TEAM REPORT

1. INTRODUCTION

Nanook is a robot made by NASA summer interns in a bootcamp led by Michael Comberiate (NASA-Mike). Three robots (Nanook, Penguin 1 and Penguin 2) were developed to explore a remote terrain, while challenging communication protocols, large data transference, and ability to work in harsh environments.

According to P. Stakem [1], the goal of the project was "to assemble a networked team of autonomous robots to be used for three-dimensional terrain mapping, high-resolution imaging, and sample collection in unexplored territories." The mothership (Nanook) was responsible for locating their auxiliar robots (Penguin 1 and 2) based on sphere detection (each auxiliar robot had unique spheres placed at their top). They tracked an area sending stitched successive images to the control center, where high level strategy was decided and sent as commands to the robots. The robots are semi-autonomous requiring a minimal command from the control center.

A lot of incredible pioneer work was done during the few weeks of the NASA Engineering Bootcamp, but a good documentation about this project was not developed concomitantly.

Almost five years after the end of Nanook's project, a reverse engineering project is being developed to document it. Characterized by "taking apart an object to see how it works in order to duplicate or enhance"[2] it, a reverse engineering project allows the reconnaissance of the entire system and its characterization for further consultancy and implementation by establishing the adequate documentation.

2. INITIAL SCENARIO

In the end of June, Michael Comberiate brought the Nanook stating that the main problems were:

I. Lack of documentation;
II. Start-up glitch;
III. Display software for the LIDAR view (it doesn't show up the scanning profile image);
IV. Lack of obstacle avoidance.

His main concerns were related to overcharge the batteries and to crash the robot into some obstacle. The battery charger was oversized for the robot's batteries capacity, which make it propitious to overcharging them. Controling the robot was supposed to be easy, but a detailed instruction of how to operate it was missing. Learning how to stop it with the GUI was a caution requirement before running it on the floor.

Two of the people who worked with Nanook before shared with us their knowledge about the robot. Their observations were that the internal code should be able to be taken

from the on-board computer. As long as a monitor and a keyboard were connected to the robot, the code could be extracted. However, it would be convoluted. Also, they told us that the Nanook was a project where different teams worked with in different years, which make it very complex. Every time new functionalities were added and some were disabled without any documentation about the changings.

Regarding the 3D image, we were assured this functionality was finished and should work properly. Someone changed the code in a way that this image wouldn't open automatically, so we would have to discover which button should be clicked to open it.

3. OBJECTIVES

The main aim of this project is to develop a reverse engineering analysis in Nanook. Due to the team efforts, the following objectives were stablished:

A. Reverse Engineering Nanook to understand how it works
B. Hardware Documentation
C. Software Documentation
D. Interface Description
E. Glitch Analysis
F. Improvements

4. RESEARCH

All the information available about Nanook was gathered in a shared folder. The attachment I summarizes the technical information obtained. Information about nanook purposes and old news were also grouped, since it could help us understanding the robot.

We highlight below a piece of information obtained in [1], which shows the reason for using dot matrix to compose the LIDAR images. The communication process between the robot and the manned computer was a great advance for that time:

"*Communicating between their* [nanook team] *offices in Maryland and the robot at the South Pole is similar to communicating to a roving robot on Mars. The engineers experience satellite synchronization issues with volumes of data as the robot takes digital dot-matrix pictures of objects it finds, similar to what they will experience when transmitting with equipment on another planet. The images are sent to the engineers, who then decide what objects require a closer look. Dot matrix is used because it will transmit faster than a digital camera image. The robot uses laser-based guidance known as LADAR (Laser Detection and Ranging) to find and take images of objects. It is semi-autonomous and has 3D scanning capability with image stitching.*"

5. VISUAL INSPECTION

After the research about Nanook, we did a visual inspection to gather more information about Nanook.

5.1. COMPONENTS IDENTIFIED

The table below lists all the components identified as being part of the robot's circuit. The datasheets are attached in the end of this report.

Table 01 - Information About Robot's Components Found During Visual Inspection

Components	Brief Description	Additional Information	Reference Link
Isolated DC/DC Converters 20W 24V TO 5V 4.0A	Converter to 5V (obsolete)	Power Converter System	http://www.mouser.com/ds/2/281/murata_tdc_wpn20r-547589.pdf
Isolated DC/DC Converters 20W 24V TO 12V 1.66A	Converter to 12V (obsolete)	Power Converter System	http://www.mouser.com/ds/2/281/murata_tdc_wpn20r-547589.pdf
CP2102	Single-Chip USB to UART Bridge	DC Motors System	https://cdn.sparkfun.com/datasheets/BreakoutBoards/CP2102_v1.2.pdf
CP210x Driver	CP210x USB to UART Bridge VCP Drivers	DC Motors System	http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx
SABERTOOTH 2X25	Motor Driver	DC Motors System	https://www.dimensionengineering.com/datasheets/Sabertooth2x25.pdf
AD4-B-S	Quadrature to RS232 Adapter	DC Motors System	http://www.digchip.com/datasheets/parts/datasheet/502/AD4-B-pdf.php

R256 Controller	Step Motor Driver	LIDAR Step Motor System	http://www.linengineering.com/drivers-Controllers/PDF/R256_Manual1.08.pdf
Intel model D945GCLF2	Motherboard	General Circuit	http://downloadmirror.intel.com/16960/eng/D945GCLF2_TechProdSpec02.pdf
Kingston V125-S2	Flash Drive	General Circuit	http://www.kingston.com/datasheets/SNV125-S2_us.pdf
SICK LMS 221-30206	LIDAR	LIDAR Circuit	http://sicktoolbox.sourceforge.net/docs/sick-lms-technical-description.pdf

5.2. ADDITIONAL INFORMATION

5.2.1. BATTERIES

We observed that one of the batteries (a blue one, in the left side) is swollen. Maybe this battery has been overcharged before. We would recommend to replace this battery soon, or at least to keep monitoring it.

Figure 01 – Battery swelled

In a further hardware inspection, we observed that the other blue battery (at the front of the robot) was also swelled as the figure below is showing.

Figure 02 – Another battery swelled

The battery voltages were measured and the values are listed below:

Results: Right Battery (Red) – 13.09 V

Left Battery (Blue) – 13.21 V

Table 02 – Batteries Voltage

Battery		**Voltage (V)**
Right	Red	13.09
Left	Blue	13.21

Both sides presented similar voltage. However, while we were working with the robot, we discovered that the red batteries discharge faster than the blue ones. We measured the voltage levels in each battery and the results are shown below.

