ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

School of Engineering

Semester project: STI robot competition

Mr.Sticky

Authors
Héloïse Boross (MT)
Claudio Fritsche (GM)
Roman Oechslin (MT)

Professor
Auke Ijspeert

Assistants
Alessandro Crespi
Petr Listov

June 16, 2017

Abstract

Mr.Sticky is a robot that has been designed and built in the framework of the STI interdisciplinary robotics competition at EPFL. The working principle of this bottle collecting robot is a sticky conveyor belt that turns around the whole chassis which is used to grab, store and release the bottles.
The data acquisition and its processing is done on board. The bottle detection uses a classifier trained by a Haar cascade using the OpenCV library. The localization of the robot is calculated through a Kalman filter which uses data from encoders and a self-built triangulation system. The latter detects four LED beacons using a camera with a parabolic lens resulting in a panoramic 360 degree view.
This project allowed us to gain experience in time and budget management as well as practical experience such as mechanical design, PCB manufacturing and software programming.

Contents

List of Figures

List of Tables

1 Introduction

In this project the goal was to develop and build an autonomous robot. The necessary tasks of this robot are specified in the rule book. The main goal is to navigate in a given arena and to collect PET bottles and return them to a designated recycling area. The work has been split up in different parts, such as electrical and mechanical design and they have been tackled separately. In order to have a fully working robot in the end, it was necessary to build prototypes of key parts and test and optimize them accordingly. Another key factor was to come up with solutions for arising problems even with a time constraint or a limiting budget.

1.1 The Team

The team consists of three students in the master degree level. Claudio is from the mechanical engineering department, more precisely in automation and control with a minor in biomedical technologies. Héloïse is from the microengineering department and follows a minor in space technologies. Roman is also in Microengineering with a specialization in robotics. The team has its assigned team coach Petr Listov who is in his PhD in control system engineering. The principal assistant responsible for the whole project and all groups is Alessandro Crespi.

1.2 Specifications of the competition

First, it has to be made clear what the robot must do. The goals and constraints are specified in the rule book and the key points are listed below.

Goals

- Grab bottles and release them in the recycling area
- Avoid obstacles encountered within the arena
- Must be completely autonomous
- Computation must be done on board
- Robot must carry its power source

Constraints

- Virtual and real budget of 2000 and 1000 CHF respectively
- Maximum clearance in arena is 50cm
- Allowed time in arena 10min
- Robot must fit in $1m^3$

The arena The arena itself is structured similarly to the following schematic.

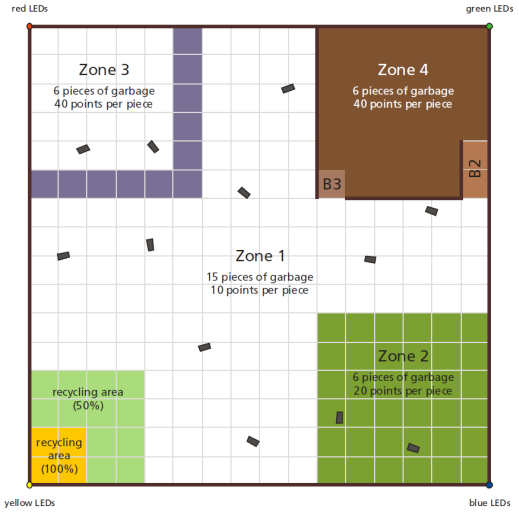


Figure 1: Layout of the arena from the rule book

There are four different zones with different characteristic. Zone 1 has a normal floor and is easy to reach, the bottles are not so worthy. The second zone is a field of grass or artificial grass like carpet. It is a bit harder to reach and could be a problem for simple wheels. The bottles within this zone count twice as much as the bottles in zone 1. The third zone is surrounded by a barrier of stones. To get to this arena is hard and the bottles count more points. And finally, the last zone is situated on a stage and surrounded by a wall. To get to this place, the robot needs to either climb the stairs, or use the ramp. The bottles within this area count as much as the bottles behind the stone barrier.
Throughout the arena there are some obstacles at initially unknown places. These obstacles are two bricks stacked on top of each other and they should be detected and avoided. Slight contact is allowed, but no interaction shall happen.

Reward For every bottle that ends up in the yellow zone of the recycling area, the full points are given as reward. In the green area only 50% is rewarded. If the time runs up and the robot still has some bottles stored, they count 25%.

2 Functional Analysis

With these goals and constraints, a thorough functional analysis can be done on the robot and the tasks to accomplish. For this the main functions are listed below.

Locomotion The robot shall go to different places. Depending on what has been chosen as locomotion principle the robot may not reach all the zones. This choice influences mainly the mechanical design and the speed and precision with which it is moving.

Grabbing To be able to score some points the robot shall have one or several grabbing techniques. This means taking a bottle and put it somewhere for storage. Here it is important to take into account that the position and orientation of the bottle might not be fully known and that the precision is limited.

Storing The robot shall be time efficient and therefore it shall have a storing mechanism or a solution to be able to go from bottle to bottle instead of returning to the recycling area after every single bottle. This storing mechanism defines the general shape and size of the robot and determines the cadence at which the robot can bring the bottles back. The robot also needs to detect when the storage is full such that it knows when to go back.

Releasing Once the robot has a bottle and has returned to the recycling area, it shall release the bottles in an area of $1m^2$ or in the worst case $4m^2$.

Bottle detection The bottles must be discriminated from the obstacles in order to be able to catch them. Therefore it is necessary to detect either one of them or even both.

Obstacle detecting The obstacles need to be detected and avoided by the robot.

Navigating A navigation and localization within the arena might be useful for time efficiency and path planning. When an obstacle is detected it has to be avoided by for example driving around it, even though the bottle might be just behind this obstacle.

Computing and Controlling All the acquired data shall be processed and interpreted such that the robot can react to its environment. The motors for the locomotion have to be controlled by at least one processing unit.

User interfacing The more user friendly the robot is, the easier it will be to test and debug it. It is not critical for the performance of the robot, but can save a lot of time.

Powering The different electrical components need to be powered and if operating at different voltages, the power buses need to be converted and regulated.

Sensing The robot shall be capable of having some sensory input. This can be for navigation as well as for bottle detection or obstacles. The sensors can be proprioceptive or exteroceptive.

3 Brainstorming

With all these functions, the main guidelines and tasks have been defined. The next step is to find various solutions for each function. In the brainstorming phase the goal was to be as creative and original as possible. Bad ideas shall not be judged to impose no barriers on thoughts and ideas.
The complete set of all the solutions that have been found can be seen in table 1.

Locomotion	**Capture**	**Storage / release**
Wheels	Sticky material	Wheelbarrow
Wheel with pivots	Net	Bag
Crawler (Caterpillar) (all versions)	Grab by sitting on it	Container with moving bottom
Tilting mechanism (reaction wheels)	Articulated gripper	Conveyor (reverse)
Articulated Leg	Vaccuum	Blow
Whegs	Blower	Throw
Jumper	Turning structure with Archimedes	Kick
Spherical Structure around robot	Chameleon tongue	Archimedes screw
Snakelike movement	Lasso	Limit number of bottles (f.ex. 6)
Anchor	Sweep / Broom	V-shape with lifting barrier
Leg with anchor	Shovel	1-by-1
Reaction wheel	Conveyor belts	Ropes
Wheels on Wheels (Epi.q)	Tunnel on Wheels	Double bottom structure
Wheel with anchor for obstacles	Pushing (f.ex. In V-shape)	Six Pack
Transforming wheels	Tentacles	Home Tower Strategy
Spider like movement	Hooks (f.ex. On conveyor belt)	
Ropes	Liquid Nitrogen	
Rails	Kicking	
	Moving barrier	
	Movable arm with vacuum/blow grip	

Table 1: All feasible ideas for the main tasks

After having analyzed the advantages and drawbacks of every possible solution, a

matrix with different weighting factors can be built (see figure 46 in appendices for the full decision matrix). These weighting factors include among others the feasibility, originality, reusability, expected performance and difficulty of control. The best solutions were explored and implemented in different combinations.
Out of this, four main solutions have aroused. Since a lot of time has been put into evaluating the solutions, they shall quickly be explained below.

3.1 Plucking robot

This robot uses a rectangular frame that can be lowered down to the bottles. On this frame, there are some bristles that point inwards and the bottles is plucked by the force of the frame towards the ground. The big advantage of this robot is that the orientation of the bottle is not needed and that the design is very innovative. The disadvantage is the complexity and the risk of mechanical malfunction.

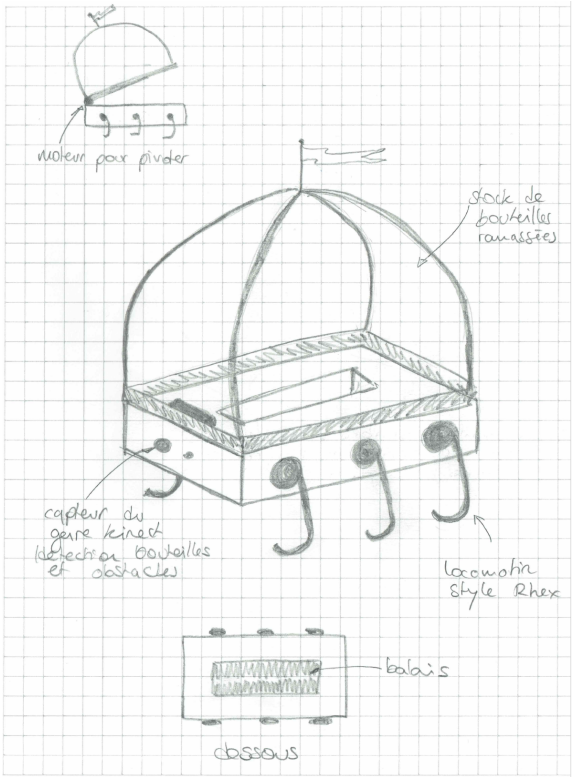


Figure 2: Rough schematic of the plucking robot design

3.2 Tunnel on Tracks

This robot is relatively simple and inspired by the vehicles cleaning the roads. It is a big tunnel on wheels and automatically sweeps all bottles on its way into its body. The storage and capture technique is relatively easy. For storing a lot of bottles the robot quickly becomes very big.

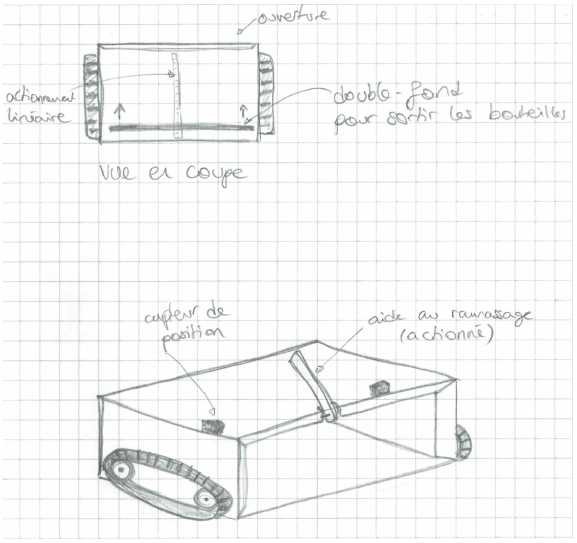


Figure 3: Rough schematic of the tunnel on tracks robot design

3.3 Snow plow

With a robot similar to a snow plow, one can drive blindly around the arena and collect everything that is no obstacle. This is mechanically the simplest solution and did not seem challenging. The storage capacity is very limited with this solution. It has a very low risk factor and could always be used as backup plan.

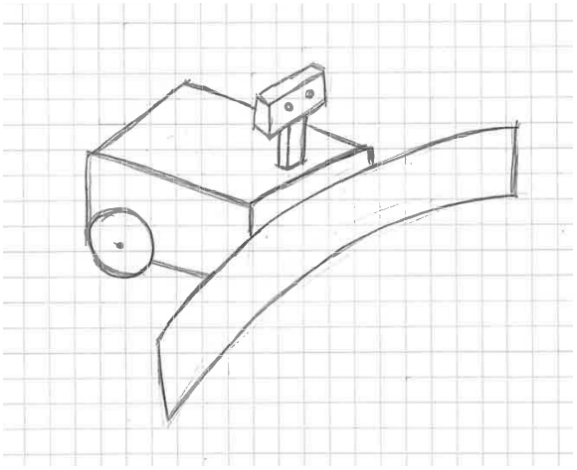


Figure 4: Rough schematic of the snow plow robot design

3.4 Sticky conveyor belt

This design implements a sticky conveyor belt that turns around the robot and stores the bottle on the robot itself. This solution is mechanically more challenging and there are some risks linked with the conveyor belt and the sticky material. The innovation aspect of this design is very interesting. The advantage of this design is that the release mechanism can be implemented with no additional requirements. It is sufficient to turn the belt further into the same direction to release all bottles.

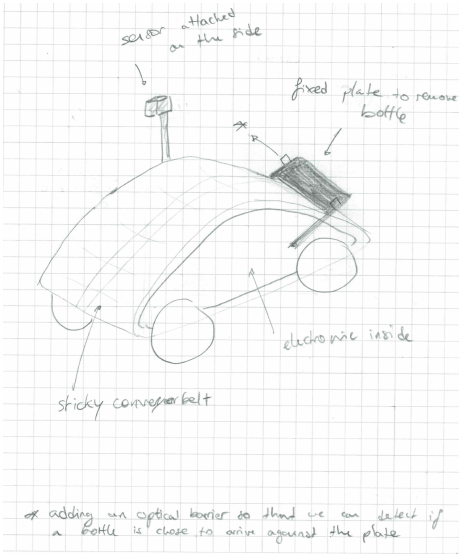


Figure 5: Rough schematic of the sticky robot design

3.5 Final design

Looking at the advantages and drawbacks of every design and after carefully evaluating and analyzing the risks, the team has made the decision to go with the design of the conveyor belt. Thus Mr.Sticky was born. Even though it is mechanically harder to build, the challenge shall be accepted. The first proposed mechanical design of a such principle can be seen in figure 6.

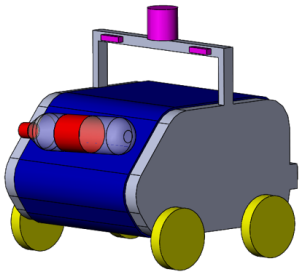


Figure 6: First mechanical design

After the first design review with the expert for the mechanical domain, Norbert Crot, the design has been adapted to ease manufacturing and avoid foreseeable assembly problems.

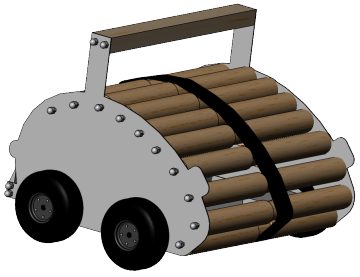


Figure 7: Advanced mechanical design

3.6 Strategy Options

With this grabbing method, there were two possibilities for the release mechanism. Either a system that exploits the mechanical features of the robot with the sticky conveyor belt, or something that needs to be implemented alongside it. The latter would have made the robot much more complicated, both from a mechanical but

also an electrical point of view. Therefore, it was obvious to go without an additional release mechanism, such that the high unnecessary risk could be avoided.
Furthermore, using the conveyor belt as release mechanism implied a bottle storage by means of the belt as well. However, this left only so much space for the electrical components on the inside of the conveyor belt. Having a robot with its electronics on the outside, or a constellation of several robots has been refused.
For the locomotion principle there were two choices left that go well along with the sticky belt. These were the caterpillars or common wheels. The wheels seemed to be appropriate for the different zones and there was no need to implement the caterpillars. The group has decided to opt for all four zones, to show the robots all terrain ability. Though at the competition itself, the time constraint would not allow to visit all zones. Another aim was to guarantee a certain precision and keep the costs low. These aspects were the main reasons for choosing wheels in the end.
Different chassis and robot base plates have been analyzed but considered too expensive or not suitable for the necessary tasks. Therefore a four-wheel, all-terrain robot has been designed that could be manufactured at EPFL and standardized off-the-shelf parts could be bought.
The bottle recognition shall be done with a camera using a Haar cascade in openCV. The camera data could later be used for other important tasks such as navigation or obstacle detection. An additional camera can be used for navigation with the four LED-beacons that are located on the corners of the arena. The obstacle detection could be done with a range of infrared sensors at the hight of the obstacles. Additional infrared sensors are needed to detect if bottles are on top of the robot or if it is time to go back to the recycling area.

