitors the Get_Dist port. When a request is detected a 10 ms pulse is generated to the Trig port, and a pulse from the Echo port is expected. After measuring the pulse, the signal duration divided by 58 and a ready signal to measure SDist are set to the Dist output port.

At the reset signal the module stops measure taking and outputs a zero level to the LED port.

For clarity, the distance value can be displayed on the LED (LED port) using PWM; the closer the object, the brighter the glow of the LED. The PWM signal is 8-bit, i.e, the valid values are in the range 0 ... 255.

If we take the LED level as y, the distance to the object as x, then:

$0 < x < D1 \rightarrow y = 255$

$D1 < x < D2 \rightarrow y = -A*x+B$

$D2 < x < \infty \rightarrow y = 0$

If we take the distance range from 20 cm to 110 cm (10 steps, 10 cm each):

D1 = 20 cm

D2 = 110 cm

The 8-bit PWM range is from 0 to 255; let’s take 10 steps 25 apiece: 255… 30.

From the system of equations:

$20 = -A*255+Btht$

$110 = -A*30+B$

EnduranceRobots.com
gf@endurancerobots.com
+7 916 2254302

We get:

$$y = -2.5*x+305$$

The interface module

UART_Interf

A technical description

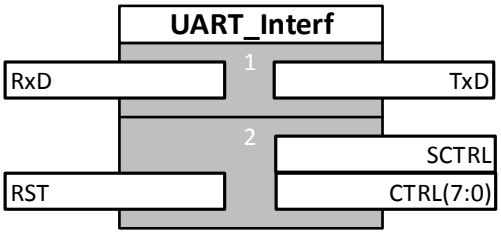


Fig. 1. UART_Interf module

Table 1. Module ports

Name	Description		I/O
Interface ports (1)			
RxD	Incoming data port		I
	0	Low level	
	1	High level	
TxD	Outgoing data port		O
	0	Low level	
	1	High level	
Control (2)			
RST	Reset signal		I
	0	Normal operation	
	1	Module reset	
SCTRL	Data relevance system CTRL (1cycle)		O

Name	Description		I/O
	0	Non relevant	
	1	Relevant	
CTRL(7:0)	Control command byte		O

Table 2. Module parameters

Parameter	Limit	Default value	Description
Given parameters			
Speed	0 < X < 2^30	57600	Speed UART,
Calculated parameters			
-	-		-

A general description of the module operation

The UART_Interf module:

- Interacts with the external device via the UART protocol;
- Checks the received packet data integrity;
- Outputs control commands.

The module receives data via the UART, collecting them in bytes. The bytes sequence for the command decoding must comply with the protocol of the exchange with the external device. The data reception scheme provides the received packet structure checking, and, when correct, outputs control byte commands, covered with the SCTRL signal.

At the reset signal the module should re-initiate its logics and reset the values at the output ports.

The protocol of interaction with external devices

The interaction with external devices takes place via the UART at the speed of 57600 baud.

The data packet consists of 8 bytes. The first 7 bytes constitute the prefix and are constant in every packet. The eighth byte is a command byte.

Table 3. The packet structure

Byte #	1	2	3	4	5	6	7	8
Command symbol	t	x	_	c	o	M	=	
Command byte	0x74	0x78	0x5F	0x63	0x6F	0x36	0x3D	

Table 4. Control commands (the eighth byte in the packet)

Command symbol	Command byte	Command
W	0x57	Forward
S	0x53	Backward
A	0x41	Left turn
D	0x44	Right turn
U	0x55	Reserve
J	0x4A	Reserve
H	0x48	Reserve
K	0x4B	Reserve
X	0x78	Any motor stop
C	0x43	Sound signal generation
V	0x56	Headlights on
B	0x42	Headlights off
1	0x31	Speed setting, level 1
…	…	…
9	0x39	Speed setting, level 9

Since the exchange protocol is fixed, the reception circuit, to avoid failures, receives each new byte coming via the UART, compares it with the expected and if the packet structure matches, the last byte is considered a command and sent to the module port . If the byte is not expected, the reception circuit is reset. In case of the exchange failure, this ensures the correct reception of the next packet (following the spoilt) without the reception circuit reset upon timeout that determines the end of data reception.