Table 03 – Batteries Voltage after Using the Robot for a While

Battery Number Identification	Battery Connections Number	Battery Color	Voltage (V)
1	1-2	Blue	13.01
2	3-4	Blue	12.93
3	5-6	Red	7.75
4	7-8	Red	8.68

The blue batteries are from a better brand than the red ones, which explains the lower voltages in the red batteries. While the blue batteries can maintain the adequate voltage much longer, the red batteries discharge faster. As both sets of batteries (two blue and two red batteries) are connected in parallel, the robot cannot work properly when the red ones are discharged. The normal procedure in this case is to charge the batteries, which makes the blue batteries overcharge. The overcharge of the batteries explains why they are both swollen.

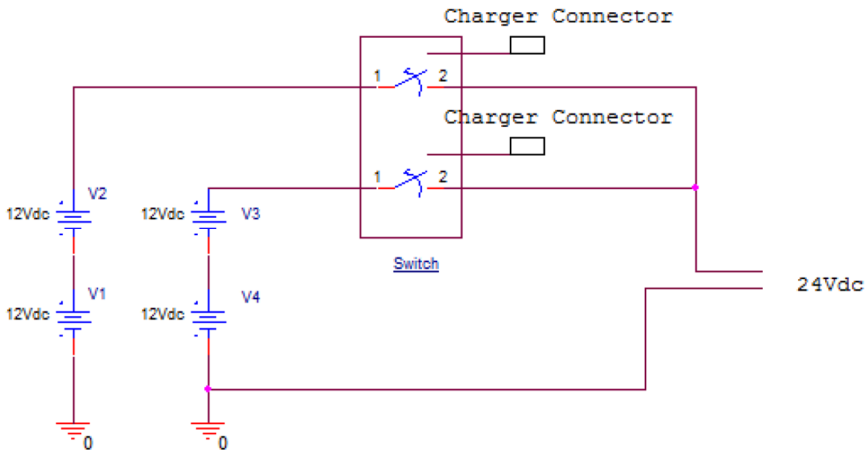


Figure 03 – Battery Parallel Connection Schematic

The recommended action is to change the swollen batteries and substitute for batteries of the same model. By doing this, we optimize the chance that the batteries are going to maintain the same voltage, avoiding to overcharge them.

As we do not have four blue batteries (the best ones), we changed the blue batteries for red ones. By doing this, the robot stayed with four red batteries. However, the two red batteries placed at the robot were old ones and they were not working properly.

To testify the malfunction of the added red batteries, we try to work with the robot after the changing of the batteries. First, one of the batteries couldn't charge more than 11V.

Then, when we tried to make the robot move, the initial movement of the robot was not adequate as it should be. Only after almost 30 minutes trying, the robot worked properly. The batteries are more efficient when they are warm. As the batteries were not good, the temperature affected them for a long time and strong enough to make the robot not work properly for a while.

Our final decision was to remove the bad batteries (now the added red ones) and substitute for gray batteries that we have stored. In order to not have the same problem as before (different batteries in parallel), we put one red batteries in series with one gray battery in one side, and repeated the same configuration in the other side.

5.2.2. PHYSICAL INFORMATION

The robot weights 147 lbs (66.8 kg) and its dimensions are roughly 51x70x60 cm.

6. PROCEDURES

In the following pages, our work is detailed following the methodology we took to achieve good results.

I - SPACE TO WORK

First, we defined a space in the laboratory to work. We select a table, some screens, mouses, and keyboards.

II - MANAGEMENT TASKS

After defining our work space, we implemented some management tasks, as described below:

i) A group connecting the people working on the project was created;

ii) All the data available was put in the same place: a folder in the google drive;

iii) All members have access to all the data available through the link: <https://drive.google.com/folderview?id=0Bycg1GnzcvjSfndPMEUwRGZrSVFNYk1HWXNUQVFvdVYyT2lGSXB5cTA3Y284RkMxWDN3dUE&usp=sharing>;

iv) A division of tasks and responsabilities were defined. Although everybody worked in the entire project, attributing some leadership for the tasks facilitate the management of the project. The individual attributions are listed below:

Table 04 – Team members and their responsibilities

Team Member	Responsibility
Ana Reboucas	Team Lead
Nadson Sousa	Software Lead
Thiago Alves	Hardware Lead

III - COMPUTER SOFTWARE

The laptop provided to us was analysed. We could detect a folder with some Nanook information and the past code versions. We have also identified that the executable software present in the Desktop was not from the most recent version of the code.

IV - POWER SUPPLY

We hooked an external power supply to the motherboard, which is important because by using a power supply connected directly to the motherboard we can guarantee that just the part we are interested in will be powered on. We do not have interest to power the mechanical part to just get the on-board code. In fact, this would be less safe.

Some difficulties overcomed were:

i) Finding a power supply with the correct connections:

We needed a male-male connection to connect the power supply in the system as it is designed. We had to adapt the original power supply to be able to do this connection.

ii) Problems to turn on the embedded system using the computer power supply:

Right now, we cannot say exactly what the power issue was. At the initial moment, we were just pulling the button and releasing it to start and it was not working. We tried to measure the voltage in the power supply, but we forgot to connect a jumper in one of the connectors to test the voltage. As the power supply was not starting because of that, the multimeter wasn't showing any voltage. After we discovered the power supply was working, we connected it again to the robot. This time, we also pulled the button holding it for a while. Doing this, everything worked fine. We tested again, but we haven't had any issue. Even pulling the button for a second was enough. Disconnecting the power supply this time was harder than the previous times, which may let us think there were a problem with the connection. We would require more tests to figure out what was the real problem.

V – CONNECTING TO THE MOTHERBOARD

To be able to connect to the motherboard, keyboard, mouse, and monitor were used, so it was not necessary to use wireless connection.

There are two available USB ports and PS/2 mouse and keyboard ports in the robot. We used a USB keyboard to connect a flash drive to get the code. The other USB port was used to connect a keyboard and the mouse was connected in the PS/2 port. An important note is that as the mouse was connected in the PS/2 port, we had to reinitialize the computer before being able to use the mouse after it was connected.

After power up the CPU from a standard PC power supply, the code was pulled from the hard drive to a flash drive.

VI - DOWNLOADING VISUAL STUDIO

We discovered that the version of Visual Studio used to develop the software project which is embedded in the robot was Visual Studio 2008. In order to conclude that, we opened one of the VCPROJ files using notepad and obtained the following description in the head of the file: Version="9.00". After researching this information in the internet, we discovered the version 9.00 corresponds to Visual Studio 2008.

We downloaded this version to analyze the code. It's important to note that we could have used the newest version of Visual Studios. However, the configuration files would be automatically converted to the corresponding version. If someone want to use the older version again, it woul not be possible. That's why we prefered to preserve the original development environment.

We have done a copied version of the code to use in the Visual Studios 2013 Ultimate to be able to generate the UML diagrams. However, the original code was maintained safely in another folder.

VII - HARDWARE INSPECTION

There was a first general inspection on 6/29/2015 to figure out the schematics of the robot hardware in order to better understand it. The results are described in the next pages. These were our first analysis and comprehension about the hardware. Some information was missing at the time we finished the first inspection, that was complemented in further inspections.