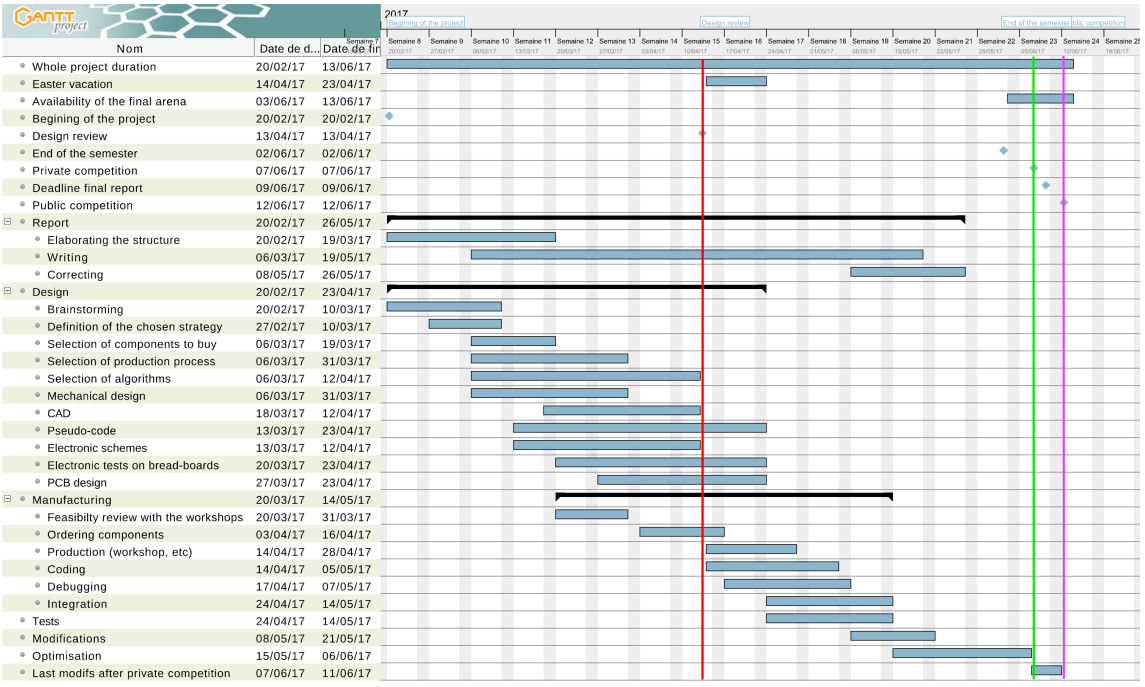
3.7 Concept of Operation

During the competition the robot shall start in the recycling area and head directly to one of the zones were the bottles have the highest score. Once it is in this zone, it shall search for bottles and grab them with its sticky conveyor belt. If the belt is full and no more bottle can fit on it, the robot shall head home to release the bottles. Then the rocky area shall be tackled and the bottles on it shall be captured. When it is full again, it shall return and continue in zone 1 for the rest of the time. During all these phases the obstacle avoidance shall be active and allow the robot to reach its target without hitting an obstacle.
The navigation shall be done visually and with encoders, coupled by for example a Kalman filter. In some cases the robot may use its on board inertial measurement unit (IMU) to verify heading or inclination angle.
All the computation shall be done on board.

4 Project management

4.1 Time Management

Since this is a huge project it had to be planned carefully. With the experience from other projects, a gantt chart respecting the deadlines that had to be met, has been built.



Nom	Date de d...	Date de fin
Whole project duration	20/02/17	13/06/17
Easter vacation	14/04/17	23/04/17
Availability of the final arena	03/06/17	13/06/17
Begining of the project	20/02/17	20/02/17
Design review	13/04/17	13/04/17
End of the semester	02/06/17	02/06/17
Private competition	07/06/17	07/06/17
Deadline final report	09/06/17	09/06/17
Public competition	12/06/17	12/06/17
Report	20/02/17	26/05/17
Elaborating the structure	20/02/17	19/03/17
Writing	06/03/17	19/05/17
Correcting	08/05/17	26/05/17
Design	20/02/17	23/04/17
Brainstorming	20/02/17	10/03/17
Definition of the chosen strategy	27/02/17	10/03/17
Selection of components to buy	06/03/17	19/03/17
Selection of production process	06/03/17	31/03/17
Selection of algorithms	06/03/17	12/04/17
Mechanical design	06/03/17	31/03/17
CAD	18/03/17	12/04/17
Pseudo-code	13/03/17	23/04/17
Electronic schemes	13/03/17	12/04/17
Electronic tests on bread-boards	20/03/17	23/04/17
PCB design	27/03/17	23/04/17
Manufacturing	20/03/17	14/05/17
Feasibilty review with the workshops	20/03/17	31/03/17
Ordering components	03/04/17	16/04/17
Production (workshop, etc)	14/04/17	28/04/17
Coding	14/04/17	05/05/17
Debugging	17/04/17	07/05/17
Integration	24/04/17	14/05/17
Tests	24/04/17	14/05/17
Modifications	08/05/17	21/05/17
Optimisation	15/05/17	06/06/17
Last modifs after private competition	07/06/17	11/06/17

Figure 8: Gantt chart of the organization of the semester

For keeping track of how the project goes, meetings at every 2 to 3 days have been fixed. In addition to that, a weekly meeting with the assistant has been arranged.

4.2 Task Distribution

Already in the early phase of the development the tasks have been split up and distributed among the team members. The three main pillars were software (Claudio), mechanical design (Héloïse) and electrical design (Roman). However, important decisions have always been made by all the members and feedback has been given in

every meeting. There have been regular meetings with the assigned experts from the robot competition as well.

4.3 Budget Management

The budget is composed of virtual and real budget. The virtual budget is of 2000 CHF and can be spent internally at EPFL for everything that is needed. The real budget is of 1000 CHF and with this money payments can be made everywhere. As an initial planning method, it has been repartitioned and assigned to different categories.

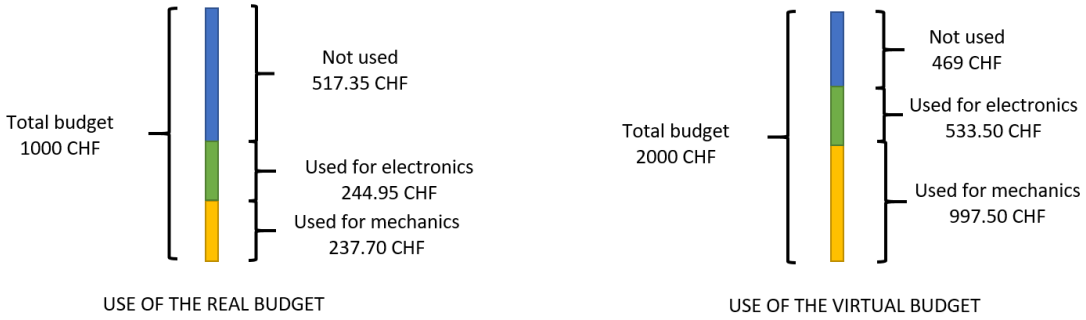


Figure 9: Status of the budget at the end of the project

	Allocated real budget [CHF]	Used real budget [CHF]	Description	Allocated virtual budget [CHF]	Used virtual budget [CHF]	Description
Mechanics	400	15.4	Order FreshAndHype.ch	900	34.95	172:1 DC motor with encoder [#10]
		17	Order Item24		34.95	172:1 DC motor with encoder [#6]
		48.35	Order RS Components (13-19.4.2017)		34.95	172:1 DC motor with encoder [#15]
		21.8	Wooden bars		34.95	172:1 DC motor with encoder [#11]
		24.95	Profiles and velcro		34.95	172:1 DC motor with encoder [#7]
		35.8	Tape and chain		34.95	172:1 DC motor with encoder [#4]
		6.55	Plastic tube		374.8	3D printing: 809.69 cm^3 model, 132.68 cm^3 support
		16.9	Glue		413.00	workshop (approximation: bill never recieved)
		30	Wheels			
		20.95	Screws			
Subtotal		237.7			997.5	
Margin with allocated		162.3			-97.5	
Electronic	400	129.55	Order MakerShop (13.4.2017)	900	36.2	Raspberry Pi 2 (model B, v. 1.1) [#2]
		50.3	Order Brack (18.5.2017)		36.2	Raspberry Pi 2 (model B, v. 1.1) [#4]
		8.65	Order Distrelec (26.5.2017)		12	MicroSD card 16 GB with SD adapter [#2]
		6.4	Order RS Components (2.5.2017)		34.95	9 Degrees of Freedom IMU [#5]
		32.55	Orders RS Components (13-19.4.2017)		31.55	Raspberry Pi Camera [#12]
		17.5	Order RS Components (2.5.2017)		18.6	80 cm IR proximity sensor [#34]
					18.6	80 cm IR proximity sensor [#21]
					18.6	80 cm IR proximity sensor [#24]
					18.6	80 cm IR proximity sensor [#31]
					18.6	80 cm IR proximity sensor [#37]
					18.6	80 cm IR proximity sensor [#12]
					18.6	80 cm IR proximity sensor [#6]
					18.6	80 cm IR proximity sensor [#11]
					0	Battery connector/fuse board [#4]
					34.95	172:1 DC motor with encoder [#3]
					45	DFRobot DRI0018 2x15A motor driver [#1]
					19.95	NiMH rechargeable battery pack (7.2 V, 3000 mAh) [#4A]
					19.95	NiMH rechargeable battery pack (7.2 V, 3000 mAh) [#4]
					19.95	NiMH rechargeable battery pack (7.2 V, 3000 mAh) [#1A]
					0	Arduone connector/power board [#1]
					44.5	Arduino Mega2560 R3 [#8]
					44.5	Arduino Mega2560 R3 [#2]
					5	USB-UART converter (PL-2303) [#2]
					0	HDMI-VGA converter [#1]
Subtotal		244.95			533.5	
margin with allocated		155.05			366.5	
TOTAL USED		**482.65**			**1531**	
Margin with allocated		317.35			269	
Total at our disposal		1000			2000	
Magin with total budget		517.35			469	

Figure 10: Detailed repartition of the budget

5 Design Phase

The design of the robot has been separated in three parts. The mechanical parts allow the robot to move in the arena on the carpet, the grass, stones and up the ramp. The robot has to be capable of grabbing, storing and releasing bottles implementing a sticky conveyor belt, the chosen solution. In addition to that, it has to carry multiple sensors located at strategical points depending on their utility. The electronic part has to ensure proper power supply for all the components used during the competition (plus a margin). And it has to be able to support the number of sensors and computers desired with proper connections. The software has to process data coming from the sensors, to plan the navigation, manage the filling of the conveyor belt and also to detect bottles and obstacles. A solid and efficient communication between the electronic components also needs to be implemented.

5.1 Mechanical Parts

To ensure a good integration of the components, the whole robot, sensors and processing units were designed on Solidworks. The overall structure has been designed, such that it can be produced at EPFL's workshop with a laser cutter. Pieces with an unconventional shape have been 3D-printed and standardized elements like a profile in aluminum have been bought. The mechanical part is composed of about 50 components, not taking into account the assembling elements such as screws.

5.1.1 Chassis and general structure

The chassis is a metal plate on which motors are fixed. To be able to adjust their positions, oblong holes have been drilled. This metal plate is fixed on two aluminum profiles. Three aluminum profiles contribute to the structure of the robot. They permit to have flexibility in pieces fixed on them and to slide the components around easily. In addition to that, they bring rigidity to the structure.
Theses profiles are used to fix the chassis with the sides of the robot as well.

The robot has an open side in order to let the air pass through it for cooling the PCB and processing units inside. Moreover, it permits to access some of the components.

The dimensions have been chosen with the idea to not exceed the clearance between the obstacles, knowing that the minimal distance between them is 0.5m. The dimensions are presented on figure 11. Because the bottles are stored on the robot and not inside, the robot is relatively compact.

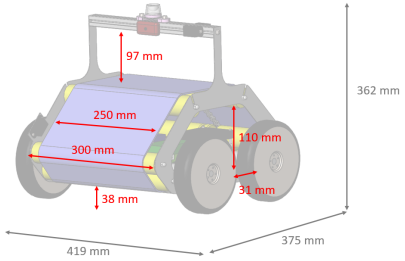


Figure 11: Functional dimensions of the robot

5.1.2 Wheels and motors

The robot uses wheels as a locomotion means. They are chosen with a big diameter such that the robot is capable of driving on all terrain. A second argument is to have the appropriate clearance under the robot. But they have a relatively thin profile to not increase the width of the robot. The four wheels are actuated to offer the capability of the robot to turn on spot and to go over the rocks easily.

5.1.3 Conveyor Belt

The conveyor belt is basically composed of three components as described on figure 12: A plastic chain to make it move, a thin sheet of plastic and a two-sided tape in order to make the conveyor belt sticky.

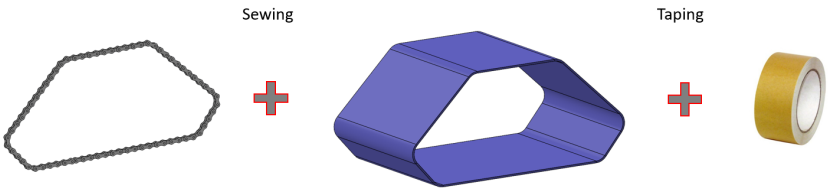


Figure 12: Composition of the conveyor belt

Because the sticky material will collect dust while operating, it is important to have the possibility to change it in a short time. The idea is to add a new layer of 2-sided tape on the previous one if needed. However, having too many layers of tape could create collisions between mechanical parts or induce stress on the belt. This is why a backup conveyor belt has been foreseen that can be switched in and replace the

dirty belt. This can be performed by removing 13 screws, as shown in figure 13. In order to adjust the tension in the conveyor belt after this operation, oblong holes have been implemented in the structure of the robot.

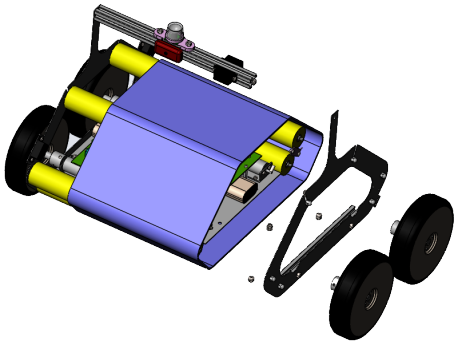


Figure 13: Changing the conveyor belt

In order to turn the conveyor belt, it is fixed on pieces called the rollers, which are composed of cylinders with a sprocket in between (see section 5.1.4).

5.1.4 Roller

A roller is shown in figure 14. They are composed of two 3D-printed cylinders supporting the conveyor belt (in grey) and a plastic sprocket (in green) adjusted for the chain of the conveyor belt. Two ball bearings are inserted in both extremities of the grey cylinder to minimize friction while turning. The sprocket is linked to the 3D-parts with a squared section aluminium axle.

Four of the rollers are exactly the same, one has an additional sprocket to be actuated by the fifth motor and the last one has its shape optimized in order to avoid mechanical conflicts with other pieces of the robot.

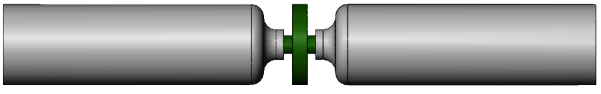


Figure 14: 3D-printed cylinders with a sprocket for the rollers

The material for the rollers has been chosen in the idea to minimize their weight and to have a good precision in length. Different solutions like wooden beams and

tubes have been considered, but it didn't seem to have as many advantages as the 3D printing, except regarding the price.

5.1.5 Supports and other pieces

To fix the sensors, many adjusted pieces have been 3D-printed. The original support of the 360° camera has been modified to fit on the top of the Raspberry Pi camera and a new support was made to have it as desired.

5.1.6 Integration of the electronics in the mechanics

All the electronic components (except the sensors for which the positioning is decided by its function), are located on a selfmade PCB. This is made in order to be able to remove all the electronic from the robot without having to unplug electrical connections. Only the motors and batteries have to be unplugged from the connectors on the PCB directly, so that it is not necessary to touch the interlinking connections between the components attached on the PCB. This support PCB can be removed from the robot by unscrewing the 4 srcews which fix it and 2 screws which hold the wheels.

The batteries are fixed on the chassis with velcro in order to be easily removed when they need to be charged. Enough velcro has been added to be able to bring the robot upside down without the batteries dropping or moving around.

5.1.7 Backup plan

The most critical part of our robot is the conveyor belt. In case a problem occurs, there still was the possibility of removing it and change the forward roller to transform the robot into a snow plow as described in the section 3.3 and depicted in figure 15. The advantage of this backup solution is that it does not need to be actuated, which reduces the chance of failure.

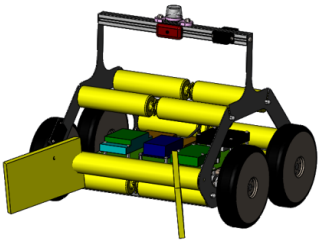


Figure 15: backup plan

5.2 Electrical Parts

5.2.1 Sensors

Bottle detection To have a robot that actually reacts to the environment, it needs to have a set of sensors. There are a variety of sensors available and most of them can be useful for this application. One of the most important sensors is the one used for bottle detection. This could be done with sonar sensors, infrared (IR) distance sensors, Lidars or a camera. The problem with sonar and infrared sensors is the high level of noise that is present in the measured data. Even with Lidars, which are light sensors that measure the distance to an object on a more sophisticated level than sonar sensors, have a noisy measurement. In addition to that, the Lidar that are commercially available are very costly.
Out of these reasons a camera has been chosen to detect the bottles. The data that is being generated can be processed with OpenCV as explained in section 5.4.2.

Obstacle detection The obstacles could also be detected with the camera, but to ensure a higher success rate of obstacle detection, a more reliable solution shall be found. An important point has been observed: the obstacles, consisting of double stacked bricks, are taller than an upright bottle. And of course, the bricks would definitely be taller than a bottle lying on the floor.
Therefore, a distance sensor is necessary. Again the IR sensors and the sonar sensors for the Arduino boards have been considered. Due to the fact that the sonar sensors have a very dispersed signal, only IR sensors shall be implemented. The goal is to have the sensors oriented horizontally at a higher altitude than the bottles, but still low enough to detect the obstacles in front. A rear sensor is not necessary, since the robot is able to turn around its center of mass and a backward motion is not part of the strategy. We considered using solutions as structured light as well which could serve for obstacle detection and bottle detection. However regarding the

computational costs due to the data generation we decided to go for many simpler and smaller sensors. This has the advantage of redundancy too.

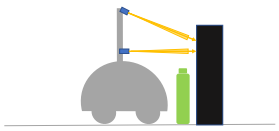


Figure 16: Side view of the IR sensors

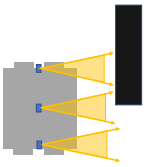


Figure 17: Top view of the IR sensors

Navigation For the navigation, the same camera as for the obstacle detection can be used. However, it is not wanted that the robot needs to turn around to navigate. Therefore an additional solution is needed that provides a vision in all directions. For this purpose a 360 degree camera or in this case a normal camera with a 360 degree lens adapter shall be placed on top of the robot to guarantee direct sight with all the LED bars within the arena. In the worst case the robot can still use its front camera to navigate by turning around itself slowly for reorientation as a backup solution. The advantage is that this method does not require any additional resources and that it can be implemented quickly enough so that there is a bit of freedom before the competition. The 360 degree camera shall be combined with the encoder readings of the wheels with a Kalman filter to have a good pose estimation. Since the navigation is a rather vital part of performing well in the arena, some backup solutions shall be implemented as well. First of all, there shall be a inertial measurement unit providing heading, acceleration and change in heading. Since good IMUs are rather expensive, it can be assumed that the data can only be so precise and that there is a drift present for example on the magnetometer. It shall therefore only be used for sensor fusion in some special cases, such as verification of the heading with low precision or for example a proof of being on the ramp with the accelerometer.
The second backup solution are two micro switches, called bumpers, on the rear side of the robot. These can be used for recalibration against a wall if necessary. This might come especially handy within the zone 3, when the calibration is distorted after passing the stone barrier.

5.2.2 Motors and Encoders

Since the decision has been made of using four actuated wheels, four motors are needed. The calculations of the torque necessary for these motors can be seen in figure 18.

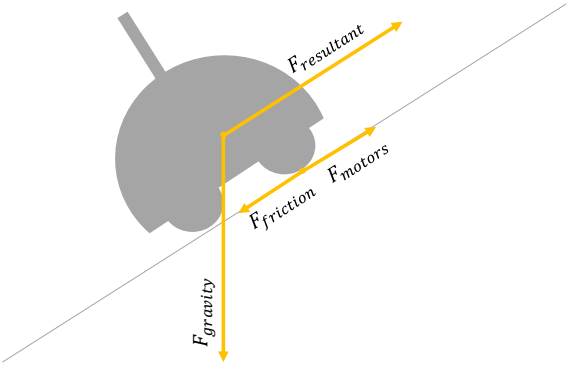


Figure 18: Schematics of the forces applied on the robot for torque calculations

To calculate the necessary torque, a matlab script has been written that basically calculates the required torque as follows:

$$F_{resultant} = m \cdot a = F_{mot} + F_{friction} + F_{gravity} \tag{1}$$

where $F_{gravity} = m \cdot g \cdot \sin(\alpha)$ and the friction force has been estimated to be the normal force times the friction coefficient $\mu = 0.6$ from rubber on wood. And the final torque is $T_{motor} = \frac{1}{4} r_{wheel} \cdot F_{mot}$. With a weight of $m = 10$kg, which is supposed to be an overestimation, this yielded a final torque of $T_{motor} = 1.4$Nm.

These calculations have been made under the assumption that the static friction of the wheels on the surface is the same as rubber on wood, and therefore the μ has been set to 0.6. Furthermore, the slope has only been estimated according to to the videos from the previous years and the angle of 16.7° has been used for the

ramp. The desired acceleration has been set to $1\frac{m}{s^2}$ so that even with a lot of internal friction, the robot would not be standing still.

With this torque needed, the Pololu 2288 DC motors have been chosen from the catalog at EPFL. The reduction gear is rather high, meaning that the robot would drive very slowly, but the torque is high enough to go up the ramp. The advantage of these motors is that the encoders are already attached to the motors. Even though it is not needed to have two encoders on each side, one on each side could also just be not interpreted. This data will help a lot for speed control of the motors and for the navigation with odometry. It is important to note at this point that the drift in odometry is very big and that it can only deliver reliable measurements in the beginning of the competition.
The control of these motors can be done with a simple motor controller unit with an H-bridge on it. For this a motor controller has been chosen from the catalog again (DFRobot Motor controller board). The symmetry of the robot allows to have the same commands on the left two wheels and on the right two wheels. This means that two motors can be connected directly to one output of the controller board. With the stall-current of 2.4 A this is not a problem. Due to imprecision of the motors and the ground, the four wheels will never turn around in perfect synchronization, even if the command of going straight forward is sent.

5.2.3 Conveyor Belt

The conveyor belt mechanism requires some sensors and motors as well. First of all a motor is needed to actuate the chain to which the sticky material is connected. It is hard to give a good estimate about the torque needed for this purpose. After a set of tests for the static friction of the chain and the rollers that turn with the chain, it seemed reasonable to take the same motor as for the wheels. This would provide the necessary torque without a problem. The major risk is that it would turn too fast, but with the high gear reduction ratio this turned out to be well adapted. Again, as for the other set of motors, it comes with an integrated encoder which is important for this task.
The disadvantage of the conveyor belt is mainly the limited capacity of bottle storage. Therefore it is vital to correctly determine if a bottle has been grabbed before blindly turning the belt. It should be avoided under any circumstance to go back to the base empty-handed. This is the reason why three IR sensors are implemented. The first detects if there is a bottle in front of the robot, the second detects whether the bottle is stuck to the belt on the upper side of the belt. If this should not be the case, the belt is rotated backwards to save some space and a grabbing phase is initiated again. And the third IR sensor shall be implemented on the rear end

of the robot. Its function is to determine whether the belt is full and the bottle that has been grabbed first reached the end of the belt. With this mechanism it is guaranteed to not drive home empty-handed.
For a feedback on how far the belt has been turned already, the encoders can be used.
This motor has to be controlled by a separated motor driver. For this reason a self-made motor driver module has been designed using the H-bridge component $L298N$. The circuit for this can be seen in the appendices.

5.2.4 User Interface

To guarantee a user friendly robot and a not so hard time in the debugging and testing phase, it is absolutely advisable to have a nice interface. This includes switches, buttons and LEDs for starting modules and control their functioning or data flow. With the choice of the conveyor belt, the electronics are not easily reachable and therefore it must be made visible from afar.

5.2.5 Computational Hardware

With these different sensors and electric components it is important to have a processing unit that connects the elements and treats the incoming data. For this purpose there are a variety of elements at disposal. Some of them are devices from Arduino, BeagleBone, Raspberry Pi and Odroid. All of them have a relatively big and active community and they show different capabilities and limitations. It was therefore necessary to implement at least two of them for having a real-time control on the motors as well as a high level data processing, for example from a camera. The devices that seemed best fitted for these two tasks were the Arduino and the Odroid. Since it was not clear whether the Odroid could handle two camera inputs at the same time, a Raspberry Pi has been chosen as well. This had the advantage that it was available from the catalog at EPFL and could be paid with the internal budget. Another advantage of this is that it makes it much more easy to separate the workload and develop on both devices individually.
The Odroid and the Raspberry Pi shall be interconnected with an Ethernet cable and the messaging service that shall be used is called Robot Operating System (ROS). The Raspberry Pi needs no further connection (except to the camera) since the Odroid has been chosen as the main device. It is connected to the Arduino through a USB cable and ROS shall be used as communication means again.
The Arduino does the low-level data measurement (encoders, sensors) and sends the

motor commands with a simple PWM signal. It is also connected to the IMU. But the IMU is operating at a lower voltage and therefore the I2C signal needs to be down-shifted to 3.3V. This is done with the schematic shown in the appendices .

The electric schematic can be seen in figure 19. The motors operate at a voltage of 6 V but one can also exceed this voltage for a short amount of time. Therefore, the battery of 7.2 V has not been down converted before it goes to the motors. All the sensors (except the cameras) consume rather little power and therefore the Arduone should be sufficient to power all of them at the same time. The amount of IR sensors used has been set to 6 in order to detect and avoid the obstacles as well as detect the bottles on the conveyor belt.

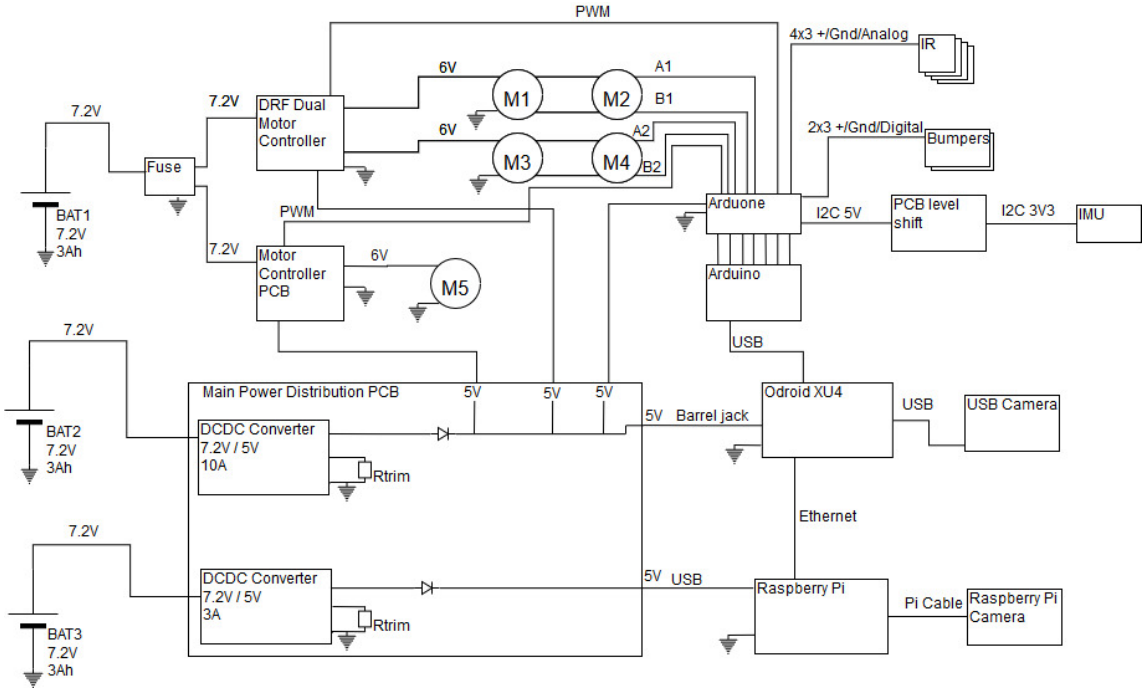


Figure 19: Electric schematic

On this figure it can be seen that the two pairs of motors are synchronized by sharing the same power lines. This is not a perfect synchronization, but still allows to turn the two motors on each side at the same nominal speed. For the fifth motor, the one that turns the conveyor belt, a selfmade motor controller was necessary. In order to guarantee the safety also for battery 2 and battery 3, fuses have been added after the DC-DC conversion.
For flexibility reasons, a backup connector for the Raspberry Pi has been created,

such that it could also be powered by the same battery that powers the Odroid (battery 2). This was made with the idea of having an extra battery for the motors, or having a 5V output available if it was needed.

5.2.6 Electrical Parts List

Knowing what will be present on the robot, one can create an electrical parts list (cf. table 5.2.6). From this, one can also find the operating voltages and the necessary (or peak) currents of the devices. Then, the energy budget can be calculated and the batteries can be identified.

5.2.7 Energy

Power budget It is needless to say that all these devices need to be powered. But it is important to keep in mind that the powering will be a crucial part of the performance of the robot. Only with a steady power supply, the different devices can operate in their nominal range. As indicated in the table 2 the total consumption is 110 W.

It is noteworthy that 65% is consumed by the motors. For calculating this the stall current of 2.4 A has been used. This is why it is safe to assume, that the nominal power consumption will be much lower. Additionally, there is the fact that the fifth motor will only run for a small amount of time during the competition. However, a margin of 50% has been added to the total power consumption and a nominal run time of 20 minutes has been assumed. This yielded the total energy needed of 9200 mAh. Therefore, three of the 3000 mAh batteries, that are available at the catalog of EPFL will be sufficient.
An advantage of having more energy is in the testing phase, to guarantee several tests before having to recharge the batteries.

Power distribution Since the different devices will be operating at different voltages and the current that is drawn can vary from very low (Arduino) to very high (Odroid) several voltage converters are necessary.
For the motors the 7.2V batteries should be no problem as long as the voltage does not exceed the nominal voltage of the motors for too long. No converter is needed at this point, but a saturation at 6V is advantageous.
The Raspberry Pi is operating at 5V and can draw up to 1 Amp. This is no problem with a simple DC-DC converter (OKR-T/3-W12-C).
The Odroid however draws much more current, that can be as high as 4 Amps.

Type	Qty	Name	Volt [V]	Peak Current [A]	Supplier	Price [CHF]
Processing	1	Raspberry Pi 2 model B	5	1.2	EPFL	36.2
Units	1	Odroid	5	4	Maker Shop	89.9
	1	Arduino Mega2560 R3	12	0.1	EPFL	44.5
Sensors	1	Raspberry Pi Camera	5	0.25	EPFL	31.55
	1	Odroid USB Camera	5	0.15	EPFL	26.9
	1	IMU SEN 11486	5	0.05	EPFL	34.95
	6	IR Distance (SHARP 2Y0A21)	5	0.05	EPFL	18.6
Shields	1	Arduone	12	0.05	EPFL	0
Powering	3	NiMH Battery			EPFL	19.95
	1	DC-DC converter 5V 10A	5	0.05	RS Components	10.55
	1	DC-DC converter 5V 3A	5	0.05	RS Components	9.11
Motors	5	Motor + encoder	6	2.4	EPFL	34.95
	1	DFRobot motor driver	5	0.05	EPFL	45
	1	H-bridge	5	0.05	RS Components	4.03

Table 2: Electronic components and consumption

A more powerful DC-DC converter (NQR010A0X4Z) is needed that delivers this power. Due to the expected high power consumption, one battery has been assigned for only this device.
For safety reasons a fuse has been foreseen after every battery.

5.3 Customized PCB

The explained circuits and connections would require quite a number of cables. Therefore, the team has opted for designing a PCB that solves as much as possible with lines on the board rather than wires. An additional advantage of such a customized PCB is the usability of the board as a mechanical component. This allows to assemble all the processing units and motor drivers on one board. This makes it much more easy to assemble, test and debug the devices. Also it allows to have all the components attached together, such that they can be mounted with little effort. The initial mounting plan can be seen in the figure 20.