Six main USB connections were identified in the motherboard. Connections identified as one and two are the feedback signal from the motors (part of the motors control system), which came through the Quadrature to RS232 converter (AD4-B-S). Because of the converter, we believe that there are encoders inside the black box in which the motors lie. A third USB connection (connection number three) goes through a USB/UART converter which sends signals to the motor controller (Sabertooth). Then the motor controller (Sabertooth) outputs the final control signals to both right and left motors. The USB connection identified as number 8 sends the control signals from the motherboard to the step motor. These signals go through the R256 motor driver before reaching the servo motor

which controls the range of inspection of the LIDAR. We discovered that the parallax servo controller is off and therefore is not being used, since the stepper motor is receiving signals directly from the motherboard from connection number 6. Finally, connection number 7 receives data from the LIDAR sensor. Also, the motherboard is connected to an external hard drive and to the power supply. Additionally, we found out that the battery voltage level is adjusted by two DC-DC power converters (one outputs 5 V and the other 12 V) before powering the main circuits. The motherboard is powered by a connector that has the voltages levels adjusted by another power converter which has 12 V as input.

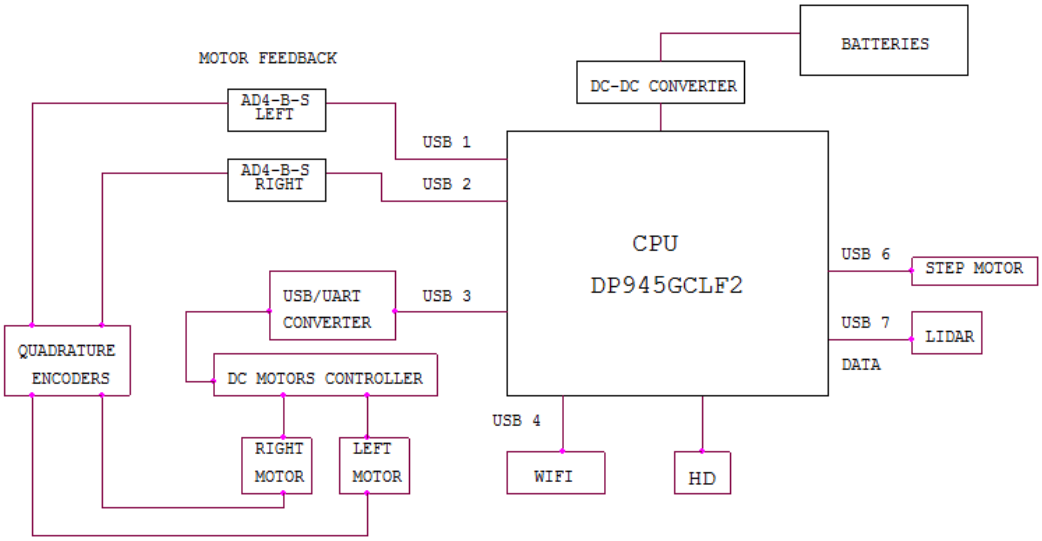


Figure 04 –System Diagram

The motherboard has 8 USB connections in total, but 2 are not being used permanently.. We can use to connect a flash drive to the system for example. They are identified by numbers from 1 to 8. The identification is a little messy for us because the same USB connection has two different numbers (for example, 1 and 3) that we couldn't understand why they were different and what was each meaning.

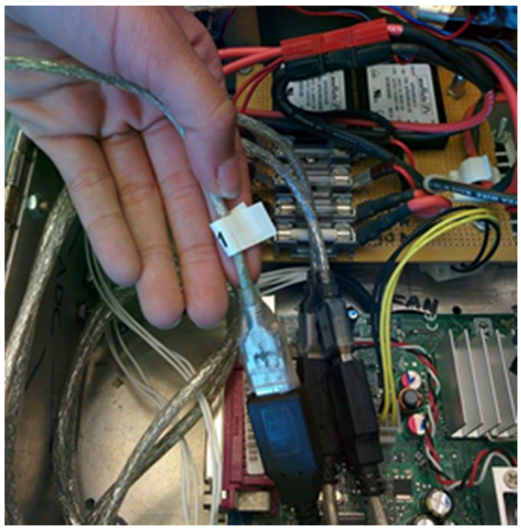


Figure 05 –Wrong tags in the circuit

Following our observations, we identified each USB in the motherboard:.

i) USB from wires in the -motherboard:

- USB 1 from cable “Right” (connected to VGA connectors without identification): Goes to AD4-B-S Left
- USB 2 from cable without identification connected to VGA connectors identified as 1E: Goes to AD4-B-S Right
- USB 3 from cable “1” – Goes to the CP2102 (USB/UART converter), which goes to SABERTOOTH 2X25 (motor controller)
- USB 4(in fact, without identification) from cable without identification – Goes to Wi-Fi Adapter

ii) USB from connectors in the motherboard:

- USB 6 – Connection from LIDAR Stepper (Stepper Motor of LIDAR, white connector)
- USB 7 – Connection from LIDAR (green connector) Receive data from LIDAR

We checked our conclusions with the USB Layout illustrated in the gray box’s lid and both were compatible.

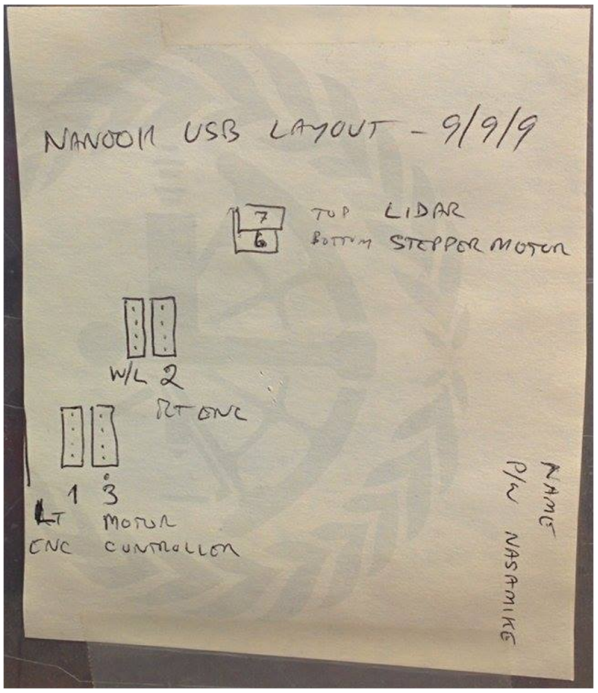


Figure 06 – USB Layout

As it is confusing to have a cable identified as "**Right**" connected to the **Left** motor through the AD4-B-S and a cable identified as "1" connected to a USB "3" while there is a USB"1", we put an additional identification to facilitate the understanding of the system's connections:

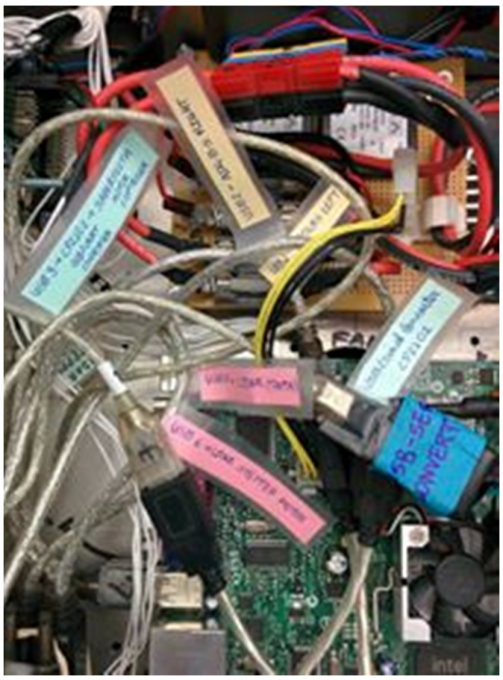


Figure 07 –New tags for the main gray box

The Sabertooth receives two wires (a green and a blue one) coming from the CP2102 (a USB/UART converter) and also the power and ground wires coming from the Converter Board, as will be explained. The sabertooth's output is composed by two pairs of red and black wires that goes to a connector in the border of the main gray box (on the right corner) and goes to the motors. Each pair goes to a motor (left or right).

Under the main circuit box, we could identify power (ground (black cable) and VCC (red cable)) for the left and right motor (for each motor, there are two cables) and a set of five small wires (yellow, blue, green, red, and black) for each motor. The power wires (red and blue thick cables) goes to the Converter System.

Those two set of colorful small cables comes from Quadrature Encoders attached to the DC motors and they link the DC motors and its respective AD4-B-S, passing through the MS3106A28-21S connector which is placed on the right corner of the gray box. The quadrature encoder captures the phase position of the DC motors in Channel A and B, and then that information is processed into the AD4-B-S, in which is converted into serial communication (RS232). At the chart below, it is possible see the connections.

By inspection of the hardware, we found out that there are encoder in the DC motors. We concluded that after we found the connections coming from the motors' black box should come from a encoder. The motor's black box is connected to the AD4-B-S, a "Quadrature to RS232 converter", before reaching the motherboard. Looking at the AD4-B-S' datasheet we discovered its input is a quadrature encoder.

Table 05 – Quadrature Encoder

AD4-B-S input (Wire number/Color)	Quadrature Encoder Outputs
1/Black	Ground
2/Green	Index
3/Yellow	Channel A
4/Red	+5VDC power
5/Blue	Channel B

Each set of small wires goes to a different AD4-B-S. One set goes to the AD4-B-S corresponding to the left motor (AD4-B-S-L) and the other one goes to the AD4-B-S corresponding to the right motor (AD4-B-S-R), tagged respectively as L and R.

In the Converter Board, the voltage level from the battery is converted to 12V and 5V in the WPN20R24S12 and WPN20R24S05, respectively. These components are obsolete now. After the converter, the VCC goes to a fuse. There is one for the 5V and one for the 12V. The 5V power is identified in the board with the letter "J" and goes to the yellow connector marked with "5". The 12V power is identified in the board with the letter "M" and

goes to the blue connector marked with “12”. There are also two fuses (one of 25A and one of 30A) that receive power.

From the yellow connector, which outputs 5 V, there are two wires coming out, one black and one red (both are 5V, even the black one). The black one gets together with another black wire (which this time is a ground) and goes to a connector entitled as “for hub”. The choice of wire colors was not the best as it is confusing and falls off the pattern commonly used by electrical engineers. The red wire (5V) goes together with another black one (ground) to the “Parallax”. From the “Parallax” there are 6 wires coming out, but end up not being used, since the control signals for the lidar step motor are coming from the motherboard (USB 6). This led to our conclusion that for this robot version the “Parallax” is not being used.

Two blue wires leave from the blue connector. They go to the AD4-B-S. A blue wire and a black wire (coming from the black connector - ground) go to the AD4-B-S-L (AD4-B-S corresponding to the left motor), and another pair of these wires goes to the ADB-4-B-S-R (AD4-B-S corresponding to the right motor).

The power that passes through the 25A fuse goes to a red connector and then is connected to the Sabertooth . The power that passes through the 30A fuse goes to a red connector and two red wires leave from it (one thick and one thin). The thick wire is connected to another connector in the board of the gray box. From this connector, the one red wire becomes three red wires. Two of them are connected to the LIDAR, the other one is connected to the lateral gray box. The thin wire leaving the connector after the 30A fuse goes to a connector and becomes a white wire connected to the DC-DC ATX power supply. This power supply is connected to the motherboard by the ATX 20-pin connector. Four wires leave from this converter: a yellow wire (+12V), a red wire (+5V), and two black wires (Ground). These wires and some wires coming from the external drive are connected to the motherboard through a female-male connection (4-pin Molex).

There is a SATA Data connection from the external drive to the motherboard (SATA0). Also, there is a SATA Power connection with the connector at the main power in the motherboard. This plug is also connected to the 2x2 power in the motherboard - only the two GND and the +12VDC (yellow wire). The model of the SSD flash storage drive is SNV125-S2/64GB.

The outside turn-on button (white button) is also connected to the motherboard.

In the lateral gray box, there’s a circuit to control the LIDAR. In this box, there is a R256 microstepping controller unit with RS232 communication kit. It is used to control the LIDAR stepper motor. First of all the CPU sends signals from the USB #6, which are converted to RS232 communication. These signals go into the box, where they pass through the RS232 to RS485 converter (which is marked in the pink box in the figure). The RS232-RS485 card is plugged to the CPU using a standard male to female DB-9 cable. The RS232-RS485 card outputs an “A” phase and a “B” phase. + 12 VDC power is supplied to both the converter card and R256.

The signals outputted in RS485 communication standard go through the R256 step controller. Then R256 outputs the final 4 signals that go to the step motor. The Red wire is A, Blue is A Bar, Green is B, and Black is B Bar.