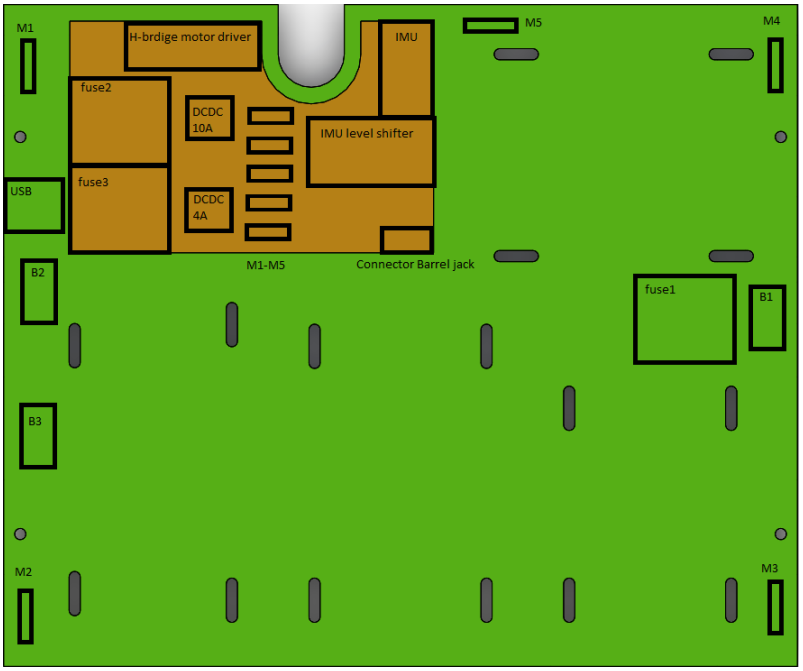


Figure 20: PCB layout with the places of the components

This board has then been adapted to make it easier to produce. These changes include mainly the half-circle cut-out which has been changed to a rectangular cut-out, as well as the oblong holes that are normal holes for the final design.

The tasks of this PCB are namely the following:

- Assemble the components
- Motor driver for the conveyor belt motor
- Battery connection via fuses to the units
- Level shift of the I2C connection with the IMU
- Down-conversion of the voltages for Odroid and Raspberry Pi

The schematics of each one of these tasks, as well as a final global connection plan can be seen in the appendices.

5.4 Software

In the end all these different components need to work together to make Mr.Sticky an autonomous robot. This is done with different software that treats and interprets the data captured by the sensors. There are four different main parts, the two algorithms that interpret the image data from the two cameras and deliver the bottle locations and the robot's actual position, the program loop running on the Arduino that processes the raw data of the sensors and sends the motor commands and finally a finite state machine that handles all the data and therefore makes the intelligent unit of the robot.

5.4.1 Processing Units

The processing units that have been used are the Arduino, the Raspberry Pi and the Odroid. Given the fact that the Arduino only has a low processing capability, the goal was to shift all the computationally intensive work, ie. the high-level commands, to the Odroid and the Raspberry Pi. The interface that has been used for this is the Robotic Operating System (ROS). This is a collection of libraries and tools that allows to have an easy way of communication in situations like for this project. Therefore, it was possible to encapsulate the tasks for the Raspberry Pi and Arduino and send only the main output, while they receive very little and basic commands in the case for the Arduino, or no command at all for the Raspberry Pi. The processing power of the Odroid alone is sufficient to do the rest of the work.
The programming environment could be chosen freely except on the Arduino. Since two programming languages are no problem to handle for ROS, Python2.7 [1] has been chosen.

5.4.2 Bottle Detection

The Open Computer Vision Library [2] is an open source library where different image processing algorithms are implemented. Image processing is a huge field with thousands of applications, such as color detection, object detection or object recognition. How is this useful for Mr.Sticky?
Using a Haar cascade [3] bottles can be detected from a video-stream. This information can be used to steer the robot towards these bottles in order to collect them. Further OpenCV commands can be used to detect colors in images. Since there are four different LED strips placed in the final arena of this competition, they can be used to do triangulation in order to calculate the robot's position and orientation.

5.4.3 Navigation

For the navigation part different algorithms have been considered. But first the question of main input for the navigation needed to be addressed. One possibility was to use a version of SLAM (Simultaneous Localization and Mapping) that is based on monocular vision and encoders. Another option is to use the triangulation system of the arena with the LED towers and filter these measurements together with the odometry from the encoders. This can be done by using a Kalman filter in order to have a good estimate of where the robot is on the map. Even though an already made implementation of SLAM has been found, the choice for the navigation was the one with the Kalman filter. This was mainly due to the reason of its success rate and also it is much easier to solve problems if some arise.

Odometry The odometry could only be done for two wheels out of the four, since the Arduino provides not enough interrupt pins and the workload had to stay low. Therefore, only two wheels could be measured, which already implies a certain imprecision when some of the wheels slip or lose ground. With the interrupts on the encoders, one can calculate the covered distance of the two wheels and predict where the robot is. This is a relatively precise method for short distances, since the motors have such a high reduction ratio, but becomes very poor after some time has passed, since the position and orientation are based on the previous predictions which includes a drift.

Triangulation The basic idea of triangulation is similar to the global positioning system (GPS). In this project the arena is a square with a length of eight meters. In every corner there is a LED light with a different color. This information can be

used to calculate the position within the arena. According to the ToTaL Algorithm [see 4] at least three angles need to be measured in order to calculate the position and the direction.
In order to apply this concept there needs to be a system that measures the angles. The idea is to use a linear camera with a mirror that allows light from 360 degrees to reach it. Finally the angles can be measured by detecting the colors within this image.

Kalman Filter The Kalman filter is a method used to merge two or more input signals, in this case the measurements of the triangulation and odometry. The filter has a model of the robot and can also use the control input to give a prediction of the propagation. The measurements have an intrinsic covariance that can change over time (as it is the case for the odometry). The model also has a covariance, which can be determined experimentally with a lot of testing and tuning.

5.4.4 Sensor readings

The implemented sensors are mainly the infrared for the obstacle detection and to determine if a bottle is stuck to the conveyor belt, as well as the bumpers and the IMU. These sensors are read only by the Arduino and the values are then transmitted to the state machine that decides on how to proceed.

5.4.5 State Machine

The state machine is used to explain and define the behavior of the robot during the competition. It shows the different states that the robot may be in and shows the conditions that need to be met to switch to another state. The main functions and conditions and the flow direction is shown in the figure 21 below.

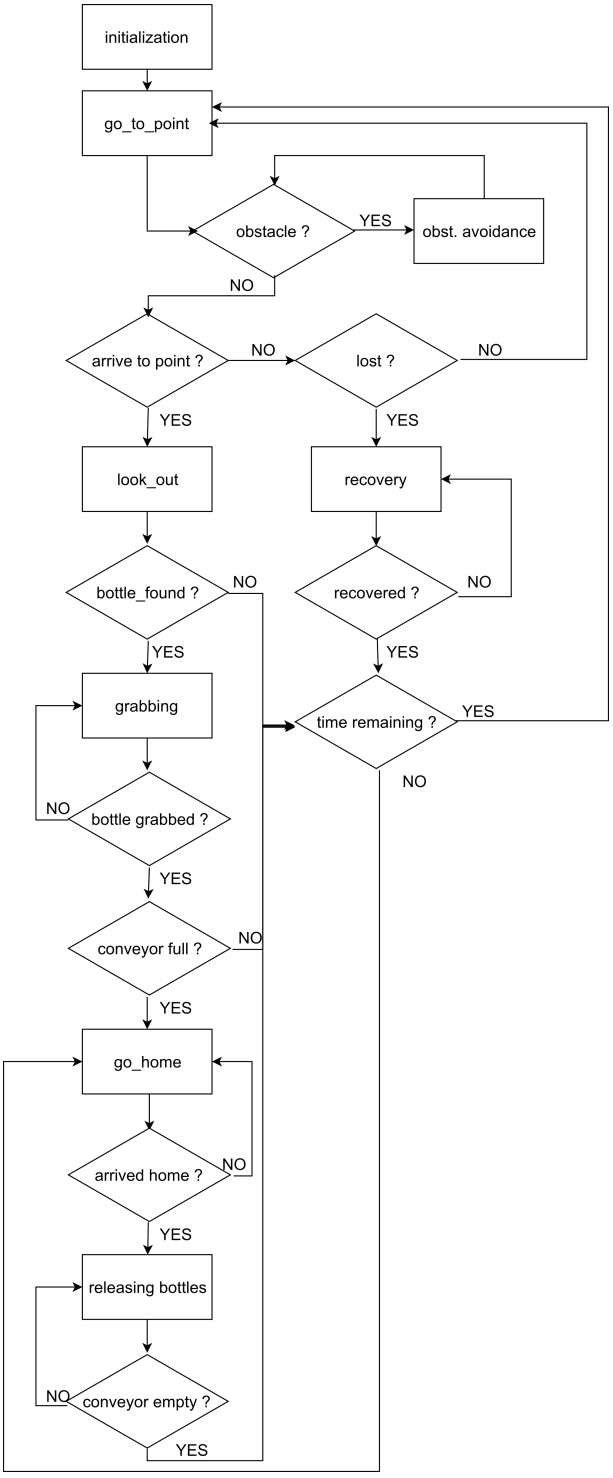


Figure 21: State machine

6 Building Phase

Once all the mechanical components have been designed and bought, the mechanical assembly process can be started. This section is dedicated to explain this in detail.

6.1 Assembling procedure

1. Fixing the motors on their brackets. These brackets are fixed on the plate of the chassis. Adding screws on the dedicated holes. The screws on the chassis are not screwed tightly in order to be able to slide the plate in the aluminum profile at the following step.

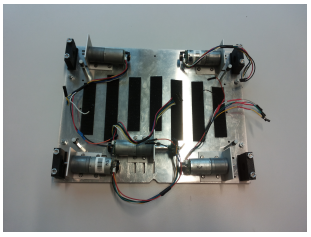


Figure 22: Motors fixed on the bottom plate

2. Sliding the chassis in the two small aluminum profiles. Screws dedicated to link the chassis with these profiles can be screwed tightly.

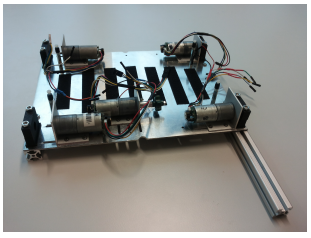


Figure 23: Assembly of the bottom plate with aluminum profile

3. Assembling the two sides structures of the robot with the plate for the top support.

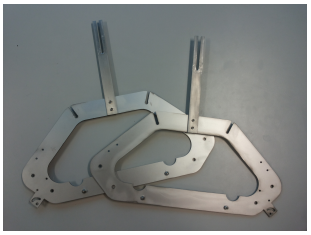


Figure 24: Two sides of the robot

4. Sliding one side structure in one of the aluminum profile of the chassis. Screwing it tightly.

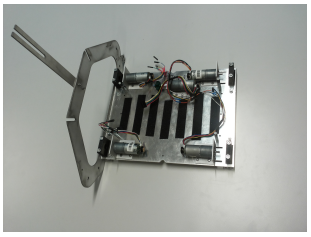


Figure 25: Fixation of one side of the robot

5. Assembling the rollers.

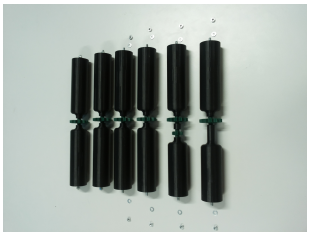


Figure 26: Assembled rollers

6. Screwing the rollers in one side of the robot.

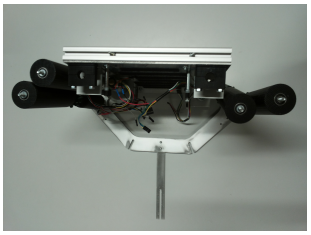


Figure 27: 4 rollers already screwed in one side of the robot

7. Assembling the conveyor belt

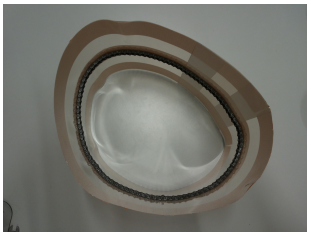


Figure 28: Conveyor belt assembled

8. Inserting the conveyor belt on the rollers
9. Screwing the second side structure of the robot

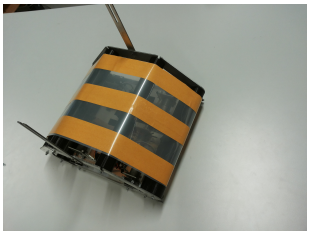


Figure 29: Conveyor belt assembled on the robot

10. Inserting sensors in the horizontal aluminum profile

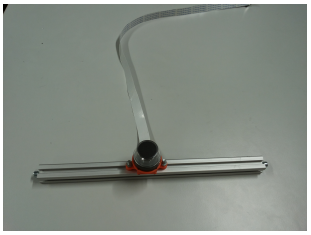


Figure 30: Camera 360 degree on the top aluminum profile

11. Screwing the third aluminum profile in the top plate, the supports of the bumpers and the blade to remove the bottle.

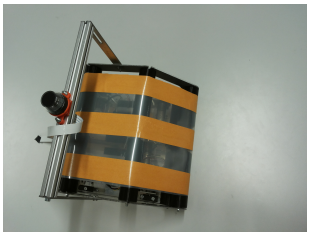


Figure 31: Top view of the robot with the sensor's support

12. Screwing the 6 IR's support at their dedicated place.
13. Inserting batteries and the electronic plate. Connecting motors and sensors.

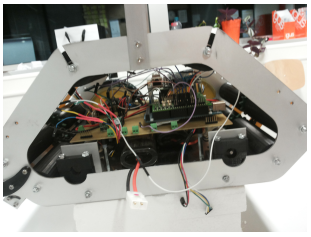


Figure 32: Insertion of the electronic plate inside of the robot

14. Screwing the 4 wheels.
15. Fixing the protection plate under the robot.
16. Removing the protection on the tape to obtain a sticky robot.

Figure 33: Final robot in the final arena

7 Software

In order for a robot to be autonomous, there needs to be a software architecture that organizes the control of the different actuators as well as the reading of the sensors. Therefore a communication system is needed, such that the different algorithms can transfer data. For this robot the Robotics Operating System (ROS) [5] has been chosen due to it's simplicity and huge online community.
The raw data coming from the various sensors needs to be processed such that it can be converted into commands for the robot. Therefore different algorithms are needed. They go from filtering different position data as far as real time processing of image data to detect bottles.

7.1 ROS - communication medium

Both, the Odroid and the Raspberry Pi 2, run Ubuntu 14.04, which is an open source operating system. ROS Jade has been installed on these two computers such that they can communicate with each other, as well as different packages that allow ROS to communicate with the Arduino. The advantage of ROS is that it handles the whole communication and process division automatically, which means that no protocols need to be defined. This comes with the cost that it is computationally expensive for the serial connection between the Arduino and the Odroid.

Nodes, Topics and Messages To guarantee a successful communication between the different parts of the software architecture, ROS uses nodes, topics, messages and services. The main algorithms are running through different nodes and publish the obtained data to topics. The publishing format is a ROS-message, which can be defined beforehand. Nodes can subscribe to a topic, and whenever something is published to that topic a callback function is called. Through services a running node can receive commands to change it's behavior.
Custom ROS-messages have been defined to make the communication easier and more readable. They can be found in the appendices.

Starting Script To automatically initialize Mr.Sticky, a shell script has been written that is launched upon booting. This uses a roslaunch file which automatically starts different nodes even across multiple machines (Odroid and Raspberry Pi). In order to function, the ssh connection needs to be set up beforehand by generating access-keys between the devices and modifying the hosts file.

7.2 Node tree graph

The interaction between the nodes is depicted in the figure below.

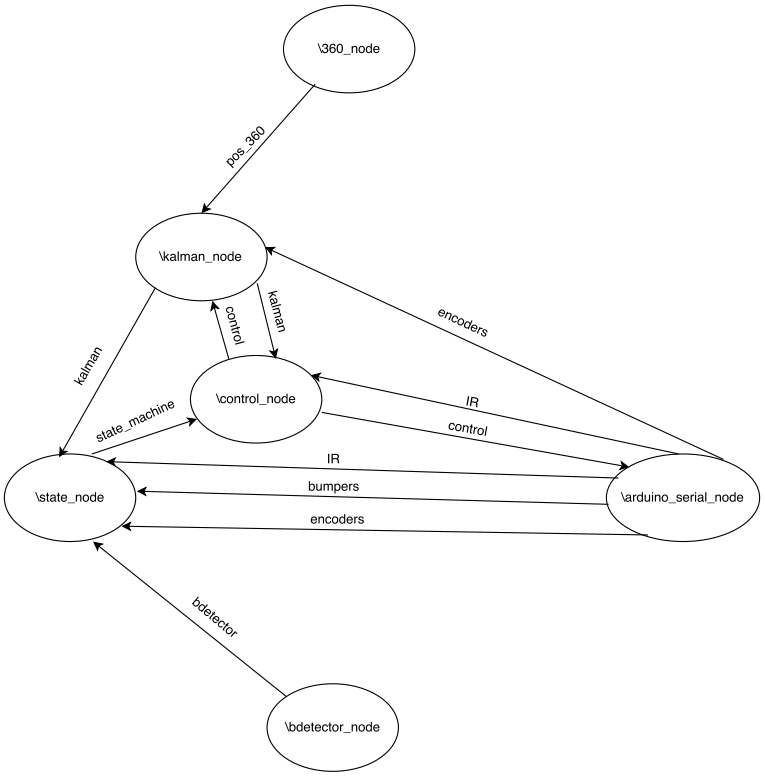


Figure 34: The node graph with nodes as ellipses and topics as arrows

The nodes are shown in ellipses and the topics are the arrows. As it is shown in the figure, the state node is the main junction of all the nodes that sends its output to the control node. The advantage of this node structure is that nodes such as the bottle detector and the triangulation nodes are publishers only. Even if one of these nodes would break down, the script would restart the separate nodes and the robot is still functional.
The messages that are published in these topics are almost all customized for a more user-friendly experience. These messages can be seen in the appendices.