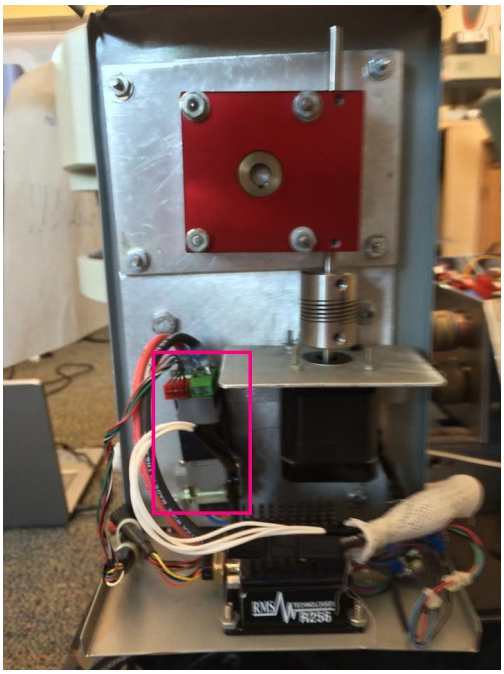


Figure 08 –Lateral box and the R256 highlighted

VIII - DETAILED HARDWARE INSPECTION

On July, 17th we finished the entire hardware inspection. The additional details we discovered are regarding the power system and encoders from the motor. This part was not accessible previously. We had to uncouple some parts of the robot (the LIDAR system, the CPU box, and also the insulation next to the batteries) to access the black box that contains batteries and motors.

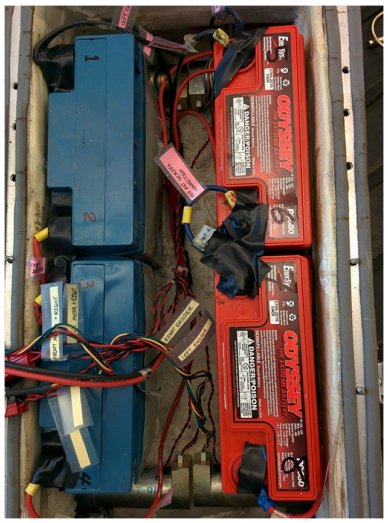


Figure 09 –Circuit under the main gray box

We found four motors, four encoders, and four batteries. Also, we were able to track the wires from the battery and from the motors. We used a mirror to be able to see the switch

connections that were still not visible for us. Two set of colorful wires (black, blue, red, yellow, and green) go to the back encoders (one for each side). The two encoders in the front of the robot are not being used for now. All the motors receive power and ground. Each set of batteries (two in each side) is connected in parallel, while each set is internally connected in series to provide 24V each. The power (VDC and GND) goes to the main gray box as a red and a black wire. These wires go to the power board, where there's two converters: from 24V to 5V and from 24V to 12V. Two red wires are connected to the board in a way that is also connected to the entrance voltage. The output of the board are the two 24V power wires, one 12V power wire, and one 5V power wire. The 12V is connected to the blue connector, the 5V is connected to the yellow connector, and the 24V is connected to the red connectors. The figure showing the back side of the power board is showed below.

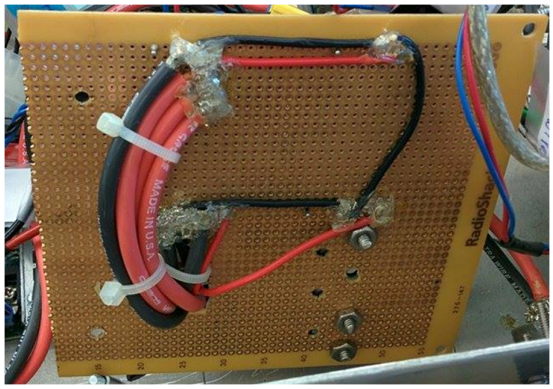


Figure 10 –Power System Board

The batteries are connected to a switch, so the system is able to alternate between the charger connection and the robot's system connection.

One of the connectors in the boundary of the main gray box was identified as a bendix LJTPQ00RT-15-35S.

By the total hardware inspection, we were able to track the entire hardware of the robot system. As a result, we develop a schematic using the AutoCad software to documentate the hardware system. A quick view of this schematic is illustrated in the figure below. The schematic file is attached to this document.

IX) HARDWARE ORGANIZATION

To help the hardware identification in the future, we put tags in the wires under the main box. We have done this in a way that is possible to track all the wires just removing the gray box (removing the insulation and the LIDAR system is not necessary anymore).

Figure 11 – Tags for the circuit under the main gray box

Tags were also put in the main gray box to facilitate the comprehension. This made easy to analyze the circuit by a simple look into the tags. Different colors separate different main circuits (for example, DC motors system has green tags, LIDAR system has pink tags).

X) SCHEMATICS

We drew the circuit's schematics in AutoCad software describing the circuits: one general schematics for the entire system and schematics for subsystems.

The general schematics is ilustrated below, but all of them are attached in the hardware report.

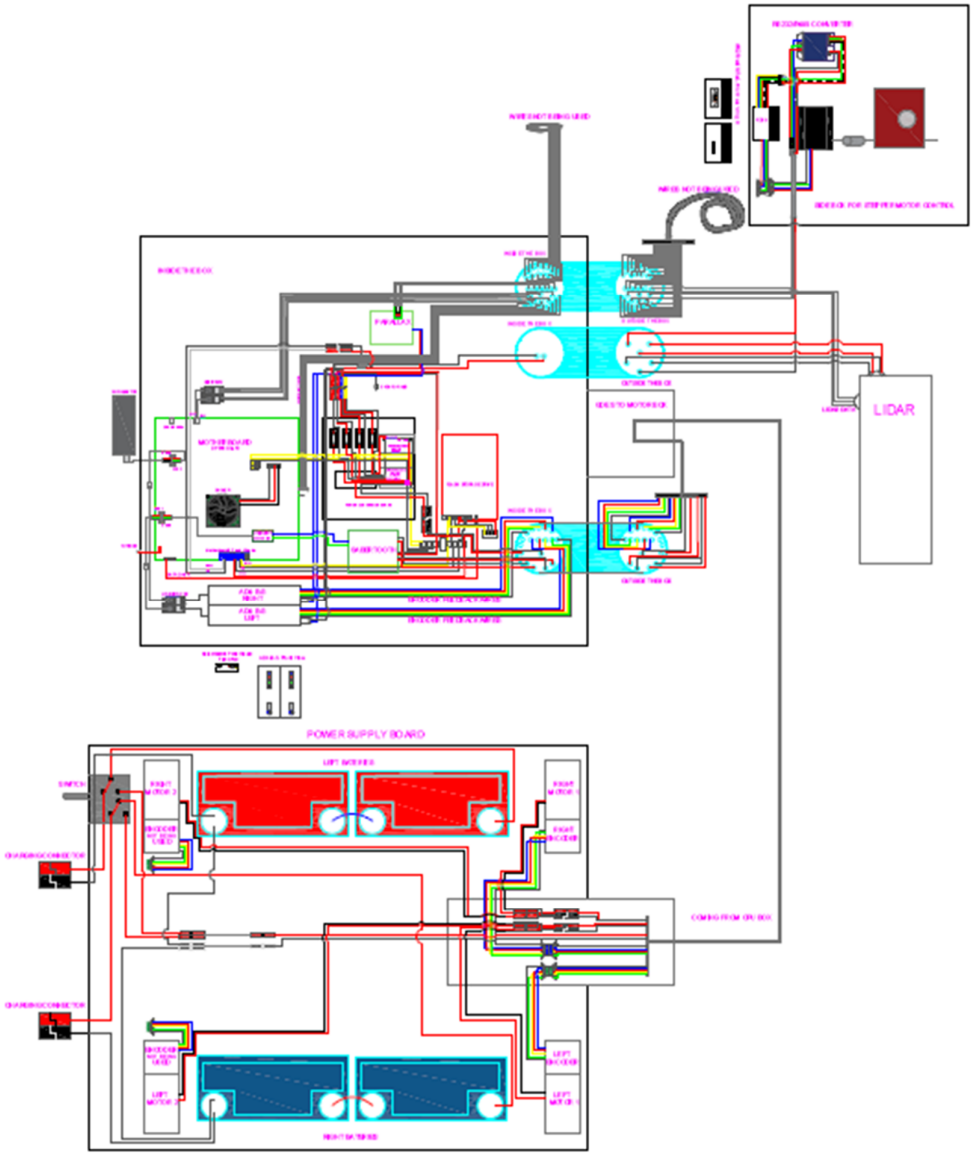


Figure 12 – General Schematics

XI) HARDWARE REPORT

We have documented the entire hardware system as a report, so anyone could be able to understand the circuits and replicate it. Also, with the full description of the system, proposing improvements is easier.

The Hardware Report is attached to this document as a module.

XII) COMPUTER SOFTWARE ANALYSIS

We analyzed the entire project of the computer software. It was written in C# using Visual Studios 2008. Our main dificulties in the begining was to learn the proramming language used to be able to understand the code. We created some useful C# help resource while we were reading the code to help all the members understanding the files. All the files were read, commented to generate documentation, and explained in a report.

As C# is an object oriented language, we explained the main structure of the code by highlighting some classes, objects and methods.

The graphical interface created was also explained with details, which can help to manipulate the robot when necessary. All the buttons and windows were defined, illustrated and clarified, so all the functions of the robot could be accessible for anyone.

XIII) EMBEDDED SOFTWARE ANALYSIS

The embedded software was analyzed following the same model as the computer software. However, in this case, the programming language used was C++. We developed a guide help in this language because it was also unknown for us. We explained the code according to the headers files (.hpp and .h), source files (.cpp), project files (.vcproj) and solution files. Then we analyzed the code understanding its functionalities.

XIV) GENERATION OF HTML DOCUMENTATION

Both codes (embedded and computer software) were commented to generate a html documentation. We used the Doxygen software, since it is open and it is a standard tool for generating documentation.

Accessing the referred documentation is possible using the link to the Nanook folder or accessing the github folder (recommended) at <https://github.com/anareboucas/nanook.git>.

XV) IMPROVEMENTS

Analyzing the code was harder because many parts of the code were unused or not functional. We have done some improvements to correct this:

i) Eliminating unused lines;

The code is now easier to understand. Although some commented lines were important because they contained some unfinished work, we have maintained the original version of the software (just including the comments for doxygen documentation). Our modifications were done in a new software version. As we put the software on github, it will be easier to track the exact changes in each version of the code from now on.

ii) Making the offline modes functional:

This made the software work properly independent of the function enabled. The offline mode (navigation planning, sphere recognition, and range image viewing) were not working, so the code was modified to correct their functionalities.

iii) Adding a help button:

A help button was added in the windows, so anyone with some doubt about how to use the software can have the questions clarified. This will prevent that some robot's functionality does not work because of the user's fault.

iv) Changing the connection dialog interface:

This modification was more aesthetics than a problem solver. We added the nanook image to the connection dialog to personalize it.

XVI) SOFTWARE REPORT

We have documented the entire software (from the embedded circuit and from the computer) as a report, so anyone could be able to understand the code and replicate it. Also, with the full description of the software, proposing improvements is easier.

The Software Report is attached to this document as a module.

XVII) GITHUB

All the documentation produced during this work, including the code, schematics, and reports were added to a repository in github. Therefore, the information about nanook is now available for everybody and allows an easy track of the changes each time they happen.

XVIII) STARTUP GLITCH

Nanook has a turn on glitch problem unsolved. Some initial suggestions about this problem were that the turn-on glitch might be from the last data sent to the controller before the robot is powered down, or from a start-up transient. Maybe some loop could be unfinished or the robot was saving cache memory. The solution if this is the case would be:

i) Add a section in the initialization code for the robot that sends zeros to the controller at start-up.

ii) Add a power-down mode, where we again send zeros to the controller, and do an orderly shut-down.

Because of these possibilities, we look into the code (both embedded and the computer software) trying to discover the specific parts where we should add the proposed routines.

While doing this, we realized that the robot starts with a pattern to identify it is working properly. The robot moves forward, backward, turn to left, turn to the right, and move the LIDAR as a indicator to the user that everything is working properly. These movements happen when we turn on the robot even if we haven't connected it to the computer. Therefore, any change in the computer software wouldn't correct the glitch problem, only the embedded software could do this.

Analyzing further, we concluded that the embedded software also was not responsible or able to solve the glitch. By looking into the hardware schematics and making some measurements, we discovered that when we turn on the robot, power from the batteries goes directly to the CPU and to the Sabertooth. This means that the Sabertooth receives power before it receives the input signal. Another factor that we found related to the glitch problem is the batteries. If the batteries are not good, they can provide voltage, but not enough current to make the motors work properly. There's a possibility that temperature affects even more bad batteries in relation to this point. If the robot is using a bad battery, as soon as we turn on it, the batteries are going to be still cold and will not provide enough current.

Our analysis and proposal solution is based on solve the problem that the sabertooth receives power before the input signals. We are assuming the robot will have good batteries, since this is a minimun requirement.

The detailed description of the startup glitch analysis is presented in the next section.

7 - STARTUP GLITCH ANALYSIS

Nanook robot has a problem that makes its utilization a little risky. Sometimes, after initialization, an error makes the motors moves in a random way. A combination of causes for this problem was identified, leading to proposed solutions and guidances for better use.