7.3 Algorithms

The third main part of the software architecture are the different algorithms that do the work. The chosen concept is to use a finite state machine. This means there

are a finite amount of predefined states, which are changed upon different inputs. These inputs are calculated using different algorithms such as the bottle detection algorithm.

7.3.1 Finite State machine

The predefined states are the following:

- init
- recovery
- go to point
- lookout
- grabbing
- homing

These states define all situations the robot could possibly be in and the state machine decides how it gets there and what it does.

init In the initializing phase all the variables are set to the initial values and the robot basically waits until it is sent to fulfill its task. When the ROS nodes are started, it automatically enters this state and only when both bumpers are pressed for 5 seconds, it switches to the first state, the go to point state.

go to point Here the robot reads a point from the actual goal list, which is a list of points of interest. This is not a fixed list and can change its size and ordering during the game. Once it is empty and the robot enters this state, it is automatically appended by the center. On the way to the wanted position, the state machine receives the input from the bottle detector, ie. the number of detected bottles and the angle of the best fitted one. If the angle is low enough (smaller than a variable that depends on the states), the robot chooses to go to the bottle first and afterwards it will redirect to the point from the goal list. Also it constantly checks if the front and back infrared sensors detect a bottle, which would launch the grabbing phase or the homing state respectively.
The points that are added to the goal list are eight predefined waypoints as can be seen in figure 35. These points have been chosen to be able to access all different areas in the arena with ease.

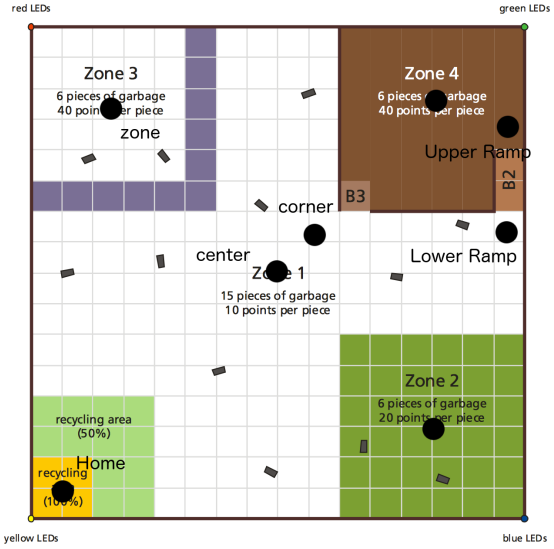


Figure 35: Waypoints within the arena

lookout When the robot has reached its point of interest, it enters the lookout state. Here it turns around itself until it finds a bottle or a certain amount of time has passed. When a bottle is detected, it will go to it and grab it and lookout for more bottles, if no bottle has been detected, the next point of interest is set as a goal.

grabbing This is the case when the front infrared sensor has detected a bottle. The robot moves forward for some centimeters and then the conveyor belt is turned, until the second infrared detects a bottle. If this does not happen after a certain time or a certain distance of the conveyor belt (detected by the encoder), the conveyor belt is turned back at its original position and the robot moves forward again and repeats its process. If no bottle has been detected for a second time, it enters the lookout state, otherwise it will go to the next point of interest. The last infrared sensor along the conveyor belt checks if a bottle is on the very end of it and if this is the case, the robot is sent home.

homing In this state, the robot is full, or too much time has passed. It goes back to the recycling area and automatically releases the stored bottles. During the testing phase it has been shown that it might be useful to move forward while releasing. After the releasing process, the robot goes to the next point again.

recovery This state is any major problem that occurred and the robot should try to localize itself or restart devices if connections are lost. This state has not been used for the competition, since testing showed no problems.

7.3.2 Control

The control is implemented as another node that receives the input from the state machine, the current pose estimation of the Kalman node and the infrared sensor readings. The goal with x and y coordinates as well as the desired orientation are saved in the state machine message. Additionally, the state machine sends a value of -1, 0 or 1 to indicate in which direction the conveyor belt should turn, if any. This is verified in the beginning. If the conveyor belt should not turn, the control is computed as a function of the current position and the goal. If the orientation of the robot towards the goal is higher than a certain threshold, this first needs to be corrected. The speeds of the left and right wheels are calculated and are of opposite sign. Once this orientation difference has been annihilated, the robot can start driving towards its goal.
The PID that has been implemented worked well on the robot to reach its goal and throttle the speed before arriving, however with abrupt changes of the goal or sudden detection of obstacles, it seemed that high differences in input might have been a reason for why the motors broke down so many times. Therefore, a slew rate has been implemented. This slew rate is simply put a limit on the current motor input based on what the previous input was. The slew rate has been tuned to be at 30 experimentally, where the motor control command is bounded between -255 and 255. Full throttle corresponds to roughly 0.5m per second.
The obstacle avoidance has first been implemented on the state machine, where it did not seem to work too well. Therefore, it has also been implemented in the control node. This is done by checking the infrared sensors and reducing or increasing the wheel speed if an obstacle is detected. This alone was not sufficient either, which is why a combination of both has been used as the final method. The values that are implemented for this obstacle avoidance have been fount to work best in an empirical manner.

7.3.3 Bottle Detection

Haar Cascade The bottles are detected through image processing using the OpenCV Library for Python. Within OpenCV there are hundreds of implemented open source algorithms. The one that is used for the bottle detection is a Haar cascade [3] which needs to be trained. In order to train the Haar cascade it needs

to be fed with a lot of positive and negative images. There are several methods to do that:

1. Generate positive images using a few original images of bottles. These images are automatically put on negative images with different angles and positions.
2. Take all images with a script and put different bottles manually.

The first method has been tried several times with different source images without delivering promising results. Therefore the second approach was used. A script has been written which automates the image capturing process, such that several hundreds of images can be taken relatively quickly. These images are then fed into the algorithm as well as negative images which are produced in a similar fashion. The train cascade algorithm from the OpenCV library then produces a Bottle Classifier (.XML file), which is finally used to detect bottles.
Two example images that have been used to train the cascade can be seen in figure 7.3.3 and 7.3.3. In order to do the training two different tutorials have been used, namely [naotoshi 6] and [pythonprogramming.net 7].

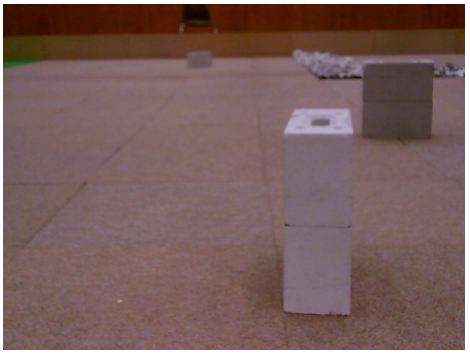


Figure 36: Negative used for training the Haar cascade

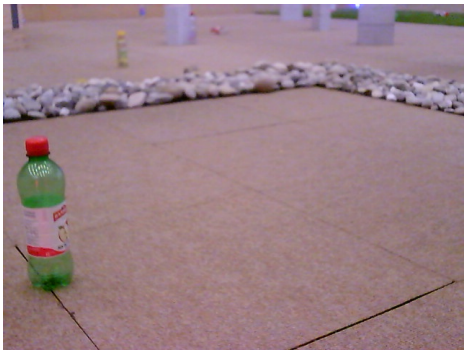


Figure 37: Positive Image used for training the Haar cascade

Detection Algorithm The algorithm uses a live stream of images and detects bottles with the classifier file. The output is an array of pixels which describe a rectangle around the bottle. These detections are saved in an array and compared to existing detections. Whenever there is a new detection which is far enough from old detections a new bottle instance is created. If the detection is close to an existing bottle instance it is added to it. Finally there will be several bottle instances with a different number of detection each and a different average pixel position of all corresponding detections.
Since there are always false positive detections there is a need for a filter that removes wrong detections. This can be done by the number of detections, since a false

bottle will be detected a lot less than a real bottle. The validate method takes care of this and outputs a list of bottle instances that passed the filter.
Finally there needs to be one bottle that is chosen. The decision method has different objective functions to choose from such as smallest angle from the heading of the robot or the number of detections per bottle instance. In the end there should only be one bottle at a time, such that Mr.Sticky can actually handle it. Therefore a decision function takes all detected bottles and decides which one is tracked. Its pixel position is converted into a real angle by knowing the field of view of the camera used.

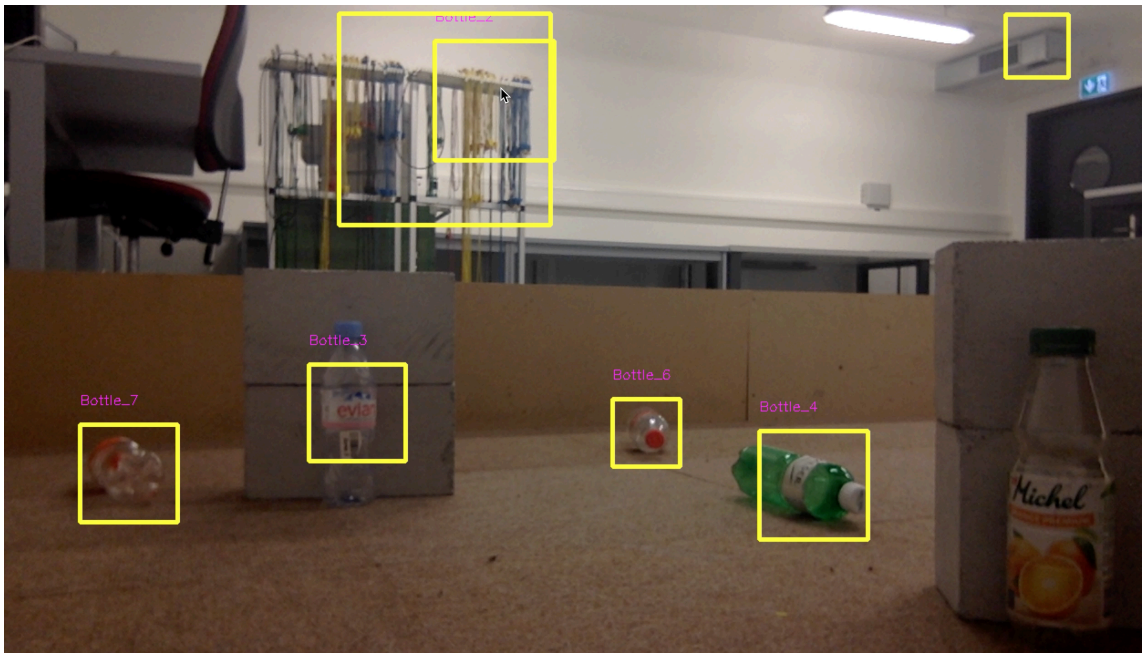


Figure 38: Visualization of the Haar Classifier in action

In figure 38 the bottle detection is visualized. One can clearly see two false positive detections in the upper part of the image. These are filtered out by cropping the image to the right ROI (Region Of Interest).

7.3.4 Obstacle Detection

For the obstacle detection 3 infrared sensors have been used. The one in the center is placed on the profile bar, where the cameras are attached. Since this height is variable and higher than the obstacles, a 3D-part has been printed to allow an adjustable tilt. The two additional IR-sensors are placed on either side of the profile

Left Sensor	Center Sensor	Right Sensor	Action Performed
			Keep going
X			Turn right
X	X		Turn hard right
X		X	Turn very hard left
	X		Turn left
	X	X	Turn hard left
		X	Turn left
X	X	X	Turn very hard left

Table 3: where X stands for a sensor that has been blocked

fixations. The algorithm for the obstacle avoidance method is relatively simple and is based on the fact that the robot only moves forward.

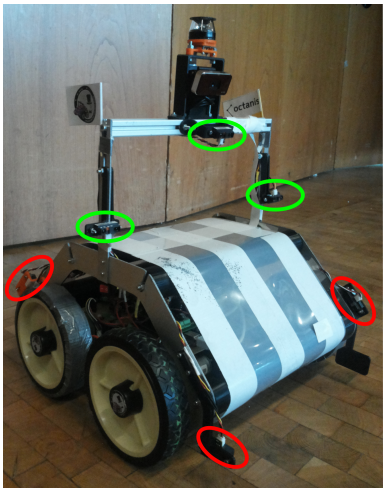


Figure 39: Location of the IR sensors on the robot, green: IR sensors for obstacle avoidance, red: IR sensors for bottle sensing in front of the robot and on the conveyor belt

7.3.5 Odometry

The odometry is done by retrieving the data from the encoders on two out of the four wheels, one on each side of the robot. We use it to know how much the robot has moved in order to be able to localize the robot. The wheels that have been used for odometry are the ones without the duct tape, since they have more grip. This is an estimate of movement, which means that the initial position must be known in

order to calculate the absolute position.
The sensors used to measure this motion are the incremental encoders mounted on the motors (Pololu 2288), with approximately 8246 counts per revolution. By counting the signals in the encoder the turns per time can be calculated and with the known diameter of the wheels this can be translated into actual motion.

To calculate the covered distance Δx of the wheel, the difference between two encoder readings Δk has been used as follows:

$$\Delta x = \Delta k \frac{2 \cdot \pi \cdot r}{CPR \cdot reduction}$$

The CPR is the intrinsic counts per revolution that is given by the quadrature encoder. In this case $CPR = 48$ [8]. The reduction is given by the gear and is fixed to 172. The radius of the wheel has been measured and corresponds to 0.075m.

7.3.6 Triangulation

The triangulation is being done with a 360 degree camera system, where the four LED's with different colors can be detected. Being able to measure the angle between these LED's the ToTal algorithm is used to calculate the position of the robot [4].

Color detection First of all a preprocessing mask is applied to the grabbed image such that only the correct ROI is considered. Once the camera is placed in its fixation a test image is used to find the camera center in pixels. From there a mask is created using two disks with different sizes. Indeed, the final image represents nicely the arena and therefore the correct ROI (see figure 7.3.6).

The second step is to convert the images into the color space HSV (Hue, Saturation and Value), where specific color ranges can be defined more easily. The hue value defines the principle color range, whereas the nuances within this range are defined through the saturation and the value. For each of the four colors there is a predefined range which can be seen in the source code. The *cv2.inRange(img, lower_bound, upper_bound)* function from OpenCV creates a mask of the input image where all the pixels within the given range are white and all others are black. This means that four masks are created, one for each color range. By doing a bitwise operation the correct pixels of the original image can be identified. Since the color detection does not always work perfectly due for example to different light conditions, the masks

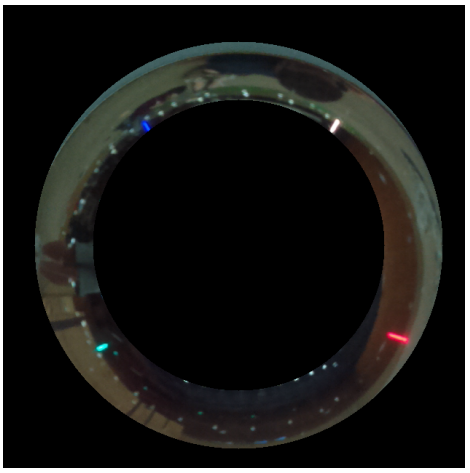


Figure 40: Frame through the 360° lens with preprocessing mask

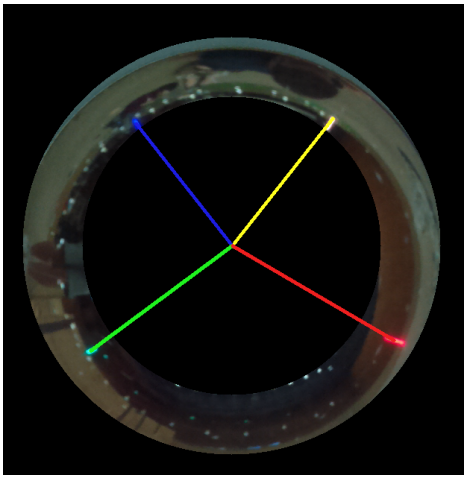


Figure 41: Visualization of LED-detection algorithm

need to be processed in a way such that the LED is detected, but nothing else what could be due to noise or disturbances. Using several morphological transformations on the images [9], the larger white zones are enlarged, whereas the smaller zones are eliminated. This is done by first opening the image and then closing it afterwards, see figures 7.3.6 and 7.3.6 (image source[1])

Figure 42: Opening example from OpenCV

Figure 43: Closing example from OpenCV

In the end the largest of the zones, which has the same elongated shape as the LED strip, is chosen to be the right detection. This is done by first finding the contours on the four masks and then by calculating the image moments using the *cv2.moments()* function from the OpenCV library. With this the centroid of the detected zone can

[1] `http://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html` (accessed: June 13th, 2017)

be determined and knowing the center of the camera the angle can be determined with a simple trigonometric calculation.
In order to make the triangulation more robust the angles are checked for errors. Because there are four different colors, the order of the colors must always be the same. If this is not the case, the image is discarded and no position is calculated. To further improve the robustness a color tracking could be implemented, but since the localization worked well enough without it, this has not been done.