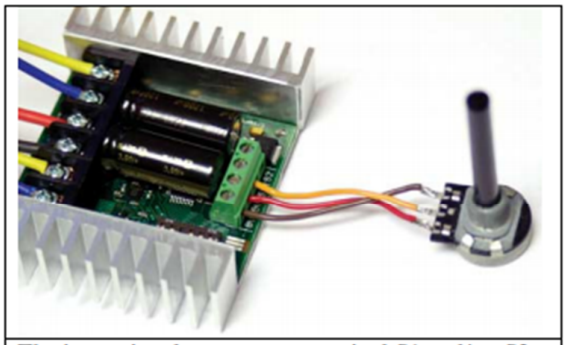
7.1. METHODOLOGY

After analysis of software and hardware, we discovered that the problem happens in the latter. The combination of two factors is most likely the reason for start glitches. What happens is that when someone turns on the power switch, the battery provides power to the motors' drivers (DC driver and stepper motor driver). The motherboard receives power too. However, someone needs to press a second button in order to initialize the CPU. This window of time between turning on both the power switch and the CPU button causes the drivers to receive floating signals and to send random signals to the motors. In addition, when the robot is turned on the first time after a long period, the batteries can still be "cold" and not provide enough power to correctly drive the motors. The proposed solution is to connect a switch that turns on the motors (DC motors and stepper motor) only after one initializes the CPU, so that it is already sending the right control signals. In addition, it is recommended to change all the batteries for 4 new batteries that are more efficient and do not take too long to heat up and provide the necessary power for the driver.

When looking further to see how we could implement our solution to the Sabertooth 2x25, we found out in its datasheet that the motors could start running unexpectedly if the power is connected in the Sabertooth before the input signals from the CPU. The figure below shows the part of the datasheet that explains this situation. Even though nanook's Sabertooth is in the *Standard Simplified Serial Mode* and not on *Analog Mode* it is very likely that this is one of the problems causing the initial glitch.

Signal Input Terminals S1 and S2

The input signals that control the Sabertooth are connected to terminals S1 and S2. If you are running in analog mode, it is important to have both the signal connected before applying power to the device. Otherwise, the motors may start unexpectedly.



The input signals connect to terminal S1 and/or S2

Figure 13 - A piece of information about Sabertooth 2x25 from its datasheet [6]

7.2 – PROPOSED SOLUTION

7.2.1 – SWITCH ADDITION

A first proposed implementation to solve the glitch problem is to add a switch the would enable/disable the motors. This is a fast and safe solution. However, it adds one more piece of hardware and more work for the user. It is important to highlight that this solution only works if the user properly follow these instructions in this exactly order:

I) Turn on the main power switch in the black box.

II) Turn on the bottom that initializes the CPU.

III) Turn on the switch that enables power for the motors.

It is important to remind that the step number three has to be always the last one made in order to this solution to be useful. A schematic below displays this solution.

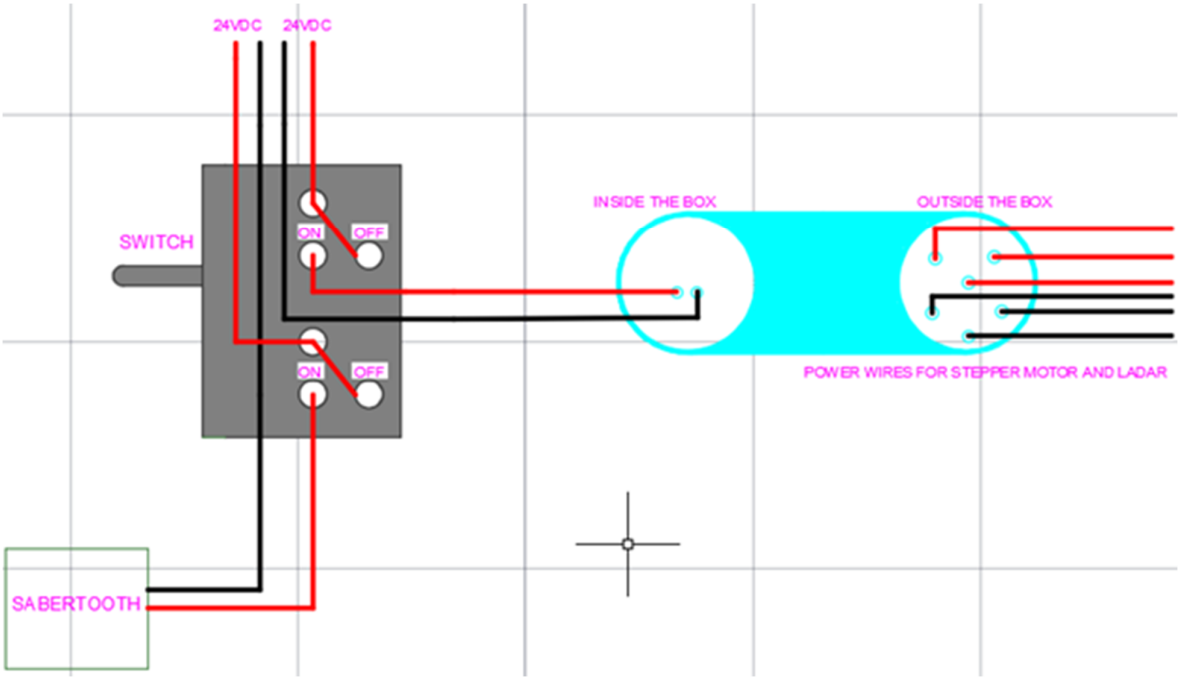


Figure 14 – Switch Solution Schematics

7.2.2. CPU CONTROLLED RELAY

The second proposal is to add a CPU controlled relay. For this, one needs to create an interface between the relay and the motherboard, in a way that the drivers for the motors only receive power after the motherboard initializes and is sending the right control signals. The figure below shows a possible implementation, using an USB controlled relay board.

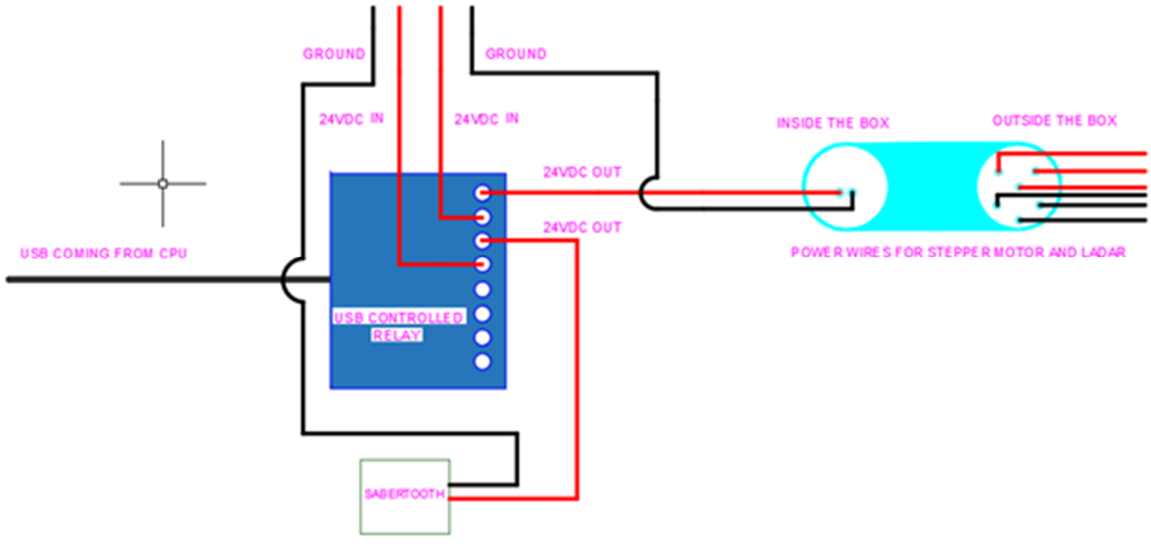


Figure 15 – Relay Solution Schematics

8. CONCLUSION

The Nanook reverse engineering project was a big challenge because we have to start from nowhere. However, a good methodology of work and a lot of efforts put together to solve the problem in a tiny deadline were enough. We have learnt to work in teams, worked in a fast-pass environment, taken difficulties decisions, learnt new skills and programming language, documented hardware and software systems as well as the work done, and proposed new improvements.

The main challenge was to stay in the frontier between being safe enough to not break the robot but invasive enough to make us discovery the entire robot's system. Our deadline was approximately one month to do all those tasks and we judge our work completed and successful.

Any person interested can now start to work with Nanook or to replicate it with minimum efforts.

9. REFERENCES

[1] Available at <http://www.theoldrobots.com/Pat_Sc.html>
[2] Available at <http://www.mouser.com/ds/2/281/murata_tdc_wpn20r-547589.pdf>
[3] Available at <http://www.mouser.com/ds/2/281/murata_tdc_wpn20r-547589.pdf>
[4] Available at <https://cdn.sparkfun.com/datasheets/BreakoutBoards/CP2102_v1.2.pdf>
[5] Available at <http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx>
[6] Available at <https://www.dimensionengineering.com/datasheets/Sabertooth2x25.pdf>
[7] Available at <http://www.digchip.com/datasheets/parts/datasheet/502/AD4-B-pdf.php>
[8] Available at <http://www.linengineering.com/drivers-Controllers/PDF/R256_Manual1.08.pdf>
[9] Available at <http://downloadmirror.intel.com/16960/eng/D945GCLF2_TechProdSpec02.pdf>
[10] Available at <http://www.kingston.com/datasheets/SNV125-S2_us.pdf>
[11] Available at <http://sicktoolbox.sourceforge.net/docs/sick-lms-technical-description.pdf>

ATTACHMENT I

Nanook Previous Information Available

Nanook System Description
6-29-15

Introduction

Nanook is a tracked robot platform, mounting a 2-d scanning lidar. It was developed at NASA-GSFC by a team of students under the lead of Mike Comberiate ("NASA-Mike"). The robot unit has been in both the Arctic and Antarctic. It is one of 3 similar tracked platforms. One of those is at Capitol Technology University.

Physical characteristics

Electronics

cpu - IBM-pc architecture motherboard, Intel model D945GCLF2. 32-bit dual core ATOM processor.

Intel 945GC chipset, GMA950 graphics

ATX power connector, using dc/dc, +12 to +5, +12.

memory – 240 pin DIMM SDRAM DDR2, 2 G max, 1.8 v

expansion – single ide and pci slot.

I/O expansion, 1000 Mbps lan RJ-45, dual SATA, Realtech Audio, line out/in, mic in.
PS/2 keyboard and mouse, serial and parallel port,
vga and s-video
8 usb ports.

Secondary storage, one pata, 2 sata.

Real time clock with CR-2032 battery

Motor Control

Control of the motors uses a 2-channel motor driver, a Sabretooth. It is provided with a direction bit and pwm information for the cpu. It is powered from the onboard batteries.

It communicates with the computer in the simplified serial mode. The output from the computer is via usb, which is level-shifted to ttl-level RS-232. There is a 2-wire interface to the Sabretooth module. The baud rate is selected by a dips switch on the Sabretooth. Commands are single byte.
Because Sabertooth controls two motors with one 8 byte character, when operating in Simplified Serial mode, each motor has 7 bits of resolution. Sending a character between 1 and 127 will control motor 1. 1 is full reverse, 64 is stop and 127 is full forward. Sending a character between 128 and 255 will control motor 2. 128 is full reverse, 192 is stop and 255 is full forward. Character 0 (hex 0x00) is a special case. Simplified serial is operated in option 1 (see user manual)

Motors

tbd, dc gearhead motors

Batteries
dual rechargeable lead acid, gell-cell, 12 volt, tbd A-H

Lidar

SICK LMS221-30206, Laser Measurement System, Outdoor Version, with fog correction, high range and 180 degree scanning angle, IP 67 (grey color)

LIDAR description

Features	
Field of application:	Outdoor
Version:	Mid Range
Light source:	Infrared (905 nm)
MTBF:	50,000 h
Laser class:	1 (EN/IEC 60825-1), eye-safe
Field of view:	180 °
Scanning frequency:	75 Hz
Angular resolution:	0.25 ° 0.5 ° 1 °
Heating:	Yes
Operating range:	0 m ... 80 m
Max. range with 10 % reflectivity:	30 m
Fog correction:	Yes

Performance	
Response time:	≥ 13 ms
Detectable object shape:	Almost any
Systematic error 1):	± 35 mm
Statistical error 2):	± 10 mm
Integrated application:	Field evaluation
Number of field sets:	2 field triples (6 fields)
Simultaneous processing cases:	1 (3 fields)

1) 2) Typical value; actual value depends on environmental conditions.

Interfaces	
Serial (RS-232, RS-422):	
Function (Serial (RS-232, RS-422)):	Host, AUX
Data transmission rate (Serial (RS-232, RS-422)):	RS-232: 9.6 / 19.2 kBd and RS-422: 9.6 / 19.2/ 38.4/ 500 kBd

Switching inputs:	1
Switching outputs:	3
Optical indicators:	0

Mechanics/electronics	
Electrical connection:	1 16-pin plug
Operating voltage:	≤ 24 V DC
Power consumption:	30 W, + 140 W heating
Housing:	Base plate aluminum diecast Covering PU
Housing color:	Gray (RAL 7032)
Enclosure rating:	IP 67
Protection class:	II (VDE 0106/IEC 1010-1) 1)
Weight:	9 kg
Dimensions:	241 mm x 351 mm x 265 mm
1) Insulated	
Ambient data	
Object remission:	1.8 % ... 10,000 %
Electromagnetic compatibility (EMC):	EN 61000-6-2 / EN 61000-6-3 / A11 (2004-07)
Vibration resistance:	IEC 68
Shock resistance:	IEC 68
Ambient operating temperature:	-30 °C ... +50 °C
Storage temperature:	-30 °C ... +70 °C

General notes	
Note on use:	Not suitable for personnel protection
Life cycle phase:	Phased out