Triangulation - ToTaL algorithm These four angles can in a further step be used to calculate Mr.Sticky's position by knowing the position of the beacons. In order to do this the ToTaL-algorithm [4] has been used. This algorithm only needs three angles to calculate the position and the orientation. Since normally four angles can be extracted the algorithm is run four times with different combinations of three angles and finally an average of the position is calculated.

7.3.7 Kalman Filter

The two measurements for the position have different characteristics and precisions. The odometry based on the encoders for example have a drift and the final pose estimate gets quite bad after a short while. The triangulation however works around the ground truth but has lower precision. From time to time it might have big jumps from one pose estimate to another. Therefore it is necessary to implement some sort of filter to use the advantages of both. This can be done with the Kalman filter. The principle is to calculate different weights for the measures based on their covariance matrices. It also uses an internal model (that itself has an uncertainty) to propagate the prediction of the pose. For this the following kinematic model has been used.

$$v_L = r \cdot \omega_L$$

and

$$v_R = r \cdot \omega_R$$

where

$$\omega = \frac{v_R - v_L}{L}$$

$$v = \frac{v_R + v_L}{2}$$

This is used for the kinematic equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2}$$

In order to integrate the speeds and get the position and orientation out of it, the Runge-Kutta method to the fourth order has been used. At first, a script has been written that allows to call an update every time a new measure has been made. Later on, this has been implemented in a node with ROS, such that it can subscribe to the encoders and the 360 degree camera.
For testing the script, simulated data has been sent and the result has been plotted with matplotlib - a python library for plotting. This can be seen in the figures 7.3.7 and 7.3.7 below.

As it is indicated on the figures, the tracking of the robot is rather precise up to several centimeters. The figure indicating the angle however, shows the singularity for the case of an orientation of 0 or $2 \cdot \pi$. In this case the numerical value changes a lot whereas the physical orientation does not change at all. Yet, the angle is tracked well enough and the true value is achieved after a short amount of time. The timestep that has been chosen for this simulation corresponds to 50ms.

7.4 Arduino - Low Level

An Arduino MEGA 2560 board is used to send motor commands and read sensor data with the help of the general usage shield Arduone (designed by A. Crespi - EPFL). For the infrared sensors the analog pins from 0 to 5 have been used, whereas the bumpers have been connected to the digital pins. The two lines for each encoder were connected to the interrupt pins and the interruption was therefore triggered on both, the falling and the rising edge. The motors could be controlled by the PWM pins sending the enable and the PWM signal. For the four motors only four pins have been used, due to synchronization. The last motor driving the conveyor belt has again two connections to the PWM pins.
In order to transmit the sensory input the rosserial connection has been used, publishing all the measurements at every iteration of the loop. The control commands are read directly from the topic *control* that correspond to the commands for the left wheels, right wheels and conveyor belt in a range of $-255 \leq cmd \leq 255$.
Besides reading these sensor values and publishing them, as well as controlling the motor according to the command signal, the Arduino does not have to do any additional computation. This shifts the workload onto the Odroid which is much more

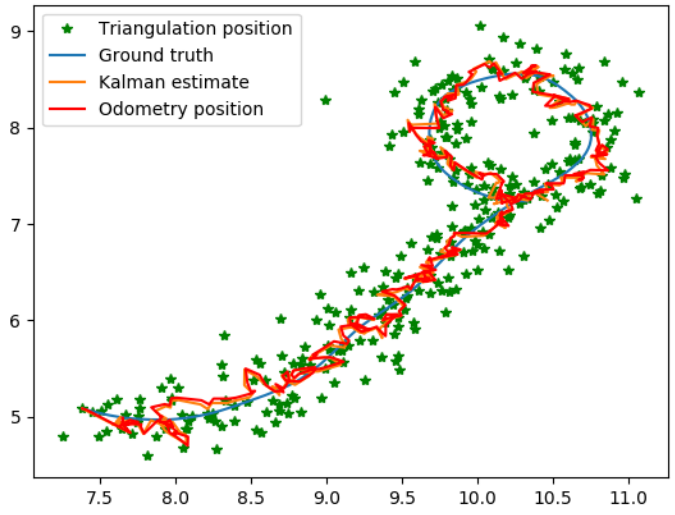


Figure 44: Simulated position of the robot with the Kalman filter

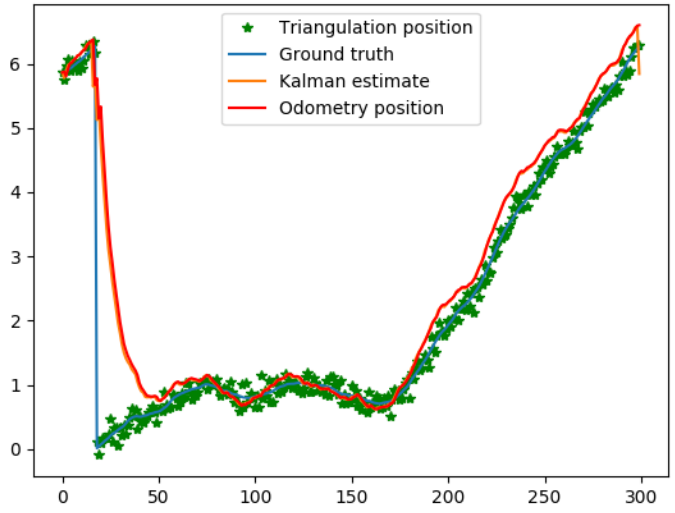


Figure 45: Simulated angle of the robot with the Kalman filter

efficient and reduces the data sent over the serial cable to a minimum.
Given the fact that the localization turned out to work quite well, the implementation of the IMU has not been done.

8 Testing Phase

In a first time, the different parts of the robot have been tested independently before being fully assembled. When the component seemed to be functional, they have been integrated all together to do a general testing.

8.1 Mechanical tests

- Driving over obstacles (tools, feet, etc)
  Issues encountered: Robot gets stuck on the obstacles, wheels spinning in the air
  Solution: Increase the speed to the maximum value
  Result: success
- Rolling up an inclined board at full speed (robot with all components inside)
  Result: success
- Turning the 4 driving motors
  Issues encountered: The wheels kept detaching.
  Solution: Glue the screw to the axle of the wheel fixations.
  Result: success
- Driving on cobbles
  Results: success
- Turning on spot
  Issues encountered: Wheels had too much grip, motors broke
  Solution: Put tape, find new wheels
  Result: partial success with tape
- Drive around in the arena
  Issues encountered: 5 motors broke.
  Solution: implementing a PID and a slew on the motor commands to reduce impact of bang bang acceleration. Adding tape on two wheels to reduce the friction while keeping the ability to turn on the spot (center of rotation corresponds to center of the robot).
  Result: success
- Turning the conveyor belt motor
  Issues encountered: The motor for the conveyor belt is driven by an H-bridge on the selfmade PCB and not on the same PCB as the motor control. But because of the design of the path on the selfmade PCB, the conveyor belt wouldn't turn

when the other motors were not turning.
Solution: This problem has been fixed through coding.
Result: success

- Turning the conveyor belt
  Issues encountered: bending of the actuated roller and stopping of motion of the roller.
  Solution: elevating the motor, glueing the roller
  Result: success

- Grabbing bottles
  Issues encountered: Conveyor belt is stuck to the protection plate on the ground.
  Solution: Leave out the protection plate entirely
  Result: success

- Grabbing bottles
  Issues encountered: Sprocket for the conveyor belt jumps teeth.
  Solution: Start with higher velocity
  Result: success (partial)

- Detecting bottles
  Issues encountered: Bottles detected got out of sight
  Solution: Adjusting and tuning the angle of the camera with respect to the robot
  Result: success

- Bottle grabbing
  Issues encountered: Bottle moves away from robot is not stuck at first contact.
  Solution: Drive an additional distance and turn the conveyor back if no bottle is detected
  Result: success

- Bottle on belt testing
  Issues encountered: Bottle is not detected.
  Solution: Fine-tune the IR thresholds and implement a time and distance limit of the turning conveyor belt
  Result: success during testing but failure at the competition

- Releasing bottles
  Issues encountered: Bottle was stuck to the conveyor belt after turning
  Solution: After covering some of the sticky parts on the conveyor belt to reduce the sticky surface, this risk has been accepted, since it worked for most of the test cases
  Result: success (partial)

- Communication between Arduino and Odroid
  Issues encountered: Input/Output Error 5
  First solution: Change cable
  Result: no success
  Second solution: Add ferrite to the Arduino for serial port filtering
  Result: no success
  Third solution: Switch out Arduino
  Result: no success
  Fourth solution: Power Arduino and Odroid directly from separated power sources
  Result: no success
  Fifth solution: Change code and reduce transmitted package sizes
  Result: no success
  Sixth solution: Switch out the trouble causing motor
  Result: Surprisingly worked - success

8.2 Electronic tests

- PCB testing
  Issues encountered: PCB burned through due to short circuits.
  Solution: Solder connecting wires as replacement for the lines
  Result: success
- Runtime of the batteries of the robot driving full power.
  Results: after 20 minutes the battery was still ok, so success
- Connecting everything
  Issues encountered: no sensor connections due to bad quality cables
  Solution: Change cables
  Result: success
- Arduino connection with the selfmade PCB
  Issues encountered: voltage on the pins of the arduone at 2.8V instead of 5V due to the destruction of the voltage converter of the Arduone
  Solution: Unsoldering the dead component
  Result: success

8.3 Software tests

- Detecting bottles
  Issues encountered: The algorithm could not detect bottles when turning too fast
  Solution: Reduce the angular velocity
  Result: success
- Detecting bottles
  Issues encountered: Several false positives are detected
  Solution: The false positive are appearing and disappearing whereas the true positive are detected relatively steadily. The bottle detection algorithm tracks the detected bottles
  Result: success
- Communication with the Raspberry Pi
  Issues encountered: The Raspberry Pi kept overheating and rebooting at some point, triangulation was infeasible
  Solution: Switch out the Raspberry Pi
  Result: success
- Obstacle avoidance
  Issues encountered: Obstacle not detected, IR not working properly.
  Solution: Switch out the IR that was not working and tune the IR threshold in the parameters
  Result: success (partial)
- Kalman tuning on simulation
  Issues encountered: Estimation was off
  Solution: Tune the covariance matrices
  Result: success
- Navigation with a precision of 10 centimeters[2]
  Issues encountered: The navigation was sometimes completely off (only on the Raspberry Pi, not the computer).
  Solution: Use the atan2() function instead of atan().
  Result: success
- Homing
  Result: success

[2]This needed a lot of work, but once it worked with the triangulation, the odometry was no problem with the Kalman.

9 Results and Discussion

In this section the results of the robotics competition on June 12th 2017 are presented. Mr. Sticky had several problems which resulted in zero points after ten minutes. He started well by recognizing a bottle and steering directly towards it. As soon as it got detected by the infrared sensor the conveyor belt was starting to move, but instead only a clicking sound arose. Unfortunately the belt was sticking to the bottom plate which was supposed to protect the sticky surface. After removing the aluminum plate the conveyor turned just fine and Mr.Sticky grabbed its first bottle. After continuing the journey in the arena it grabbed another bottle, but due to an error with the infrared sensors the belt kept on turning and therefore the two bottles got released at the wrong moment and were therefore lost.
The competition ended with Mr.Sticky going back to the recycling zone which proved the functioning localization with the triangulation and the Kalman filter.

9.1 Mechanical Parts

There were two major issues regarding the mechanics of Mr.Sticky. First there was the problem with the wheels. They are too heavy and the connection to the DC-motors has not been done properly. There should have been ball bearings around the aluminium axis such that the weight of the robot is not applied directly on the motor shafts.
Second, the protecting plate should have been mounted with a larger clearance such that there is no interfering with the sticky conveyor belt.

The first problem was solved partially by adding 3D-printed parts that imitated a plain bearing by having a smooth surface to minimize friction. With these compromises the robot was able to go to all the zones (see videos), which is an acceptable result.

9.1.1 Chassis

The chassis was stable throughout all the tests and also during the competition. Its compact size was a problem for intensive testing when access to the electronic parts was needed, however this could be solved by not mounting the conveyor belt for the first part of testing. As mentioned, the ground plate was stuck to the conveyor belt during the competition which could be removed within some seconds.

9.2 Electrical Parts

From an electrical point of view the only thing that did not work just before the competition was one of the infrared sensors necessary for the obstacle avoidance. This problem has been solved by replacing it by a new one from the internal stock at EPFL. Other than that there was no issue with the electrical parts.

9.3 Software

9.3.1 Processing Units

The processing units worked fine, after the malfunctioning Raspberry Pi has been replaced. The only thing that can be implemented for the next time is the script that should have a higher success rate of setting up the connection between the Odroid and the Raspberry Pi. Somehow this has not worked for all the cases.

9.3.2 Bottle Detection

The bottle detection worked fairly well, since Mr.Sticky managed to approach two bottles. Points of improvement are a better tracking and backup strategy without Haar cascade.

9.3.3 Navigation

The navigation worked surprisingly well which is mostly due to the triangulation. The odometry is not very precise, since it accumulates the error. In order to improve this, a calibration step should be implemented.
While testing the triangulation a precision of about ten centimeters was reached, which is a very good result considering the total length of the arena being eight meters. For further improvement one could robustify the triangulation algorithm such that it works with different light settings.

9.3.4 Control

The control worked well during the competition, since the robot navigated to the wanted position (ie. the first goal) and went back to home when it was full.

9.3.5 State Machine

The state machine was one of the weaker points of the algorithm. This could not be tested as intensively as previously planned and therefore a lot of workaround and clumsy bugfixes had to be implemented to make the robot behave according to the needs. An example for this is the double obstacle avoidance, the grabbing mechanism that had to be implemented with respect to the encoder position as well as the time that has passed and also the unfinished backup state such as recovery. Yet the switching logics and state transitions worked well and the robot new what it had to do during the competition.

9.3.6 Obstacle Avoidance

One of the major issues for the obstacle avoidance was the threshold setting, that was hard to tune. Sometimes a bottle has been detected as an obstacle which forced the robot to drive around it. With the better safe then sorry approach false positives were favoured and yet it managed to touch slightly one of the obstacles during the competition.

9.3.7 IR sensors on Belt

The infrared sensors that should have detected the bottles on the conveyor belt were the weakest part during the competition. The infrared on the bottom was the only sensor working reliably, whereas the front and back infrared sensors did not manage to see a bottle, even though this has worked well in the testing phase. For future work, such a delicate and normally easy to fix issue should be taken care of soon enough and tested with the exact same settings as during the competition

9.4 Strategy

The strategy was a rather open and flexible strategy. The robot navigates to the first goal in the list, in this case to the zone on the grass, and from there searches the bottles. Given the high number of bottles in the arena, it has been distracted by some bottles already before it reaches the goal point, or in the beginning of the lookout phase. From there it entered the new lookout stages and was distracted even more. This strategy worked well for testing and also for the real competition. Such an adaptive strategy was not only more interesting to work with but in some cases also even more efficient than the fixed path implementations.

9.5 Assembly and Integration

The downside of the assembly is that it takes a lot of time to fully prepare the robot for the competition. And since the wheels repeatedly detached during the testing phase, they have been screwed tightly onto the axles and the screw has been kept in place by glue. This glue has been applied just before the competition and was a critical point to have a 10 minute autonomous robot.

9.5.1 Energy / Batteries / Voltage Regulators, etc.

The powering of the robot was no problem. Even with a runtime of 10 minutes and fullspeed on all four motors the robot had enough energy to continue. The same was valid for a running Odroid while powering the Arduino and the Raspberry Pi. During the testing phase, it has shown that one battery was sufficient for the three processing units. Therefore only one of the voltage regulators has been used. It was a good idea to implement this already on the PCB with the backup connector for the Raspberry Pi at the end of the 10 Ampere voltage regulator. It would not have worked the other way round since the current provided by the smaller regulator was of 3 Ampere which clearly was not enough, especially during the booting of the Odroid and Raspberry Pi at the same time.

9.5.2 Budget Management

The budget was never really an issue and even some days before the competition there was enough leftovers to buy another set of wheels for testing and the replacements of the motors that have been broken. For a future project, it is advisable to invest more money into high-end motors instead of the cheap solution from Pololu. The stock of electrical components and the fact that the Ateliers counted as internal budget were very helpful to stay within the limits of the external budget.

9.5.3 Risk Analysis

The risks that have been identified could mostly be mitigated during the testing phase. However, there were constantly arising new problems even on the morning of the competition itself. At some point it was not possible to mitigate all of them which led to the fact that some of the risks simply had to be accepted.

10 Conclusion

10.1 Lessons learned

When looking back at the beginning of the project, one can say that we have learned a lot and that we would probably not build the robot in the same manner as we did. Therefore, this section is dedicated to list all the important lessons that we have learned during the development and testing phase.

Time management First of all it is important to mention that the time assigned for this project is considerably short. Due to the fact that in the end only little time could be spent on the testing phase, it is advisable to start the project earlier than in the first week. It might be helpful to begin brainstorming before the semester has started to not loose the first two weeks on this part. However, it is essential that the project and the goal is clearly defined during the brainstorming process, such that everybody is aware of what has to be achieved and what the final result should look like.

Design and Assembly The design, the manufacturing of the pieces and the assembly are time-consuming processes. Our biggest mistake for this part of the project was probably to wait for the milestone 3 before sending the pieces to the ateliers to be manufactured. This milestone was an important design review so we did not produce or order anything before this date. This resulted in late production, assembly and also a delayed testing phase.

PCB For the PCB design, the introduction courses (the one from Alessandro and from the PCB lab) were very helpful. At first the program seemed a bit overwhelming, but after some time, one could easily design a PCB for all the necessary circuits. It turned out that putting together all circuits into one PCB was a good idea, even though we might have thought about the places of the components a bit earlier.
Also it is important to label all the connectors and the parts of the components to make it easy for oneself and the people that use the PCB, to know which battery gives power to which converter, for instance. Furthermore, we have experienced one or the other short circuits where the lines of the PCB acted as a fuse and simply burned through. This could then easily be fixed by an additionally soldered wire on the bottom side of the PCB.

Software The software part has been handled all over the platform github. This is a useful tool that allowed us to work separately on different files and sometimes even separately on the same files. It also keeps track of the history and the changes and makes it easy to go back to a working project version if necessary. Also it was helpful to have at least two processing components such as the Odroid and the Arduino to distribute the workload on the team members. The third processing unit (the Raspberry Pi) was in terms of organizational reasons not really necessary.

Testing The testing phase is one of the most important stages of the project. At first, it is important to have subparts, be it mechanical or software, that work independently. Then one can start combining the subparts to make them work as a whole. This usually takes more time then foreseen, due to the fact that the interface has not been fully defined. One of the most crucial parts for our testing were the motors, since we couldn't test further without a mobile robot. Hence it is necessary to finish the hardware testing as soon as possible and to identify, analyze and mitigate possible risks.

10.2 Possibility of improvement

Even though the expectations were fully met there are some points that could be improved for the next time. First of all, the time management of all tasks needs to be shifted in advance, such that at least one week can be used for testing. It is crucial to test the fully assembled robot for all situations, since the robot competition has shown that all of a sudden previously overcome barriers seemed to be a problem again.
Another loss of time and money were the cheap motors bought from the internal budget at Pololu. An investment of more money into a high quality motor would have been worth it after the first few broken ones. To deal with these poor performing motors was a loss of valuable time and set back the whole project since without the motors other parts could not be tested.
Lastly, the coupling between the motors and the wheels and the choice of the wheels is crucial. Especially with these heavy wheels that have been chosen, a coupling was more than recommendable. Since all the moment was acting on the motor shaft this might also have been a possibility to damage the motors. In addition, with a better coupling, less friction would be present due to the selfmade plain bearings and the wheels would not detach as easily.

References

[1] *Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org.* http://www.python.org.

[2] Itseez. *Open Source Computer Vision Library.* https://github.com/itseez/opencv. 2017.

[3] *Viola Jones object detection framework.* https://en.wikipedia.org/wiki/Viola-Jones_object_detection_framework. Page Version ID: 768000189. 2017.

[4] V. Pierlot and M. Van Droogenbroeck. "A New Three Object Triangulation Algorithm for Mobile Robot Positioning". In: *IEEE Transactions on Robotics* 30.3 (June 2014), pp. 566–577. ISSN: 1552-3098. DOI: 10.1109/TRO.2013.2294061.

[5] *Robot Operating System.* http://www.ros.org/.

[6] Seo Naotoshi. *Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features).* http://note.sonots.com/SciSoftware/haartraining.html. Apr. 2017.

[7] *pythonprogramming.net.* https://pythonprogramming.net/loading-images-python-opencv-tutorial/.

[8] *"Pololu Motors 2288 Datasheet with encoder".* https://www.pololu.com/product/2288. Pololu.

[9] *The OpenCV Reference Manual.* 3.2.0.0. Itseez. 2017.

Appendices

Brainstorming

	Simplicity / Feasibility	Efficiency	Zone adaptation	Price	Speed	Expected Performance	Originality	Reusability	Real-life usability	Difficulty of Control	
Wheels	**12**	**7**	**4**	**11**	**11**	**8**	**3**	**11**	**8**	**10**	**85**
Wheel with pivots	8	6	9	8	10	10	6	10	7	8	82
Crawler (Caterpillar) (all versions)	**10**	**7**	**8**	**8**	**8**	**9**	**5**	**10**	**10**	**9**	**84**
Caterpillar advancement	6	5	7	8	6	6	9	10	7	6	70
Articulated Leg	1	4	11	5	6	2	9	10	9	4	61
Whegs	**11**	**7**	**10**	**9**	**10**	**9**	**7**	**11**	**9**	**9**	**92**
Jumper	4	1	8	7	6	2	11	10	4	4	57
Spherical Structure around robot	8	6	7	7	6	7	10	10	5	6	72
Snakelike movement	5	4	6	4	6	4	10	10	5	3	57
Anchor	9	4	8	9	5	6	12	7	5	5	70
Leg with anchor	4	4	3	4	2	3	8	9	5	3	45
Reaction wheel	2	3	4	2	4	5	12	10	3	1	46
Wheels on Wheels (Epi.q)	7	6	9	7	9	9	10	10	7	9	83
Wheel with anchor for obstacles	6	5	4	6	4	4	8	10	4	3	54
Transforming wheels	7	5	8	8	8	8	11	10	9	7	81
Spider like movement	5	4	8	4	4	6	12	8	6	3	60
Ropes	5	4	7	9	6	5	9	9	4	7	65
Rails	5	1	2	4	6	3	7	8	2	5	43
Tilting mechanism maybe with RW	5	5	6	4	6	6	9	8	5	3	57

Figure 46: Decision matrix for locomotion principles after brainstorming

	Simplicity / Feasibility	Efficiency	Zone adaptation	Price	Speed	Expected Performance	Originality	Reusablity	Real-life usablity	Difficulty of Control	
Wheelbarrow	9	6	5	6	7	9	5	10	10	9	76
Bag	11	6	4	11	9	9	5	9	9	9	82
Container with moving bottom	**10**	**7**	**6**	**9**	**10**	**10**	**6**	**10**	**10**	**11**	**89**
Conveyor (reverse)	**11**	**7**	**7**	**8**	**9**	**9**	**7**	**10**	**10**	**8**	**86**
Blow	7	4	5	7	6	5	7	9	5	4	59
Throw	9	5	6	8	8	6	4	10	7	6	69
Kick	7	4	6	8	7	6	7	10	6	6	67
Screw Archimedes	7	6	5	7	8	8	12	9	7	6	75
Limit number of bottles (f.ex. 6)	9	7	5	8	5	7	8	10	8	7	74
V-shape with lifting barrier	10	7	4	9	9	9	6	10	9	10	83
1-by-1	9	0	0	5	4	6	5	7	2	5	43
Ropes	6	2	3	7	5	1	7	7	2	1	41
Double bottom structure	**11**	**6**	**6**	**8**	**10**	**10**	**7**	**9**	**9**	**9**	**85**
Six Pack	8	5	5	5	8	7	9	10	7	5	69
Home Tower Strategy	7	5	8	5	9	7	9	10	7	4	71

Figure 47: Decision matrix for storage principles after brainstorming

	Simplicity / Feasibility	Efficiency	Zone adaptation	Price	Speed	Expected Performance	Originality	Reusability	Real-life usability	Difficulty of Control	
Sticky material	11	7	6	10	9	9	5	7	6	10	80
Net	7	5	1	11	5	4	7	10	4	2	56
Grab by sitting on it	**11**	**8**	**7**	**10**	**11**	**12**	**11**	**11**	**9**	**11**	**101**
Articulated gripper	5	5	7	4	7	9	7	9	9	3	65
Vaccuum	7	6	3	4	9	8	8	7	8	8	68
Blower	9	4	3	6	7	5	7	8	7	7	63
Turning structure with Archemedes	9	5	8	7	7	10	12	11	9	7	85
Chameleon tongue	6	3	6	7	7	5	10	7	6	4	61
Lasso	3	2	3	12	7	4	7	8	4	3	53
Sweep / Broom	**11**	**6**	**5**	**10**	**8**	**9**	**6**	**11**	**11**	**10**	**87**
Shovel	**12**	**6**	**5**	**11**	**9**	**10**	**5**	**10**	**10**	**9**	**87**
Conveyor belts	6	5	4	7	7	8	5	9	7	6	64
Tunnel on Wheels	9	6	5	9	9	9	8	11	7	10	83
Pushing (f.ex. In V-shape)	12	6	5	10	9	9	3	10	9	10	83
Tentacles	3	2	2	3	5	2	8	5	1	2	33
Hooks (f.ex. On conveyor belt)	3	4	3	3	3	3	3	3	3	2	30
Liquid Nitrogen	4	3	2	4	5	4	8	2	0	2	34
Kicking	9	4	7	9	6	6	8	10	7	7	73
Moving barrier	3	2	3	5	2	2	8	7	3	2	37
Movable arm with vacum/blow grip	7	6	7	5	4	9	6	10	7	6	67

Figure 48: Decision matrix for grabbing and releasing principles after brainstorming

ROS Messages

Table 4: Custom ROS messages

pos_msg	**ir_array_msg**	**bottle_msg**	**bumper_msg**
float32 x	int16 ir_front	int8 num_bottles	bool bumper_left
float32 y	int16 ir_bottom	float32 final_x_angle	bool bumper_right
float32 theta	int16 ir_back	float32 final_y_angle	
	int16 ir_center		
	int16 ir_left		
	int16 ir_right		
enc_array_msg	**pos_360_msg**		
int32 l_encoder	x		
int32 r_encoder	y		
int32 c_encoder	theta		
	home_angle		

PCB

H-bridge

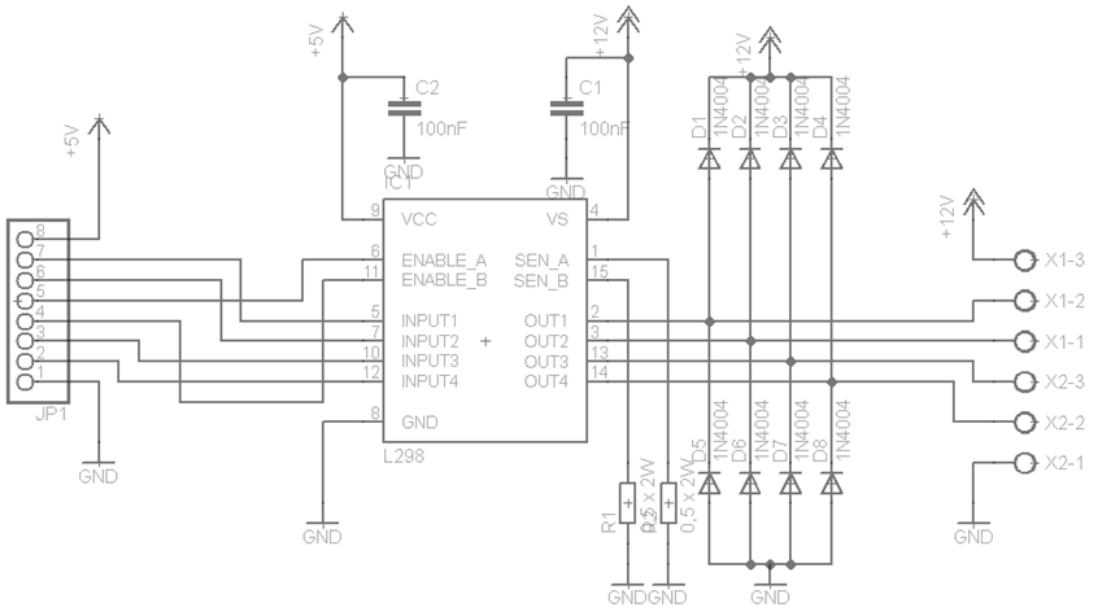


Figure 49: Schematic for the motor driver that has been used as reference

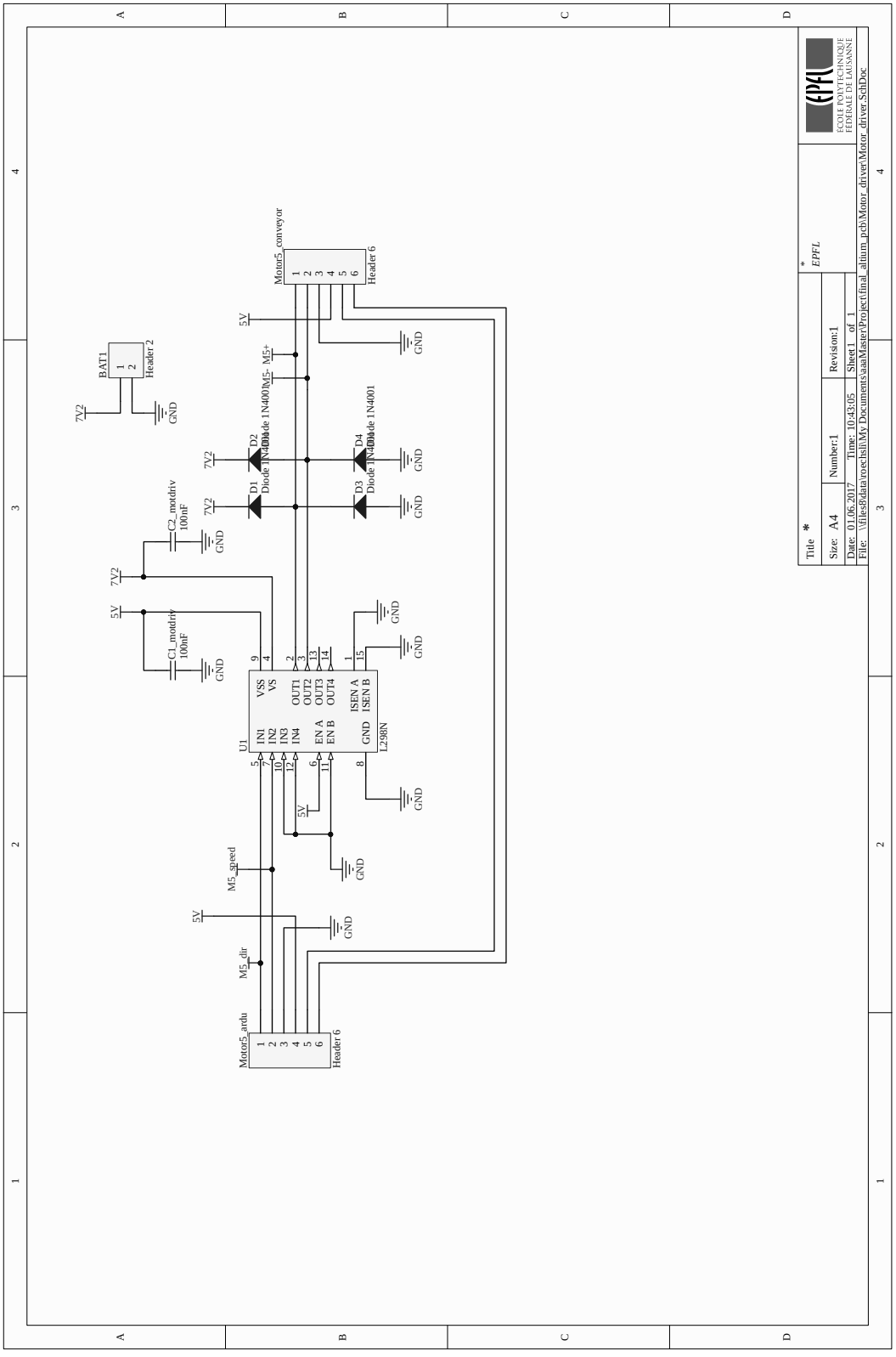


Figure 50: The motor driver circuit schematic that has been drawn

IMU circuit

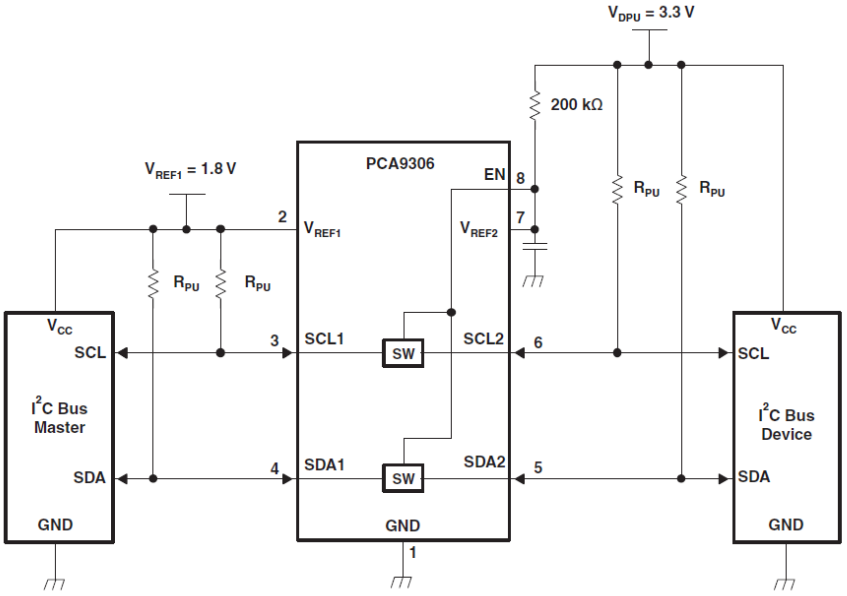


Figure 51: The IMU level shifter schematic from the datasheet that has been use as reference

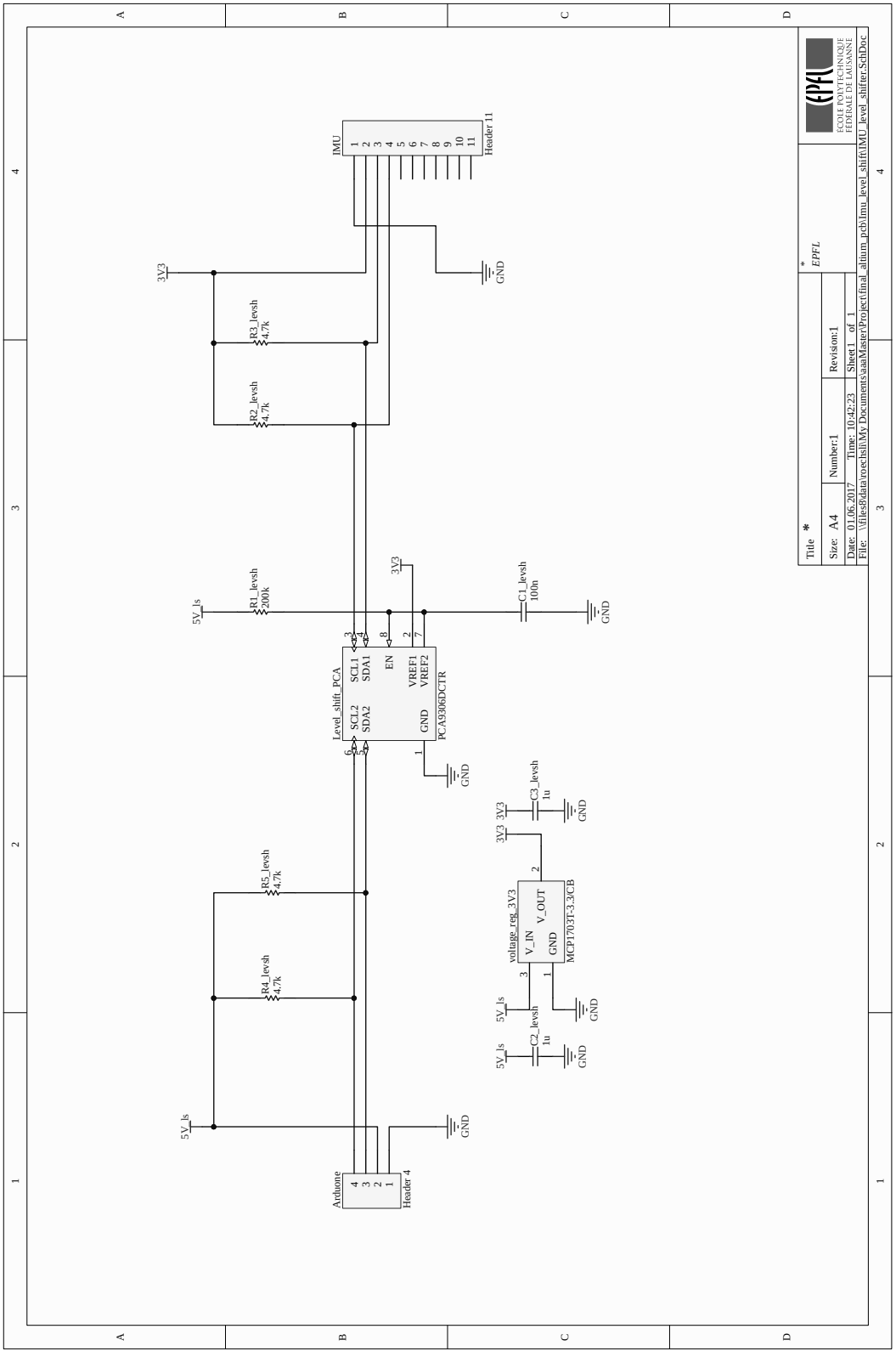


Figure 52: The IMU level shift circuit schematic that has been drawn

DC converter circuits

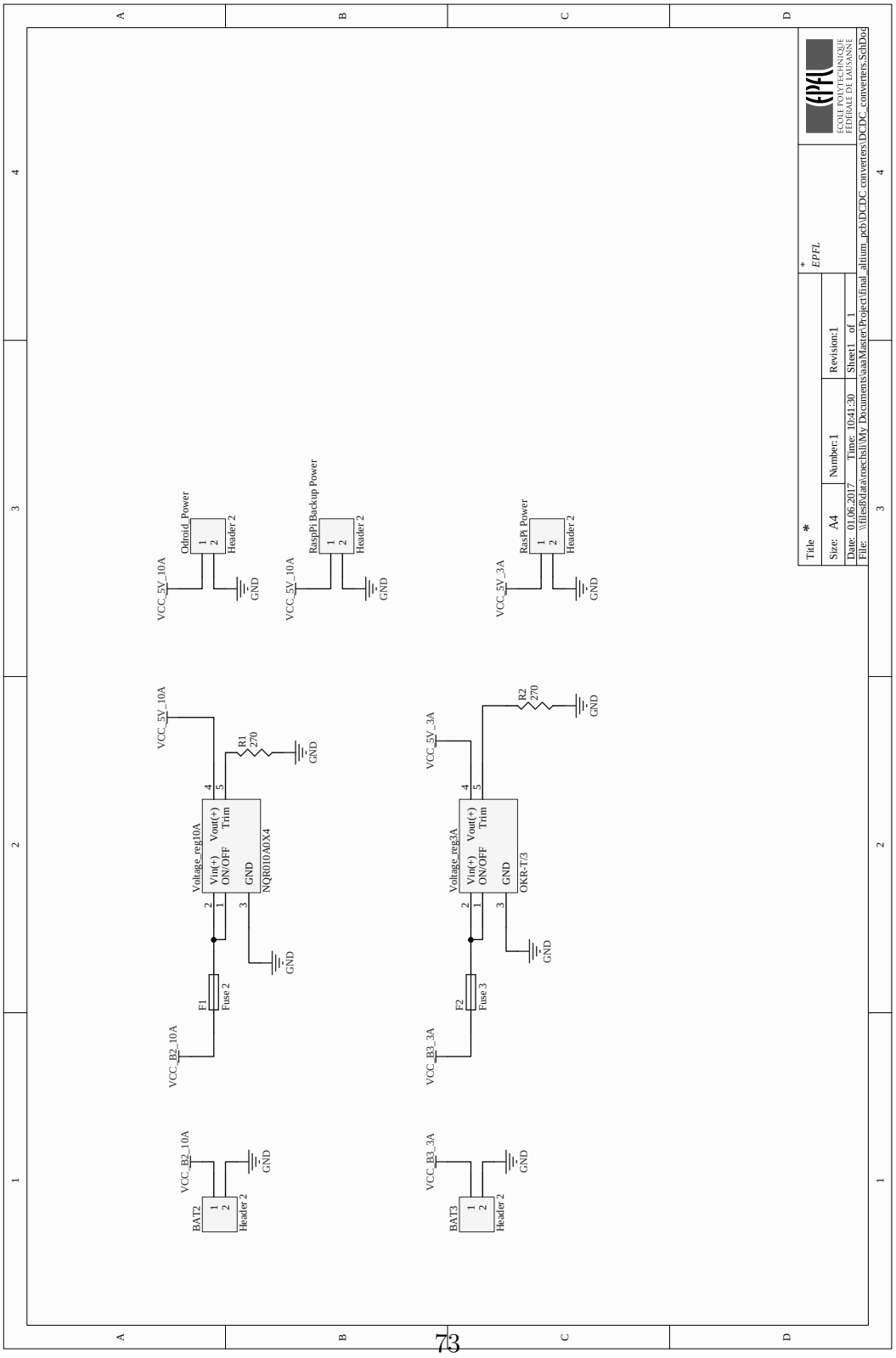


Figure 53: The DC-DC converter circuit that has been designed

Wheel motor connections

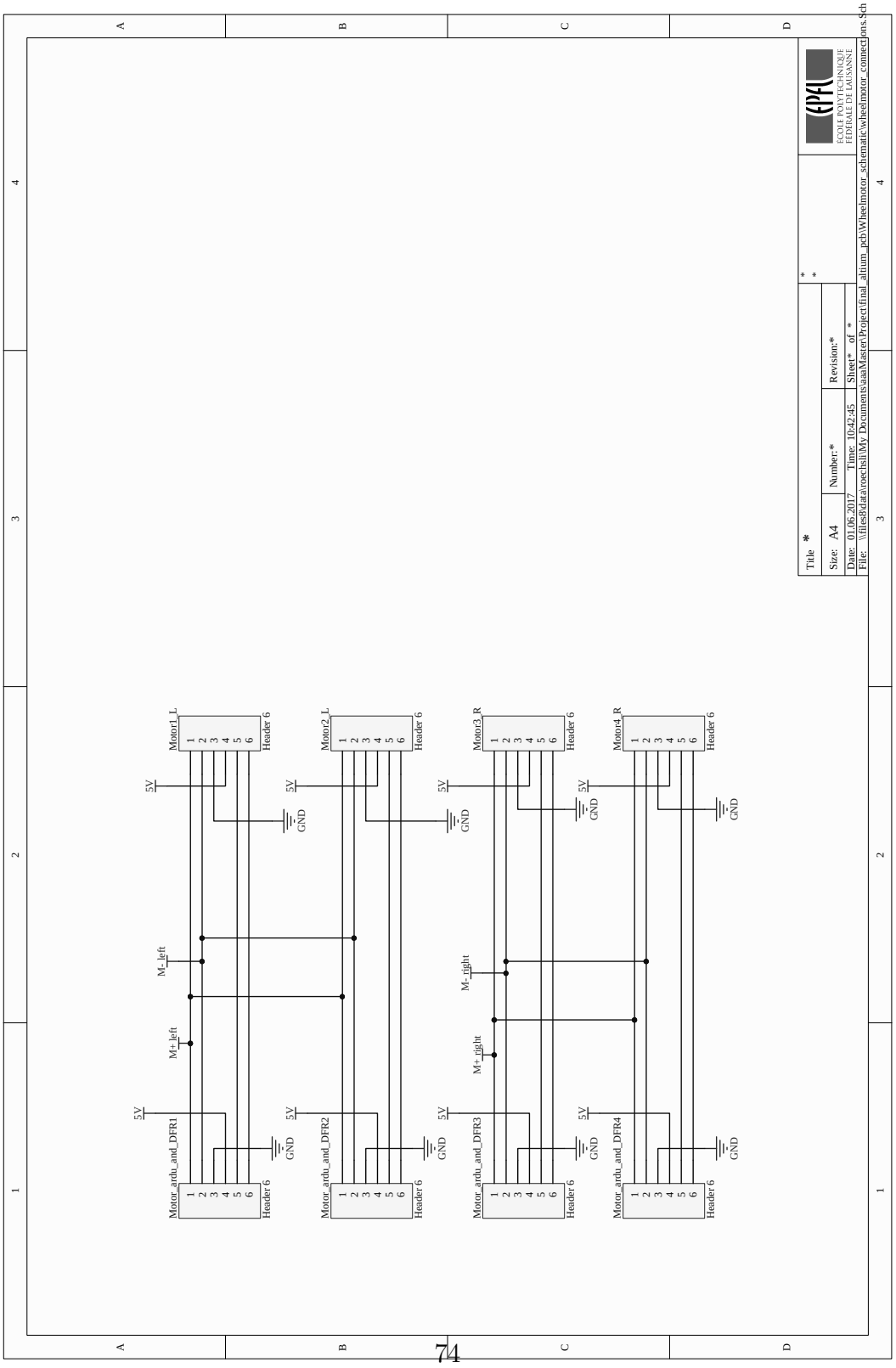


Figure 54: The motor and Arduino connecting circuit schematic that has been drawn

Final PCB design

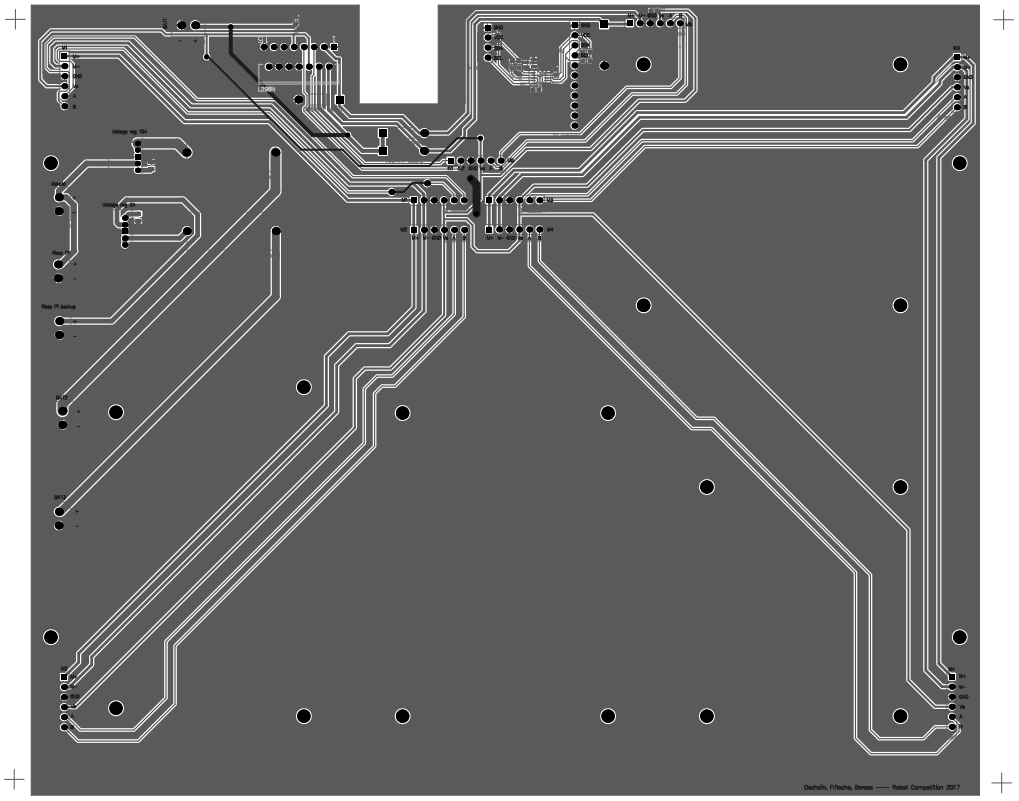


Figure 55: The final PCB ground plate layout that has been drawn